Highlights

Comparison of substructured non-overlapping domain decomposition and overlapping additive Schwarz methods for large-scale Helmholtz problems with multiple sources

Boris Martin, Pierre Jolivet, Christophe Geuzaine

- This paper presents the first detailed numerical comparison (computation time and memory use) of overlapping and non-overlapping domain decomposition methods for solving the Helmholtz equation on massively parallel supercomputers.
- Experiments are performed on a realistic geophysical test-case, the GO_3D_OBS velocity model, with high-order finite elements on adapted unstructured tetrahedral grids, with many sources.
- Both methods are shown to scale well in terms of memory and load-balancing, with a slightly better balancing for the non-overlapping method.
- The non-overlapping method is shown to outperform the overlapping method by a factor close to 2 in all problems sizes, in both memory and time.

Comparison of substructured non-overlapping domain decomposition and overlapping additive Schwarz methods for large-scale Helmholtz problems with multiple sources⋆

Boris Martin[a,1], Pierre Jolivet[b] and Christophe Geuzaine[a,*]

[a]University of Liège, Montefiore Institute, Liège 4000, Belgium
[b]Sorbonne Université, CNRS, LIP6, Paris France

ARTICLE INFO

Keywords:
Domain decomposition methods
Helmholtz equation
Optimized Schwarz Method
Optimized Restricted Additive Schwarz
High-performance scientific computing

ABSTRACT

Solving large-scale Helmholtz problems discretized with high-order finite elements is notoriously difficult, especially in 3D where direct factorization of the system matrix is very expensive and memory demanding, and robust convergence of iterative methods is difficult to obtain. Domain decomposition methods (DDM) constitute one of the most promising strategies so far, by combining direct and iterative approaches: using direct solvers on overlapping or non-overlapping subdomains, as a preconditioner for a Krylov subspace method on the original Helmholtz system or as an iterative solver on a substructured problem involving field values or Lagrange multipliers on the interfaces between the subdomains. In this work we compare the computational performance of non-overlapping substructured DDM and Optimized Restricted Additive Schwarz (ORAS) preconditioners for solving large-scale Helmholtz problems with multiple sources, as is encountered, e.g., in frequency-domain Full Waveform Inversion. We show on a realistic geophysical test-case that, when appropriately tuned, the non-overlapping methods can reduce the convergence gap sufficiently to significantly outperform the overlapping methods.

1. Introduction

Full Waveform Inversion (FWI) is a powerful technique for seismic imaging [1], which can estimate subsurface parameters such as velocities by fitting simulated waveform data to observed data. The fitting problem is formulated as an optimization problem, which is typically solved using gradient-based methods, with the gradient computed using adjoint methods. In the frequency-domain, both the misfit evaluation and the adjoint problem require solving a Helmholtz-like PDE with the current subsurface model, e.g., with finite differences or finite elements, resulting in solution of sparse, complex and indefinite linear systems. The size of these linear systems is directly related to desired imaging resolution and thus the frequency: in 3D the size grows at least cubically with frequency. Direct solvers are robust but scale poorly [2], although recent progress such as Block Low-Rank compression [3, 4, 5, 6, 7, 2] can somewhat alleviate these issues. When the number of unknowns is on the order of the hundreds of millions, iterative methods become an interesting alternative, despite the challenges of designing efficient preconditioners for indefinite problems [8]. Domain decomposition methods (DDM) are a popular choice, leveraging the power of sparse solvers on smaller subdomains. Their main drawback is that, being iterative methods, each RHS requires the full iterative procedure to be applied, making these methods a priori less well-suited for a large number of sources. Nonetheless, DDM has been shown to be able to outperform direct solvers in some cases, even with a fairly large number (130) of sources [9].

There exists a wide range of DDMs, with either overlapping or non-overlapping subdomains. On the one hand, the overlap tends to help achieve faster convergence, but at the cost of larger subdomains. On the other hand, non-overlapping methods work on a substructured problem involving a reduced set of unknowns defined on the interfaces between subdomains, resulting in smaller problem sizes. Thus, a non-overlapping approach can theoretically mitigate

*Corresponding author
boris.martin@uliege.be (B. Martin); pierre@joliv.et (P. Jolivet); cgeuzaine@uliege.be (C. Geuzaine)
https://boris-martin.github.io/ (B. Martin); https://joliv.et/ (P. Jolivet); https://people.montefiore.uliege.be/geuzaine/ (C. Geuzaine)
ORCID(s): 0000-0003-1398-091X (B. Martin); 0009-0000-3410-0884 (P. Jolivet); 0000-0001-9970-358X (C. Geuzaine)
[1]B. Martin is funded by a FRIA doctoral fellowship of the F.R.S-FNRS, Belgium.

the computational expense associated with Krylov methods, which require the storage and orthogonalization of a Krylov subspace basis.

Applying an iterative procedure to a large number of sources essentially leads to a cost proportional to that number, but the efficiency of the process can be improved with a parallel implementation that batches large portions of the computations (called *pseudo-block Krylov Methods*), and by using iterative methods that can reduce the total number of iterations for multiple sources. This can be achieved by *recycling* a part of the subspace [10] from previous solves, or, when solving for all sources simultaneously, by using a block Krylov method, where information is shared between the different sources [11]. Essentially, these methods reduce the iteration count, at the expense of additional work per iteration (e.g., additional inner products). Depending on the problem and the context, they may or may not offer a favorable trade-off compared to pseudo-block methods. In particular, the reduction in iteration count is difficult to predict, and may depend on the properties of the matrix and the space spanned by the right-hand sides, the latter being physically related to the choice of source locations [12].

In this work, we compare the Optimized Restricted Additive Schwarz (ORAS) method [13, 14, 15], which is an overlapping preconditioner for the Helmholtz equation, with the non-overlapping Optimized Schwarz method (OSM) [16, 17, 18, 15, 19], also known as FETI-2LM [10]. We focus on analyzing computational times and memory usage for increasingly large 3D problems solved with high-order finite elements on adapted unstructured tetrahedral grids. Meshes are partitioned using METIS [20] and GMRES serves as the Krylov subspace solver.

The paper is organized as follows. We first introduce the Helmholtz problem and its discretization, as well as our test case of interest in section 2. The two domain decomposition methods of interest are presented in section 3, while the convergence criterion and the parameters for both methods are discussed in section 4. Section 5 then analyzes the results of the numerical experiments in terms of both computational time and memory usage. Conclusions and perspectives for future work are finally given in section 6.

2. Problem setting

2.1. Helmholtz problem

We focus on the constant-density, variable-velocity and damping-free Helmholtz equation in a parallelepipedic domain Ω. For simplicity, we consider a low-order approximation of the Sommerfeld radiation condition, in the form of a Robin boundary condition on the boundary $\Gamma^\infty = \partial\Omega$ of Ω. With a point source located at $x = x_s \in \Omega$, the problem writes:

$$\begin{cases} -\Delta u - k^2 u = \delta(x - x_s) & \text{in } \Omega \quad \text{(Helmholtz equation)}, \\ \partial_{\boldsymbol{n}} u - \imath k u = 0 & \text{on } \Gamma^\infty \quad \text{(radiation condition)}. \end{cases} \tag{1}$$

The wavenumber k is a function of space and frequency, and is defined as $k = \frac{\omega}{c(x)}$, where ω is the angular frequency and $c(x)$ is the velocity field. Our main interest is to solve the equation for multiple source positions x_s, which will translate as different right-hand sides in the linear system after finite element discretization. We use a standard (continuous) Galerkin formulation on a tetrahedral mesh adapted to the local wavelength with third-order hierarchical basis functions [21]. The resulting linear system is denoted $A\vec{u} = \vec{f}$, where $\vec{u}$ is the unknown vector, $\vec{f}$ the discretized source, and A is a sparse, complex, symmetric (but not Hermitian) matrix. Except at the lowest frequencies, the matrix is indefinite.

2.2. A geophysical benchmark: the GO_3D_OBS velocity model

We focus on the GO_3D_OBS velocity model [22], a publicly available 3D geomodel representing a subduction zone, inspired by the geology of the Nankai Trough. The model represents a $175\,\mathrm{km} \times 100\,\mathrm{km} \times 30\,\mathrm{km}$ domain with a layer of water and complex geological structures. We focus on the $102\,\mathrm{km} \times 20\,\mathrm{km} \times 28\,\mathrm{km}$ subset of the model, called the target, suggested in [22] and illustrated on Fig. 1. Velocity ranges from $1500\,\mathrm{m\,s^{-1}}$ (water) to $8500\,\mathrm{m\,s^{-1}}$ in some areas. The velocity data is available as point values on a regular grid with a 25 m spacing, which is interpolated trilinearly on the nodes of the adapted tetrahedral finite element mesh (cf. Fig. 2). From this trilinear interpolation, we project $\frac{1}{c(x)}$ such that k is piecewise linear and continuous on the finite element mesh.

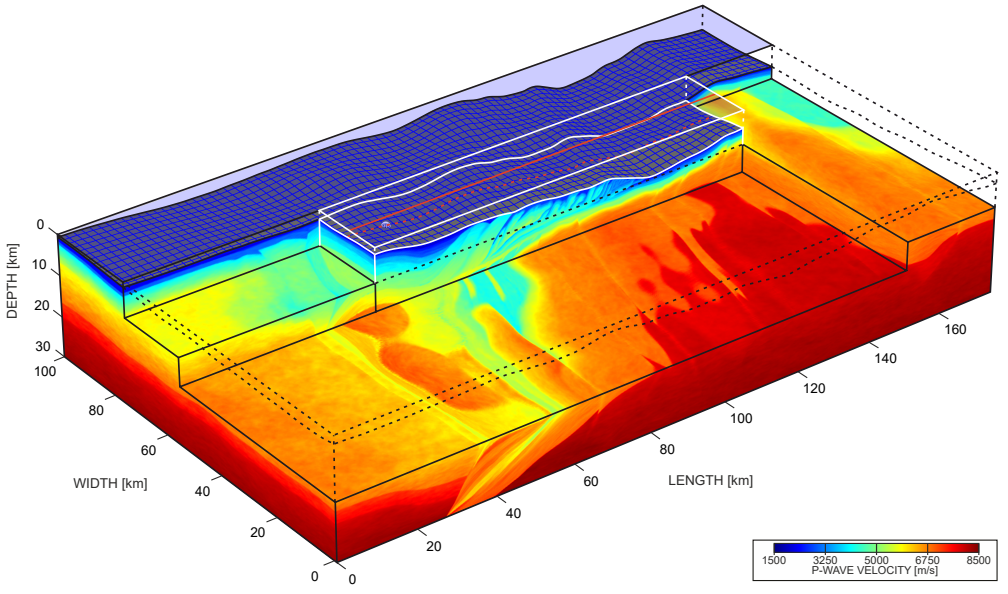

Figure 1: The GO_3D_OBS velocity model and the target region [22].

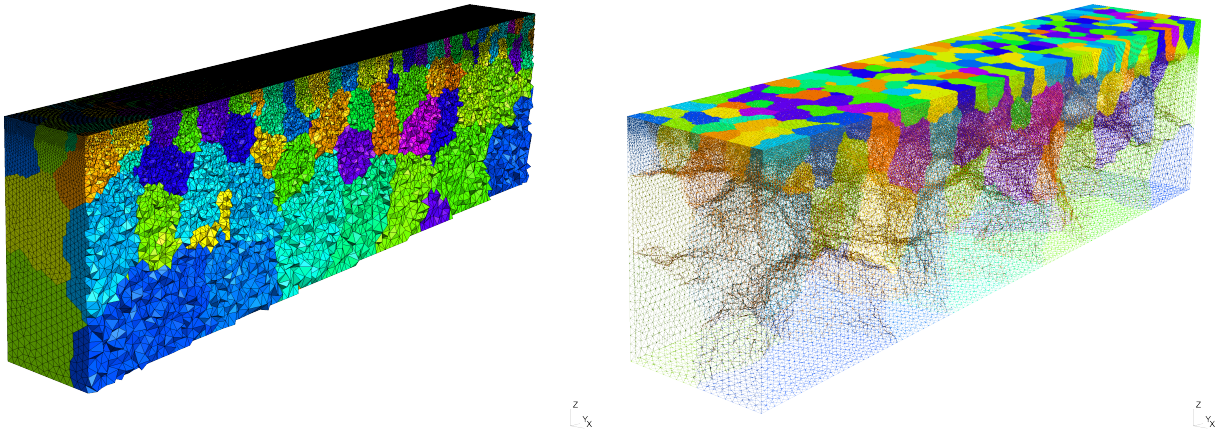

Figure 2: Typical mesh (here with 256 partitions), with element sizes adapted to the local wavelength at 2 Hz.

3. Domain decomposition methods

Let us partition the domain Ω into N subdomains Ω_i (for $i = 1, .., N$). The subdomains can either be overlapping, or non-overlapping. In both cases, the boundary of Ω_i is the union of a (possibly empty) portion of Γ^∞ and the *artificial* interface $\Sigma_i = \partial\Omega_i \setminus \Gamma^\infty$. If $\mathcal{O}(i)$ denotes the set of indices of the neighbors of subdomain i, then $\Sigma_i = \cup_{j\in\mathcal{O}(i)}\Sigma_{ij}$, where:

- in the non-overlapping case, $\Sigma_{ij} = \Sigma_{ji}$ is the surface separating Ω_i and Ω_j;
- in the overlapping case, Σ_{ij} is the part of $\partial\Omega_i$ inside Ω_j, i.e., $\Sigma_{ij} = \partial\Omega_i \cap \Omega_j$; it is thus distinct from Σ_{ji}, which lies in Ω_i.

On each subdomain, we solve a local problem with impedance-type transmission conditions on the artificial interface:

$$\begin{cases} -\Delta u_i - k^2 u_i = f_i & \text{in } \Omega_i, & \text{(Helmholtz equation)}, \\ \partial_{\boldsymbol{n}_i} u_i - \imath k u_i = 0 & \text{on } \partial\Omega_i \cap \Gamma^\infty & \text{(radiation condition)}, \\ \partial_{\boldsymbol{n}_i} u_i - \mathcal{S} u_i = \partial_{\boldsymbol{n}_i} u_j - \mathcal{S} u_j & \text{on } \Sigma_{ij},\ \forall j \in \mathcal{O}(i) & \text{(transmission condition)}, \end{cases} \tag{2}$$

where $f_i = \delta(x - x_s)$. The operator $\mathcal{S}$ is chosen to improve convergence of DDM. The simplest choice is $\mathcal{S} = \imath k$ [23, 16]. In this paper, we consider second-order operators $\mathcal{S} = \alpha + \nabla_\Sigma \cdot (\beta \nabla_\Sigma)$, where the two complex-valued coefficients α and β are optimized numerically, and ∇_Σ is the surface gradient operator.

Equation (2) can yield a simple iterative procedure (a Schwarz method) where, at step $m+1$, one can compute $u_i^{(m+1)}$ as a function of $u_j^{(m)}$ for all $j \in \mathcal{O}(i)$. The process is repeated until convergence. In practice, depending on whether the subdomains are overlapping or not, two common algorithms [15] are the Optimized Restricted Additive Schwarz (ORAS) preconditioner (with overlap), and the substructured solver with unknowns on the interfaces, called hereafter the Optimized Schwarz Method (OSM). We recall both methods in what follows.

3.1. Overlapping subdomains: the ORAS preconditioner

From the local solutions u_i on each subdomain, we can compute an approximation of the global solution u by gluing the local solutions together using a partition of unity. If $E_i(u_i)$ is the extension by zero of u_i on Ω and $\chi_i(x)$ forms a partition of unity, the global solution can be expressed as:

$$u(x) = \sum_{i=1}^{N} \chi_i(x) E_i(u_i). \tag{3}$$

The ORAS procedure in its continuous form consists of building a sequence $u^{(m)}$ where $u^{(m+1)}$ is computed by restricting $u^{(m)}$ to each subdomain, solving the local problem eq. (2) (with right-hand side provided by the residuals related to $u^{(m)}$ in the overlap, transmitting information between subdomains) and then summing the results using eq. (3). This procedure is a fixed-point iteration that will converge if the transmission conditions are appropriately chosen.

At the discrete level, the restrict, solve, extend, and combine procedure has a natural algebraic equivalent. Let R_i be the restriction matrix that selects the entries related to subdomain Ω_i from a global vector. R_i^T then extends by zero a local vector to the global space. The discrete partition of unity is a set of diagonal matrices D_i such that $\sum_{i=1}^{N} R_i^T D_i R_i = I$. Typically, the partition is chosen as Boolean, and each degree of freedom (DoF) is *owned* by a single subdomain. Each subdomain solves a problem involving its owned DoFs and some neighboring non-owned DoFs, and outputs new values for the owned ones.

Combining the restrictions, local solves, and extensions weighted by the partition of unity, the ORAS method is expressed as follows:

$$\vec{u}^{(k+1)} = \vec{u}^{(k)} + M_{\text{ORAS}}^{-1}(\vec{f} - A\vec{u}^{(k)}),$$

with the ORAS preconditioner M_{ORAS}^{-1} defined by

$$M_{\text{ORAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i.$$

The matrix A_i is the matrix arising from the discretization of the subproblems eq. (2), with the transmission conditions. This fixed-point iteration can be seen as a preconditioned Richardson iteration, where the preconditioner is the ORAS method. This preconditioner can be used with other Krylov methods, such as GMRES. In that case, we solve by right-preconditioning:

$$A M_{\text{ORAS}}^{-1} \vec{y} = \vec{f},$$

and then $\vec{u} = M_{\text{ORAS}}^{-1} \vec{y}$ is the solution of $A\vec{u} = \vec{f}$.

To summarize, the ORAS method is used as a preconditioner to solve the complete system with GMRES or another Krylov method. The residual involves the FEM matrix A and the preconditioner application involves solving each subproblem once (with A_i) and then using the extension and partition of unity to obtain a global vector as output. When using GMRES, the computational cost resides mostly in the preconditioner application, the sparse matrix-vector product and the Gram–Schmidt procedure to build an orthonormal basis of the Krylov subspace.

3.2. Non-overlapping case: substructured Schwarz method (OSM)

In the absence of overlap, the interfaces Σ_{ij} and Σ_{ji} are the same, with opposite normals. On each interface, we define $g_{ji} = \partial_{\boldsymbol{n}_i} u_j - \mathcal{S} u_j$. We denote g (without subscript) as the tuple of all coupling variables containing, for all subdomains i, the g_{ij} corresponding to each subdomain j neighboring i. Using these notations, eq. (2) can be rewritten in the form of eq. (4), whose unknowns are the local fields u_i and a tuple of traces g:

$$\begin{cases} -\Delta u_i - k^2 u_i = f_i & \text{in } \Omega_i & \text{(Helmholtz equation)}, \\ \partial_{\boldsymbol{n}_i} u_i - \imath k u_i = 0 & \text{on } \partial\Omega_i \cap \Gamma^\infty & \text{(radiation condition)}, \\ \partial_{\boldsymbol{n}_i} u_i - \mathcal{S} u_i = g_{ji} & \text{on } \Sigma_{ij},\ \forall j \in \mathcal{O}(i) & \text{(transmission condition)} \end{cases} \tag{4}$$

At convergence (i.e., for the correct g), the solution is continuous, and $u_i = u_j$ on Σ_{ij}. Combined with the opposite signs of the normals, we obtain the following equation that links the two auxiliary variables (called *interface fields*) and the wavefield, for all pairs of neighboring subdomains:

$$g_{ij} + g_{ji} = -2\mathcal{S} u \text{ on } \Sigma_{ij}. \tag{5}$$

This relation allows us to design a (parallel) Schwarz method that iterates on the interface fields. We first solve, for all i, the subdomain eq. (4) for the current values of g_{ji} (for all relevant values of j), and then, using eq. (5), we compute new values of g_{ij} necessary to solve the problem on subdomain j.

Krylov acceleration The solution of a local problem eq. (4) depends on two sources: f_i and the interface fields g_{ji}. By linearity, these can be split as $u_i^{(m)} = v_i + \tilde{u}_i^{(m)}$, where the first part comes from f_i and is constant at each iteration, and the second part represents the wavefield generated by the waves coming from other subdomains, i.e., $\tilde{u}_i$ depends linearly on all coupling variables g_{ji}. The interface update can be rewritten as the affine fixed-point iteration eq. (6) for each coupling field:

$$g_{ij}^{(m+1)} = -g_{ji}^{(m)} - 2\mathcal{S}\tilde{u}_i^{(m)} - 2\mathcal{S} v_i. \tag{6}$$

Combining this for each interface equation, the global update can be written as $g^{(m+1)} = \mathcal{T} g^{(m)} + b$ where b represents the outgoing waves generated by all the sources f_i and $\mathcal{T}$ is an exchange operator acting on the tuple of coupling variables [15, Sec. 2.3.1]. This can be reformulated as the solution of the following substructured problem in terms of g:

$$(\mathcal{I} - \mathcal{T}) g = b. \tag{7}$$

The vector b is obtained by solving the local problems with homogeneous boundary conditions. Then, to apply $\mathcal{T}$, one solves the subproblems without f_i and with inhomogeneous boundary conditions.

At the discrete level, this system can be solved with a Krylov method such as GMRES (and the fixed-point iteration is actually the Richardson iteration applied to it). Once the interface fields are recovered, the complete wavefield can be computed by solving the local problems with both these boundary conditions and the local source. It is worth noting that eq. (7) can also be seen as a Schur complement: combining eq. (4) for all subdomains and eq. (5) for all interfaces yields a global system, and the substructured problem is actually its Schur complement with respect to all the u_i, and the fixed-point iteration is a block-Jacobi iteration applied to the Schur complement.

In this approach, GMRES is applied to the interface problem, without preconditioner, but with an operator that contains local solves. In this workflow, the cost is spread between the local solves, the assembly of inhomogeneous terms and the construction of an orthonormal basis of reduced dimension of the Krylov subspace.

3.3. A priori comparison of ORAS and OSM

Both ORAS and OSM are based on the principle of solving subproblems with transmission boundary conditions until the full solution is recovered. Thus, the main computational cost in both cases will be to factorize the local matrices (once) and to solve the resulting triangular systems at each iteration. The workflow is however different: ORAS is a preconditioner for solving the full problem iteratively, whereas in OSM the local solves are part of the operator itself, and no preconditioner is used. The OSM method should a priori be more economical for several reasons:

- The absence of overlap reduces the subdomain size, making the local solves cheaper, and the factorization less memory-intensive.
- The iterative method is applied on the interface fields, which leads to fewer unknowns (typically between 5 and 10 times less) than the full problem. This reduces the cost (in both time and memory) of the Gram–Schmidt orthogonalization procedure in GMRES.
- The global sparse matrix-vector product disappears. However, it is replaced by a sparse matrix-vector product (SpMV) on the local interface fields to compute the right-hand side of the local problems at the next iteration. Again, this should be an order of magnitude cheaper.

However, ORAS can converge faster thanks to the overlap, and is more flexible as it is a preconditioner. In particular, inexact subdomain solves can be used safely: they can slow down convergence but not yield an incorrect result. Conversely, an inexact solve in OSM will impact the resulting solution as it introduces an error in the operator instead of the preconditioner.

In the following sections, we will evaluate how the convergence rates differ between the two methods. Afterwards, we will run experiments on large problems with multiple sources to assess the efficiency of the two methods in terms of time and memory consumption in realistic application settings.

3.4. Implementation

For our experiments, we used Gmsh [24] to generate tetrahedral meshes, which are then partitioned using METIS [20]. We extended Gmsh to generate the overlaps needed for ORAS. The problem assembly is done with GmshFEM and GmshDDM [25, 26], a Gmsh-based FEM solver and its extension designed for non-overlapping domain decomposition methods. For both ORAS and OSM, GMRES is used through PETSc [27] and its binding to HPDDM [28], which provides a pseudo-block GMRES implementation suited for multiple RHSs. Both approaches use MUMPS [29, 30] as a solver (called through PETSc) to compute a LDL^T factorization of the subproblem matrices and solve the resulting triangular problems.

While ORAS with multiple RHS is straightforward with these tools (as the additive Schwarz preconditioner of PETSc is already adapted for multiple RHS), OSM is more complex. GmshDDM was first implemented as a custom matrix-vector product provided to PETSc. We upgraded it to handle matrix-matrix products, i.e., applying the $\mathcal{I} - \mathcal{T}$ operator to a batch of vectors. This required reformulating the artificial sources assembly as a mass-like linear operator that can be precomputed then applied to a batch of incoming waves efficiently. Finally, in OSM, the local problems have a block-triangular 2×2 structure, as the local wavefield can be computed first, and the updated outgoing wavefield depends on it. Solving directly this problem is inefficient, as we lose the symmetry and have a larger system. Exploiting the triangular structure allows us to take advantage of the symmetry of the two diagonal blocks (the Helmholtz operator and a mass-matrix on the interface), and can be done through PETSc with a *fieldsplit* preconditioner. We extended this preconditioner to handle efficiently the multiple RHS case, as the current version of PETSc simply loops over all of these. Links to the modified source code are available in section 7.

4. Choice of parameters and convergence criterion

It is well-known that appropriate transmission conditions are critical to achieve good convergence in DDM, both without overlap [31, 18] and with overlap [32]. The simplest choice is $\mathcal{S} = \imath k$ but more complex conditions can be chosen: second-order operators using the surface Laplacian, Perfectly Matched Layers [33, 34] or high-order rational functions [35, 36]. We choose to use second-order operators of the form , $\mathcal{S} = \alpha + \nabla_\Sigma \cdot (\beta \nabla_\Sigma)$, which provide a good compromise between cost and performance. The complex coefficients α and β should be chosen to balance the convergence rate of all the wave modes (propagating, grazing, evanescent). In this work we numerically optimized them to find the coefficients leading to the minimal number of Krylov iterations. In addition, for ORAS we always choose

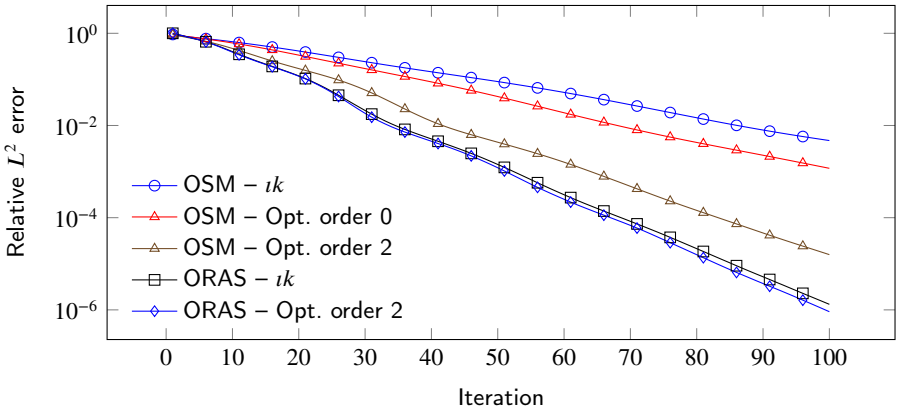

Figure 3: Convergence of the L^2 error for the homogeneous medium with 256 subdomains. Order 0 and 2 refer to the optimized transmission conditions of that order, and $\imath k$ to the basic impedance condition. In the ORAS case, $\imath k$ is actually the most effective 0th-order condition, so no additional curve is shown.

the minimal overlap size, i.e., each subdomain for ORAS is constructed by adding one layer of elements (connected by at least one node) to the elements of the corresponding OSM subdomain.

Since ORAS and OSM have different residuals (preconditioned or unpreconditioned residual from the matrix in ORAS, residual on the interface problem in OSM), we need to define a common convergence criterion. The evolution of the L^2 error over iterations is measured as well as the residuals, to find stopping criteria that yield similar accuracies with both approaches. With ORAS, the unpreconditioned residual (from the right-preconditioned GMRES) is used, as all experiments showed that right-preconditioned GMRES was slightly faster, and required a looser tolerance. In this section, we perform our experiments with double precision scalars. Single precision scalars give similar results until we lower the stopping residual below about 10^{-4}.

4.1. Homogeneous medium

We begin by simulating the $102\,\mathrm{km} \times 20\,\mathrm{km} \times 28\,\mathrm{km}$ target from GO_3D_OBS model by considering that it is entirely made of water ($c = 1500\,\mathrm{m\,s^{-1}}$), with a source at 1 Hz positioned near the surface and impedance conditions applied on the 6 faces of the box. This leads to around $68 \times 13.3 \times 18.3$ wavelengths in the three directions.

The box is discretized with an unstructured tetrahedral mesh with prescribed uniform mesh density of 4 points per wavelength. With third-order finite element basis functions this leads to around 20 million unknowns. The METIS graph partitioner is used to create 256 subdomains.

Fig. 3 shows the convergence of the L^2 error with respect to the converged solution of the discrete problem for both methods with different transmission conditions: the basic impedance condition (with $\alpha = \imath k, \beta = 0$), the optimized zeroth-order condition with $\beta = 0$ ("Opt. order 0") and the optimized second-order condition ("Opt. order 2"). In the ORAS case, no zeroth-order optimization is shown, as the basic impedance condition outperformed all the trials we performed. The optimized coefficient for OSM with zeroth-order condition is $\alpha = \imath k \frac{1+i}{\sqrt{2}}$, while for the second-order conditions we used $\alpha = \imath k \frac{1}{\sqrt{2}}, \beta = \frac{\imath}{2k}\frac{1-\imath}{\sqrt{2}}$ in OSM and $\alpha = \imath k \frac{1}{\sqrt{2}}, \beta = -\frac{\imath}{2k}\frac{1-\imath}{\sqrt{2}}$ in ORAS.

Clearly, ORAS has a better convergence than OSM, whether the transmission conditions are optimized or not. However, optimizing them dramatically reduces the gap, making OSM almost as effective. Fig. 4 shows the relationship between the relative error and the relative residual. We observe that, for a target error, the OSM tolerance should be similar to the ORAS one. Furthermore, the gains of using second-order conditions with ORAS are negligible, whereas they are crucial for OSM to achieve good convergence.

4.2. Heterogeneous medium

We now consider the same geometry but with the velocity field from the GO_3D_OBS dataset. The mesh size is locally adapted to keep 4 points per wavelength. The adaptivity allows us to use fewer elements than in the homogeneous case, as the wavelength is around 5 times smaller in water than in soil. In order to keep a problem of similar size, the

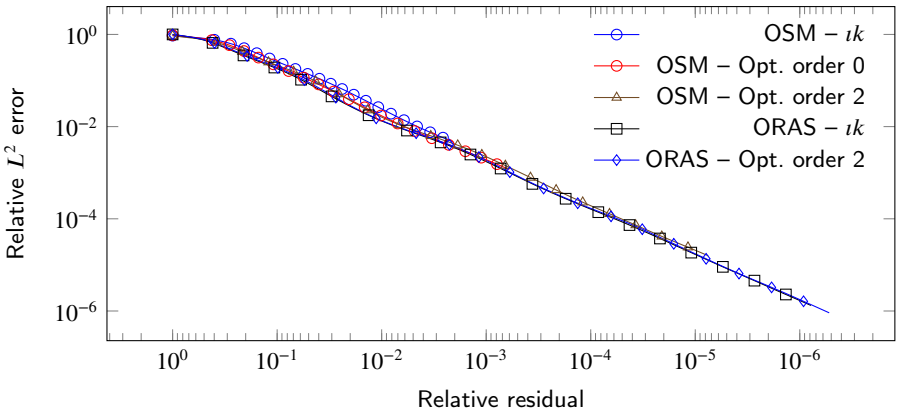

Figure 4: Relationship between the relative residual and the relative L^2 error for the homogeneous medium with 256 subdomains.

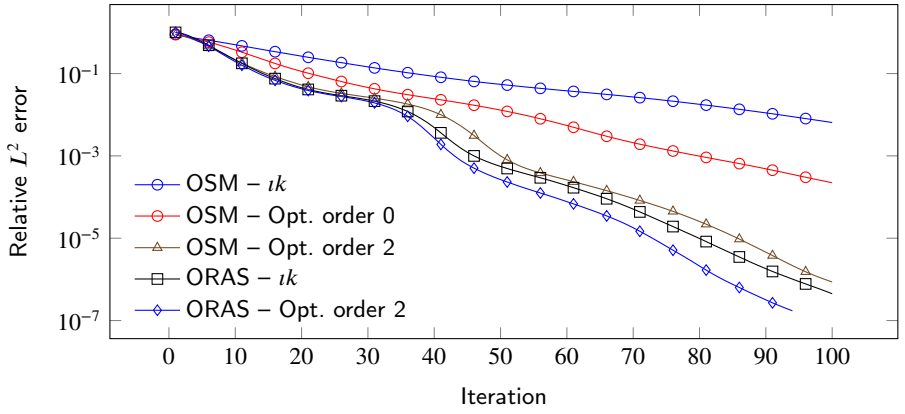

Figure 5: Convergence of the L^2 error for the heterogeneous medium with 256 subdomains.

frequency is increased to 2 Hz, which again leads to about 20 million unknowns. We again look at the L^2 convergence (Fig. 5) as well as the relationship between the L^2 error (again, to the discrete solution) and the residuals (Fig. 6). While ORAS exhibits again superior convergence, using optimized conditions makes the difference smaller. Compared to the homogeneous case, the residual/error curves are slightly different, with OSM yielding slightly more accurate results for a given tolerance.

5. Comparison on large-scale problems with multiple right-hand sides

When solving multiple sources, the setup cost (matrix assembly and factorization) is shared, whereas the iteration cost is multiplied. On paper, the cost for solving N_s sources should be proportional to N_s for the iterative part, but there is a large potential for batching. In particular, the subdomain triangular solves can handle multiple right-hand sides efficiently using vectorized operations (replacing level 2 BLAS calls with level 3 BLAS) and better cache locality, as the factors must be loaded once instead of N_s times. As shown in [11], having larger groups of RHS can be more efficient, especially when the solve is multi-threaded.

For these reasons, even when block Krylov methods or subspace recycling are not used, a good treatment of the multiple RHS is critical. When N_s is large, one can either solve the N_s RHS simultaneously, solve each of them sequentially, or split them into smaller groups (e.g., 8 or 16 at once), called the *batch size*. The larger the batch size, the more efficient the solve, at the expense of a higher memory consumption, as more vectors must be stored in the

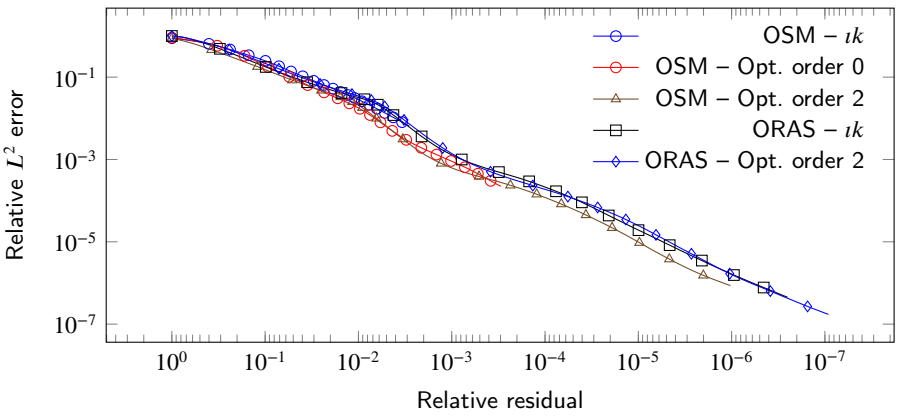

Figure 6: Relationship between the relative residual and the relative L^2 error for the heterogeneous medium with 256 subdomains.

Krylov method. Careful trade-offs are necessary, as using a large batch size might consume memory that could be better allocated to other purposes, such as increasing subdomain sizes or setting a higher GMRES restart parameter.

Overall, both memory and computational time should be considered. In what follows, we focus on minimizing the computational time for a given hardware, which limits the available memory. We first discuss the memory consumption of both methods to evaluate how large the problems we solve can be, given the available hardware and implementation. Then, we compare the computational times of the two methods for increasingly larger problems, with different batch sizes.

All experiments were run on the CPU partition of the LUMI supercomputer[1]. Each node consists of 2 AMD EPYC 7763 CPUs, each with 64 cores, so that a node has 128 cores and 256 GB of RAM. In each case, we used a tolerance of 10^{-4} on the relative residual, and a GMRES restart parameter of 50. In this case, we use single precision scalars, which are sufficient to reach the stopping criterion and allow large savings. Experiments showed that double precision led to similar convergence behavior, with iteration counts within 5% of the single precision ones for the required tolerance.

5.1. Memory analysis of a typical case

Besides fixed costs such as mesh storage and indexing, the memory cost of a simulation is primarily dominated by the factorization and the subspace storage in GMRES. When using a large number of small subdomains, a reduced amount of memory is required compared to using few large subdomains. However, this reduction in memory usage comes with a slower convergence rate. Memory usage from matrix factorizations depends essentially on the subdomain size, whereas the cost of GMRES can be controlled through the batch size and the restart parameter. The latter can however impact convergence, and smaller batches may limit the algorithm efficiency.

The Krylov subspace to store is significantly smaller with OSM than with ORAS, as it contains vectors representing interface data rather than volume data. This efficiency can be leveraged by using the freed resources to increase the batch size, the restart parameter, or the problem size. It is however worth noting that this interface problem grows larger as the number of partitions increases, and the difference with ORAS can become negligible for extremely small subdomain sizes.

Numerical experiments To estimate orders of magnitude, we measure memory consumption for a 2 Hz simulation of our test model, with 64 sources. This leads to around 20 million complex-valued unknowns. Since the LUMI cluster has nodes with 256 GB and 128 cores, we aim for a partitioning where each subdomain can be solved on a single-thread process and less than 2 GB. We partition into either 256, 512, or 1024 subdomains and run the simulations accordingly.

Table 1 reports for each partitioning and either OSM or ORAS the memory needed for matrix factorization, both as peak consumption and to store the resulting coefficients. In the absence of overlap, the factorization is significantly cheaper: depending on the subdomain sizes, the number of overlapping degrees of freedom ranges from 30% to 50% of the total number of DoFs in a subdomain, despite the overlap being chosen as small as possible. In both OSM and

[1] https://www.lumi-supercomputer.eu/

Method	N	Factorization (MiB)			Factors only (MiB)		
		Mean	Min	Max	Mean	Min	Max
ORAS	256	732	544	899	497	381	594
	512	341	262	417	226	171	277
	1024	163	108	206	106	70	128
OSM	256	439	346	494	305	267	338
	512	175	139	200	120	98	133
	1024	71	52	86	48	38	54

Table 1
Memory usage for factorization, and for storing the factors (i.e., after freeing the intermediate buffers) for a 20M unknowns problem. Values are provided as minimum, maximum, and average over all processes, with each process managing a subdomain.

Method	N	Local vector size		Memory for 64 RHSs (MiB)	
		Min	Max	Min	Max
ORAS	256	74,701	88,621	1824	2164
	512	36,490	45,013	891	1099
	1024	17,736	23,296	433	569
OSM	256	4,923	15,129	120	369
	512	3,226	10,413	79	254
	1024	2,019	6,832	49	167

Table 2
Local size of the vector over which GMRES iterates, and memory needed to store a full batch (i.e., 64×50 vectors with single-precision complex scalars) for the 20M unknowns problem.

Method	N	Peak memory per process (MiB)		
		Mean	Min	Max
ORAS	256	3703	3489	4003
	512	2014	1839	2398
	1024	1336	1024	1706
OSM	256	1532	1324	1727
	512	1032	1026	1089
	1024	1027	1022	1045

Table 3
Maximum resident set size (RSS) per process, measured at runtime on our implementation.

ORAS, a finer partitioning leads to smaller factorization storage, in a way which is consistent with the known results that, for a 3D problem of size N, memory scales asymptotically as $N^{\frac{4}{3}}$ [2]: every time we double the number of subdomains, the memory cost is reduced by a factor larger than 2. Overall, we see that large subdomains can be quite expensive, especially with ORAS: the partition in 256 subdomains leads us to having a peak memory of 899 MiB, which is almost half the budget of one process.

Table 2 reports the local size of the vector over which GMRES iterates (local part of the full vector in ORAS, local interface fields in OSM) as well as the memory needed to do a full batch simulation of 64 sources and a restart parameter of 50. Scalars are stored as single-precision complex numbers, taking 8 bytes. It highlights another advantage of using OSM: since GMRES iterates on the interface fields, the Krylov subspace has a reduced size. This again leads to a smaller memory footprint, especially for larger subdomains. As smaller subdomains lead to more interfaces, the *total* number of interface DoFs increases in OSM when the partitioning gets finer. Thus, the memory advantage of OSM is especially attractive for large subdomains, as the memory saving can be a factor larger than 6. It is however worth noting that interface DoFs suffer from a suboptimal load balancing, as the number of interface unknowns can vary significantly between subdomains due to their irregular shapes. As the table shows, the ratio between the subdomain with the most and the least interface unknowns can reach 3. One could optimize memory usage by distributing domains among nodes such that each node has some domains with many interfaces and some with fewer interfaces.

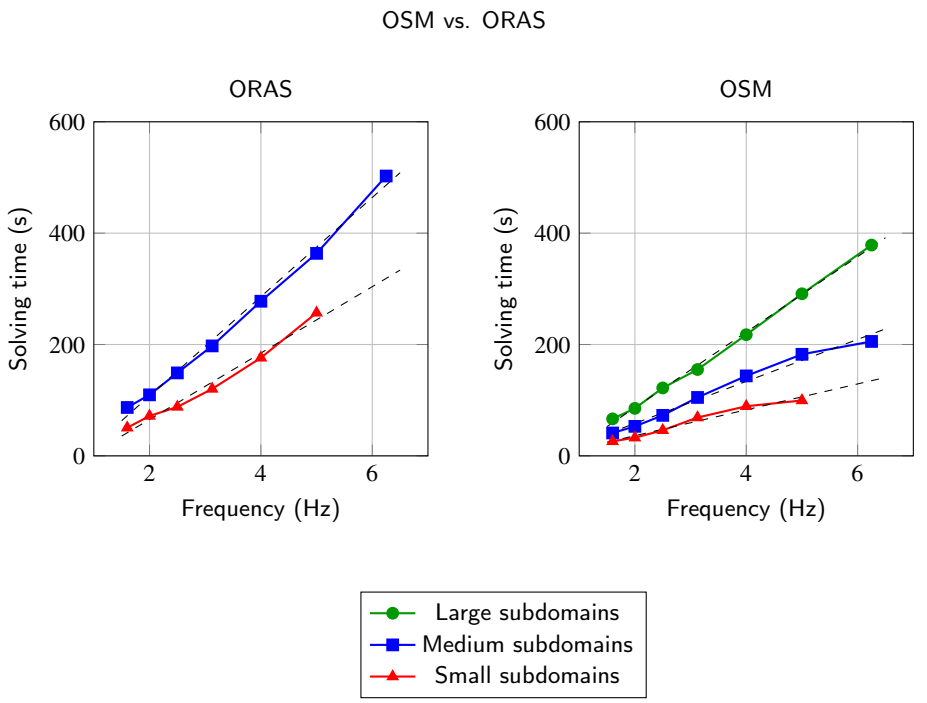

Figure 7: Maximum resident set size (RSS) over all processes, vs. batch size for OSM and ORAS, grouped by number of subdomains N.

Finally, Table 3 presents the total memory usage at runtime. It includes not only the factors and GMRES space discussed above, but also all quantities we neglected beforehand: intermediary buffers, mesh data, the matrices, the velocity model, etc. With OSM, large subdomains easily fit into one process (2 GB), unlike in the ORAS case. This issue can be mitigated by reducing the batch size or the restart parameter. However, even without a large Krylov subspace to store, the ORAS case with only 256 subdomains remains too large to fit in single-thread MPI processes. The 512 subdomains case almost fits on average, and can be made to fit using a batch size of 32 instead of 64. We will thus focus on cases with either two threads per process, or a fine enough partitioning when using ORAS. Fig. 7 showcases different values of the maximum memory (over all processes) for different batch sizes, with a dotted line representing the available memory per process.

Overall, we see that OSM is substantially more efficient in terms of memory, thanks to both the substructuring and the smaller subdomain size. It allows us to use larger batch sizes at a reasonable cost and to solve larger problems for a given hardware. In particular, we can solve problems with 20 million unknowns using only 2 nodes of the LUMI cluster, leading to one node per 10 million unknowns. If we keep the subdomain size constant, this means huge problems (over a billion unknowns) could be solved using only a fraction of the cluster.

5.2. Computational time analysis on large cases

Influence of the batch size We first compare different batch sizes for a 4 Hz problem (165 million unknowns) on 4096 subdomains and 2 threads per MPI process, which ensures enough memory is available. As Fig. 8 shows, increasing the batch size leads to faster computations on all aspects. However, moving from 32 to 64 does not yield additional speedup. This is consistent with findings from [9] where the authors mention an optimal size around 20.

Weak scaling We evaluate the scaling of both methods on larger cases, using similar domain sizes as before but with increasing frequency and mesh size inversely proportional to it. For OSM, we use 128, 256, or 512 subdomains on the smallest problem, but only 256 and 512 with ORAS, as 128 subdomains cannot fit in memory. For larger problems, we scale with constant subdomain-size by partitioning proportionally to the number of unknowns. Since this number is proportional to f^3, we multiply the frequency by $1.25 \approx \sqrt[3]{2}$ each time we double the number of subdomains. The typical subdomain sizes are as follows:

- large subdomains (OSM only): around 85,000 DoFs;

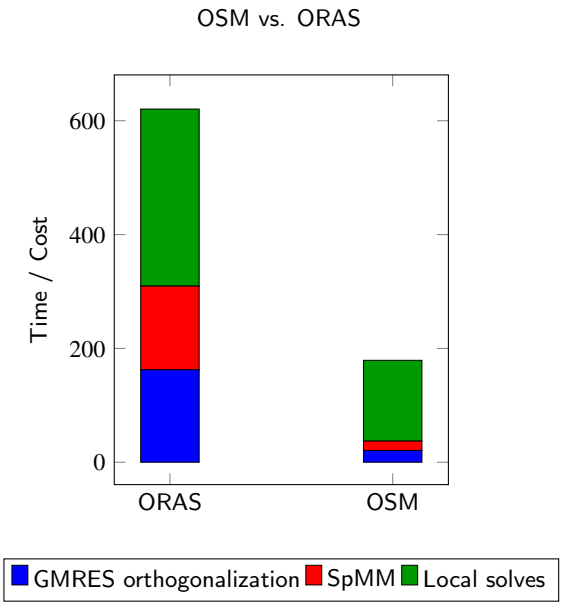

Figure 8: Time per iteration (normalized for full batch) for ORAS and OSM, for the 4 Hz problem and 4096 subdomains.

- medium subdomains: around 42,500 DoFs for OSM and 70,000 DoFs for ORAS;
- small subdomains: around 22,000 DoFs for OSM and 40,000 DoFs for ORAS.

We can see that even with minimal overlap, ORAS has significantly more DoFs per subproblem, which is why large subdomains could not be used with ORAS with the allocated resources.

As before, 64 sources are simulated, with full batch in case of OSM, and a batch size of 16 for ORAS. Each subdomain gets assigned 2 threads. Table 4 shows the result with OSM and Table 5 with ORAS. In both tables we report time spent in factorization, local (triangular) solves, in the GMRES orthogonalization process, and in sparse matrix products (the batched form of either the global matrix-vector product in ORAS, or the local SpMV in OSM). For these four quantities, we report the maximum wall time over all ranks, provided by PETSc. We also report the total wall time, which includes other minor components such as communications but excludes the factorization time. We also indicate the total time in CPU-hours, estimated as the total time multiplied by the number of threads, which itself is twice the number of subdomains. It is an approximation of the total amount of work as well as the energy consumption.

These tables show that OSM is consistently faster than ORAS, with a factor around 2, despite the tighter tolerance. The absence of overlap allows for faster subdomain solves, the substructuring reduces the cost of GMRES, and replacing the global sparse matrix-vector product by local, interface-located operations all contribute to the speedup. The larger batch size also helps, although it could have been higher with ORAS with sufficient memory.

Despite the number of cores per unknown being kept stable, the scalability is not ideal, irrespective of the method. This well-known issue is due to the local nature of communications in one-level domain decomposition methods, and should be mitigated using two-level methods, which are out of the scope of this paper. For a fixed subdomain size, computation time seems to scale linearly with the frequency (especially with OSM), i.e., as the cubic root of the number of unknowns. Overall, this yields a complexity of $\mathcal{O}(f^4)$ for the total amount of work: $\mathcal{O}(f^3)$ processes working for a duration proportional to f. This linear scaling with frequency is shown on Fig. 9.

Comparison with a direct solver We also solved the system using MUMPS as a direct solver with BLR enabled ($\epsilon_{BLR} = 10^{-4}$). For the 1.6 Hz case (about 11 million DoFs) with 256 domains (and 512 threads, on 4 nodes), a global factorization is computed in 74 s, and then 6 s are needed to solve the 64 sources, leading to a relative residual around 10^{-3}. With the same resources, OSM achieved a solution in 41 seconds. A direct solve is thus slower but has a smaller marginal cost per RHS, which could make it competitive when the number of sources is huge. A simple affine extrapolation shows that for this small-size problem the break-even point is around 150 sources, which is not rare in real FWI cases, especially when the adjoint problems are taken into account, or when the factorization is often reused,

f (Hz)	#MDoFs	N	T_{f} (s)	T_{loc} (s)	T_{g} (s)	T_{SpMM} (s)	T_{s} (s)	T_{ch} (h)	#its
1.6	11.0	128	6.2	53.9	3.6	4.0	65.3	4.6	53
1.6	11.0	256	2.2	31.6	3.1	2.8	41.5	5.9	70
1.6	11.0	512	0.8	17.6	3.0	2.1	25.8	7.3	91
2.0	21.1	256	6.0	70.5	5.1	5.7	86.7	12.3	73
2.0	21.1	512	1.9	39.3	4.7	3.6	53.1	15.1	93
2.0	21.1	1024	0.7	22.3	4.2	2.8	33.0	18.8	118
2.5	41.0	512	5.8	96.7	8.5	8.4	120.8	34.4	102
2.5	41.0	1024	1.9	54.9	7.0	5.3	74.2	42.2	131
2.5	41.0	2048	0.7	33.4	6.0	3.9	47.8	54.4	166
3.125	79.3	1024	5.4	121.0	11.3	11.0	153.7	87.4	135
3.125	79.3	2048	2.0	70.7	10.0	7.2	97.6	111.1	177
3.125	79.3	4096	0.7	44.8	9.6	6.1	69.1	157.3	254
4.0	165.2	2048	5.9	171.2	16.0	15.9	216.7	246.6	179
4.0	165.2	4096	2.0	105.6	15.2	11.0	145.7	331.6	249
4.0	165.2	8192	0.7	59.7	12.0	8.9	91.1	414.6	311
5.0	326.4	4096	5.4	243.2	22.9	22.0	307.7	700.1	247
5.0	326.4	8192	1.9	135.3	18.1	13.3	188.8	859.4	313
5.0	326.4	16384	0.8	71.7	14.1	9.9	108.7	989.7	371
6.25	630.1	8192	5.3	304.7	29.3	27.9	388.7	1768.9	323
6.25	630.1	16384	2.2	164.6	23.0	16.9	227.7	2072.3	396

Table 4
Time measured with OSM. f is the frequency, #MDoFs the number of million DoFs, N the number of subdomains, T_{f} the factorization time, T_{loc} the time spent in local subproblem solves, T_{g} the GMRES orthogonalization time, T_{SpMM} the time spent in sparse matrix - dense matrix products, T_{s} the total wall time (including setup), T_{ch} the total core-hours, and #its the number of iterations.

f (Hz)	#MDoFs	N	T_{f} (s)	T_{loc} (s)	T_{g} (s)	T_{SpMM} (s)	T_{s} (s)	T_{ch} (h)	#its
1.6	11.0	256	4.9	51.9	23.4	24.2	88.1	12.5	53
1.6	11.0	512	2.0	31.2	13.4	15.2	51.1	14.5	64
2.0	21.1	512	4.9	67.3	27.4	29.7	109.6	31.2	69
2.0	21.1	1024	2.1	43.1	22.6	20.7	70.8	40.3	89
2.5	41.0	1024	4.8	92.8	42.4	45.9	153.9	87.5	97
2.5	41.0	2048	2.0	56.4	27.8	27.4	90.3	102.7	116
3.125	79.3	2048	4.6	121.0	53.4	56.3	193.9	220.6	129
3.125	79.3	4096	2.0	75.1	39.1	36.0	122.5	278.9	162
4.0	165.2	4096	4.9	174.8	85.4	76.8	280.4	638.1	175
4.0	165.2	8192	2.3	109.0	58.5	53.3	178.5	812.5	224
5.0	326.4	8192	4.5	237.1	119.6	106.3	370.3	1685.1	240
5.0	326.4	16384	1.9	127.3	69.5	65.9	208.0	1893.4	274
6.25	630.1	16384	4.3	303.0	156.0	133.6	475.4	4327.2	324

Table 5
Time measured with ORAS. f is the frequency, #MDoFs the number of million DoFs, N the number of subdomains, T_{f} the factorization time, T_{loc} the time spent in local subproblem solves, T_{g} the GMRES orthogonalization time, T_{SpMM} the time spent in sparse matrix - dense matrix products, T_{s} the total wall time (including setup), T_{ch} the total core-hours, and #its the number of iterations.

such as with second-order adjoint methods [37] used in FWI. However, it is known that standard multifrontal direct solvers such as MUMPS do not scale as well as domain decomposition methods with problem size: full-rank MUMPS has a quadratic complexity with respect to the number of unknowns for 3D problems [2], which is however partially mitigated by the BLR approach. In practical FWI applications, using direct solves on low frequencies and iterative methods on high frequencies thus seems to be the best compromise, as already suggested in [7].

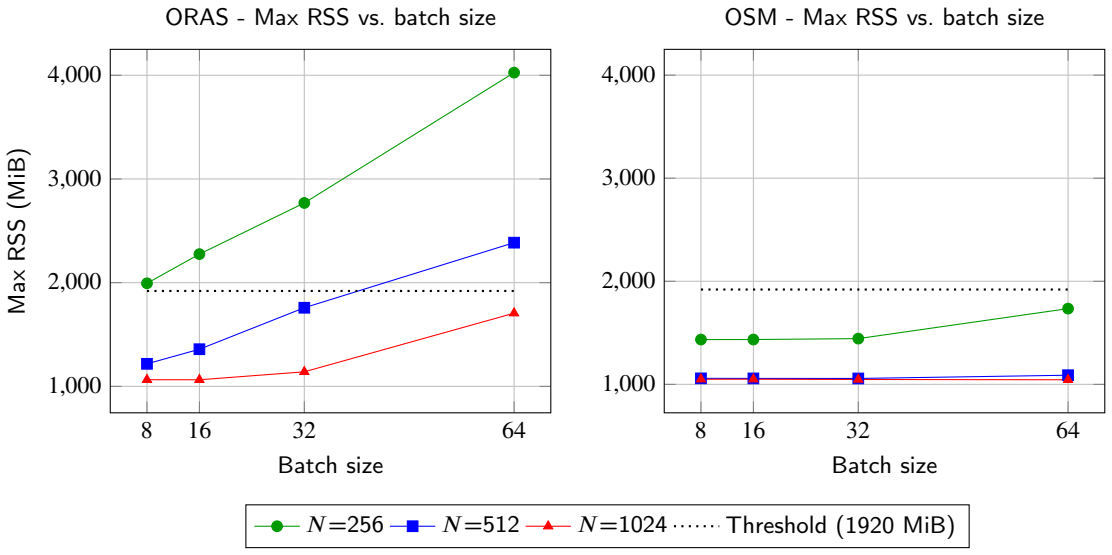

Figure 9: Weak scaling comparison of OSM and ORAS with constant subdomain size. Dashed lines represent affine functions fitting each curve.

Computation time breakdown during the solving phase For both methods, the main component of the computation (once the subproblem matrices have been factorized) is the repeated solution of each subproblem. However, a significant amount of time is also spent on two additional tasks. First, GMRES needs to build an orthonormal basis of the Krylov subspace using a Gram–Schmidt procedure. Second, at each iteration, the right-hand side of the subproblems must be computed. In ORAS, they are the restriction of the vector to which the preconditioner is applied, i.e., the product of the FEM matrix A and the latest Krylov vector. In OSM, it is the inhomogeneous source coming from other subdomains which must be assembled. In both cases, this step can be represented by sparse matrix-vector products. In OSM, these matrices are smaller (and rectangular) since they map interface variables to the source of the local Helmholtz problem. Thus, there are three main components to consider and all of them have smaller dimensions in OSM, which explains the superior performance of the method. A more detailed analysis presented in Fig. 10 shows which portion of the time is spent in these three parts for our largest test case. As expected, the GMRES part and the sparse algebra are radically cheaper in OSM due to the substructuring, allowing the algorithm to spend most of its resources on the local solves. Combined with the smaller subdomains that are faster to solve, the whole method becomes significantly more efficient.

6. Conclusion and perspectives

We tested and compared optimized implementations of the ORAS preconditioner and the OSM non-overlapping DDM for the Helmholtz equation, with a focus on solving large-scale geophysical problems with multiple right-hand sides, using one of the world's largest supercomputer, LUMI. We first showed that, while ORAS has a substantially faster convergence with basic impedance conditions, a smarter choice of parameters allows OSM to almost bridge the gap. With optimized transmission conditions the OSM method becomes consistently faster and more memory-efficient than ORAS: having smaller subdomain sizes, applying GMRES to the substructured equation, and replacing the global sparse matrix-vector products by localized interface operations all contribute to this difference. Furthermore, OSM reduces the memory footprint of large batches, which can provide additional speedup.

While OSM is quite clearly the most appropriate one-level method in this case, two-level methods must be considered for larger problems, to solve the scalability issues. Geometric coarse-grids were successfully used for ORAS [9] and have a moderate cost (essentially limited to storing the coarse problems), and recent progress has been made on spectral coarse grids [38, 39, 40], which however are designed for overlapping methods. Developing a two-level method for OSM (either geometric or spectral) is a promising avenue for future work and is currently being investigated.

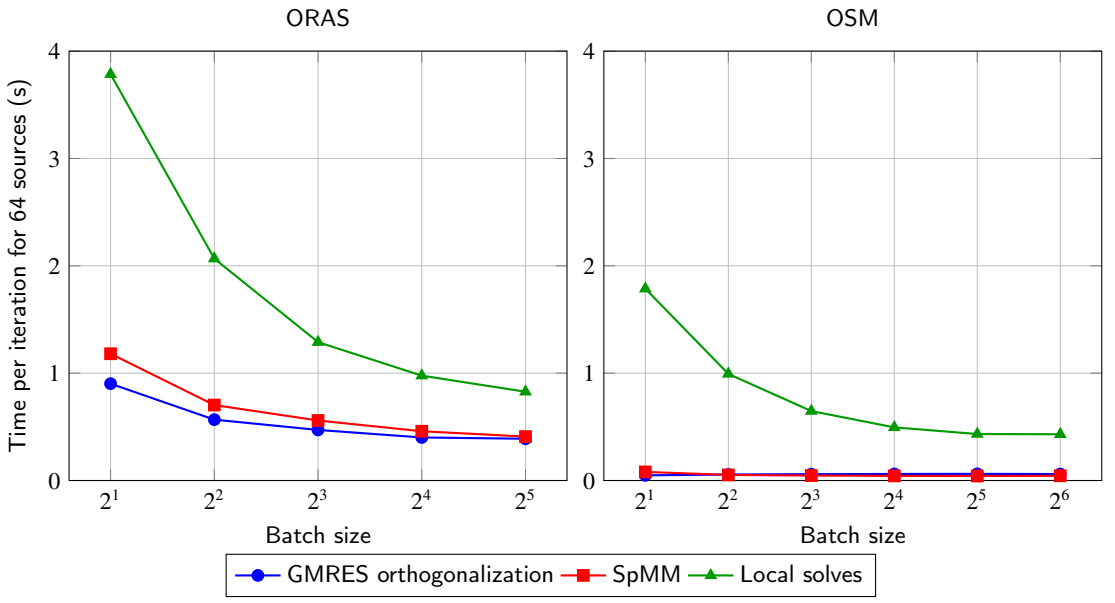

Figure 10: Time repartition in the ORAS (left) and OSM (right) methods for the 600M unknowns problem on 16 384 domains. Actual total times are 502.6 s and 205.5 s respectively.

We focused on GMRES, but other Krylov methods could be considered as well. Since we have multiple right-hand sides, block Krylov solvers could accelerate convergence. However, our experiments were disappointing as the effect on convergence was too small to justify the overhead. This seems to depend heavily on the problem, and some different geometries or material properties could yield systems where block GMRES and related algorithms do have a significant advantage [11]. It is particularly interesting to note that using block GMRES instead of GMRES is drastically cheaper with OSM than with ORAS thanks to the substructuring. For similar reasons, recycling solvers such as GCRODR and its block variant could be helpful. We could also investigate short recurrences such as BiCGSTAB [41], IDR(s) [42], and CGNR, which benefit from cheaper orthogonalization and storage costs, at the cost of additional iterations. In our cases with BiCGSTAB, the additional iteration count clearly outweighed the reduced orthogonalization cost, and was not considered further. However, if one were to use smaller subdomains (e.g., to apply the methods to GPU solvers), the orthogonalization cost could become dominant and justify using these families of methods.

Finally, similar investigations should be carried for other frequency-domain wave equations, such as time-harmonic Maxwell or time-harmonic linear elasticity (also known as elastic Helmholtz), where the same principles apply. OSM should be more economical than ORAS per-iteration for the same reasons (smaller domains, substructuring, no global sparse matrix-vector product), but a convergence analysis is needed to ensure the difference in iteration count does not outweigh these advantages. Furthermore, even in the acoustic case, we focused on geophysical cases, with a relatively coarse mesh (4 points per wavelength and third-order polynomials). Other applications might require different mesh sizes, leading to different convergence rates or optimal transmission conditions. In our experience, the transmission condition we used in OSM is competitive with ORAS for various numbers of points per wavelength, but adaptive meshing is critical. Indeed, OSM with overly refined meshes is expected to struggle with second-order transmission conditions. This is related to the higher number of evanescent modes on overly refined meshes, which are easily damped by overlaps. An alternative approach is to use OSM with PMLs [34] or High-Order Absorbing Conditions [35], but these methods assume a structured partitioning and have additional auxiliary variables, increasing their cost.

7. Reproducibility

The source code to reproduce the numerical results is available at `https://gitlab.onelab.info/boris-martin/oras_vs_osm`. Instructions for building and running the tests are available in the README.md file.

Acknowledgments

The first author gratefully acknowledges financial support from the Fonds de la Recherche Scientifique FNRS (Belgium) through a FRIA doctoral fellowship. The research was funded in part by the Walloon Region through the Win2Wal EXPANSION project (Grant No. 2010161). Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247. We acknowledge LUMI-BE[2] for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through a LUMI-BE Regular Access call.

CRediT authorship contribution statement

Boris Martin: Methodology, Software, Numerical Experiments, Writing - Original Draft. **Pierre Jolivet:** Methodology, Software support and expertise, Writing - Reviewing. **Christophe Geuzaine:** Methodology, Software support and expertise, Supervision, Writing - Reviewing.

References

[1] R. G. Pratt, Seismic waveform inversion in the frequency domain, part 1: Theory and verification in a physical scale model, Geophysics 64 (3) (1999) 888–901. doi:10.1190/1.1444597.

[2] T. Mary, Solveurs multifrontaux exploitant des blocs de rang faible : complexité, performance et parallélisme, Ph.D. thesis, Université Toulouse 3, Toulouse, France (2017).
URL http://www.theses.fr/2017TOU30305/document

[3] R. Nies, M. Hoelzl, Testing performance with and without Block Low Rank Compression in MUMPS and the new PaStiX 6.0 for JOREK nonlinear MHD simulations (Jul. 2019).
URL http://arxiv.org/abs/1907.13442

[4] P. Amestoy, R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, M. Alain, S. Operto, Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea, Geophysics 81. doi:10.1190/geo2016-0052.1.

[5] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, C. Weisbecker, Improving multifrontal methods by means of block low-rank representations, Research Report RR-8199, INRIA (2013).
URL https://inria.hal.science/hal-00776859

[6] P. Amestoy, A. Buttari, J.-Y. L'Excellent, T. Mary, Performance and scalability of the Block Low-Rank multifrontal factorization on multicore architectures, ACM Transactions on Mathematical Software 45 (1) (2019) 1–23. doi:10.1145/3242094.

[7] S. Operto, P. Amestoy, S. Beller, A. Buttari, L. Combe, V. Dolean, M. Gerest, G. Guo, P. Jolivet, J.-Y. L'Excellent, F. Mamfoumbi, T. Mary, C. Puglisi, A. Ribodetti, P.-H. Tournier, Is 3D frequency-domain FWI of full-azimuth/long-offset OBN data feasible? The Gorgon case study, Leading Edge 42 (3) (2023) 173–183. doi:10.1190/tle42030173.1.

[8] O. G. Ernst, M. J. Gander, Why it is difficult to solve Helmholtz problems with classical iterative methods, in: I. G. Graham, T. Y. Hou, O. Lakkis, R. Scheichl (Eds.), Numerical Analysis of Multiscale Problems, Vol. 83 of Lecture Notes in Computational Science and Engineering, Springer, Berlin, Heidelberg, 2012, pp. 325–363. doi:10.1007/978-3-642-22061-6_10.

[9] P.-H. Tournier, P. Jolivet, V. Dolean, H. Aghamiry, S. Operto, S. Riffo, Three-dimensional finite-difference finite-element frequency-domain wave simulation with multi-level optimized additive Schwarz domain-decomposition preconditioner: A tool for FWI of sparse node datasets, Geophysics 87 (5) (2022) 1–84. doi:10.1190/geo2021-0702.1.

[10] F.-X. Roux, A. Barka, Block Krylov recycling algorithms for FETI-2LM applied to three-dimensional electromagnetic wave scattering and radiation, IEEE Transactions on Antennas and Propagation PP (2017) 1–1. doi:10.1109/TAP.2017.2670541.

[11] P. Jolivet, P.-H. Tournier, Block iterative methods and recycling for improved scalability of linear solvers, in: SC16 - International Conference for High Performance Computing, Networking, Storage and Analysis, Proceedings of SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, Utah, United States, 2016.
URL https://hal.science/hal-01357998

[12] T. Gabriel, Acceleration of frequency-domain full wave inversion through Krylov reuse, Master's thesis, Université de Liège, Liège, Belgique (2023).
URL https://matheo.uliege.be/handle/2268.2/18323

[13] A. St-Cyr, M. J. Gander, S. J. Thomas, Optimized restrictive additive Schwarz methods, in: O. B. Widlund, D. E. Keyes (Eds.), Domain Decomposition Methods in Science and Engineering XVI, Vol. 55 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, 2007, pp. 213–220. doi:10.1007/978-3-540-34469-8_22.

[2] LUMI-BE is joint effort from BELSPO (federal), SPW Économie, Emploi, Recherche (Wallonia), Department of Economy, Science & Innovation (Flanders) and Innoviris (Brussels).

[14] S. Gong, M. J. Gander, I. G. Graham, E. A. Spence, A variational interpretation of restricted additive Schwarz with impedance transmission condition for the Helmholtz problem, in: S. C. Brenner, E. Chung, A. Klawonn, F. Kwok, J. Xu, J. Zou (Eds.), Domain Decomposition Methods in Science and Engineering XXVI, Springer International Publishing, Cham, 2022, pp. 291–298.

[15] V. Dolean, P. Jolivet, F. Nataf, An Introduction to Domain Decomposition Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015. doi:10.1137/1.9781611974065.

[16] J.-D. Benamou, B. Desprès, A Domain Decomposition Method for the Helmholtz Equation and Related Optimal Control Problems, Research Report RR-2791, INRIA, projet IDENT (1996).
URL https://inria.hal.science/inria-00073899

[17] F. Nataf, Interface connections in domain decomposition methods, in: A. Bourlioux, M. J. Gander, G. Sabidussi (Eds.), Modern Methods in Scientific Computing and Applications, Vol. 75 of NATO Science Series, Springer, Dordrecht, 2002, pp. 323–364. doi:10.1007/978-94-010-0510-4_9.
URL https://doi.org/10.1007/978-94-010-0510-4_9

[18] M. J. Gander, F. Magoulès, F. Nataf, Optimized Schwarz methods without overlap for the Helmholtz equation, SIAM Journal on Scientific Computing 24 (1) (2002) 38–60. doi:10.1137/S1064827501387012.

[19] X. Antoine, C. Geuzaine, Optimized schwarz domain decomposition methods for scalar and vector Helmholtz equations, in: D. Lahaye, J. Tang, K. Vuik (Eds.), Modern Solvers for Helmholtz Problems, Geosystems Mathematics, Birkhäuser, Cham, 2017, pp. 189–213. doi:10.1007/978-3-319-28832-1_8.
URL https://doi.org/10.1007/978-3-319-28832-1_8

[20] G. Karypis, V. Kumar, METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices (September 1998).

[21] P. Solin, K. Segeth, I. Dolezel, Higher-Order Finite Element Methods, 1st Edition, Chapman and Hall/CRC, 2003. doi:10.1201/9780203488041.

[22] A. Górszczyk, S. Operto, GO_3D_OBS: the multi-parameter benchmark geomodel for seismic imaging method assessment and next-generation 3d survey design (version 1.0), Geoscientific Model Development 14 (3) (2021) 1773–1799. doi:10.5194/gmd-14-1773-2021.

[23] B. Després, Méthodes de décomposition de domaine pour les problèmes de propagation d'ondes en régime harmonique : Le théorème de Borg pour l'équation de hill vectorielle, Ph.D. thesis, Université Paris IX – Dauphine, Paris, France (1991).

[24] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331. doi:10.1002/nme.2579.

[25] A. Royer, E. Béchet, C. Geuzaine, Gmsh-fem: An efficient finite element library based on Gmsh, in: 14th World Congress on Computational Mechanics (WCCM), ECCOMAS Congress 2020, Scipedia, 11 March 2021. doi:10.23967/wccm-eccomas.2020.161.

[26] A. Royer, Efficient finite element methods for solving high-frequency time-harmonic acoustic wave problems in heterogeneous media, Ph.D. thesis, ULiège - Université de Liège [Faculté des Sciences Appliquées], Belgique, Belgium (2023).

[27] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, H. Suh, S. Zampini, H. Zhang, J. Zhang, PETSc/TAO users manual, Tech. Rep. ANL-21/39 – Revision 3.23, Argonne National Laboratory (2025). doi:10.2172/2476320.

[28] P. Jolivet, J. E. Roman, S. Zampini, KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners, Computers & Mathematics with Applications 84 (2021) 277–295. doi:10.1016/j.camwa.2021.01.003.

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

[30] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Computing 32 (2) (2006) 136–156.

[31] V. Dolean, M. J. Gander, E. Veneros, H. Zhang, Optimized Schwarz methods for heterogeneous Helmholtz and Maxwells equations, in: Domain Decomposition Methods in Science and Engineering XXIII, Springer, 2017, pp. 145–152.

[32] M. J. Gander, H. Zhang, Optimized Schwarz Methods with overlap for the Helmholtz equation, SIAM Journal on Scientific Computing 38 (5) (2016) A3195–A3219. doi:10.1137/15M1021659.

[33] N. Bootland, S. Borzooei, V. Dolean, P.-H. Tournier, Numerical assessment of PML Transmission Conditions in a Domain Decomposition Method for the Helmholtz equation, in: Z. Dostál, T. Kozubek, A. Klawonn, U. Langer, L. F. Pavarino, J. Šístek, O. B. Widlund (Eds.), Domain Decomposition Methods in Science and Engineering XXVII, Springer Nature Switzerland, Cham, 2024, pp. 445–453.

[34] A. Royer, C. Geuzaine, E. Béchet, A. Modave, A non-overlapping domain decomposition method with perfectly matched layer transmission conditions for the Helmholtz equation, Computer Methods in Applied Mechanics and Engineering 395 (2022) 115006. doi:10.1016/j.cma.2022.115006.

[35] Y. Boubendir, X. Antoine, C. Geuzaine, A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation, Journal of Computational Physics 231 (2) (2012) 262–280. doi:10.1016/j.jcp.2011.08.007.

[36] A. Modave, A. Royer, X. Antoine, C. Geuzaine, A non-overlapping domain decomposition method with high-order transmission conditions and cross-point treatment for Helmholtz problems, Computer Methods in Applied Mechanics and Engineering 368 (2020) 113162. doi:10.1016/j.cma.2020.113162.

[37] L. Métivier, R. Brossier, S. Operto, J. Virieux, Second-order adjoint state methods for Full Waveform Inversion, in: EAGE 2012 - 74th European Association of Geoscientists and Engineers Conference and Exhibition, Copenhagen, Denmark, 2012.
URL https://hal.science/hal-00826614

[38] V. Dolean, M. Fry, I. G. Graham, M. Langer, Schwarz preconditioner with H_k-GenEO coarse space for the indefinite Helmholtz problem (2024). arXiv:2406.06283.

[39] N. Bootland, V. Dolean, P. Jolivet, P.-H. Tournier, A comparison of coarse spaces for Helmholtz problems in the high frequency regime, Computers & Mathematics with Applications 98 (2021) 239–253.

[40] L. Conen, V. Dolean, R. Krause, F. Nataf, A coarse space for heterogeneous Helmholtz problems based on the Dirichlet-to-Neumann operator, Journal of Computational and Applied Mathematics 271 (2014) 83–99. doi:10.1016/j.cam.2014.03.031.

[41] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (2) (1992) 631–644. doi:10.1137/0913035.

[42] P. Sonneveld, M. B. van Gijzen, IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, SIAM Journal on Scientific Computing 31 (2) (2009) 1035–1062. doi:10.1137/070685804.