

## A MATRIX-FREE PARALLEL SOLUTION METHOD FOR THE THREE-DIMENSIONAL HETEROGENEOUS HELMHOLTZ EQUATION\*

J. CHEN<sup>†</sup>, V. DWARKA<sup>†</sup>, AND C. VUIK<sup>†</sup>

**Abstract.** The Helmholtz equation is related to seismic exploration, sonar, antennas, and medical imaging applications. It is one of the most challenging problems to solve in terms of accuracy and convergence due to the scalability issues of the numerical solvers. For 3D large-scale applications, high-performance parallel solvers are also needed. In this paper, a matrix-free parallel iterative solver is presented for the three-dimensional (3D) heterogeneous Helmholtz equation. We consider the preconditioned Krylov subspace methods for solving the linear system obtained from finite-difference discretization. The Complex Shifted Laplace Preconditioner (CSLP) is employed since it results in a linear increase in the number of iterations as a function of the wavenumber. The preconditioner is approximately inverted using one parallel 3D multigrid cycle. For parallel computing, the global domain is partitioned blockwise. The matrix-vector multiplication and preconditioning operator are implemented in a matrix-free way instead of constructing large, memory-consuming coefficient matrices. Numerical experiments of 3D model problems demonstrate the robustness and outstanding strong scaling of our matrix-free parallel solution method. Moreover, the weak parallel scalability indicates our approach is suitable for realistic 3D heterogeneous Helmholtz problems with minimized pollution error.

**Key words.** Helmholtz equation, parallel computation, matrix-free, geometric multigrid, preconditioner, scalability

**AMS subject classifications.** 65Y05, 65F08, 35J05

**1. Introduction.** The Helmholtz equation, describing the phenomena of time-harmonic wave scattering in the frequency domain, finds applications in many scientific fields, such as seismic problems, advanced sonar devices, medical imaging, and many more. To solve the Helmholtz equation numerically, we discretize it and obtain a linear system  $Ax = b$ . The linear system matrix is sparse, symmetric, complex-valued, non-Hermitian, and indefinite. Instead of a direct solver, iterative methods and parallel computing are commonly considered for a large-scale linear system resulting from a 3D problem. However, the indefiniteness of the linear system brings a great challenge to the numerical solution method, especially for large wavenumbers. The convergence rate of many iterative solvers is affected significantly by increasing wavenumber. An increase in the wavenumber leads to a dramatic increase in iterations. Moreover, the general remedy for minimizing the so-called pollution error, driven by numerical dispersion errors due to discrepancies between the exact and numerical wavenumber, is to refine the grid such that the condition  $k^3 h^2 < 1$  is satisfied [2]. Therefore, the research problem of how to solve the systems efficiently and economically while at the same time maintaining a high accuracy by minimizing pollution error arises in this field. A wavenumber-independent, convergent, and parallel scalable iterative method could significantly enhance the corresponding research in electromagnetics, seismology, and acoustics.

Many efforts have been made to solve the problem in terms of accuracy and scalable convergence behavior. One of the main concerns is the spectrum of the system matrix, which is closely related to the convergence of Krylov subspace methods. The preferable idea is to preprocess the system with a preconditioner. By applying a preconditioner to the linear system, the solution remains the same, but the coefficient matrix has a more favorable distribution of eigenvalues. Many preconditioners are proposed for the Helmholtz problem so far, such as incomplete Cholesky (IC), incomplete LU (ILU) factorization, shifted Laplacian

---

\*Received November 14, 2022. Accepted November 20, 2023. Published online on January 8, 2024. Recommended by M. Donatelli.

<sup>†</sup>Department of Applied Mathematics, Delft University of Technology, Delft, the Netherlands  
({j.chen-11|v.n.s.r.dwarka|c.vuik}@tudelft.nl).

preconditioners [5, 19, 20], and so on. The industry standard, also known as the Complex Shifted Laplace Preconditioner (CSLP) [19, 20] does show good properties for medium wavenumbers. Nevertheless, the eigenvalues shift to the origin as the wavenumber increases. These near-zero eigenvalues have an unfavorable effect on the convergence speed of the Krylov-based iterative solvers. Therefore, Sheihk [42] further included deflation techniques to accelerate the convergence. Recently, a higher-order approximation scheme to construct the deflation vectors was proposed to reach close to wavenumber-independent convergence [16].

The development of scalable parallel Helmholtz solvers is also ongoing. One approach is to parallelize existing advanced algorithms. Kononov and Riyanti [33, 38] first developed a parallel version of Bi-CGSTAB preconditioned by multigrid-based CSLP. Knibbe et al. [30] further introduced parallel versions of CSLP-preconditioned Bi-CGSTAB and IDR(s) which run on GPU accelerators. Dan and Rachel [27] parallelized their so-called CARP-CG algorithm (Conjugate Gradient acceleration of CARP) blockwise. The block-parallel CARP-CG algorithm shows improved scalability as the wavenumber increases. Calandra et al. [7, 8] proposed a geometric two-grid preconditioner for 3D Helmholtz problems, which shows the strong scaling property in a massively parallel setup.

Another approach is the Domain Decomposition Method (DDM), which originates from the early Schwarz Methods. DDM has been widely used to develop parallel solution methods for Helmholtz problems. For comprehensive surveys, we refer the reader to [6, 9, 12, 14, 17, 24, 25, 35, 41, 44, 45, 46] and references therein.

This work is interested in parallelizing Krylov subspace methods, such as the Generalized Minimal RESidual method (GMRES), Bi-CGSTAB, and IDR(s), preconditioned by the multigrid-based CSLP for the Helmholtz equation. We consider the CSLP preconditioner because it is the first and most popular method where the number of iterations scales linearly within medium wavenumbers. However, CSLP is not wavenumber-independent. The standard configuration of CSLP+GMRES works less efficiently if we want to solve highly heterogeneous problems with minimized pollution error. Finer grids and more iterations, hence more memory, will be needed. Our contribution is the development and validation of a matrix-free parallel framework of CSLP-preconditioned Krylov subspace methods in the context of solving large-scale 3D Helmholtz problems with minimized pollution error. To the best of our knowledge, this has not been previously reported in the literature. The earlier variants proposed by Kononov and Riyanti et al. [33, 38] mainly parallelized the sequential program based on the data-parallel concept. It results in a row-wise domain partition and a 3D multigrid method with 2D semi-coarsening. In contrast, this work starts with a block-wise domain partition and implements a standard 3D multigrid method in a matrix-free way. Our method contributes to a robust and scalable parallel CSLP-preconditioned solver for realistic 3D applications. Numerical experiments on typical 3D model problems show that the matrix-free parallel solution method can effectively save memory for storing global sparse matrices and show good parallel performance.

The rest of this paper is organized as follows. Section 2 describes the mathematical model and discretization technique that we will discuss. All numerical methods we use are described in Section 3. Section 4 describes the parallel implementation. The numerical performance is explored in Section 5. Finally, Section 6 contains our conclusions.

**2. Mathematical models.** The Helmholtz equation can model the wavefield in heterogeneous media in the frequency domain. Suppose a parallelepipedal domain  $\Omega \subset \mathbb{R}^3$  with boundary  $\Gamma = \partial\Omega$ . The Helmholtz equation reads

$$-\Delta u(\mathbf{x}) - k(\mathbf{x})^2 u(\mathbf{x}) = b(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, x_3) \in \Omega,$$

supplied with one of the following boundary conditions:

$$(2.1) \quad \begin{aligned} &\text{Dirichlet: } u(\mathbf{x}) = g(\mathbf{x}), \text{ on } \partial\Omega, \\ &\text{First-order Sommerfeld: } \frac{\partial u(\mathbf{x})}{\partial \vec{n}} - ik(\mathbf{x})u(\mathbf{x}) = \mathbf{0}, \text{ on } \partial\Omega, \end{aligned}$$

where  $i$  is the imaginary unit.  $u$  represents the pressure wavefield.  $b$  is the source function,  $\vec{n}$  and  $g$  represent the outward unit normal vector and the given data on the boundary, respectively. The wavenumber  $k(\mathbf{x})$  and the frequency  $f$  are related by

$$k(\mathbf{x}) = \frac{2\pi f}{c(\mathbf{x})},$$

where  $c(\mathbf{x})$  is the space-variant acoustic-wave velocity due to changes in the material of the domain.

**2.1. 3D closed-off problem.** This problem is a constant wavenumber model with a given solution to validate the numerical methods. A parallelepipedal homogeneous domain  $\Omega = [0, 1]^3$  is considered. The source function is specified by

$$b(x_1, x_2, x_3) = (21\pi^2 - k^2) \sin(\pi x_1) \sin(2\pi x_2) \sin(4\pi x_3) - k^2.$$

It is supplied with the following Dirichlet conditions

$$u(x_1, x_2, x_3) = 1, \text{ on } \partial\Omega.$$

The analytical solution is given by

$$u(x_1, x_2, x_3) = \sin(\pi x_1) \sin(2\pi x_2) \sin(4\pi x_3) + 1.$$

**2.2. 3D wedge problem.** Most physical problems of geophysical seismic imaging describe a heterogeneous medium. The so-called wedge problem [37] is a typical problem with a simple heterogeneity. It mimics three layers with different acoustic-wave velocities, hence different wavenumbers. As shown in Figure 2.1, the parallelepipedal domain  $\Omega = [0, 600] \times [0, 600] \times [-1000, 0]$  is split into three layers. Suppose the acoustic-wave velocity  $c$  is constant within each layer but different from each other. A point source is located at  $(x_1, x_2, x_3) = (300, 300, 0)$ .

The problem is given by

$$\begin{cases} -\Delta u(x_1, x_2, x_3) - k(x_1, x_2, x_3)^2 u(x_1, x_2, x_3) = b(x_1, x_2, x_3), & \text{on } \Omega, \\ b(x_1, x_2, x_3) = \delta(x_1 - 300, x_2 - 300, x_3) & x_1, x_2, x_3 \in \Omega, \end{cases}$$

where  $k(x_1, x_2, x_3) = \frac{2\pi f}{c(x_1, x_2, x_3)}$  and  $f$  is the frequency. The wave velocity  $c(x_1, x_2, x_3)$  is shown in Figure 2.1 and  $\delta(x_1, x_2, x_3)$  is a Dirac delta function. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

**2.3. 3D SEG/EAGE salt model.** The 3D SEG/EAGE salt model [1] is a velocity field model containing salt domes, which mimics the typical Gulf Coast salt structure. As shown in Figure 2.2, it is defined in a parallelepipedal physical domain of size  $13\,520\text{ m} \times 13\,520\text{ m} \times 4200\text{ m}$ . The acoustic-wave velocity varies from  $1500\text{ m s}^{-1}$  to  $4482\text{ m s}^{-1}$ . The model is considered challenging due to the inclusion of complex geometries (salt domes) and a realistic large-size computational domain. We consider a computational domain

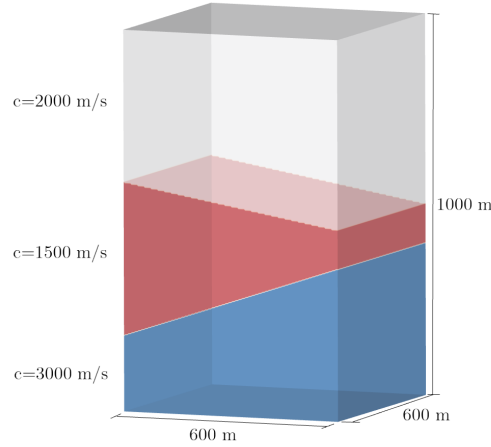


FIG. 2.1. The velocity distribution of the 3D wedge problem.

$\Omega = (0, 12800) \times (0, 12800) \times (0, 3840)$  with grid size  $641 \times 641 \times 193$ , allowing multi-level geometric coarsening. A point source is located at  $(x_1, x_2, x_3) = (3200, 3200, 0)$ .

The problem is given by

$$\begin{cases} -\Delta u(x_1, x_2, x_3) - k(x_1, x_2, x_3)^2 u(x_1, x_2, x_3) = b(x_1, x_2, x_3), & \text{on } \Omega, \\ b(x_1, x_2, x_3) = \delta(x_1 - 3200, x_2 - 3200, x_3) & x_1, x_2, x_3 \in \Omega, \end{cases}$$

where  $k(x_1, x_2, x_3) = \frac{2\pi f}{c(x_1, x_2, x_3)}$  and  $f$  is the frequency. The wave velocity  $c(x_1, x_2, x_3)$  is shown in Figure 2.2. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

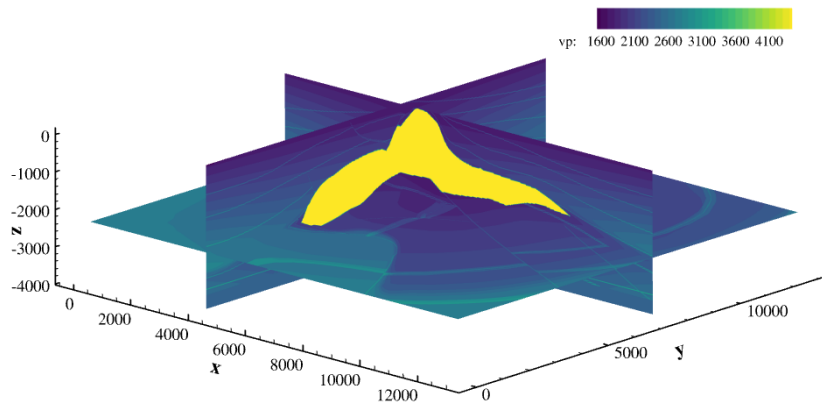


FIG. 2.2. The velocity distribution of the 3D SEG/EAGE Salt Model. The velocity varies from  $1500 \text{ m s}^{-1}$  to  $4482 \text{ m s}^{-1}$ .

**2.4. Finite-difference discretization.** Structural vertex-centered grids are used to discretize the computational domains. Suppose the grid widths in  $x_1$ ,  $x_2$ , and  $x_3$  directions are all equal to  $h$ . A second-order finite difference scheme for a 3D Laplacian operator has the following stencil

$$(2.2) \quad [-\Delta_h] = \frac{1}{h^2} \left[ \begin{array}{c} \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{array} \right]_{x_3-h} \\ \left[ \begin{array}{ccc} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{array} \right]_{x_3} \\ \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{array} \right]_{x_3+h} \end{array} \right].$$

The discrete Helmholtz operator  $A_h$  can be obtained by adding the diagonal matrix  $-k^2 I_h$  to the Laplacian operator  $-\Delta_h$ , i.e.,

$$A_h = -\Delta_h - k^2 I_h.$$

Therefore, the stencil of the discrete Helmholtz operator is

$$(2.3) \quad [A_h] = \frac{1}{h^2} \left[ \begin{array}{c} \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{array} \right]_{x_3-h} \\ \left[ \begin{array}{ccc} 0 & -1 & 0 \\ -1 & 6 - k^2 h^2 & -1 \\ 0 & -1 & 0 \end{array} \right]_{x_3} \\ \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{array} \right]_{x_3+h} \end{array} \right].$$

In case the Sommerfeld radiation condition (2.1) is used, the discrete schemes for the boundary points need to be defined. We can introduce ghost points located outside the boundary points. For instance, assume  $u_{0,i_2,i_3}$  is a ghost point on the left of  $u_{1,i_2,i_3}$ , the normal derivative can be approximated by

$$\frac{\partial u}{\partial \vec{n}} - iku = \frac{u_{0,i_2,i_3} - u_{2,i_2,i_3}}{2h} - iku_{1,i_2,i_3} = 0.$$

We can rewrite it as

$$(2.4) \quad u_{0,i_2,i_3} = u_{2,i_2,i_3} + 2hiku_{1,i_2,i_3}.$$

Then one can eliminate the ghost point in the stencil for the boundary points.

In addition, for the discretization of the Dirac function  $\delta(x_1 - x_0, x_2 - y_0, x_3 - z_0)$ , we can set the right-hand side (RHS) as

$$b_h(i_1, i_2, i_3) = \begin{cases} \frac{1}{h^3}, & x_{1i_1} = x_0, x_{2i_2} = y_0, x_{3i_3} = z_0, \\ 0, & x_{1i_1} \neq x_0, x_{2i_2} \neq y_0, x_{3i_3} \neq z_0. \end{cases}$$

The finite-difference discretization of the partial differential equation on geometric grids results in a system of linear equations

$$A_h \mathbf{u}_h = \mathbf{b}_h.$$

With Sommerfeld boundary conditions, the resulting matrix is sparse, symmetric, complex-valued, indefinite, and non-Hermitian for a sufficiently large wavenumber  $k$ .

Note that  $kh$  is an important parameter that can indicate how many grid points per wavelength are used. The grid width  $h$  can be determined by the rule of thumb of including at least  $N_{pw}$  (e.g., 10 or 30) grid points per wavelength. One has the following relationships

$$kh = \frac{2\pi h}{\lambda} = \frac{2\pi}{N_{pw}}.$$

For example, if at least 10 grid points per wavelength are required, one has to satisfy the condition  $kh \leq 0.625$ .

**3. Preconditioned Krylov method.** For a large, sparse system matrix  $A$ , Krylov subspace methods are popular. This section will specify every component of the preconditioned Krylov subspace methods that we use to solve the linear system.

**3.1. Krylov subspace methods.** The Krylov subspace methods are established on a collection of iterations in the subspace

$$K^k(A; \mathbf{r}^0) := \text{span} \{ \mathbf{r}^0, A\mathbf{r}^0, \dots, A^{k-1}\mathbf{r}^0 \},$$

where  $K^k$  is at most a  $k$ -dimensional Krylov space with respect to matrix  $A$  and initial residual  $\mathbf{r}^0$ . For a basic iterative method, after  $k$  iterations,  $\mathbf{u}^k$  will be an element of  $\mathbf{u}^0 + K^k(A, \mathbf{r}^0)$ . Some representative Krylov methods, like Conjugate Gradient (CG) [28], CGNR [39], MINRES [36], BICG [22], Bi-CGSTAB [47], GMRES [40], are developed so far. Among these, the CG method is a basic one. The error is minimized in the A-norm, and it only needs three vectors in memory during iterations. However, this algorithm is mainly designed for a symmetric and positive definite system matrix. In contrast, Bi-CGSTAB and GMRES can be used for consistent problems that are indefinite and non-symmetric. They are suitable choices for the Helmholtz equation. Compared with full GMRES, Bi-CGSTAB has short recurrences and better parallel properties.

Also, the IDR(s), which is developed by Sonneveld, Van Gijzen [43], is an efficient alternative to Bi-CGSTAB for Helmholtz problems [30]. In IDR(s),  $s$  pre-defined vectors are used to enhance the convergence. It was showed in [43] that IDR(1) has similar computational complexity and memory requirements as Bi-CGSTAB. With higher values of  $s$ , IDR(s) exhibits a performance close to GMRES, but with more storage requirements. For example, we need to store 17 vectors for IDR(4), while Bi-CGSTAB needs to store 7 vectors. In general, IDR(4) is sufficient for most of the problems. Thus, we employ IDR(4) as a representation of IDR(s) in our parallel framework. For a detailed analysis of the influence of  $s$  on the algorithm convergence properties, we refer the reader to the work of Sonneveld and Van Gijzen et al. [11, 43]

We first focus on implementing the matrix-free parallelization of GMRES, then it can be directly generalized to Bi-CGSTAB and IDR(s).

**3.2. Preconditioning.** The coefficient matrix of a non-Hermitian linear system is preferred to have a spectrum located in a bounded region that excludes the origin in the complex plane, which results in fast convergence for iterative methods. We can incorporate a preconditioner to enhance the convergence of Krylov subspace methods, that is, to pre-multiply the linear system with a preconditioning matrix  $M_h^{-1}$ . It can be implemented by left preconditioning

$$M_h^{-1}A_h\mathbf{u}_h = M_h^{-1}\mathbf{b}_h,$$

or right preconditioning,

$$A_hM_h^{-1}\tilde{\mathbf{u}}_h = \mathbf{b}_h,$$

where  $\mathbf{u}_h = M_h\tilde{\mathbf{u}}_h$ . It can be proved that there is no essential difference between both preconditioning methods with respect to convergence behavior. Anyway, it is worth noting that the residual vectors computed by left preconditioning and right preconditioning correspond to the preconditioned and actual residuals, respectively. Thus, it may cause some difficulties for left preconditioning if a stopping criterion needs to be based on the actual residual instead of the preconditioned one.

**3.3. Complex Shifted Laplace Preconditioner.** We focus on CSLP motivated by its nice performance and easy setup. The CSLP is defined by

$$(3.1) \quad M_h = -\Delta_h - (\beta_1 - \beta_2 i) k^2 I_h,$$

where  $\beta_2$  is the so-called complex shift. It is proved that a small magnitude of the complex shift is necessary for convergence if the wavenumber  $k$  is large [23]. In practice, it is also expensive to invert the preconditioner. Standard multigrid methods [19, 34] are usually employed to invert a shifted Laplacian preconditioner. However, as the complex shift becomes small, one cannot ensure that the standard multigrid methods will be effective [10]. Besides, the computational burden to approximate the preconditioner will significantly increase. Thus, it is necessary to choose a proper complex shift. In the numerical experiments of this paper, unless noted otherwise,  $\beta_1 = 1$  and  $\beta_2 = -0.5$  will be used [48], and the inverse of the preconditioner will be approximated by one geometric multigrid V-cycle.

**3.4. 3D multigrid for the preconditioner solve.** One needs to compute the inverse of the preconditioner  $M$  in the preconditioned Krylov-based algorithms. A direct solver is a good choice if the preconditioner is simple, such as the block-Jacobi preconditioner. However, it is usually too costly to directly invert a preconditioner like CSLP. One idea is to approximately solve the preconditioner by using the multigrid method [19].

A 3D multigrid method involves several components that need a careful design to achieve excellent convergence. The first ingredient of multigrid methods is the smoothing property of basic iterative methods. For example, the Gauss-Seidel and  $SOR(\omega)$  methods can serve as efficient smoothers. As coarser grids are involved, going back and forth between different hierarchies of grids is required. Specifically, the inter-grid transfer operations include restriction and prolongation operators. Besides, the construction of coarse grid operators also needs careful attention.

**3.4.1. Smoothers.** Classical iteration methods such as the Gauss-Seidel or damped Jacobi iterations can be used as smoothers. The impact on the convergence properties of varying the relaxation parameter for the damped Jacobi smoother and using different smoothers, has already been extensively investigated by [18] and [29]. It has been shown that, based on an actual situation, the use of smoothers and the choice of relaxation parameters can be flexible within a certain range. In this framework, we will mainly use the damped Jacobi smoother, as it is easy to parallelize and has been shown to be effective in the before-mentioned papers. We will fix the relaxation parameter at  $\omega = 0.8$ , as this value was found to work well for our test problems.

**3.4.2. Restriction.** Without loss of generality, the inter-grid operations will be presented in a simple case of two grids, i.e., a fine, and a coarse grid. Two sets of uniform grids with size  $h$  and  $H = 2h$  are used to discretize a regular computational domain  $\Omega = (0, 1) \times (0, 1) \times (0, 1)$ . Figure 3.1 shows part of a 3D fine grid with a coarse grid obtained by standard coarsening.

The 3D full-weighting restriction operator has a stencil given by

$$I_h^H = \frac{1}{64} \left[ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right]_h^H \left[ \begin{array}{ccc} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{array} \right]_h^H \left[ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right]_h^H.$$

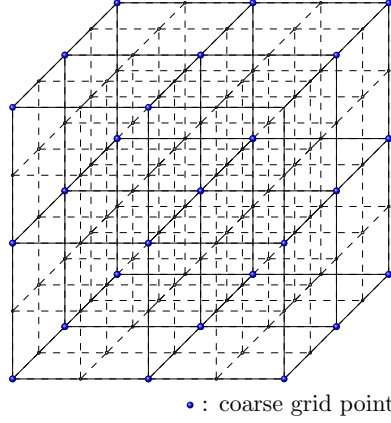


FIG. 3.1. Vertex-centered 3D standard coarsening.

Let  $b^H = I_h^H r^h$ , where  $b^H$  is the right-hand side of the coarse grid and  $r^h$  is the residual of the fine grid. The restriction can be implemented in a matrix-free way as follows

$$\begin{aligned}
 b_{i_1, i_2, i_3}^H &= I_h^H r_{2i_1, 2i_2, 2i_3}^h = \frac{1}{64} \left( 8r_{2i_1, 2i_2, 2i_3}^h \right. \\
 &+ 4r_{2i_1-1, 2i_2, 2i_3}^h + 4r_{2i_1+1, 2i_2, 2i_3}^h + 4r_{2i_1, 2i_2-1, 2i_3}^h \\
 &+ 4r_{2i_1, 2i_2+1, 2i_3}^h + 4r_{2i_1, 2i_2, 2i_3-1}^h + 4r_{2i_1, 2i_2, 2i_3+1}^h \\
 &+ 2r_{2i_1-1, 2i_2-1, 2i_3}^h + 2r_{2i_1+1, 2i_2-1, 2i_3}^h + 2r_{2i_1-1, 2i_2+1, 2i_3}^h + 2r_{2i_1+1, 2i_2+1, 2i_3}^h \\
 &+ 2r_{2i_1-1, 2i_2, 2i_3-1}^h + 2r_{2i_1+1, 2i_2, 2i_3-1}^h + 2r_{2i_1-1, 2i_2, 2i_3+1}^h + 2r_{2i_1+1, 2i_2, 2i_3+1}^h \\
 &+ 2r_{2i_1, 2i_2-1, 2i_3-1}^h + 2r_{2i_1, 2i_2+1, 2i_3-1}^h + 2r_{2i_1, 2i_2-1, 2i_3+1}^h + 2r_{2i_1, 2i_2+1, 2i_3+1}^h \\
 &+ r_{2i_1-1, 2i_2-1, 2i_3-1}^h + r_{2i_1+1, 2i_2-1, 2i_3-1}^h + r_{2i_1-1, 2i_2+1, 2i_3-1}^h + r_{2i_1+1, 2i_2+1, 2i_3-1}^h \\
 &\left. + r_{2i_1-1, 2i_2-1, 2i_3+1}^h + r_{2i_1+1, 2i_2-1, 2i_3+1}^h + r_{2i_1-1, 2i_2+1, 2i_3+1}^h + r_{2i_1+1, 2i_2+1, 2i_3+1}^h \right),
 \end{aligned}$$

where  $(i_1, i_2, i_3) \in \Omega^H$ .

**3.4.3. Interpolation.** The interpolation operator  $I_H^h$  transfers grid vectors from the coarse to the fine grid. The 3D trilinear interpolation operator stencil can be written as

$$I_H^h = \frac{1}{8} \left[ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right]_H^h \left[ \begin{array}{ccc} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{array} \right]_H^h \left[ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right]_H^h.$$

Let  $u^h = I_H^h u^H$ , where  $u^H$  is the solution of the coarse grid and  $u^h$  is the correction for



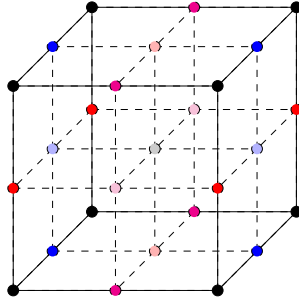


FIG. 3.2. The allocation map of interpolation operator.

the fine grid. As shown in Figure 3.2, the interpolation can be implemented as follows

$$(3.2) \quad I_H^h u_{i_1, i_2, i_3}^H = \begin{cases} u_{i_1, i_2, i_3}^H, & \bullet \\ \frac{1}{2} (u_{i_1, i_2, i_3}^H + u_{i_1+1, i_2, i_3}^H), & \bullet \\ \frac{1}{2} (u_{i_1, i_2, i_3}^H + u_{i_1, i_2+1, i_3}^H), & \bullet \\ \frac{1}{2} (u_{i_1, i_2, i_3}^H + u_{i_1, i_2, i_3+1}^H), & \bullet \\ \frac{1}{4} (u_{i_1, i_2, i_3}^H + u_{i_1+1, i_2, i_3}^H + u_{i_1, i_2+1, i_3}^H + u_{i_1+1, i_2+1, i_3}^H), & \bullet \\ \frac{1}{4} (u_{i_1, i_2, i_3}^H + u_{i_1+1, i_2, i_3}^H + u_{i_1, i_2, i_3+1}^H + u_{i_1+1, i_2, i_3+1}^H), & \bullet \\ \frac{1}{4} (u_{i_1, i_2, i_3}^H + u_{i_1, i_2+1, i_3}^H + u_{i_1, i_2, i_3+1}^H + u_{i_1, i_2+1, i_3+1}^H), & \bullet \\ \frac{1}{8} (u_{i_1, i_2, i_3}^H + u_{i_1, i_2+1, i_3}^H + u_{i_1+1, i_2, i_3}^H \\ + u_{i_1+1, i_2+1, i_3}^H + u_{i_1, i_2, i_3+1}^H + u_{i_1+1, i_2, i_3+1}^H \\ + u_{i_1, i_2+1, i_3+1}^H + u_{i_1+1, i_2+1, i_3+1}^H), & \bullet \end{cases}$$

where  $(i_1, i_2, i_3) \in \Omega^H$ .

**3.4.4. Coarse grid operator.** The coarse grid matrix  $A_H$  can be built in two ways. The first way is to obtain  $A_H$  by re-discretizing on the coarse grid in the same way as the matrix  $A_h$  is obtained on the fine grid, which is known as the discretized coarse grid operator (DCG). The second approach,  $A_H = I_h^H A_h I_H^h$ , is known as Galerkin Coarsening. Galerkin coarse-grid operator GCG is more general in its range of applicability, but it has an associated expense in terms of growing stencils. Considering the parallel implementation, we choose DCG as our method. Boundary conditions of the preconditioner operator are set identically to the corresponding model problems.

**3.4.5. Multigrid cycles.** Based on the ingredients above, a classical two-Grid cycle is given in Algorithm 1, where  $S_h$  denotes the smoothing procedure, and  $v_1$  and  $v_2$  are the numbers of pre- and post-smoothing steps, respectively.

---

**Algorithm 1:** A two-grid cycle.

---

- 1  $v_1$  pre-smoothing sweeps:  $\mathbf{u}_h^{i+1/3} = S_h^{v_1}(\mathbf{u}_h^i, A_h, \mathbf{b}_h)$ ;
  - 2 Residual computation:  $\mathbf{r}_h = \mathbf{b}_h - A_h \mathbf{u}_h^{i+1/3}$ ;
  - 3 Restriction of the residual to  $G_H$ :  $\mathbf{r}_H = I_h^H \mathbf{r}_h$ ;
  - 4 Determination of the error on  $G_H$ :  $\mathbf{e}_H = (A_H)^{-1} \mathbf{r}_H$ ;
  - 5 Prolongation of the error to  $G_h$ :  $\mathbf{e}_h = I_h^h \mathbf{e}_H$ ;
  - 6 Correction of the last solution iterate:  $\mathbf{u}_h^{2/3} = \mathbf{u}_h^{1/3} + \mathbf{e}_h$ ;
  - 7  $v_2$  post-smoothing sweeps:  $\mathbf{u}_h^{i+1} = S_h^{v_2}(\mathbf{u}_h^{i+2/3}, A_h, \mathbf{b}_h)$ .
- 

Two-grid methods are rarely practical because the coarse-grid problem may still be too large to be solved with a direct method. The idea to apply the two-grid idea to  $A_H$  recursively and to continue until the coarse linear system can be solved with negligible computational costs gives rise to a genuine multigrid method.

Different *cycle* types can be distinguished based on the sequence in which the grids are traversed within a single multigrid iteration. The V-cycle and W-cycle are achieved by calling the two-grid method once or twice, respectively, on each coarse grid. The V-cycle method involves traversing from the finest grid to the coarsest grid, performing a single pre-smoothing step at each level, and then moving back up to the finest grid, performing a single post-smoothing step at each level. The F-cycle lies between the V-cycle and the W-cycle. It begins with the restriction to the coarsest grid. During the prolongation process, after reaching each level for the first time, an additional V-cycle to the coarsest grid is performed. In the numerical experiments, both V-cycle and F-cycle methods will be involved.

For solving the coarse grid problems (i.e., step 4 in Algorithm 1), a direct solver is not easy to be parallelized in a matrix-free way. Thus, we solve the final coarse-grid problem iteratively by GMRES.

**3.5. Matrix-free method.** We can implement the Krylov subspace methods in a matrix-free way instead of constructing the coefficient matrices explicitly. The matrix-vector multiplication can be replaced by stencil computations. Likewise, the preconditioning matrix  $M$  and its stencil can be obtained analytically by its definition (3.1). One does not need to construct  $M$  explicitly since the result of  $M\mathbf{x}$  can be calculated by its corresponding stencil.

Considering any grid point  $(i_1, i_2, i_3)$ , define  $ap, aw, ae, as, an, ad,$  and  $au,$  as the multipliers of  $u(i_1, i_2, i_3), u(i_1 - 1, i_2, i_3), u(i_1 + 1, i_2, i_3), u(i_1, i_2 - 1, i_3), u(i_1, i_2 + 1, i_3), u(i_1, i_2, i_3 - 1),$  and  $u(i_1, i_2, i_3 + 1),$  respectively. When physical boundary conditions are encountered, it only needs to eliminate the ghost grid points as given in equation (2.4).

For the Helmholtz operator (2.3), we have

$$ap = \frac{6 - k^2 h^2}{h^2}, \quad aw = ae = as = an = au = ad = -\frac{1}{h^2}.$$

As we invert the preconditioner  $M$  by the multigrid method,  $y = M_h x$  and  $y = M_H x$  need to be performed in the smoother and coarsest grid solver. For these CSLP operators, according to equation (2.2) and (3.1), we will have

$$ap = \frac{6 - (\beta_1 - \beta_2 i) k^2 h^2}{h^2} \quad aw = ae = as = an = au = ad = -\frac{1}{h^2}.$$

The results of  $\mathbf{v} = A_h \mathbf{u}, \mathbf{y} = M_h \mathbf{x}$  and  $\mathbf{y} = M_H \mathbf{x}$  can be computed in a matrix-free way by Algorithm 2.

Besides, the matrix-free restriction and prolongation operators in the multigrid method can be implemented according to equations (3.4.2) and (3.2), respectively.

---

**Algorithm 2:** Matrix-free Matrix-Vector Multiplication.
 

---

```

Input: 3D Array  $u(1 : n_1, 1 : n_2, 1 : n_3)$ 
1 Initiate  $ap, aw, ae, as, an, ad,$  and  $au$  ;
2 Initiate 3D Array  $v(1 : n_1, 1 : n_2, 1 : n_3)$ ;
3 Exchange the interface boundaries data of  $u$ ;
4 for  $i_3 := 1, 2, \dots, n_3$  do
5   for  $i_2 := 1, 2, \dots, n_2$  do
6     for  $i_1 := 1, 2, \dots, n_1$  do
7       if physical boundary grid point then
8         | Reset  $ap, aw, ae, as, an, ad,$  and  $au$ ;
9       end
10       $v(i_1, i_2, i_3) = ap * u(i_1, i_2, i_3) + ae * u(i_1 + 1, i_2, i_3) + aw * u(i_1 -$ 
11      |  $1, i_2, i_3) + an * u(i_1, i_2 + 1, i_3) + as * u(i_1, i_2 - 1, i_3) + au *$ 
12      |  $u(i_1, i_2, i_3 + 1) + ad * u(i_1, i_2, i_3 - 1);$ 
13    end
14  end
15 end
16 Return  $v$ .
  
```

---

**4. Parallel implementation.** A parallel Fortran 90 code is developed to solve the 3D heterogeneous Helmholtz problems. The MPI standard is employed for data communications among the processes. Therefore, the design of an MPI topology will be the basis. Further, the domain partition and the data structure within the processes will determine the implementation of the matrix-vector multiplication and dot product. Finally, the parallelization of the Krylov subspace methods and the multigrid cycle are carried out.

**4.1. Parallel setup.** For the MPI setup, the first step is to determine the total number of processes (denotes as  $np$ ) and the number of processes in each direction. In 3D cases, the number of processes in the  $x$ -,  $y$ -, and  $z$ - direction is denoted as  $np_x0$ ,  $np_y0$ , and  $np_z0$ , respectively. As shown in Figure 4.1, the entire computational domain is partitioned into  $3 \times 4 \times 5$  blocks. Each block is seen as an MPI process, each of which is assigned to a CPU core. Due to the star-type computation stencil, the communication between processes exists only between the adjacent blocks in each coordinate direction. After acquiring  $np$  processes and creating a parallel computing environment using MPI, each process will own its MPI rank (0 to  $np - 1$ ). We assign each subdomain to each process according to the  $x$ -line lexicographic order. Indicating to different processes whether their corresponding subdomains contain physical boundaries or interface boundaries, we define  $np_x$ ,  $np_y$ , and  $np_z$ , to describe the position in the  $x$ -,  $y$ -, and  $z$ - directions for every process. For example,  $np_x$  is in the range of  $[0, np_x0 - 1]$ . When  $np_x$  is equal to 0, the process needs to deal with the left physical boundary. When  $np_x$  equals  $np_x0 - 1$ , the process needs to handle the right physical boundary.

In domain partitioning, we choose to partition the domain between two grid points, i.e., along the red dotted line shown in Figure 4.3. Therefore, the boundary points of adjacent subdomains are adjacent grid points in the global grids. Note that the Helmholtz operator and the CSLP have a similar stencil that only needs the data from the adjacent grid points. Thus, for each block, we introduce one layer of overlapping grid points outward at each interface boundary to represent the adjacent grid points, i.e., the blue grid in Figure 4.2.

In the program, the grid unknowns are stored as an array based on the grid ordering  $(i_1, i_2, i_3)$  instead of a column vector based on  $x$ -line lexicographic ordering. We store the number of grid points in each dimension within each subdomain as  $nx$ ,  $ny$ , and  $nz$ , respectively. To store and use the data from adjacent processors, the local arrays are extended

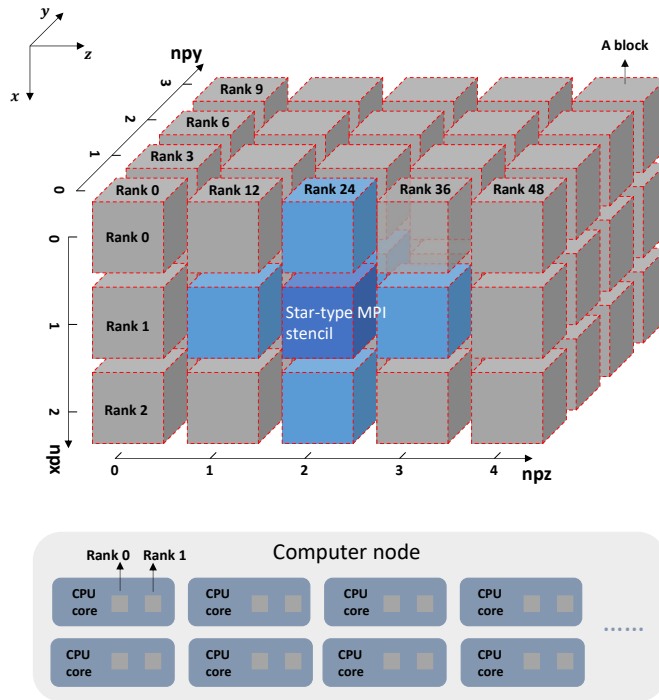


FIG. 4.1. The schematic diagram of the MPI topology for  $np_x0 \times np_y0 \times np_z0 = 3 \times 4 \times 5$ .

based on the subdomain grid structure as shown in Figure 4.2. Thus, the indices range of the array  $\mathbf{u}$  becomes  $(1 - lap : nx + lap, 1 - lap : ny + lap)$ . For second-order finite-difference discretization, the number of overlapping grid points  $lap$  is 1. Within a certain subdomain, the operations and array updates are limited to the range  $(1 : nx, 1 : ny, 1 : nz)$ . The data  $u(i_1, i_2, i_3)$  for  $i_1 = 1$  or  $nx$ ,  $i_2 = 1$  or  $ny$ , and  $i_3 = 1$  or  $nz$ , are sent to adjacent processors. The data received from adjacent processors are stored in the corresponding extended grid points, which are called during the operations concerning interface grid points.

**4.2. Parallel multigrid method based on global grid.** We will consider the parallel implementation of the multigrid iteration based on the original global grid, as shown in Figure 4.3, where an arbitrary grid size is chosen for demonstration purposes.

According to the relationship between the fine grid and the coarse grid, the parameters of the coarse grid are determined by the grid parameters of the fine one. For example, point  $(i_{1c}, i_{2c}, i_{3c})$  in the coarse grid corresponds to point  $(2i_{1c} - 1, 2i_{2c} - 1, 2i_{3c} - 1)$  in the fine grid. The restriction, as well as the interpolation of the grid variables, can be implemented according to the stencils based on the index correspondence between the coarse and fine grid. The grid-based matrix-free multigrid preconditioner starts at the finest grid, and recursively performs a two-grid cycle until the bottom level is reached, where a certain number of pre-defined grid points are left in one of the directions, denoted as the stopping criterion for coarsening, i.e.,  $nx_{\text{coarsest}} \times ny_{\text{coarsest}} \times nz_{\text{coarsest}}$ . The coarsest-grid problem is solved by parallel GMRES. On the one hand, direct solvers are not easy to parallelize and an exact solution for the coarse grid

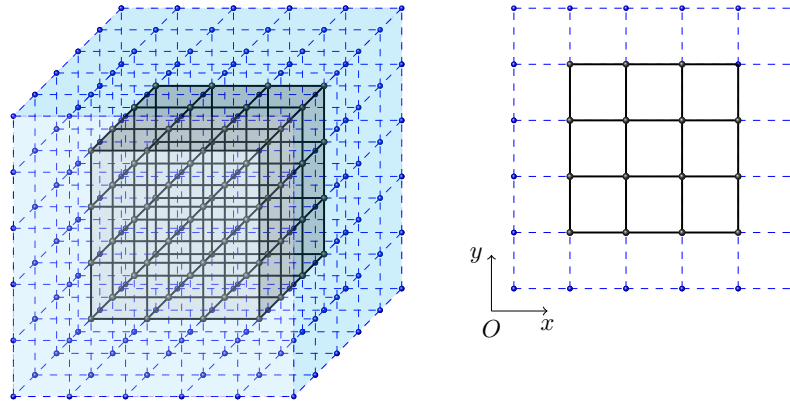


FIG. 4.2. *The schematic diagram of the local grid structure.*

problem is not necessary for our preconditioner. On the other hand, we developed this work for massively parallel computing of large-scale practical problems. Agglomerating to one process and using a direct solver is not an economical option, as the all-gather communication would be expensive and the rest of the processes are idle. Thus, our idea is to stop coarsening at a certain level, on which the communication load is still much less than the computational load and we can solve the coarse-grid problem in parallel with fairly good efficiency. In this way, we can take advantage of the parallelism available in the system and achieve better overall performance, because it allows for more accurate solutions at coarser levels, which can then be used to inform the solutions at finer levels. This is particularly important for practical applications where we need to balance both performance and accuracy. However, the optimal selection of the coarsest grid size depends on both the number of iterations and processes. Due to our limitations in computing resources and machine time, we set the stopping criterion for coarsening as  $17 \times 17 \times 17$  to ensure that each process has at least 1–2 coarsest grid points. It should be noted that if the computational domain is not a cube, this setting means that coarsening will stop when the number of grid points in a certain direction is less than 17. This setup allows a relatively practical lower bound for reference. The related topic of optimal coarsest grid size will be further investigated in our future work.

**5. Numerical experiments.** The numerical experiments are primarily performed on the Linux supercomputer DelftBlue [13]. DelftBlue runs on the Red Hat Enterprise Linux 8 operating system. Each compute node is equipped with two Intel Xeon E5-6248R processors with 24 cores at 3.0 GHz, 192 GB of RAM, a memory bandwidth of 132 GByte/s per socket, and a 100 Gbit/s InfiniBand card. In our experiments, the solver is developed in Fortran 90. On DelftBlue, the code is compiled using GNU Fortran 8.5.0 with the compiler options `-O3` for optimization purposes. Open MPI library (version 4.1.1) is employed for message passing. HDF5 1.10.7 is used for massively parallel I/O.

For the iterative methods we use in this section, the number of matrix-vector multiplications (denoted as #Matvec), which only includes matrix-vector products with the system matrix  $A_h$ , will be the main amount of work. Unless mentioned, the following convergence criterion of the preconditioned GMRES algorithm is used.

$$\frac{\|M_h^{-1}\mathbf{b}_h - M_h^{-1}A_h\mathbf{u}_h^k\|_2}{\|M_h^{-1}\mathbf{b}_h\|_2} \leq 10^{-6}.$$

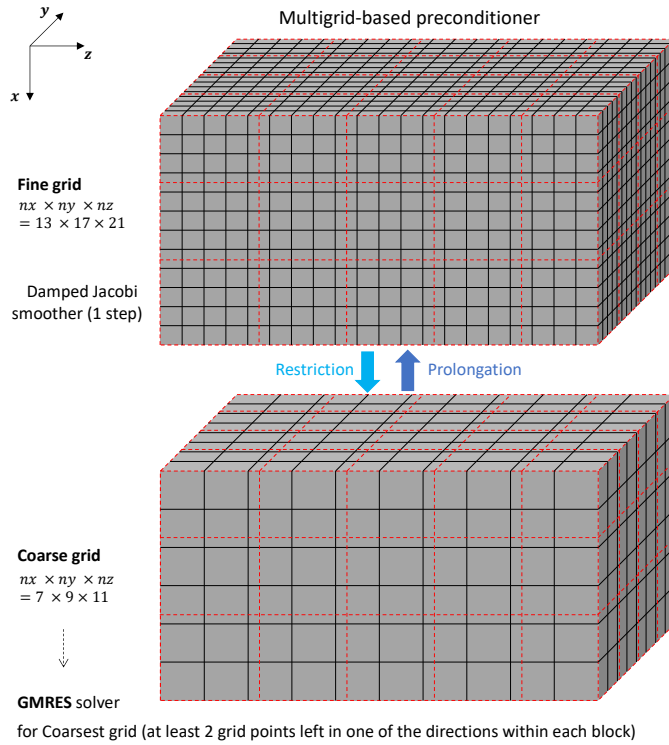


FIG. 4.3. The schematic diagram of parallel multigrid based on global-grid coarsening. Grid size is chosen at random for demonstration purposes.

The convergence criterion for the preconditioned Bi-CGSTAB and IDR(s) is

$$\frac{\|\mathbf{b}_h - A_h \mathbf{u}_h^k\|_2}{\|\mathbf{b}_h\|_2} \leq 10^{-6}.$$

The different stopping criteria for GMRES, Bi-CGSTAB, and IDR(4) are chosen in relation to the left and right preconditioning strategies discussed in Section 3.2. It is intended to illustrate the flexibility of our framework in handling various solvers, and preconditioning techniques.

For the coarsest-grid problem solver, according to our pre-experiments, the stopping criterion should be 3 orders of magnitude smaller than the stopping criterion for the outer iteration. This work employs full GMRES to reduce the relative residual to  $10^{-11}$ , ensuring an accurate approximation for  $M_H^{-1}$ . Since it is only performed on the coarsest grid, it will not lead to a high computational cost.

The wall-clock time for the preconditioned Krylov solver to reach the stopping criterion is denoted as  $t$ . The speedup  $S_p$  is defined by

$$S_p = \frac{t_r}{t_p},$$

where  $t_r$  and  $t_p$  are the wall-clock times for reference and parallel computations, respectively. The parallel efficiency  $E_p$  is given by

$$E_p = \frac{S_p}{np} = \frac{t_r}{t_p \cdot np},$$

where  $np$  is the number of processors. It should be noted that when performing computations within a single compute node, the reference time is the wall-clock time of sequential computation. When performing distributed computing across multiple compute nodes, the reference time will be the wall-clock time of computation on a single fully utilized node.

In this section, we first validate the numerical accuracy and algorithmic flexibility of our parallel framework for three test problems. To evaluate the performance of our solver, we then focus on weak scaling, which examines the solver performance when the problem size and the number of processing elements increase proportionally. This allows us to assess our parallel solver ability to solve large-scale heterogeneous Helmholtz problems with minimized pollution error. We next extensively investigate the strong scaling of our parallel solution method, examining the speedup and parallel efficiency as the number of processing elements increases while the problem size remains constant. Additional performance analysis for each part of the parallel framework can be found in Appendix A.

**5.1. Validation.** We validate the numerical experiments by comparing results from both serial and parallel computations with analytical solutions, as well as by observing the wave propagation patterns of model problems with non-constant wavenumber. Additionally, we conducted a preliminary exploration of the flexibility of various Krylov-based iterative methods and multigrid cycle types in the parallel framework.

**5.1.1. Sequential and parallel computing.** For the accuracy validation, our parallel solver is used to solve the so-called 3D closed-off problem, which has an analytical solution. According to the analytical solution given by equation (2.2), we can estimate the magnitude of the error between the numerical and analytical solutions. Figure 5.1 shows the logarithmic error  $\log_{10}(|u - u_{\text{analy}}|)$  of numerical results obtained by CSLP-preconditioned IDR(4). The left one is the result obtained by sequential computing, and the right one is obtained by parallel computing. Within the error tolerance, the numerical results agree well with the analytical solution, and the parallel computing setup does not introduce additional errors. It should be noted that the small but visible differences arise due to the accumulation of the rounding error. For example, the dot product  $[a_1 \ a_2 \ a_3 \ a_4] \cdot [b_1 \ b_2 \ b_3 \ b_4]^T$  calculated by  $(a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4)$  and  $(a_1b_1 + a_3b_3) + (a_2b_2 + a_4b_4)$  may yield small differences, especially for massively parallel computing. Additionally, in the aspect of the matrix, blockwise domain partitioning actually performs some elementary row and column transformations on the matrix compared to sequential computing, leading to further differences in the results. Extensive numerical experiments indicate that these differences are within an acceptable error range and do not significantly affect the convergence and accuracy.

**5.1.2. Model problems with non-constant wavenumber.** The so-called wedge problem is a typical model with non-constant wavenumbers. Figure 5.2 shows a reasonable wave diffraction pattern of the wedge problem at  $f = 40$  Hz and 80 Hz, which are obtained by the parallel CSLP-preconditioned IDR(4) solver. Note that the wavefront is significantly curved at the slow-to-fast interface. The wave is mainly reflected in that transition region. It illustrates that the parallel framework also works for the case of non-constant wavenumbers.

For practical application, we further consider the challenging 3D SEG/EAGE Salt model including complex geometries (salt domes) and a real large-size computational domain. To allow multi-level geometric coarsening, we utilize a domain with grid size  $641 \times 641 \times 193$ ,



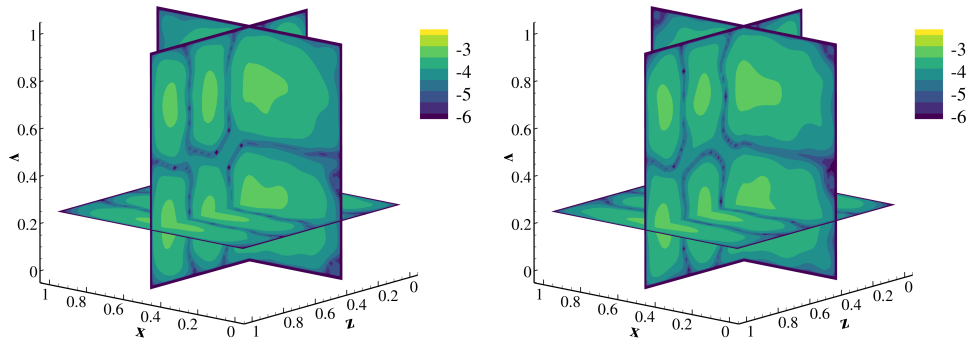


FIG. 5.1. The logarithmic error  $\log_{10}(|u - u_{analy}|)$  of numerical results obtained by CSLP-preconditioned IDR(4), with  $np = 1$  (left) and  $np = 3 \times 3 \times 3$  (right), for 3D closed-off problem grid size  $65 \times 65 \times 65$ ,  $k = 40$ .

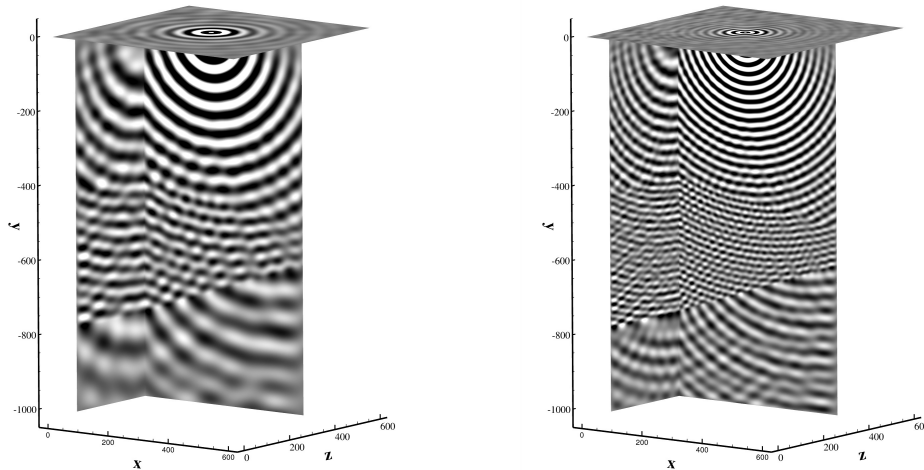


FIG. 5.2. Real part of numerical solutions for 3D wedge problem with grid size  $193 \times 321 \times 193$  at  $f = 40$  Hz (left), and  $385 \times 641 \times 385$  at 80 Hz (right).

as it has a high degree of divisibility by powers of two in each dimension. In this case, with the coarsest-grid criterion given by  $17 \times 17 \times 17$ , the coarsest grid will have dimension  $41 \times 41 \times 13$ . A point source is located at  $(x_1, x_2, x_3) = (3200, 3200, 0)$ . Figure 5.3 gives a reasonable wave diffraction pattern of this model at  $f = 5Hz$ . It is obtained by our parallel CSLP-preconditioned IDR(4) solver.

**5.1.3. Krylov methods.** Various Krylov-based iterative methods can be implemented in our parallel framework; see Table 5.1. For the 3D closed-off problem with Dirichlet boundary conditions, GMRES requires the least number of iterations, and IDR(4) is closer to GMRES than Bi-CGSTAB. These relative relationships can be expected in the theoretical analysis of these algorithms. One can find that the count of matrix-vector products is constant with respect to the number of processors for GMRES, while it is not for Bi-CGSTAB and IDR(4). It is because GMRES is a very robust and stable algorithm, whereas Bi-CGSTAB and IDR(s)



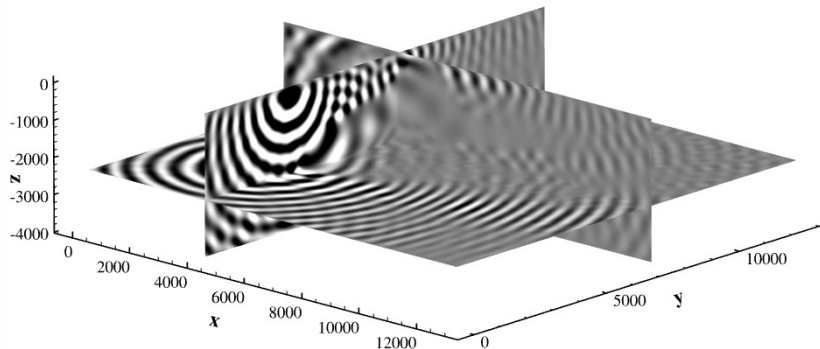


FIG. 5.3. Real part of the solutions for 3D SEG/EAGE Salt Model with grid size  $641 \times 641 \times 193$ , at  $f = 5$  Hz.

are sensitive to the accumulation of rounding errors and the re-ordering of the linear system. Additionally, IDR(s) is initialized by random vectors and, therefore, one cannot obtain a totally constant number of iterations if the seed is not manually chosen. As Bi-CGSTAB requires a much larger number of matrix-vector multiplications, the parallel performance of GMRES and IDR(4) will be mainly considered.

TABLE 5.1

*The convergence behavior of different iterative methods for 3D closed-off problem, grid size  $65 \times 65 \times 65$ ,  $k = 40$ .*

$np$	#Matvec		
	GMRES	Bi-CGSTAB	IDR(4)
1	145	359	192
$2 \times 1 \times 1$	145	359	211
$1 \times 2 \times 3$	145	403	192
$3 \times 3 \times 3$	145	407	201

**5.1.4. Multigrid cycle types.** Our parallel framework can solve the CSLP preconditioner not only using one multigrid V-cycle, but also F-cycle. Let us take the 3D wedge problem as an example. We found that if  $\beta_2 = -0.5$ , even for small frequency  $f = 10$  Hz with grid size  $73 \times 193 \times 73$ , the convergence result cannot be obtained by using the F-cycle, but it is possible by using the V-cycle. However, the Krylov solvers using one multigrid F-cycle should converge faster than those using one V-cycle, theoretically. Thus, the effect on convergence of the complex shift  $\beta_2$  in CSLP is studied. As shown in Figure 5.4, when the grid is  $193 \times 321 \times 193$  and  $f = 40$  Hz, if a larger complex shift ( $-1 \leq \beta_2 < -0.5$ ) is used, Krylov solvers can obtain better convergence results by using the F-cycle. Figure 5.4 confirms the introduction in Section 3.3, namely, the choice of the complex shift parameter in the preconditioner indeed has a significant impact on the performance of multigrid methods including F-cycle as well as V-cycle. In the case of CSLP, a smaller complex shift generally leads to a better preconditioner, but the complex shift has to be large enough for standard multigrid methods to converge; see [10, 21, 23]. For a small complex shift, moving from a V-cycle to an F-cycle may not improve the convergence. One potential explanation could be the damped Jacobi smoother,

used in the multigrid method, in principle diverges for Helmholtz-type problems. Therefore, using more smoothing steps, as in the F-cycle, may exacerbate the problems and lead to slower convergence compared to V-cycle. There is also evidence of this in Chapter 9 of [15].

When the complex shift is  $-1$ , as shown in Figure 5.5, it can be seen that the convergence of Krylov solvers using the F-cycle is faster than that using the V-cycle. For example, with  $np = 8$ , utilizing the F-cycle yields a wall-clock time of 336.60 s for GMRES and 101.03 s for IDR(4), while implementing the V-cycle results in a wall-clock time of 796.31 s for GMRES and 149.16 s for IDR(4). In general, the F-cycle involves more computation than the V-cycle, but the improved convergence can still make it a viable option in certain cases.

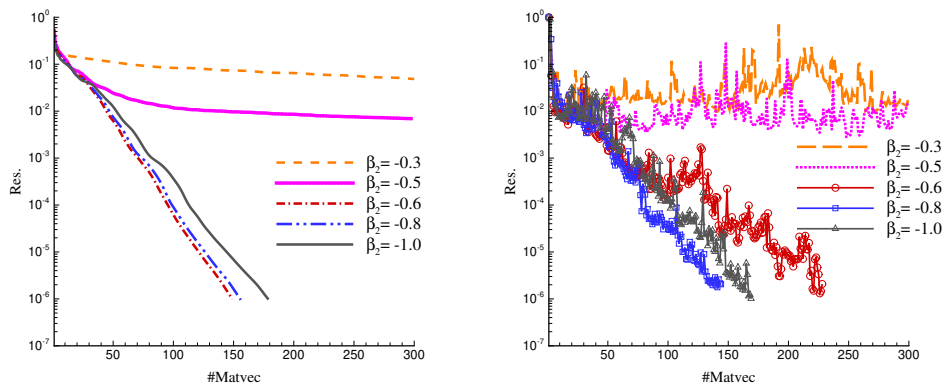


FIG. 5.4. Convergence behavior of the parallel CSLP-preconditioned Krylov solvers, GMRES (left) and IDR(s) (right). One multigrid F-cycle is used for preconditioner solving. The 3D wedge problem with grid size  $193 \times 321 \times 193$  at  $f = 40$  Hz is solved. Different complex shifts of CSLP ( $\beta_2$ ) are compared.

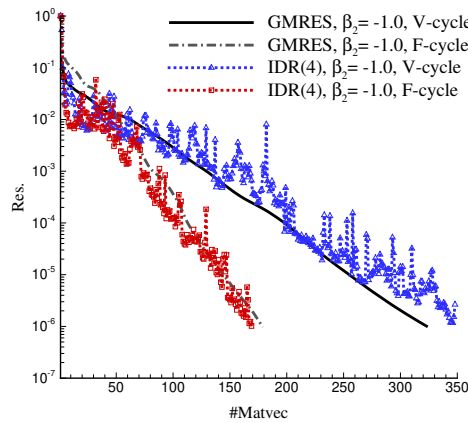


FIG. 5.5. Convergence behavior of the parallel CSLP-preconditioned Krylov solver using one multigrid V- or F-cycle for preconditioning. The complex shift of CSLP is  $\beta_2 = -1.0$ . The 3D wedge problem with grid size  $193 \times 321 \times 193$  at  $f = 40$  Hz is solved.

**5.2. Weak scaling.** The first parallel property that we are interested in is the weak scalability of our parallel solver. In the experiments, we increase the problem size and the number of processors proportionally, to maintain the same grid size for each processor.

The weak scaling results for the 3D closed-off problem are presented in Table 5.2. One can find that parallel GMRES does not show satisfying parallel scalability and consumes more CPU time than IDR(4), especially for large grid sizes. It is because GMRES needs long recurrences and more dot-product operations, hence more global communications. As the problem size increases, the sequential computation load which concerns the process of calculating the Given rotation matrix in GMRES becomes the bottleneck.

For IDR(4), when the number of processors and grid size increases proportionally, that is maintaining the same grid size for each processor, the wall-clock time remains relatively stable, indicating that our parallel framework can effectively handle large-scale Helmholtz problems with minimized pollution error. Since GMRES does not demonstrate weak scalability for problems requiring a large number of iterations, we will primarily use IDR(4) to solve the following increasingly complex model problems.

TABLE 5.2

*Weak scaling analysis of parallel CSLP-preconditioned GMRES and IDR(4) for 3D closed-off problem,  $k = 40$ .*

grid size	$np$	GMRES		IDR(4)	
		#Matvec	$t(s)$	#Matvec	$t(s)$
$129 \times 129 \times 129$	8	146	30.19	199	21.03
$193 \times 193 \times 193$	27	137	80.10	187	26.93
$257 \times 257 \times 257$	64	143	97.20	190	25.22

**5.3. Strong scaling.** In this section, we perform numerical experiments on a fixed problem size while varying the number of processors. The speedup and parallel efficiency are then calculated to assess the strong scaling of our parallel solver.

TABLE 5.3

*Comparisons with PETSc implementation for one GMRES iteration. Parallel CSLP-preconditioned GMRES is used to solve the 3D closed-off problem with grid size  $129 \times 129 \times 129$ ,  $k = 40$ .*

$np$	PETSc			Present		
	$t(s)$	$S_p$	$E_p$	$t(s)$	$S_p$	$E_p$
1	30.698	-	-	1.310	-	-
4	8.802	3.49	0.87	0.334	3.92	0.97
8	5.280	5.81	0.73	0.205	6.37	0.80
16	2.759	11.13	0.69	0.117	11.20	0.70

**5.3.1. 3D closed-off problem.** We initiate our analysis by comparing the strong scaling performance of our parallel framework with a PETSc [3, 4] implementation, specifically employing PETSc version 3.19.0 with complex support and optimized mode enabled. The application is compiled and tested in the same environment as our program. We use the CSLP-preconditioned GMRES algorithm to solve the 3D closed-off problem with a grid size of  $129 \times 129 \times 129$  and wavenumber  $k = 40$ . In the PETSc implementation, matrices are explicitly constructed, including the matrix defining the linear system and the matrix employed for constructing the CSLP preconditioner. The CSLP is implemented in a so-called Shell preconditioner of PETSc, where CSLP is inverted approximately by a default preconditioned GMRES solver. Both methods exhibit similar convergence properties. The

number of iterations required for the PETSc implementation is 155 for sequential computing and that of our implementation is 146. Table 5.3 presents the average wall-clock time for each outer GMRES iteration and the corresponding strong scaling. Our implementation demonstrates less time consumption and better parallel performance.

**5.3.2. 3D wedge problem.** For the 3D wedge problem, the performance of the parallel CSLP-preconditioned IDR(4) on one compute node and multiple compute nodes are shown in Table 5.4 and Table 5.5, respectively. The results show that the parallel framework exhibits good performance for non-constant wavenumbers and is scalable across multiple compute nodes as well as within a single node. With a growing number of processors, we observe a moderate decrease in parallel efficiency. This is mainly due to increased communication, which leads to a decrease in the computation/communication ratio.

TABLE 5.4

*Performance of the parallel CSLP-preconditioned IDR(4) for 3D wedge problem with grid size  $193 \times 321 \times 193$  at  $f = 40$  Hz.*

$np_x \times np_y \times np_z$	#Matvec	$t(s)$	$S_p$	$E_p$
$1 \times 1 \times 1$	395	1033.11		
$1 \times 2 \times 1$	421	557.63	1.85	0.93
$2 \times 2 \times 2$	377	152.88	6.76	0.84

TABLE 5.5

*Performance of the parallel CSLP-preconditioned IDR(4) for 3D wedge problem with grid size  $385 \times 641 \times 385$  at  $f = 80$  Hz.*

$np_x \times np_y \times np_z$	Nodes	#Matvec	$t(s)$	$S_p$	$E_p$
$4 \times 4 \times 3$	1	835	1313.82		
$4 \times 6 \times 4$	2	821	952.94	1.38	0.69
$6 \times 8 \times 4$	4	825	418.74	3.14	0.78
$6 \times 8 \times 6$	6	832	298.33	4.40	0.73

**5.3.3. 3D SEG/EAGE salt model.** We are interested in the scalability properties of our parallel framework in realistic applications. Hence, we solve this model problem at a fixed frequency on a growing number of compute nodes. Table 5.6 collects the number of matrix-vector multiplications and the wall-clock time versus the number of compute nodes. The fact that solving a linear system with approximately 79.3 million unknowns within hundreds of seconds shows the capability of solving a realistic 3D high-frequency problem with limited memory and time consumption. One can also find the nice property that the number of matrix-vector multiplications keeps independent of the number of compute nodes. Taking the wall-clock time on a single computing node as a reference, we can observe fairly good scaled parallel efficiency (around 0.8) for such a large-scale complex model.

Without loss of generality, we also tried to obtain the parallel performance of our method on a different platform as a complement to this study. In addition to DelftBlue (which includes 48 cores per compute node), we used a commercial supercomputer named Magic Cube 3\* (which includes 32 cores per compute node) to perform this numerical experiment. Magic Cube 3 is a supercomputer managed by Shanghai Supercomputer Center. It runs on CentOS Linux release 7.5. Each compute node is equipped with two Intel Xeon Gold 6142 processors

\*Magic Cube 3: <https://www.ssc.net.cn/en/resource-hardware.html>

with 16 cores at 2.6 GHz, and  $12 \times 16\text{G}$  DDR4 2666MHz ECC REG memory. The cluster is equipped with an Intel Omni-Path high-speed network with a transmission bandwidth of 100 Gb/s for interconnectivity between compute nodes and storage systems. On Magic Cube 3, Intel Fortran 17.0.4 and Intel MPI 17.4.239 will be used instead.

TABLE 5.6

*Performance of the parallel CSLP-preconditioned IDR(4) on DelftBlue for 3D SEG/EAGE Salt Model with grid size  $641 \times 641 \times 193$  at  $f = 5$  Hz.*

$np_x \times np_y \times np_z$	Nodes	#Matvec	$t(s)$	$S_p$	$E_p$
$6 \times 4 \times 2$	1	413	897.25		
$6 \times 8 \times 2$	2	423	510.56	1.76	0.88
$6 \times 8 \times 4$	4	423	298.86	3.00	0.75
$9 \times 8 \times 4$	6	404	203.31	4.41	0.74

TABLE 5.7

*Performance of the parallel CSLP-preconditioned IDR(4) on Magic Cube 3 for 3D SEG/EAGE Salt Model with grid size  $641 \times 641 \times 193$  at  $f = 5$  Hz.*

$np_x \times np_y \times np_z$	Nodes	#Matvec	$t(s)$	$S_p$	$E_p$
$4 \times 4 \times 2$	1	405	505.14		
$4 \times 4 \times 4$	2	418	287.60	1.76	0.88
$8 \times 8 \times 2$	4	390	155.64	3.25	0.81

As shown in Table 5.7, we can still achieve satisfactory parallel performance. A moderate decrease in terms of the scaled parallel efficiency should be due to the different bandwidths of the platforms. These results indicate that our parallel framework can be used on different computational platforms. This adaptability is crucial for realistic applications.

In summary, our parallel solver exhibits good weak and strong scaling for a variety of test problems, including large-scale, complex, and realistic applications. The results demonstrate the solver potential for solving challenging 3D large-scale heterogeneous Helmholtz problems with limited memory and time consumption.

**6. Conclusions.** In this paper, we developed a matrix-free parallel framework of CSLP-preconditioned Krylov subspace methods, such as GMRES, Bi-CGSTAB, and IDR(s), for 3D large-scale Helmholtz problems in heterogeneous media. The preconditioning operator is approximately inverted by a standard 3D multigrid method. We validate the numerical accuracy by comparisons with an analytical solution, as well as observations of the wave pattern. Both weak and strong scaling properties of our parallel framework for typical non-constant wavenumber model problems are studied. Additionally, a POP performance analysis has been provided in Appendix A to address the bottleneck of this framework.

To sum up, our research presents a novel matrix-free parallelization of the CSLP preconditioner for the Helmholtz problems. Our parallel implementation maintains good convergence properties without excessive memory consumption, making it an effective alternative to traditional matrix-based preconditioners. Moreover, our work provides a robust and scalable matrix-free parallel-computing framework for solving the Helmholtz problem in increasingly complex 3D scenarios, allowing for the use of various Krylov subspace methods and multigrid methods. Its weak scalability makes it possible to solve the Helmholtz problems with minimized pollution error by using a very large grid size. It also provides an implementation direction for researchers to further develop parallel scalable iterative solvers with wavenumber-independent convergence. Finally, we demonstrate the effectiveness and scalability of our

solver/preconditioner combinations in solving the Helmholtz equation on a large-scale parallel architecture, which may be beneficial to engineers to solve large-scale heterogeneous Helmholtz problems.

Future work will focus on further improving the parallel efficiency of the solver and exploring the use of advanced preconditioning techniques to enhance the convergence of Krylov-based iterative methods.

**Acknowledgments.** We would like to acknowledge the support of the CSC scholarship No. 202006230087. The authors are grateful to the referees for their helpful comments and suggestions that improved the quality of the paper considerably.

### Appendix A. POP performance analysis.

With traditional performance metrics such as speed-up and efficiency, it is difficult to understand the actual execution behavior of a parallel program and identify the cause of poor performance and where it occurs. In order to explore the bottleneck of our parallel framework and which part can be improved, this section will consider the performance assessment of our code, using the methodology of the Performance Optimisation and Productivity (POP) provided by the EU HPC Centre of Excellence (CoE)<sup>†</sup>. The POP methodology can help us build a quantitative picture of application behavior by a set of POP performance metrics, including parallel efficiency (PE), load balance (LB), communication efficiency (CommE), serialization efficiency (SerE), transfer efficiency (TE), and so on. The metrics are computed as efficiencies ranging from 0 to 1, where higher values are more desirable. In general, efficiencies 0.8 are considered acceptable, while lower values signal performance concerns that warrant further investigation. We use the following open-source tools: Score-P [32] for profiling and tracing, Scalasca [26] for extended analyses, and CUBE [31] for presentation.

TABLE A.1

*Execution efficiency of the parallel CSLP-preconditioned IDR(4) on different numbers of compute nodes of DelftBlue for 3D wedge model problem with grid size  $385 \times 641 \times 385$  at 80 Hz.*

Compute nodes <i>np</i>	1	2	6
Parallel Efficiency (PE) <sup>1</sup>	0.83	0.78	0.32
+ Load Balance (LB) <sup>2</sup>	0.91	0.82	0.78
+ Communication Efficiency (CommE) <sup>3</sup>	0.92	0.96	0.41
++ Serialisation Efficiency (SerE) <sup>4</sup>	0.92	0.96	0.42
++ Transfer Efficiency (TE) <sup>5</sup>	0.99	1.00	0.97

<sup>1</sup> *Parallel efficiency* is the ratio of mean computation time to total runtime of all processes

<sup>2</sup> *Load balance* is the mean/maximum ratio of computation time outside of MPI

<sup>3</sup> *Communication efficiency* is the ratio of maximum computation time to total runtime.

<sup>4</sup> *Serialisation efficiency* is estimated from idle time within communications where no data is transferred.

<sup>5</sup> *Transfer efficiency* relates to essential time spent in data transfers.

Table A.1 summarizes the performance assessment of the parallel CSLP-preconditioned IDR(4) on different numbers of compute nodes of DelftBlue for the 3D wedge model problem with grid size  $385 \times 641 \times 385$  at 80 Hz. The parallel efficiency drops from 0.83 to 0.5 when the number of processes reaches 288. We find that among the two factors contributing to parallel efficiency, a high load balance is maintained, but communication efficiency decreases.

<sup>†</sup>POP CoE: <https://www.pop-coe.eu>

Among the two aspects of communication efficiency, excellent transfer efficiency is maintained, while the most significant inefficiency is serialization, which concerns processes waiting at communication points due to temporal imbalance.

To determine which part contributes to the poor serialization efficiency, we further study the performance of preconditioning, matrix-vector multiplication, and dot-product operations, as shown in Tables A.2, A.3, and A.4, respectively. It can be seen that parallel matrix-vector multiplications can maintain a fairly good efficiency. The preconditioning component, exhibiting similar behavior to the whole framework in Table A.1, can be considered as the main factor that affects the overall efficiency. The results in Table A.4 reveal that the dot product operation is one of the main reasons for the low serialization efficiency. Thus, we can conclude that the preconditioning step becomes the bottleneck, because it uses full GMRES to solve coarse grid problems, which involve numerous inner product operations. Further efficiency optimization in this direction will be implemented in future work.

TABLE A.2

*Execution efficiency of preconditioning for parallel IDR(4) on different numbers of compute nodes of DelftBlue.*

Compute nodes	1	2	6
<i>np</i>	48	96	288
Parallel Efficiency (PE)	0.80	0.79	0.25
+ Load Balance (LB)	0.90	0.86	0.79
+ Communication Efficiency (CommE)	0.89	0.92	0.31
++ Serialisation Efficiency (SerE)	0.90	0.92	0.32
++ Transfer Efficiency (TE)	0.99	0.99	0.97

TABLE A.3

*Execution efficiency of matrix-vector multiplications of parallel IDR(4) on different numbers of compute nodes of DelftBlue.*

Compute nodes	1	2	6
<i>np</i>	48	96	288
Parallel Efficiency (PE)	0.70	0.62	0.55
+ Load Balance (LB)	0.90	0.77	0.82
+ Communication Efficiency (CommE)	0.77	0.81	0.67
++ Serialisation Efficiency (SerE)	0.78	0.82	0.67
++ Transfer Efficiency (TE)	0.99	0.99	0.99

TABLE A.4

*Execution efficiency of dot-product operations of parallel IDR(4) on different numbers of compute nodes of DelftBlue.*

Compute nodes	1	2	6
<i>np</i>	48	96	288
Parallel Efficiency (PE)	0.53	0.21	0.18
+ Load Balance (LB)	0.70	0.55	0.65
+ Communication Efficiency (CommE)	0.75	0.38	0.28
++ Serialisation Efficiency (SerE)	0.76	0.38	0.29
++ Transfer Efficiency (TE)	1.00	0.99	1.00



## REFERENCES

- [1] F. AMINZADEH, J. BRAC, AND T. KUNZ, *SEG/EAGE 3D salt and overthrust models*, in SEG/EAGE 3-D Modeling Series, No. 1: Distribution CD of Salt and Overthrust Models, SEG Book Series, 1997.
- [2] I. M. BABUSKA AND S. A. SAUTER, *Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?*, SIAM J. Numer. Anal., 34 (1997), pp. 2392–2423.
- [3] S. BALAY, S. ABHYANKAR, M. F. ADAMS, S. BENSON, J. BROWN, P. BRUNE, K. BUSCHELMAN, E. 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, S. ZAMPINI, H. ZHANG, H. ZHANG, AND J. ZHANG, *PETSc/TAO users manual*, Tech. Report ANL-21/39 - Revision 3.19, Argonne National Laboratory, 2023.
- [4] S. BALAY, W. D. GROPP, L. C. MCINNES, AND B. F. SMITH, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Basel, 1997, pp. 163–202.
- [5] A. BAYLISS, C. I. GOLDSTEIN, AND E. TURKEL, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.
- [6] Y. BOUBENDIR, X. ANTOINE, AND C. GEUZAIN, *A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation*, J. Comput. Phys., 231 (2012), pp. 262–280.
- [7] H. CALANDRA, S. GRATTON, X. PINEL, AND X. VASSEUR, *An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media*, Numer. Linear Algebra Appl., 20 (2013), pp. 663–688.
- [8] H. CALANDRA, S. GRATTON, AND X. VASSEUR, *A geometric multigrid preconditioner for the solution of the Helmholtz equation in three-dimensional heterogeneous media on massively parallel computers*, in Modern Solvers for Helmholtz Problems, D. Lahaye, J. Tang, and K. Vuik, eds., Springer, Cham, 2017, pp. 141–155.
- [9] Z. CHEN AND X. XIANG, *A source transfer domain decomposition method for Helmholtz equations in unbounded domain*, SIAM J. Numer. Anal., 51 (2013), pp. 2331–2356.
- [10] P. H. COCQUET AND M. J. GANDER, *How large a shift is needed in the shifted Helmholtz preconditioner for its effective inversion by multigrid?*, SIAM J. Sci. Comput., 39 (2017), pp. A438–A478.
- [11] T. P. COLLIGNON AND M. B. VAN GIJZEN, *Minimizing synchronization in IDR(s)*, Numer. Linear Algebra Appl., 18 (2011), pp. 805–825.
- [12] F. COLLINO, S. GHANEMI, AND P. JOLY, *Domain decomposition method for harmonic wave propagation: a general presentation*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 171–211.
- [13] DELFT HIGH PERFORMANCE COMPUTING CENTRE (DHPC), *DelftBlue Supercomputer (Phase 1)*. <https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1>, 2022.
- [14] J. DOUGLAS JR AND D. B. MEADE, *Second-order transmission conditions for the Helmholtz equation*, in Ninth International Conference on Domain Decomposition Methods, P. E. Bjørstad, M. S. Espedal and D. Keyes, eds., DDM.org, 1998, pp. 434–440.
- [15] V. DWARKA, *Iterative methods for time-harmonic waves: towards accuracy and scalability*, PhD. Thesis, Delft University of Technology, Delft, 2022.
- [16] V. N. S. R. DWARKA AND C. VUIK, *Scalable convergence using two-level deflation preconditioning for the Helmholtz equation*, SIAM J. Sci. Comput., 42 (2020), pp. A901–A928.
- [17] B. ENQUIST AND L. YING, *Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers*, Multiscale Model. Simul., 9 (2011), pp. 686–710.
- [18] Y. A. ERLANGGA, *A robust and efficient iterative method for the numerical solution of the Helmholtz equation*, PhD. Thesis, Delft University of Technology, Delft, 2005.
- [19] Y. A. ERLANGGA, C. W. OOSTERLEE, AND C. VUIK, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492.
- [20] Y. A. ERLANGGA, C. VUIK, AND C. W. OOSTERLEE, *On a class of preconditioners for solving the Helmholtz equation*, Appl. Numer. Math., 50 (2004), pp. 409–425.
- [21] O. G. ERNST AND M. J. GANDER, *Why it is difficult to solve Helmholtz problems with classical iterative methods*, in Numerical Analysis of Multiscale Problems, I. G. Graham, T. Y. Hou, O. Lakkis, and R. Scheichl, eds., Springer, Berlin, Heidelberg, 2012, pp. 325–363.
- [22] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, G. A. Watson, ed., Berlin, Heidelberg, 1976, Springer, pp. 73–89.
- [23] M. J. GANDER, I. G. GRAHAM, AND E. A. SPENCE, *Applying GMRES to the Helmholtz equation with shifted laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed?*, Numer. Math., 131 (2015), pp. 567–614.
- [24] M. J. GANDER, F. MAGOULES, AND F. NATAF, *Optimized Schwarz methods without overlap for the Helmholtz equation*, SIAM J. Sci. Comput., 24 (2002), pp. 38–60.
- [25] M. J. GANDER AND H. ZHANG, *A class of iterative solvers for the Helmholtz equation: Factorizations*,



- sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods, *SIAM Rev.*, 61 (2019), pp. 3–76.
- [26] M. GEIMER, F. WOLF, B. J. N. WYLIE, E. ÁBRAHÁM, D. BECKER, AND B. MOHR, *The Scalasca performance toolset architecture*, *Concur. Comput.-Pract. Exp.*, 22 (2010), pp. 702–719.
- [27] D. GORDON AND R. GORDON, *Robust and highly scalable parallel solution of the Helmholtz equation with large wave numbers*, *J. Comput. Appl. Math.*, 237 (2013), pp. 182–196.
- [28] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, *J. Res. Nat. Bur. Stand.*, 49 (1952), pp. 409–436.
- [29] L. R. HOCKING AND C. GREIF, *Optimal complex relaxation parameters in multigrid for complex-shifted linear systems*, *SIAM J. Matrix Anal. Appl.*, 42 (2021), pp. 475–502.
- [30] H. KNIBBE, C. W. OOSTERLEE, AND C. VUIK, *GPU implementation of a Helmholtz Krylov solver preconditioned by a shifted Laplace multigrid method*, *J. Comput. Appl. Math.*, 236 (2011), pp. 281–293.
- [31] M. KNOBLOCH, P. SAVIANKOU, M. SCHLÜTTER, A. VISSER, AND B. MOHR, *A picture is worth a thousand numbers—enhancing Cube’s analysis capabilities with plugins*, in *Tools for High Performance Computing 2018 / 2019*, H. Mix, C. Niethammer, H. Zhou, W. E. Nagel, and M. M. Resch, eds., Springer, Cham, 2021, pp. 237–259.
- [32] A. KNÜPFER, C. RÖSSEL, D. A. MEY, S. BIERSDORFF, K. DIETHELM, D. ESCHWEILER, M. GEIMER, M. GERNDT, D. LORENZ, A. MALONY, W. E. NAGEL, Y. OLEJNIK, P. PHILIPPEN, P. SAVIANKOU, D. SCHMIDL, S. SHENDE, R. TSCHÜTER, M. WAGNER, B. WESARG, AND F. WOLF, *Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir*, in *Tools for High Performance Computing 2011*, H. Brunst, M. S. Müller, W. E. Nagel, and M. M. Resch, eds., Springer, Berlin, Heidelberg, 2012, pp. 79–91.
- [33] A. V. KONONOV, C. D. RIYANTI, S. W. DE LEEUW, C. W. OOSTERLEE, AND C. VUIK, *Numerical performance of a parallel solution method for a heterogeneous 2D Helmholtz equation*, *Comput. Vis. Sci.*, 11 (2007), pp. 139–146.
- [34] S. P. MACLACHLAN AND C. W. OOSTERLEE, *Algebraic multigrid solvers for complex-valued matrices*, *SIAM J. Sci. Comput.*, 30 (2008), pp. 1548–1571.
- [35] L. C. MCINNES, R. F. SUSAN-RESIGA, D. E. KEYES, AND H. M. ATASSI, *Additive Schwarz methods with nonreflecting boundary conditions for the parallel computation of Helmholtz problems*, *Contemp. Math.*, 218 (1998), pp. 325–333.
- [36] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, *SIAM J. Numer. Anal.*, 12 (1975), pp. 617–629.
- [37] R. E. PLESSIX AND W. A. MULDER, *Separation-of-variables as a preconditioner for an iterative Helmholtz solver*, *Appl. Numer. Math.*, 44 (2003), pp. 385–400.
- [38] C. RIYANTI, A. KONONOV, Y. ERLANGGA, C. VUIK, C. OOSTERLEE, R.-E. PLESSIX, AND W. MULDER, *A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation*, *J. Comput. Phys.*, 224 (2007), pp. 431–448.
- [39] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.
- [40] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, *SIAM J. Sci. Stat. Comput.*, 7 (1986), pp. 856–869.
- [41] A. SCHÄDLE AND L. ZSCHIEDRICH, *Additive Schwarz method for scattering problems using the PML method at interfaces*, in *Domain Decomposition Methods in Science and Engineering XVI*, Springer, Berlin, 2007, pp. 205–212.
- [42] A. H. SHEIKH, D. LAHAYE, L. G. RAMOS, R. NABBEN, AND C. VUIK, *Accelerating the shifted Laplace preconditioner for the Helmholtz equation by multilevel deflation*, *J. Comput. Phys.*, 322 (2016), pp. 473–490.
- [43] P. SONNEVELD AND M. B. VAN GIJZEN, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, *SIAM J. Sci. Comput.*, 31 (2009), pp. 1035–1062.
- [44] C. C. STOLK, *A rapidly converging domain decomposition method for the Helmholtz equation*, *J. Comput. Phys.*, 241 (2013), pp. 240–252.
- [45] M. TAUS, L. ZEPEDA-NÚÑEZ, R. J. HEWETT, AND L. DEMANET, *L-sweeps: A scalable, parallel preconditioner for the high-frequency Helmholtz equation*, *J. Comput. Phys.*, 420 (2020), Art. 109706 (32 pages).
- [46] A. TOSELLI, *Overlapping methods with perfectly matched layers for the solution of the Helmholtz equation*, in *Eleventh International Conference on Domain Decomposition Methods*, C. Lai, P. Bjorstad, M. Cross, and O. Widlund, eds., DDM.org, 1999, pp. 551–558.
- [47] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, *SIAM J. Sci. Stat. Comput.*, 13 (1992), pp. 631–644.
- [48] M. B. VAN GIJZEN, Y. A. ERLANGGA, AND C. VUIK, *Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted laplacian*, *SIAM J. Sci. Comput.*, 29 (2007), pp. 1942–1958.