High-Performance Iterative Methods for the Helmholtz Equation

Chen, J.

DOI
10.4233/uuid:8e7c44b5-091a-4a14-98d7-762be3ee59dd

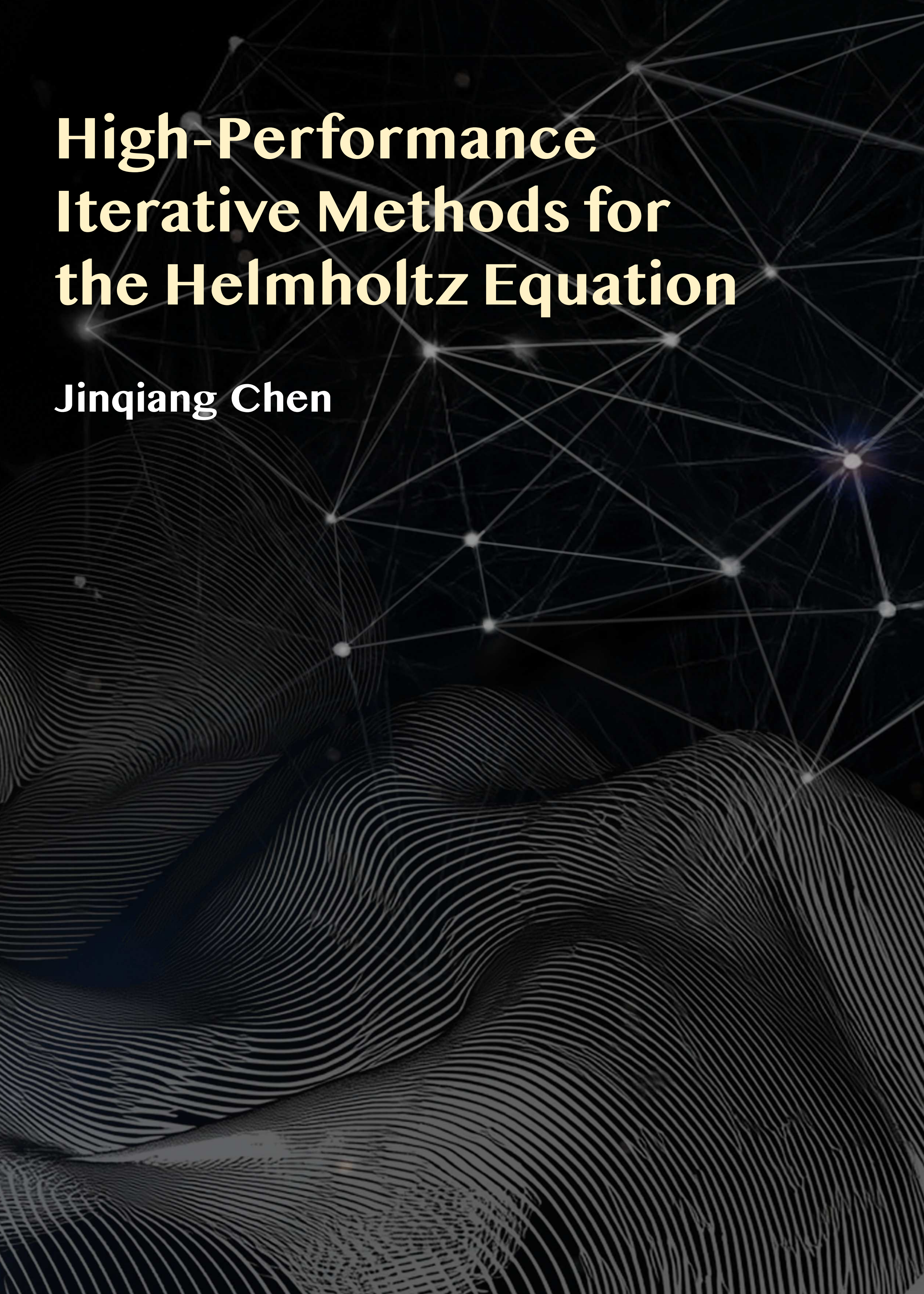
Publication date
2025

Document Version
Final published version

Citation (APA)
Chen, J. (2025). *High-Performance Iterative Methods for the Helmholtz Equation*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:8e7c44b5-091a-4a14-98d7-762be3ee59dd

High-Performance Iterative Methods for the Helmholtz Equation

Jinqiang Chen



High-Performance Iterative Methods for the Helmholtz Equation

HIGH-PERFORMANCE ITERATIVE METHODS FOR THE HELMHOLTZ EQUATION

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Wednesday 2 July 2025 at 10:00 o'clock

by

Jinqiang CHEN

Master of Science in Power Engineering, Shanghai Jiao Tong University, China
born in Guangdong, China

This research was supported by the China Scholarship Council (CSC).

Keywords:	Helmholtz, CSLP, Deflation, Parallel Computing, Scalable, Matrix-Free, Multigrid
Printed by:	Ridderprint | www.ridderprint.nl
Cover by:	studio shu | shu.artdesign@gmail.com

ISBN 978-94-6522-403-9

An electronic copy of this dissertation is available at https://repository.tudelft.nl/.

To my parents,
while I accelerate algorithms,
how I wish I could decelerate time's passage upon you.

Contents

Summary

The numerical solution of the Helmholtz equation presents significant challenges in computational mathematics and scientific computing, particularly for high-frequency problems in heterogeneous media. This dissertation addresses these challenges through the development of high-performance iterative methods, focusing on the critical balance between numerical efficiency and practical implementation on modern computing architectures.

The research is motivated by the growing computational demands in seismic imaging and other wave propagation applications, where increasing frequencies and larger domains necessitate more efficient solution strategies. Traditional approaches often struggle with the combined challenges of wavenumber-dependent convergence, pollution errors, and substantial memory requirements, particularly for three-dimensional problems in heterogeneous media.

This work presents a comprehensive framework for solving large-scale Helmholtz problems through matrix-free parallel implementations of preconditioned iterative methods. The framework combines Complex Shifted Laplace Preconditioner (CSLP) with advanced deflation techniques, implemented in a manner that eliminates the need for explicit matrix storage while maintaining computational efficiency. A key innovation is the development of matrix-free implementations for higher-order deflation methods combined with the CSLP preconditioner, achieved through carefully designed re-discretization schemes that preserve the advantages of Galerkin coarsening.

The methodology progresses from two-dimensional implementations to fully three-dimensional frameworks, incorporating increasingly sophisticated preconditioning techniques. A significant achievement is the development of a matrix-free parallel multilevel deflation preconditioner that exhibits near wavenumber-independent convergence while maintaining excellent parallel scalability. The implementation utilizes a hybrid MPI+OpenMP parallelization strategy, effectively addressing both computational and memory challenges in extreme-scale scenarios.

Extensive numerical experiments validate the effectiveness of these methods across a range of problem types, from academic test cases to industrial-scale applications. Notably, the framework successfully resolves a challenging seismic model, involving approximately 3.8 billion degrees of freedom, while achieving 86% parallel efficiency when scaling to 2304 CPU cores. This demonstration of practical viability for large-scale heterogeneous problems represents a significant advance in computational capabilities for seismic imaging applications.

The research makes several fundamental contributions to the field of numerical analysis and scientific computing. First, it establishes new approaches for matrix-free implementation of state-of-the-art preconditioners, significantly reducing memory requirements while maintaining numerical efficiency. Second, it demonstrates the achievement of close-to wavenumber-independent convergence through carefully designed deflation strategies in a parallel computing environment. Third, it provides a comprehensive framework for solving extreme-scale Helmholtz problems that combines numerical robustness with practical applicability.

The methodologies developed in this work contribute to the broader field of scientific computing, demonstrating how careful algorithm design, combined with modern computing architectures, can address previously intractable problems in wave propagation modeling.

Samenvatting

De numerieke oplossing van de Helmholtzvergelijking vormt een belangrijke uitdaging in de computationele wiskunde en het wetenschappelijk rekenen, vooral voor hoogfrequente problemen in heterogene media. Dit proefschrift adresseert deze uitdagingen door de ontwikkeling van hoogwaardige iteratieve methoden, met focus op de cruciale balans tussen numerieke efficiëntie en praktische implementatie op moderne computerarchitecturen.

Het onderzoek wordt gemotiveerd door de groeiende computationele eisen in seismische beeldvorming en andere golfvoortplantingstoepassingen, waar toenemende frequenties en grotere domeinen efficiëntere oplossingsstrategieën vereisen. Traditionele benaderingen worstelen vaak met de gecombineerde uitdagingen van golfgetal-afhankelijke convergentie, vervuilingsfouten en substantiële geheugenbehoeften, vooral voor driedimensionale problemen in heterogene media.

Dit werk presenteert een uitgebreid raamwerk voor het oplossen van grootschalige Helmholtzproblemen door matrix-vrije parallelle implementaties van gepreconditioneerde iteratieve methoden. Het raamwerk combineert de Complex Shifted Laplace Preconditioner (CSLP) met geavanceerde deflatietechnieken, geïmplementeerd op een manier die de noodzaak van expliciete matrixopslag elimineert terwijl de computationele efficiëntie behouden blijft. Een belangrijke innovatie is de ontwikkeling van matrix-vrije implementaties voor hogere-orde-deflatiemethoden in combinatie met de CSLP-preconditioner, bereikt door zorgvuldig ontworpen her-discretisatieschema's die de voordelen van Galerkin-coarsening behouden.

De methodologie ontwikkelt zich van tweedimensionale implementaties naar volledig driedimensionale toepassingen, met steeds geavanceerdere preconditioneringstechnieken. Een belangrijke prestatie is de ontwikkeling van een matrix-vrije parallelle multilevel-deflatie-preconditioner die bijna golfgetal-onafhankelijke convergentie vertoont terwijl de uitstekende parallelle schaalbaarheid behouden blijft. De implementatie maakt gebruik van een hybride MPI+OpenMP parallellisatiestrategie, die effectief zowel de computationele als geheugenuitdagingen in zeer grote scenario's aanpakt.

Uitgebreide numerieke experimenten valideren de effectiviteit van deze methoden voor verschillende probleemtypes, van academische testcases tot industriële toepassingen. Met name lost het raamwerk met succes een uitdagend seismisch model op met ongeveer 3,8 miljard vrijheidsgraden, waarbij 86% parallelle efficiëntie wordt bereikt bij schaling naar 2304 CPU-kernen. Deze demonstratie van praktische haalbaarheid voor grootschalige heterogene problemen vertegenwoordigt een significante vooruitgang in computationele mogelijkheden voor seismische beeldvormingstoepassingen.

Het onderzoek levert verschillende fundamentele bijdragen aan het gebied van numerieke wiskunde en wetenschappelijk rekenen. Ten eerste introduceert het nieuwe benaderingen voor matrix-vrije implementatie van geavanceerde preconditioners, wat de geheugenbehoeften aanzienlijk vermindert terwijl de numerieke efficiëntie behouden blijft. Ten tweede demonstreert het het bereiken van bijna golfgetal-onafhankelijke convergentie door zorgvuldig ontworpen deflatiestrategieën in een parallelle rekenomgeving. Ten derde biedt het een uitgebreid raamwerk voor het oplossen van zeer grote Helmholtzproblemen dat numerieke robuustheid combineert met praktische toepasbaarheid.

De methodologieën ontwikkeld in dit werk dragen bij aan het bredere gebied van wetenschappelijk rekenen, en demonstreren hoe zorgvuldig algoritme-ontwerp, gecombineerd met moderne computerarchitecturen, voorheen onoplosbare problemen in golfvoortplantingsmodellering kan aanpakken.

1

Introduction

1.1. THE HELMHOLTZ EQUATION

Wave phenomena play a fundamental role in numerous physical processes and technological applications, from seismic wave propagation to electromagnetic wave transmission. The mathematical description of such phenomena in frequency domain through the Helmholtz equation, introduced by Hermann von Helmholtz in 1860 [1], has proven instrumental in understanding and modeling wave behavior. Over the past century, applications of the Helmholtz equation have expanded dramatically, particularly in geophysical exploration, medical imaging, and telecommunications. The evolution of solution approaches for the Helmholtz equation closely mirrors the advancement of computational capabilities. From early analytical solutions limited to simple geometries, to modern numerical methods capable of handling complex three-dimensional problems, this progression has continuously enabled new applications while simultaneously presenting new challenges.

1.1.1. MATHEMATICAL FORMULATION

The Helmholtz equation is a fundamental partial differential equation (PDE) arising in various fields of physics and engineering. It is typically expressed as:

$$-\Delta u - k^2 u = 0, \tag{1.1}$$

where $\Delta = \sum_{j=1}^{n} \partial_{x_j}^2$ denotes the Laplace operator in n variables for $n \in 1,2,3$, $u = u(\mathbf{x})$ is the unknown scalar field as a function of spatial coordinates $\mathbf{x} \in \mathbb{R}^n$, and $k > 0$ is real parameter called wavenumber, which relates the wave properties as follow

$$k = \frac{\omega}{c} = \frac{2\pi f}{c} = \frac{2\pi}{\lambda} \tag{1.2}$$

Here, ω represents the angular frequency of the wave, f is the frequency, λ stands for the wavelength, and c is the phase velocity in the medium. In many applications involving the modeling of phenomena propagating through heterogeneous media, a spatially varying velocity $c(\mathbf{x})$ hence a non-constant wavenumber $k(\mathbf{x})$ is implemented to accurately represent diverse velocity distributions. The Helmholtz equation thus models wave phenomena under time-harmonic conditions, where all field variables oscillate sinusoidally with a fixed frequency f.

The non-homogeneous Helmholtz equation commonly appears in practical applications. This formulation incorporates a source function $b(\mathbf{x})$ on the right-hand side of Equation (1.1), expressed as

$$b(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x_s}) \tag{1.3}$$

wherein $\mathbf{x_s}$ represents the spatial coordinates of the source within the specified domain. $\delta(\mathbf{x})$ is a Dirac delta function and satisfies

$$\int \delta(\mathbf{x})\, d\mathbf{x} = 1. \tag{1.4}$$

BOUNDARY CONDITIONS

When solving the Helmholtz equation within a bounded domain $\Omega \subset \mathbb{R}^n$, it is essential to specify appropriate boundary conditions on the boundary $\partial\Omega$ to ensure the well-posedness of the problem, as the equation inherently models the indefinite propagation of waves. The choice of boundary conditions reflects the physical nature of the problem and the behavior of waves at the domain boundaries. Several types of boundary conditions are commonly employed in the context of the Helmholtz equation.

Dirichlet Boundary Conditions The Dirichlet boundary condition prescribes the value of the solution on the boundary:

$$u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \tag{1.5}$$

where $f(\mathbf{x})$ is a known function defined on the boundary $\partial\Omega$. The homogeneous Dirichlet boundary condition, that is $f(\mathbf{x}) = 0$, models scenarios where the acoustic pressure vanishes at the boundary, effectively representing a perfectly absorbing or pressure-release surface.

Neumann Boundary Conditions The Neumann boundary condition, representing sound-hard obstacles, specifies the normal derivative of the solution on the boundary:

$$\frac{\partial u(\mathbf{x})}{\partial n} = f(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \tag{1.6}$$

where $\mathbf{n}$ is the outward unit normal vector to the boundary $\partial\Omega$. The homogeneous Neumann boundary condition, that is $f(\mathbf{x}) = 0$, models a perfectly reflecting or sound-hard surface, indicating that there is no net flux of the wave field across the boundary.

Sommerfeld Radiation Boundary Conditions In many applications, the domain of interest is unbounded, and waves radiate to infinity. The Sommerfeld radiation condition is imposed to ensure that the solution represents outgoing waves at infinity and that no incoming waves originate from infinity:

$$\lim_{r\to\infty} r^{(n-1)/2}\left(\frac{\partial u}{\partial r} - \mathtt{i}ku\right) = 0, \tag{1.7}$$

where r is the radial distance from a fixed point, $\mathtt{i}$ is the imaginary unit, and n is the spatial dimension. This condition mathematically enforces that the solution behaves asymptotically like an outgoing spherical wave. The Sommerfeld condition ensures uniqueness of the solution. But it is impractical to implement directly in numerical computations due to the necessity of modeling an infinite domain. To overcome this, approximate radiation or absorbing boundary conditions are applied on the artificial boundary $\partial\Omega$ of a truncated computational domain. A commonly used first-order Sommerfeld boundary condition is

$$\frac{\partial u(\mathbf{x})}{\partial n} - \mathtt{i}ku(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega. \tag{1.8}$$

This condition is derived as an approximation of the Sommerfeld condition and allows outgoing waves to exit the computational domain with minimal reflection. It effectively simulates an open domain by absorbing waves at the boundary, thereby reducing artificial reflections that can contaminate the solution within the domain.

Higher-order absorbing boundary conditions and perfectly matched layers (PML) are more sophisticated techniques developed to further minimize reflections from the artificial boundaries in numerical simulations.

Mixed Boundary Conditions In some situations, different types of boundary conditions are applied on different portions of the boundary, leading to mixed boundary conditions. For example, part of the boundary may be reflective, while another part is absorbing:

$$\begin{cases} u(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_D, \\ \frac{\partial u(\mathbf{x})}{\partial n} - \mathrm{i}ku(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_S, \end{cases} \tag{1.9}$$

where Γ_D and Γ_S are disjoint subsets of $\partial\Omega$ corresponding to Dirichlet and Sommerfeld radiation boundary conditions, respectively. Mixed boundary conditions allow for greater flexibility in modeling complex physical scenarios, such as wave scattering by obstacles where different physical properties are present on various parts of the boundary.

The appropriate selection and implementation of boundary conditions are crucial for accurately modeling wave phenomena using the Helmholtz equation. Boundary conditions influence the existence and uniqueness of the solution and affect the stability and accuracy of numerical methods. In the context of numerical simulations, improper boundary conditions can lead to non-physical solutions, spurious reflections, and significant errors.

1.1.2. Physical Significance

The Helmholtz equation is a cornerstone in the mathematical modeling of time-harmonic wave phenomena across various fields of physics and engineering. Its significance stems from its ability to describe the spatial variation of wave fields under steady-state or oscillatory conditions, where the temporal dependence is sinusoidal.

Acoustics

In acoustics, the Helmholtz equation arises from the linearization of the fundamental fluid dynamics equations governing sound propagation. Starting with the continuity equation and the equation of motion for a compressible, inviscid fluid, and assuming small perturbations around a constant mean state, we obtain the acoustic wave equation:

$$\frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} - \nabla^2 U = 0, \tag{1.10}$$

where c is the speed of sound in the medium, and $U(\mathbf{x}, t)$ can represent the acoustic pressure, the density, and, under suitable assumptions, the velocity potential and the components of the velocity. Assuming time-harmonic solutions of the form $U(\mathbf{x}, t) = u(\mathbf{x})e^{-i\omega t}$, substitution into equation (1.10) yields the Helmholtz equation

$-\Delta u(\mathbf{x}) - \frac{\omega^2}{c^2} u(\mathbf{x}) = 0$. This formulation allows for the analysis of steady-state wave propagation and is fundamental in applications such as architectural acoustics, noise control engineering, and sonar technology.

ELECTROMAGNETISM

In electromagnetism, Maxwell's equations describe the behavior of electric and magnetic fields. In homogeneous, source-free media, and under the assumption of time-harmonic fields, Maxwell's equations reduce to the time-harmonic Maxwell' s equations

$$\nabla \times \mathbf{E} - \mathrm{i}\omega\mu\mathbf{H} = 0, \tag{1.11}$$

$$\nabla \times \mathbf{H} + \mathrm{i}\omega\epsilon\mathbf{E} - \sigma\mathbf{E} = 0. \tag{1.12}$$

where $\mathbf{E}$ and $\mathbf{H}$ represents the time-independent electric field and magnetic field, respectively. The electric permittivity (ϵ), magnetic permeability (μ), and conductivity (σ) characterize the electromagnetic properties of the propagation medium. In the context of a homogeneous and isotropic medium, both ϵ and μ maintain positive constant values, while σ represents a non-negative constant parameter. The conductivity is 0 in a perfect insulator, whereas it assumes positive values in electrically conducting media.

The system consists of two PDEs with vector-valued functions as their unknown variables. Eliminating $\mathbf{H}$, one obtains the second-order time-harmonic Maxwell's equations governing the electric field, wherein each component satisfies the Helmholtz equation with $k^2 = \omega^2\epsilon\mu + \mathrm{i}\omega\sigma\mu$:

$$\Delta E_j + k^2 E_j = 0, \quad \text{for} j = 1,2,3. \tag{1.13}$$

The speed of propagation of electromagnetic waves (e.g. of light) is $c = \frac{1}{\sqrt{\epsilon\mu}}$. This reduction is essential for analyzing electromagnetic wave propagation, including antenna design, waveguide analysis, and optical fiber communications.

ELASTODYNAMICS

In elastodynamics, the propagation of mechanical waves in homogeneous elastic solids is governed by the Navier equations:

$$\rho\frac{\partial^2 \mathbf{U}}{\partial t^2} = (\lambda + 2\mu)\nabla(\nabla\cdot\mathbf{U}) - \mu\nabla\times(\nabla\times\mathbf{U}), \tag{1.14}$$

where $\mathbf{U}$ is the displacement vector field, ρ is the density, and λ, μ are Lamé's constants characterizing the material's elastic properties. Assuming time-harmonic displacements $\mathbf{U}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x})e^{-\mathrm{i}\omega t}$, equation (1.14) transforms into:

$$-(\lambda + 2\mu)\nabla(\nabla\cdot\mathbf{u}) + \mu\nabla\times(\nabla\times\mathbf{u}) + \omega^2\rho\mathbf{u} = 0. \tag{1.15}$$

By decomposing $\mathbf{u}$ into longitudinal and transverse components, each satisfying a separate Helmholtz equation with different wave speeds, this framework captures both pressure wave (P-wave) and shear wave (S-wave) in solids, which is critical in seismology, nondestructive testing, and materials science.

The ability to model such a wide array of physical phenomena underpins the importance of Helmholtz equation in applied mathematics. It serves as a fundamental equation for understanding and predicting the behavior of waves in various media and under different conditions. Moreover, it provides a foundation for advanced numerical methods, which are indispensable tools for solving complex wave propagation problems in practical engineering applications.

1.1.3. APPLICATIONS

Initially applied primarily in acoustics and electromagnetics, the scope of Helmholtz equation applications expanded with technological advances in the 20th century. The development of seismic imaging techniques in 1920s [2] and medical ultrasound diagnostics in 1940s [3] established the equation's crucial role in modern imaging technologies. The advent of computational methods in 1960s enabled a pivotal shift from analytical to numerical solutions, enabling the treatment of more complex, realistic problems [4].

ACOUSTIC ENGINEERING AND NOISE CONTROL

In acoustic engineering, the Helmholtz equation is fundamental for analyzing sound propagation in various environments, which is essential for designing acoustic spaces and controlling noise.

Solving the Helmholtz equation offers valuable insights into modal characteristics and resonance phenomena within architectural acoustic environments such as concert halls and performance venues. Through rigorous mathematical analysis of these solutions, architectural designers can systematically optimize acoustic parameters to enhance sound clarity, timbral richness, and spatial distribution. Such comprehensive evaluations enable precise understanding of sound wave propagation and interaction with structural elements, ultimately facilitating the creation of auditory spaces that effectively realize the aesthetic and acoustic objectives of musical and theatrical performances. The equation also aids in designing enclosures and barriers that effectively attenuate sound, contributing to environmental noise control and occupational safety.

FULL WAVEFORM INVERSION IN GEOPHYSICAL EXPLORATION

In geophysics, accurate simulation of seismic waves is crucial for resource exploration and understanding geological structures.

Full Waveform Inversion (FWI) is a technique that utilizes the complete seismic data to construct high-resolution subsurface models. Frequency-domain FWI involves solving the Helmholtz equation across a range of frequencies and iteratively updating the Earth's parameter models to minimize the difference between observed and simulated seismic data [5–7].

Mathematically, FWI aims to find the model parameters (e.g., wave speed $c(\mathbf{x})$) that minimize the objective function:

$$J(\mathbf{m}) = \frac{1}{2}\sum_{\omega} \|d_{\text{obs}}(\omega) - d_{\text{sim}}(\omega;\mathbf{m})\|^2, \tag{1.16}$$

where d_{obs} is the observed seismic data, d_{sim} is the simulated data obtained by solving the Helmholtz equation. The model $\mathbf{m}$ represents discretized physical properties of the subsurface. By minimizing the misfit between observed and simulated wave fields, which involves solving a nonlinear optimization problem, researchers can iteratively reconstruct subsurface velocity models with unprecedented detail.

Medical Imaging

In medical diagnostics, modern imaging techniques leverage frequency-domain wave models to generate detailed anatomical representations with high spatial resolution.

For instance, transcranial ultrasound imaging is used for non-invasive diagnostics and therapeutic interventions within the brain. By mathematically modeling wave propagation through heterogeneous biological media, researchers can precisely predict acoustic field distributions within complex anatomical structures like the human skull. The Helmholtz equation enables quantitative analysis of wave interactions, accounting for critical factors such as sound speed variations, wave distortions, and transmission losses through different tissue interfaces. Advanced computational techniques for solving the Helmholtz equation allow researchers to generate high-resolution wave propagation models that can guide non-invasive therapeutic interventions [8].

Similarly, microwave tomography represents a novel medical imaging approach for brain stroke detection [9], leveraging the Helmholtz equation to model wave propagation through biological tissues. By analyzing the complex permittivity variations between healthy and diseased brain tissues, this technique enables rapid, non-invasive stroke identification and monitoring.

Optical Instrumentation

In optics, the Helmholtz equation underpins the analysis of diffraction patterns, which is fundamental in optical instrumentation and imaging technologies.

The paraxial approximation of the Helmholtz equation yields Gaussian beam solutions that describe laser beam propagation [10]. It is crucial for optical engineers to predict beam behavior in optical systems and designing laser resonators, cavities, and diode modes.

Besides the traditional paraxial approximation, the unidirectional Helmholtz equation (UHE) is particularly significant for modeling sharply focused laser beams, especially in turbid media that mimic biological tissues [11]. This approach allows for a more accurate simulation of light behavior in complex, multi-layered scattering environments, which is crucial for applications such as optical coherence tomography (OCT), multiphoton microscopy, and confocal microscopy.

The breadth and depth of applications underscore the Helmholtz equation's central role in modeling and understanding wave phenomena across multiple disciplines. Its study not only advances theoretical knowledge but also drives technological innovation, contributing to developments in acoustic design, geophysical exploration, medical diagnostics, and so forth. By providing precise and robust mathematical descriptions, the Helmholtz equation facilitates the exploration of complex physical systems and the development of sophisticated tools and techniques.

1.2. MODEL PROBLEMS

To evaluate the solution methods for the Helmholtz equation, we present a suite of model problems that capture essential challenges in Helmholtz applications. These test cases progress from homogeneous problems to heterogeneous configurations, incorporating varying wavenumbers and material properties in both two and three dimensions. The selected problems serve to assess convergence behavior and computational efficiency while representing practical scenarios in wave propagation phenomena.

1.2.1. TWO-DIMENSIONAL MODELS

MODEL PROBLEM 1 - CLOSED-OFF PROBLEM

This problem is the so-called 2D closed-off problem. A rectangular homogeneous domain $\Omega = [0,1]^2$ is considered. The source function is specified by

$$b\left(x,y\right) = \left(5\pi^2 - k^2\right)\sin\left(\pi x\right)\sin\left(2\pi y\right) - k^2, \ \ \Omega = [0,1] \tag{1.17}$$

It is supplied with the following Dirichlet conditions

$$u = 1, \ \text{ on } \partial\Omega \tag{1.18}$$

Thus, we can have the exact solution given by

$$u(x,y) = \sin\left(\pi x\right)\sin\left(2\pi y\right) + 1. \tag{1.19}$$

MODEL PROBLEM 2 - CONSTANT-WAVENUMBER PROBLEM WITH POINT SOURCE

We also consider a problem with a point source, which can be given by

$$b\left(x,y\right) = \delta\left(x - x_0, y - y_0\right), \ \ \Omega = [0,1] \tag{1.20}$$

In this problem, the point source is imposed at the center $(x_0, y_0) = (0.5, 0.5)$. The wave propagates outward from the center of the domain. The Dirichlet boundary conditions (denoted as MP-2a) or the first order Sommerfeld radiation conditions (denoted as MP-2b) are imposed, respectively.

The analytical solution for MP-2a is given by [12]

$$u\left(x,y\right) = \frac{4}{L}\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{\sin\left(\frac{i\pi x}{L}\right)\sin\left(\frac{i\pi x_0}{L}\right)\sin\left(\frac{j\pi y}{L}\right)\sin\left(\frac{j\pi y_0}{L}\right)}{\frac{i^2\pi^2 + j^2\pi^2}{L^2} - k^2} \tag{1.21}$$

where $k^2 \neq i^2\pi^2 + j^2\pi^2$. L is the side length of the square domain.

For MP-2b, the analytical solution is

$$u(\vec{r}) = \frac{\mathrm{i}}{4} H_0^{(1)}(k|\vec{r}|) \tag{1.22}$$

where $\vec{r} = (x - x_0, y - y_0)$, $H_0^{(1)}$ is a Hankel function. It should be noticed that if $\vec{r} = \vec{0}$, the Hankel function will be infinite. Therefore, the value of the Hankel function when $|\vec{r}| = h/2$ will be used in place of the value for $\vec{r} = \vec{0}$ when calculating the analytical solution.

MODEL PROBLEM 3 - WEDGE PROBLEM

Most physical problems of geophysical seismic imaging describe a heterogeneous medium. The so-called wedge problem [13] is a typical problem with a simple heterogeneity. It mimics three layers with different velocities hence different wavenumbers. As shown in Figure 1.1, the rectangular domain $\Omega = [0, 600] \times [-1000, 0]$ is split into three layers. Suppose the wave velocity c is constant within each layer but different from each other. A point source is located at $(x, y) = (300, 0)$.

The problem is given by

$$\begin{cases} -\Delta u(x,y) - k(x,y)^2 u(x,y) = b(x,y), & \text{on } \Omega = (0,600) \times (-1000,0) \\ \qquad b(x,y) = \delta(x - 300, y) & x, y \in \Omega \end{cases} \tag{1.23}$$

where $k(x, y) = \frac{2\pi f}{c(x,y)}$, f is the frequency. The wave velocity $c(x, y)$ is shown in Figure 1.1. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

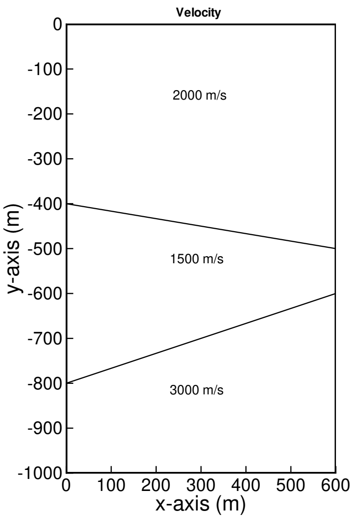


Figure 1.1: The velocity distribution of the wedge problem

MODEL PROBLEM 4 - MARMOUSI PROBLEM

For industrial applications, the fourth model problem is the so-called Marmousi problem [14], a well-known benchmark problem. The geometry of the problem stems from the section of the Kwanza Basin through North Kungra. It contains 158 horizontal layers in the depth direction, making it highly heterogeneous.

The problem is given by

$$\begin{cases} -\Delta u(x,y) - k(x,y)^2 u(x,y) = b(x,y), & \text{on } \Omega = (0,9200)\times(-3000,0) \\ b(x,y) = \delta(x-6000,y) & x,y\in\Omega \end{cases} \tag{1.24}$$

where $k(x,y) = \frac{2\pi f}{c(x,y)}$. The wave velocity $c(x,y)$ over the domain is shown in Figure 1.2. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

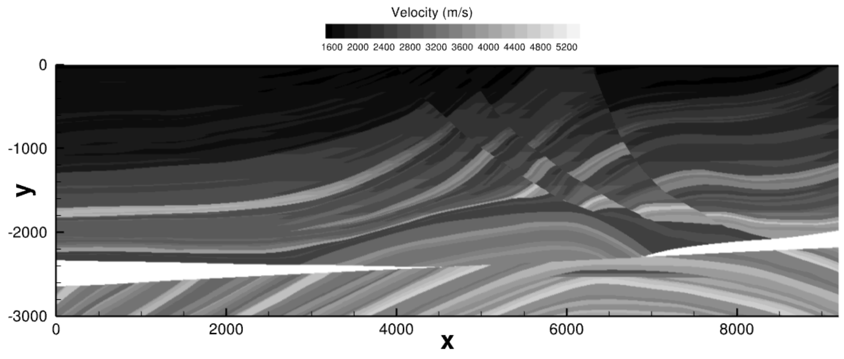


Figure 1.2: The velocity distribution of the Marmousi problem

1.2.2. THREE-DIMENSIONAL MODELS

3D CLOSED-OFF MODEL PROBLEM

This problem is a constant wavenumber model with a given solution to validate the solution methods for 3D problems. A cubic homogeneous domain $\Omega = [0,1]^3$ is considered. The source function is specified by

$$b\left(x,y,z\right) = \left(21\pi^2 - k^2\right)\sin(\pi x)\sin\left(2\pi y\right)\sin(4\pi z) - k^2,\ \ \Omega = [0,1]^3. \tag{1.25}$$

It is supplied with the following Dirichlet conditions

$$u(x,y,z) = 1,\ \text{on } \partial\Omega. \tag{1.26}$$

The analytical solution is given by

$$u(x,y,z) = \sin(\pi x)\sin\left(2\pi y\right)\sin(4\pi z) + 1. \tag{1.27}$$

3D Constant-Wavenumber Problem with Point Source

To establish a baseline for our numerical methods and to verify the accuracy of our solutions, we consider a 3D problem with constant wavenumber k in a rectangular homogeneous domain $\Omega = [0,1]^3$. A central point source is given by $\delta\left(x-0.5, y-0.5, z-0.5\right)$, where $\delta\left(x,y,z\right)$ is a Dirac delta function which satisfies

$$\int\int\int \delta(x)\cdot\delta\left(y\right)\cdot\delta(z)dxdydz = 1. \tag{1.28}$$

The wave propagates outward from the center of the domain. To simulate an unbounded domain, we apply first-order Sommerfeld radiation conditions on all boundaries, which allow outgoing waves to exit the computational domain with minimal reflections.

For this constant wavenumber problem, a fundamental solution exists [15], providing a valuable reference for assessing the accuracy of our numerical methods. The analytical solution is given by:

$$u\left(x,y,z\right) = \frac{e^{\mathrm{i}k\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}}}{4\pi\sqrt{(x-x_0)^2+\left(y-y_0\right)^2+(z-z_0)^2}}, \tag{1.29}$$

where $\left(x_0, y_0, z_0\right)$ is the location of the source.

With this simplified problem, we aim to establish the validity and efficiency of our methods, particularly our multilevel deflation approach.

3D Wedge Model Problem

This 3D model extends the 2D wedge problem to mimic three layers with different acoustic-wave velocities in three dimensions. As shown in Figure 1.3, 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 $\left(x,y,z\right) = (300,300,0)$.

The problem is given by

$$\begin{cases} -\Delta u(x,y,z) - k(x,y,z)^2 u(x,y,z) = b(x,y,z), & \text{on } \Omega \\ b(x,y,z) = \delta(x-300, y-300, z) & x,y,z \in \Omega \end{cases}, \tag{1.30}$$

The wave velocity $c(x,y,z)$ is shown in Figure 1.3. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

3D SEG/EAGE Salt Model Problem

The 3D SEG/EAGE salt model [16] is a velocity field model containing salt domes, which mimics the typical Gulf Coast salt structure. As shown in Figure 1.4, it is defined in a parallelepipedal physical domain of size 13520 m × 13520 m × 4200 m. The acoustic-wave velocity varies from 1500 m s^{-1} to 4482 m s^{-1}. The model is considered challenging due to the inclusion of complex geometries (salt domes) and a realistic large-size computational domain. A point source is located at $\left(x,y,z\right) = (3200,3200,0)$.

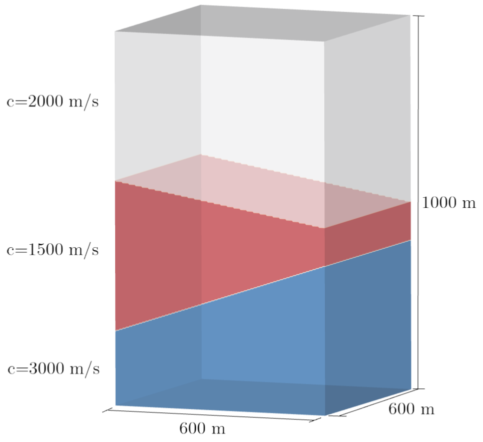


Figure 1.3: The velocity distribution of the 3D wedge problem.

The problem is given by

$$\begin{cases} -\Delta u(x,y,z) - k(x,y,z)^2 u(x,y,z) = b(x,y,z), & \text{on } \Omega \\ b(x,y,z) = \delta(x-3200, y-3200, z) & x,y,z \in \Omega \end{cases}, \tag{1.31}$$

The wave velocity $c(x,y,z)$ is shown in Figure 1.4. The first-order Sommerfeld boundary conditions are imposed on all boundaries.

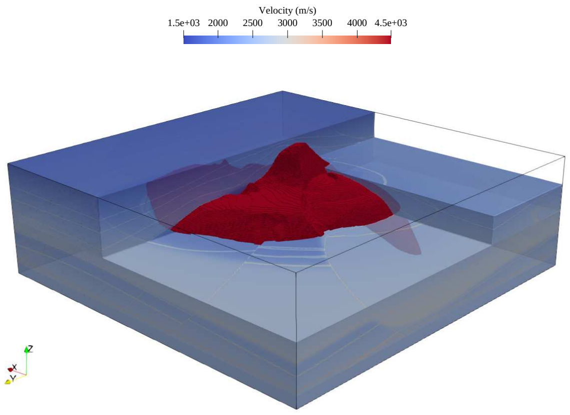


Figure 1.4: The velocity distribution of the 3D SEG/EAGE Salt Model.

GO_3D_OBS Model Problem

In a more realistic geophysical context, we consider the GO_3D_OBS model inspired by the geology of the Nankai Trough [17]. It is a high-resolution 3D subduction-zone geomodel developed to address the long-standing challenge of detailed reconstruction of deep crustal targets by seismic methods. It covers a continental margin at a regional scale, encompassing the wider range of wave speeds found in the crust and upper mantle. This model serves as a crucial benchmark for evaluating and minimizing the acquisition footprint in high-resolution imaging processes such as full waveform inversion, especially in the context of marine environments where controlled-source seismic surveys typically employ sparse arrays of ocean bottom seismometers (OBSs).

Figure 1.5 illustrates the acoustic velocity profile of the GO_3D_OBS model, which encompasses a wide range of velocities from a minimum of $1500\,\mathrm{m\,s^{-1}}$ to a maximum of $8600\,\mathrm{m\,s^{-1}}$. The point source is positioned at (10000 m, 10000 m, 900 m).

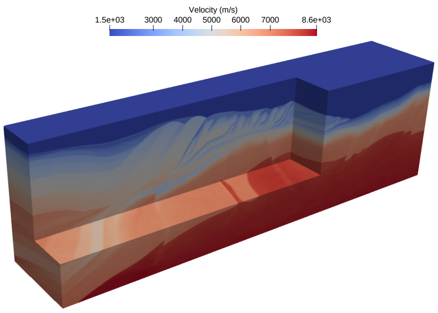


Figure 1.5: The target of the regional GO_3D_OBS model

1.3. Numerical Methods

The numerical solution of the Helmholtz equation demands a sophisticated translation from continuous partial differential representation to a discrete computational framework – a process fundamentally grounded in numerical discretization. Multiple discretization strategies, including finite difference, finite element, and spectral methods, can be used according to specific problem characteristics [18]. Having simple geometries in geophysical seismic imaging applications, this work mainly employs a second-order finite difference method.

1.3.1. Finite Difference Method

Employing the finite difference method, the computational domain is discretized utilizing vertex-centered structural grids. A geometric visualization of these grid configurations is depicted in Figure 1.6.

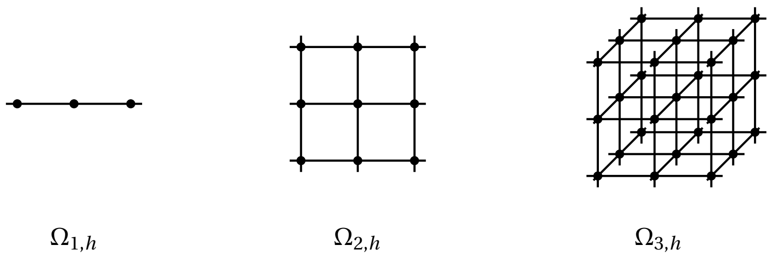


Figure 1.6: Uniform finite-difference grid discretizations for 1D, 2D, and 3D domains.

ONE-DIMENSION

To illustrate the concept of numerical discretization, we begin with the following one-dimensional (1D) example.

$$-\frac{d^2u(x)}{dx^2} - k^2u(x) = b(x), \quad x \in \Omega = [0, L] \tag{1.32}$$

We consider the continuous problem defined on a finite domain $\Omega_{1,h} = [0, L]$, which is discretized using a uniform grid comprising n equally spaced nodes. Given the interval, we obtain the numerical domain with a step size $h = \frac{L}{n-1}$:

$$\Omega_{1,h} = \{x_i | x_i = (i-1)h, h = \frac{L}{n-1}, 1 \le i \le n, n \in \mathbb{N} \setminus \{0\}\}. \tag{1.33}$$

Spatial grid vectors are introduced to approximate the source function $b(x)$ and the wave function $u(x)$ on $\Omega_{1,h}$.

$$u(x) \approx u(x_i) = u_{h,i}, \tag{1.34}$$
$$b(x) \approx b(x_i) = b_{h,i}, \tag{1.35}$$
$$x \in \Omega_{1,h}. \tag{1.36}$$

By approximating the continuous second-order derivatives with central finite difference approximations, we have:

$$-\frac{d^2u(x)}{dx^2} \approx \frac{-u_{h,i-1} + 2u_{h,i} - u_{h,i+1}}{h^2}, \quad 2 \le i \le n-1. \tag{1.37}$$

which can achieve second-order accuracy $O(h^2)$ for smooth solutions on uniform grids. It can be denoted by the so-called *stencil* notion

$$\frac{1}{h^2}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}. \tag{1.38}$$

For the 1D Helmholtz equation, we have

$$\frac{-u_{h,i-1} + 2u_{h,i} - u_{h,i+1}}{h^2} - k^2u_{h,i} = b_{h,i}, \quad 2 \le i \le n-1. \tag{1.39}$$

Thus, the stencil of the discrete Helmholtz operator is

$$\frac{1}{h^2}\begin{bmatrix} -1 & 2-k^2h^2 & -1 \end{bmatrix}. \tag{1.40}$$

Discretization of the source function For the discretization of a common source function given by the Dirac function $\delta(x - x_s)$, according to its definition (1.3), we can set the right-hand side (RHS) as

$$b_{h,i} = \begin{cases} \frac{1}{h}, & x_i = x_s \\ 0, & x_i \neq x_s \end{cases} . \tag{1.41}$$

Discretization of the Boundary Conditions Discretization of the Dirichlet boundary conditions (1.5) is straightforward:

$$u_{h,1} = f(0), \quad u_{h,n} = f(L) \tag{1.42}$$

For discretization of Sommerfeld boundary conditions, a ghost point located outside the boundary points can be introduced. For the 1D case of equation (1.8), for instance, suppose $u_{h,0}$ is a ghost point on the left of $u_{h,1}$, the normal derivative can be approximated by

$$\frac{\partial u}{\partial \vec{n}} - iku \approx \frac{u_{h,0} - u_{h,2}}{2h} - iku_{h,1} = 0 \tag{1.43}$$

The ghost point can be represented by observing that

$$u_{h,0} = u_{h,2} + 2hiku_{h,1}. \tag{1.44}$$

For consistency, we can also include a ghost point for the Dirichlet boundary condition. For instance, the ghost point $u_{h,0}$ can be introduced by the following relationship

$$u_{h,1} = \frac{u_{h,0} + u_{h,2}}{2} \tag{1.45}$$

which can be rewritten as

$$u_{h,0} = 2u_{h,1} - u_{h,2} = 2f(0) - u_{h,2}. \tag{1.46}$$

As a result, the indices in the discretization scheme (1.39) can range from $j = 1$ to $j = n$ whenever the boundary nodes are included.

We can assemble the unknown grid values $u_{h,i}$ and $b_{h,i}$ into column vectors of dimension n. Consequently, we construct the linear system of equations as follows:

$$[A_h u_h]_{1 \le i \le n} = \frac{1}{h^2} \begin{bmatrix} -1 & 2 - k^2 h^2 & -1 \end{bmatrix} [u_h]_{1 \le i \le n} = [b_h]_{1 \le i \le n}. \tag{1.47}$$

Thus, we have transformed the continuous ordinary differential Helmholtz equation into a linear system of equations. Solving the Helmholtz boundary value problem now reduces to solving the system:

$$A_h u_h = b_h, \quad A_h \in \mathbb{R}^{n \times n}, \quad u_h, b_h \in \mathbb{R}^n. \tag{1.48}$$

TWO-DIMENSIONAL CASE

The discretization for a two-dimensional (2D) problem can be naturally extended. In the 2D case, the finite domain becomes a square $\Omega_{2,h} = [0, L] \times [0, L]$. By employing a ghost point extrapolation technique derived from the boundary conditions and implementing a second-order finite difference discretization approach, we have

$$\frac{-u_{h,(i,j-1)} - u_{h,(i-1,j)} + 4u_{h,(i,j)} - u_{h,(i+1,j)} - u_{h,(i,j+1)}}{h^2} - k^2 u_{h,(i,j)} = b_{h,(i,j)}, \quad 1 \le i, j \le n. \tag{1.49}$$

in stencil notation

$$[A_h u_h] = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 - k^2 h^2 & -1 \\ 0 & -1 & 0 \end{bmatrix} [u_h]_{1 \le i,j \le n} = [b_h]\ _{1 \le i,j \le n}. \tag{1.50}$$

In finite difference methods, stencil notation provides a compact representation of the discretized differential operator at a specific grid point. The stencil depicts the geometric pattern of neighboring points and their associated coefficients used in the numerical approximation.

When assembling the global linear system, each row of the matrix corresponds to the application of this stencil at a specific grid point. For interior points, the stencil pattern directly maps to the matrix entries: the diagonal element contains the center value $(4 - k^2 h^2)/h^2$, while off-diagonal elements contain the $-1/h^2$ contributions from neighboring points.

By implementing an x-line lexicographic ordering of the grid nodes, the unknown grid values $u_{h,(i,j)}$ and $b_{h,(i,j)}$ can be assembled into column vectors of dimension n^2, yielding a linear system

$$A_h u_h = b_h, \quad A_h \in \mathbb{R}^{n^2 \times n^2}, \quad u_h, b_h \in \mathbb{R}^{n^2}. \tag{1.51}$$

THREE-DIMENSIONAL CASE

For three-dimensional cases, the discrete Helmholtz operator has a $3 \times 3 \times 3$ stencil

$$[A_h] = \frac{1}{h^2} \left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{z-h} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 - k^2 h^2 & -1 \\ 0 & -1 & 0 \end{bmatrix}_{z} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{z+h} \right]. \tag{1.52}$$

For the 3D case in (1.52), the stencil extends to three dimensions. The subscripts $z-h$, z, and $z+h$ denote three 3×3 planes corresponding to the z-coordinate below, at, and above the current point, respectively. The center plane contains the central coefficient $(6 - k^2 h^2)/h^2$ and four $-1/h^2$ coefficients for neighbors in the x-y plane, while the planes above and below each contribute one $-1/h^2$ coefficient in the z-direction. This yields a seven-point stencil that, when assembled into matrix form, creates a sparse matrix with seven non-zero entries per row for interior points.

Grid Resolution

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

$$kh = \frac{2\pi h}{\lambda} = \frac{2\pi}{N_{pw}}. \tag{1.53}$$

For example, if at least 10 grid points per wavelength are required, one has to satisfy the condition $kh \le 0.628$.

1.3.2. Linear System Properties

The coefficient matrix A resulting from the discretization of the Helmholtz equation has several characteristic properties. The matrix's nature, whether real or complex, depends on the boundary conditions used, specifically Dirichlet or Sommerfeld conditions. With homogeneous Dirichlet boundary conditions, the matrix is normal and self-adjoint. When Sommerfeld boundary conditions are imposed, matrix A becomes complex symmetric but non-normal and non-Hermitian.

In essence, the coefficient matrix A represents a discretized Laplace operator augmented by a wavenumber-dependent term involving k^2. Closed-form eigenvalue expressions exist for matrices with homogeneous Dirichlet boundary conditions, but not for those with Sommerfeld conditions. For a 1D problem, we can express the continuous eigenvalues λ_i and the discrete eigenvalues $\hat{\lambda}_i$ as follows

$$\lambda_i = i^2\pi^2 - k^2, \quad i = 1,2,3,\dots \tag{1.54}$$

$$\hat{\lambda}_i = \frac{1}{h^2}(2-2\cos(i\pi h)) - k^2, \quad i = 1,2,\dots,n-2. \tag{1.55}$$

It should be noted that the corresponding matrix A is constructed by eliminating the boundary grid points. The corresponding eigenvectors are given by

$$\hat{v}_{i,h} = \begin{pmatrix} \sin(\pi i h) \\ \sin(2\pi i h) \\ \vdots \\ \sin((n-2)\pi i h) \end{pmatrix}, \quad i = 1,2,\dots,n-2. \tag{1.56}$$

For a 2D problem with homogeneous Dirichlet boundary conditions and elimination of the boundary grid points, the eigenvalues of the coefficient matrix are given by

$$\lambda_{i,j} = i^2\pi^2 + j^2\pi^2 - k^2, \quad i,j = 1,2,3,\dots \tag{1.57}$$

$$\hat{\lambda}_{i,j} = \frac{1}{h^2}\left(4-2\cos(i\pi h) - 2\cos\left(j\pi h\right)\right) - k^2, \quad i,j = 1,2,\dots,n-2. \tag{1.58}$$

One can observe that it is imperative to ensure that $i^2\pi^2 \neq k^2$ and $i^2\pi^2 + j^2\pi^2 \neq k^2$, as such conditions would indicate resonance and lead to unbounded oscillations in the absence of dissipation.

For sufficiently large k^2, the matrix becomes highly indefinite. This characteristic manifests in negative real components of the matrix's (complex) eigenvalues, consequently inducing a large condition number for matrix A. Thus, solving the Helmholtz problems numerically pose complex computational challenges due to their inherently indefinite and ill-conditioned mathematical nature.

1.3.3. COMPUTATIONAL CHALLENGES

Solving the Helmholtz equation, especially in high-frequency regimes, presents several computational challenges that can significantly impact the accuracy and efficiency of numerical methods. These challenges include pollution effects, the indefinite nature of the resulting linear systems, and the substantial resource requirements associated with large-scale three-dimensional problems.

One of the primary obstacles is the *pollution effect*, which is a significant issue when discretizing the Helmholtz equation at higher frequencies. As the wavenumber k increases, numerical accuracy deteriorates, necessitating increasingly refined discretizations. This phenomenon is notably pronounced in both finite difference and finite element methods, where the relationship between the analytical wavenumber and its discrete counterpart becomes problematic, which is related to the so-called numerical dispersion errors. As k increases, solutions exhibit more oscillatory behaviors, leading to phase discrepancies between the numerical solution and the analytical solution. A critical measure of pollution error is given by the ratio of the mesh size h to the wavenumber k. Smaller values of h mitigate the pollution error; however, this also enlarges the corresponding linear system, making computations more intensive and challenging. Consequently, there are stringent conditions that must be satisfied for discretization, such as the condition $k^3h^2 \le 1$, which underscores the necessity for careful grid refinement to maintain solution accuracy [19, 20].

While the number of grid points needed to retain that accuracy grows along with the wavenumber, it grows slower than the order of accuracy of the schemes. In particular, if we let k denote the wavenumber, n the problem size in one-dimension and p the order of a finite difference or finite element scheme, then

$$n = Ck^{\left(\frac{p+1}{p}\right)} \tag{1.59}$$

where C is a constant that only depends on the accuracy achieved [21]. Various discretization methods have been developed to address the accuracy issue. New higher-order compact finite difference schemes [22–25] or dispersion-minimized discretization schemes [26–28] have been formulated. Further advancements emerged through wave-ray-based optimal stencils [29] or wavenumber correction [30]. Recent developments have evolved to include sophisticated techniques such as combined discretization on rotated grids [31], yielding efficient wavelength-adaptive 27-point compact stencils. Dwarka and Vuik [12] have provided deeper insights into the nature of dispersion error through eigenvalue analysis, leading to novel correction methods. These developments have significantly reduced the number of grid points required per wavelength while maintaining solution accuracy.

Another computational challenge lies in the properties of the linear systems derived from discretizing the Helmholtz equation. As the resulting linear systems are often indefinite, traditional iterative solvers, which generally assume positive definiteness, are not effective [32]. This requires the development of specialized solution techniques to handle the unique characteristics of these systems, complicating the solution process further.

Additionally, the scalability of numerical methods becomes a pressing concern when dealing with large-scale three-dimensional problems. The computational resource requirements for high-frequency simulations can be substantial to achieve the necessary accuracy. The increasing oscillatory nature of the solution as k rises mandates finer discretizations, which in turn lead to larger linear systems requiring significant computational power and memory [33]. This escalates challenges related to both execution time and resource allocation, essential considerations for practical computational implementations.

The solutions of the Helmholtz equation, particularly in seismic imaging and medical diagnostics, demand unprecedented computational capabilities. The seismic studies and applications routinely require solutions for problems involving millions of unknowns, while medical imaging applications need real-time solutions for wave propagation in heterogeneous media. These practical demands have exposed limitations in existing solution methodologies.

Despite significant advances in both algorithms and computing hardware, several fundamental challenges persist. The inherent difficulties in solving high-frequency wave problems, combined with the massive scale of realistic applications, create a compelling need for continued research and development of solution methods.

1.4. Linear System Solvers

The discretization of the Helmholtz equation typically results in large, sparse, and indefinite linear systems that pose significant computational challenges for numerical solution strategies. Direct numerical methods, such as lower-upper (LU) factorization, become increasingly impractical as problem dimensions and frequencies increase. For two-dimensional problems, direct solvers can still be feasible, but three-dimensional scenarios rapidly expose their computational limitations. In three-dimensional spatial domains, direct numerical solvers exhibit theoretical upper-bound complexities of $\mathcal{O}(N^7)$ for time and $\mathcal{O}(N^5)$ for memory requirements, where N represents the discretization points along each dimension of a cubic grid of size N^3. While [34] demonstrate reduced practical complexities of $\mathcal{O}(N^6)$ and $\mathcal{O}(N^4)$ for their implementation, these asymptotic scalings nonetheless quantify an growing computational burden with increasing dimensionality. Consequently, even state-of-the-art direct solvers remain impractical for high-resolution and high-frequency numerical simulations.

Traditional iterative methods, such as standard stationary methods like Jacobi or Gauss-Seidel, demonstrate poor convergence characteristics when applied to the indefinite linear systems arising from Helmholtz equation discretization. The spectral properties of these matrices, particularly their indefinite nature and close-to-zero eigenvalues, dramatically inhibit convergence. Consequently, basic iterative techniques often fail to provide reliable solutions or may even diverge completely [32].

These computational constraints necessitate more sophisticated solution strategies that can efficiently handle the complex spectral properties of Helmholtz discretizations. Krylov subspace methods emerge as a promising approach, offering a systematic framework for developing iterative solvers that can potentially overcome the limitations of traditional techniques.

In the subsequent sections of this chapter, we omit the subscript " $_h$ " from matrices and vectors for convenience unless it is involved in multigrid concepts.

1.4.1. KRYLOV SUBSPACE METHODS

For a large, sparse system matrix A, the Krylov subspace methods are generated on a collection of iterants in the subspace

$$K^k(A;\mathbf{r}_0) := \operatorname{span}\left\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\right\} \tag{1.60}$$

where K^k is a k-dimensional Krylov space with respect to matrix A and initial residual $\mathbf{r}_0$. The concept of Krylov subspace methods can be summarized as follows. Starting from an initial solution $\mathbf{u}_0$, the solution $\mathbf{u}$ is approximated at each iteration by $\mathbf{u}_k$, which satisfies

$$\mathbf{u}_k \in \mathbf{u}_0 + K^k(A,\mathbf{r}_0), \quad k > 1. \tag{1.61}$$

Suppose the Krylov subspace K^k is spanned by the basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$, where

$$V_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k] \in \mathbb{C}^{n\times k}. \tag{1.62}$$

With residual $\mathbf{r}_k = b - A\mathbf{u}_k$, an expression for the residual at the k-th step is given by

$$\mathbf{r}_k = \mathbf{r}_0 - AV_k\mathbf{y}_k \tag{1.63}$$

where $\mathbf{y}_k \in \mathbb{C}^k$ and $\mathbf{u}_k = \mathbf{u}_0 + V_k\mathbf{y}_k$. In Krylov subspace methods, the key steps involve creating an orthonormal basis for V_k and generating the vector $\mathbf{y}_k$. This can be accomplished using either Arnoldi or Lanczos methods for basis construction, and employing a residual projection or norm minimization method for vector construction.

Some representative Krylov methods, such as the Conjugate Gradient method (CG) [35], Conjugate Gradient Normal Residual method (CGNR) [36], Minimal Residual method (MINRES) [37], Biconjugate Gradient method (BICG) [38], Biconjugate Gradient Stabilized method (Bi-CGSTAB) [39], Generalized Conjugate Residual method (GCR) [40], Generalized Minimal RESidual method (GMRES) [41], GMRES Recursive method (GMRESR) [42], Induced Dimension Reduction method (IDR(s)) [43], and others, have been developed so far. Among these, the CG method is a basic one, which minimizes the error $\|\mathbf{u} - \mathbf{u_k}\|$ in the A-norm and uses the Lanczos method. This method only needs three vectors in memory during iterations. However, this algorithm is chiefly designed for the symmetric and positive definite system matrix. In contrast, GMRES, Bi-CGSTAB, and IDR(s) can be used for non-singular problems that are indefinite and non-symmetric as well, which make them suitable choices for the Helmholtz equation.

GMRES

The GMRES method is an iterative method for nonsymmetric matrices, which minimizes the residual norm over the Krylov subspace. In this method, Arnoldi's method is used for computing an orthonormal basis of the Krylov subspace. The GMRES algorithm is shown in Algorithm 1.

Algorithm 1 GMRES for system $A\mathbf{u} = \mathbf{b}$.

```
1:  $\mathbf{u}_0$ is an initial guess
2:  Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$, $\beta = \|\mathbf{r}_0\|_2$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
3:  for $j = 1,2,\dots,k$ or until convergence do
4:      $\mathbf{w} = A\mathbf{v}_j$
5:      for $i := 1,2,\dots,j$ do
6:          $h_{i,j} = (\mathbf{w},\mathbf{v}_i)$
7:          $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$
8:      end for
9:      $h_{j+1,j} := \|\mathbf{w}\|_2$
10:     $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
11: end for
12: Store $V_k = [\mathbf{v}_1,\dots,\mathbf{v}_k]$; $\bar{H}_k = \{h_{i,j}\}$, $1 \le i \le j+1$, $1 \le j \le k$
13: Compute minimizer $\mathbf{y}_k$ over $\|\beta\mathbf{e}_1 - \bar{H}_k\mathbf{y}\|_2$, $\mathbf{e}_1 = (1,0,\cdots,0)^\mathsf{T}$
14: Update approximated solution $\mathbf{u}_k = \mathbf{u}_0 + V_k\mathbf{y}_k$
```

The GMRES algorithm can terminate early if the residual vector becomes zero at iteration step j, which indicates that an exact solution has been found. A theorem provides insight into the algorithm's convergence.

Theorem 1.4.1 (Convergence of GMRES)**.** *Let P_k be the space of all polynomials of degree less than k and A has the spectrum $\sigma = \{\lambda_1,\dots,\lambda_N\}$. Suppose that A is diagonalizable so that $A = X\Lambda X^{-1}$ and let*

$$\varepsilon^k = \min_{p\in P_k, p(0)=1} \max_{\lambda_i\in\sigma} |p(\lambda_i)| \tag{1.64}$$

Then the residual norm of the k-th iterate satisfies

$$\|r_k\|_2 \le \varepsilon^k K(X)\|r_0\|_2. \tag{1.65}$$

If all eigenvalues are enclosed by a circle centered at $C > 0$, $C \in \mathbb{R}$ and having radius $R < C$ (so the circle does not enclose the origin), then

$$\varepsilon^k \le \left(\frac{R}{C}\right)^k. \tag{1.66}$$

Pursuing certain optimality properties with the long recurrences, the GMRES algorithm becomes computationally impractical for a large number of iterations. To mitigate memory and computational constraints, we can limit iterations to m, form an approximate solution, and use this as the starting vector for subsequent GMRES applications. This restarted GMRES is denoted by the GMRES(m) procedure.

BI-CGSTAB

For non-symmetric matrices, the Bi-CGSTAB algorithm provides an iterative method based on constructing two biorthogonal bases for Krylov subspaces $K(A, \mathbf{r}_0)$ and $L(A^*, \tilde{\mathbf{r}}_0)$, where A^* is the conjugate transpose of A and $\tilde{\mathbf{r}}_0$ is the initial residual of the associated dual system. The algorithm includes a technique where the dual system vectors $\tilde{\mathbf{r}}_k$ can be constructed using specific polynomial relations, eliminating the need for direct multiplication by A^*. The Bi-CGSTAB method addresses convergence issues in earlier algorithms like CGS by introducing a stabilizing polynomial $P_k(A)$.

The Bi-CGSTAB algorithm is shown in Algorithm 2. An advantage of the Bi-CGSTAB method is that it uses short recurrences. The algorithm suffers from semi-optimality, requiring more matrix-vector products and lacking global error minimization. Rounding errors may significantly impact performance, and small modifications can introduce numerical instabilities. Thus, it is always necessary to compare the norm of the updated residual to the exact residual $\|\mathbf{b} - A\mathbf{u}_k\|$.

Algorithm 2 Bi-CGSTAB Algorithm for system $A\mathbf{u} = \mathbf{b}$

```
1: $\mathbf{u}_0$ is an initial guess, $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$
2: $\hat{\mathbf{r}}_0$ be an arbitrary vector such that $(\hat{\mathbf{r}}_0,\ \mathbf{r}_0) \neq 0$
3: $\rho_0 = \alpha = \omega_0 = 1$
4: $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$
5: for $k = 1, 2, \ldots, \text{max\_iter}$ do
6:     $\rho_k = (\hat{\mathbf{r}}_0,\ \mathbf{r}_{k-1})$
7:     if $\rho_k = 0$ then
8:         break
9:     end if
10:    $\beta = (\rho_k / \rho_{k-1}) \cdot (\alpha / \omega_{k-1})$
11:    $\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$
12:    $\mathbf{v}_k = A\mathbf{p}_k$
13:    $\alpha = \rho_k / (\hat{\mathbf{r}}_0,\ \mathbf{v}_k)$
14:    $\mathbf{s} = \mathbf{r}_{k-1} - \alpha\mathbf{v}_k$
15:    $\mathbf{t} = A\mathbf{s}$
16:    $\omega_k = (\mathbf{t},\ \mathbf{s}) / (\mathbf{t},\ \mathbf{t})$
17:    $\mathbf{u}_k = \mathbf{u}_{k-1} + \alpha\mathbf{p}_k + \omega_k\mathbf{s}$
18:    if $\mathbf{u}_k$ is accurate enough then quit;
19:    $\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$
20: end for
21: return $\mathbf{u}_k$
```

IDR(S)

The IDR(s) method is an efficient alternative to Bi-CGSTAB for Helmholtz problems [44]. IDR(s) is a memory-efficient method to solve large non-symmetric systems of linear equations. In IDR(s), s pre-defined vectors are used to enhance the convergence. [43] showed that IDR(1) has similar computational complexity and memory requirements as Bi-CGSTAB. With higher values of s, IDR(s) shows performance close to GMRES but with more storage requirements compared to Bi-CGSTAB. For example, we need to store 17 vectors for IDR(4), while Bi-CGSTAB needs storing 7 vectors. Van Gijzen and Sonneveld [45] further imposed a bi-orthogonalization condition on the iteration vectors and obtained a more robust variant of the IDR(s) algorithm.

1.4.2. Preconditioned Krylov Methods

As we mentioned, iterative methods for solving large-scale linear systems, particularly the Helmholtz equation, often suffer from slow convergence and numerical instability. Preconditioning emerges as a critical technique to address these fundamental challenges, transforming the original linear system into an equivalent problem with more favorable computational properties. The primary objective of preconditioning is to modify the spectral characteristics of the coefficient matrix to accelerate the convergence of Krylov subspace methods while maintaining the original solution [36]. For non-normal problems solved using GMRES, while eigenvalues typically play the dominant role [46], both the eigenvectors of the preconditioned matrix and the initial residual also affect convergence rates [47].

The essential strategy of preconditioning involves introducing an invertible matrix M that approximates the original system matrix A in a way that makes the transformed system more amenable to iterative solution. Mathematically, this can be implemented through two primary approaches: left preconditioning:

$$M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}, \tag{1.67}$$

or right preconditioning:

$$AM^{-1}\mathbf{x} = b, \ \mathbf{x} = M\mathbf{u} \tag{1.68}$$

While theoretically equivalent, these approaches can exhibit subtle differences in practical implementation, particularly regarding residual computation and stopping criteria.

For Krylov subspace methods, preconditioning can be applied to several prominent iterative algorithms.

Left-Preconditioned GMRES

The straightforward application of GMRES to the linear system (1.67) yields the preconditioned version of GMRES in Algorithm 3.

The algorithm computes and tracks preconditioned residuals $M^{-1}(\mathbf{b} - A\mathbf{u}_j)$, making unpreconditioned residuals inaccessible without explicit calculation. This may complicate implementing stopping criteria based on actual, rather than preconditioned, residuals.

Right-Preconditioned GMRES

The right preconditioned GMRES algorithm is based on solving the system (1.68). The new variable can be handled without explicit invocation. After computing the initial residual $\mathbf{r}_0 = \mathbf{b} - AM^{-1}\mathbf{x}_0 = \mathbf{b} - A\mathbf{u}_0$, subsequent Krylov subspace vectors can

Algorithm 3 Left-Preconditioned GMRES for system $A\mathbf{u} = \mathbf{b}$

```
1: Choose initial guess $\mathbf{u}_0$
2: Compute $\mathbf{r}_0 = M^{-1}(\mathbf{b} - A\mathbf{u}_0)$; $\beta = \|\mathbf{r}_0\|$; $\mathbf{v}_1 = \mathbf{r}_0/\beta$
3: for $j = 1,2,\dots,k$ or until convergence do
4:     $\mathbf{w} = M^{-1}A\mathbf{v}_j$
5:     for $i = 1,2,\dots,j$ do
6:         $h_{i,j} = (\mathbf{w},\mathbf{v}_i)$
7:         $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$
8:     end for
9:     $h_{j+1,j} := \|\mathbf{w}\|$
10:    $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
11: end for
12: Store $V_k = [\mathbf{v}_1,\dots,\mathbf{v}_k]$; $\bar{H}_k = \{h_{i,j}\}$, $1 \le i \le j+1$, $1 \le j \le k$
13: Compute minimizer $\mathbf{y}_k$ of $\|\beta\mathbf{e}_1 - \bar{H}_k\mathbf{y}\|$; Update $\mathbf{u}_k = \mathbf{u}_0 + V_k\mathbf{y}_k$
```

be generated without referencing **x**-variables. Thus, unlike the left-preconditioned version, this right-preconditioned GMRES requires the preconditioning operation only at the end of the outer loop, as shown in Algorithm 4. The residual norm is now relative to the initial system, distinguishing this approach from left-preconditioned GMRES.

Algorithm 4 Right-Preconditioned GMRES for system $A\mathbf{u} = \mathbf{b}$

```
1: Choose initial guess $\mathbf{u}_0$
2: Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$; $\beta = \|\mathbf{r}_0\|$; $\mathbf{v}_1 = \mathbf{r}_0/\beta$
3: for $j = 1,2,\dots,k$ or until convergence do
4:     $\mathbf{w} = AM^{-1}\mathbf{v}_j$
5:     for $i = 1,2,\dots,j$ do
6:         $h_{i,j} = (\mathbf{w},\mathbf{v}_i)$
7:         $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$
8:     end for
9:     $h_{j+1,j} := \|\mathbf{w}\|$
10:    $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
11: end for
12: Store $V_k = [\mathbf{v}_1,\dots,\mathbf{v}_k]$; $\bar{H}_k = \{h_{i,j}\}$, $1 \le i \le j+1$, $1 \le j \le k$
13: Compute minimizer $\mathbf{y}_k$ of $\|\beta\mathbf{e}_1 - \bar{H}_k\mathbf{y}\|$; Update $\mathbf{u}_k = \mathbf{u}_0 + M^{-1}V_k\mathbf{y}_k$
```

FLEXIBLE GMRES

The preconditioner M^{-1} discussed above is assumed fixed. However, in many practical scenarios, the preconditioning operation may vary, potentially being the result of another iterative process. Flexible iterations accommodate these preconditioner variations, allowing the preconditioner to change from step to step.

For the GMRES algorithm, this means instead of using a constant M^{-1}, we now allow a changing preconditioner M_j^{-1}. The key modification is in computing preconditioned vectors:

$$\mathbf{z}_j = M_j^{-1}\mathbf{v}_j \tag{1.69}$$

The approximate solution is then computed as $\mathbf{x}_k = \mathbf{x}_0 + Z_k\mathbf{y}_k$, where $Z_k = [\mathbf{z}_1, \ldots, \mathbf{z}_k]$. These are the modifications transforming the right preconditioned GMRES algorithm into flexible GMRES (FGMRES), as described in Algorithm 5.

Algorithm 5 Flexible GMRES

```
1: Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$, $\beta = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
2: for $j = 1, \ldots, k$ do
3:     Compute $\mathbf{z}_j := M_j^{-1}\mathbf{v}_j$
4:     Compute $\mathbf{w} := Az_j$
5:     for $i = 1, \ldots, j$ do
6:         $h_{i,j} := (\mathbf{w}, \mathbf{v}_i)$
7:         $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$
8:     end for
9:     Compute $h_{j+1,j} = \|\mathbf{w}\|_2$ and $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
10: end for
11: Define $Z_k := [z_1, \ldots, z_k]$, $\bar{H}_k = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le m}$
12: Compute minimizer $\mathbf{y}_k$ of $\|\beta\mathbf{e}_1 - \bar{H}_k\mathbf{y}\|$; Update $\mathbf{u}_k = \mathbf{u}_0 + Z_k\mathbf{y}_k$
```

The flexible variant requires additional memory to store the set of vectors $\{z_j\}_{j=1,\cdots,k}$, effectively doubling the storage needed compared to GMRES.

The effectiveness of preconditioned Krylov methods critically depends on the design of the preconditioning matrix M. Ideal preconditioners should satisfy two primary requirements: (i) the linear system $Mx = b$ should be efficiently solvable, typically with computational complexity significantly lower than the original system, and (ii) the preconditioner should effectively cluster the eigenvalues of the transformed matrix to accelerate convergence. This delicate balance requires sophisticated mathematical analysis and problem-specific insights.

While the specific construction of preconditioners is explored in subsequent sections, it is crucial to understand that preconditioning represents a transformative approach to iterative linear system solution. By strategically manipulating the spectral properties of the coefficient matrix, preconditioned Krylov methods can overcome many of the convergence limitations inherent in standard iterative techniques, making them particularly powerful for challenging problems like the discretized Helmholtz equation.

1.4.3. Multigrid Methods

Multigrid methods originate from the idea of achieving better convergence by solving the given problem with a hierarchy of discretizations and using relaxation techniques. The principle of divide and conquer is the foundation of multigrid techniques. The iteration error is decomposed into components in the spatial frequency domain and each component is sequentially treated on its most appropriate scale of discretization.

A multigrid method involves several components that need a careful design to achieve its excellent convergence. To demonstrate the multigrid method's framework, we use 2D scenarios as an example.

The first ingredient of multigrid methods is the smoothing property of basic iterative methods. For example, the Gauss-Seidel and SOR(ω) methods can serve as efficient smoothers for elliptic problems. 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 the coarse-grid operator also needs careful attention.

SMOOTHERS

When analyzing and processing the iteration error, we can divide the iteration error into high and low frequency modes. By applying several steps of basic iterative methods (BIMs), the high frequency components will decay rapidly while the low frequency component does not. Classical iteration methods such as the Gauss-Seidel or damped Jacobi iterations can be used as smoothers.

RESTRICTION

The inter-grid operations will be presented in a simple case of two meshes, i.e. fine and coarse grids. Two sets of uniform mesh with size h and $H = 2h$ are used to discretize a square computational domain Ω_2.

The restriction operator is usually a rectangular matrix that can carry out the transfer of grid vectors from the fine mesh to the coarse one. Let us denoted it as I_h^H (One can read the subscripts from bottom to top)

$$I_h^H : u_h \in \mathbb{R}^{n_h} \to u_H = I_h^H u_h \in \mathbb{R}^{n_H} \tag{1.70}$$

The injection operator has a stencil given by

$$I_h^H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}_h^H \tag{1.71}$$

Alternatively, one can also take a weighted average of the fine grid nodes. The so-called full weighting restriction operator has a stencil given by

$$I_h^H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H \tag{1.72}$$

If a lexicographic ordering is used, the assembled restriction operator will be a a full-row-rank $n_H \times n_h$ matrix. Higher-dimensional stencils will be presented as needed.

Interpolation/Prolongation

The interpolation operator I_H^h transfers grid vectors from the coarse to the fine grid, i.e.,

$$I_H^h : u_H \in \mathbb{R}^{n_H} \rightarrow u_h = I_H^h u_H \in \mathbb{R}^{n_h} \tag{1.73}$$

In the bilinear interpolation operator stencil is given by

$$I_H^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_H^h \tag{1.74}$$

Using a lexicographic ordering of the grid nodes as before, the interpolation operator can be assembled into a rectangular $n_h \times n_H$ matrix with full column rank. The bilinear interpolation and the full weighting restriction are related by the following variational condition

$$I_H^h = 4\left(I_h^H\right)^{\mathsf{T}} \tag{1.75}$$

coarse-grid operator

The coarse grid matrix A_H can be built from the fine grid matrix A_h in two ways. The first way is to obtain A_H by rediscretizing on the coarse mesh in the same way that the matrix A_h is obtained on the fine mesh. It is known as discretization coarse-grid operator (DCG). The second way is Galerkin Coarsening. The matrix A_H in the Galerkin approach is constructed algebraically using the relation

$$A_H = I_h^H A_h I_H^h. \tag{1.76}$$

The Galerkin coarse-grid operator GCG is more general in its range of applicability, but it has an associated expense in terms of growing stencils (except in the case of finite-volume methods with the lowest order restriction and prolongation). Its sparsity may change for complex domains. For this reason, it often pays to check out if DCG can be satisfactorily applied in a given situation.

Two Grid cycles

Suppose the iteration matrix corresponding to the (damped Jacobi, Gauss-Seidel or other) smoother is S_h, and this matrix is applied v times, so that we have

$$\mathbf{e}^{k+v} = (S_h)^v\,\mathbf{e}^k, \text{where } S_h = I_{n_h\times n_h} - (M_h)^{-1} A_h \tag{1.77}$$

where $\mathbf{e}$ denotes the error vector. This iteration can be written equivalently for the iterant $\mathbf{u}^{k+v}$. Thus, we now have all ingredients to define an iterative two-grid method. Algorithm 6 is a classical two-grid method. The resulting algorithm described in Step 2 up to Step 6 in Algorithm 6 is called the coarse grid correction (CGC) iteration. It can be written as a stationary iterative scheme with iteration matrix B_{CGC} given by

$$\mathbf{e}^{k+1} = B_{CGC}\mathbf{e}^k, \text{where } B_{CGC} = I_{n_h\times n_h} - I_H^h (A_H)^{-1} I_h^H A_h \tag{1.78}$$

Then the iteration matrix of two grid method (TGM) is given by

$$\mathbf{e}^{k+1} = B_{TGM}\mathbf{e}^{k}, \text{where } B_{TGM}(\upsilon_1, \upsilon_2) = (S_h)^{\upsilon_2} B_{CGC} (S_h)^{\upsilon_1} \tag{1.79}$$

The coarse grid correction can be regarded as a special case of the two-grid operator without smoother, i.e. $\upsilon_1 = \upsilon_2 = 0$. Note that the preconditioner associated with this iteration is $I_H^h (A_H)^{-1} I_h^H$. If we use the Galerkin approach to construct the coarse grid matrix, the coarse grid correction operator will be a projector which is orthogonal with respect to the A_h-inner product.

Algorithm 6 A Two-Grid Cycle

```
1: procedure TWO-GRID CYCLE($\mathbf{u}_h^i, \mathbf{b}_h, A_h, M_h, I_h^H, I_H^h$)
2:     $\mathbf{u}_h^{1/3} = S_h^{\upsilon_1}\mathbf{u}_h^i + (M_h)^{-1}\mathbf{b}_h$      ▷ $\upsilon_1$ pre-smoothing sweeps
3:     $\mathbf{r}_h = \mathbf{b}_h - A_h\mathbf{u}_h^{1/3}$      ▷ Residual computation
4:     $\mathbf{r}_H = I_h^H\mathbf{r}_h$      ▷ Restriction of the residual to $G_H$
5:     $\mathbf{e}_H = (A_H)^{-1}\mathbf{r}_H$      ▷ Exact determination of the error on $G_H$
6:     $\mathbf{e}_h = I_H^h\mathbf{e}_H$      ▷ Prolongation of the error to $G_h$
7:     $\mathbf{u}_h^{2/3} = \mathbf{u}_h^{1/3} + \mathbf{e}_h$      ▷ Correction of the last solution iterate
8:     $\mathbf{u}_h^{i+1} = S_h^{\upsilon_2}\mathbf{u}_h^{2/3} + (M_h)^{-1}\mathbf{b}_h$      ▷ $\upsilon_2$ post-smoothing sweeps
9: end procedure
```

Since it may still be too expensive to solve the coarse-grid problem exactly, the two-grid methods are seldom use in practice.

V-CYCLES AND W-CYCLES

The idea to apply the two-grid idea to A_H recursively until the coarse-grid problem can be solved with insignificant computational costs gives rise to a genuine multigrid method such as W-cycle and V-cycle described in Algorithm 7.

For solving linear symmetric positive definite (SPD) systems obtained from the discretization of partial differential equations, multigrid methods are usually regarded as one of the most efficient algorithms due to the grid-size-independent convergence and a complexity of $\mathcal{O}(N \log N)$. However, for the Helmholtz equation, standard multigrid methods may not maintain a complexity of $\mathcal{O}(N \log N)$ and even fail to converge. This challenge arises from the oscillatory nature of the solution and the difficulty in smoothing high-frequency error components. The problem becomes particularly pronounced when dealing with wavenumbers k that are large relative to the mesh resolution.

To address these limitations, researchers have developed specialized multigrid-based preconditioning techniques tailored to the Helmholtz equation, such as the Complex Shifted Laplacian Preconditioner (CSLP).

Algorithm 7 V-cycle: $\mathbf{u}_h = V\text{-cycle}(A_h, \mathbf{u}_h^0, \mathbf{b}_h, h, H_0)$

```
1: procedure V-Cycle($A_h, \mathbf{u}_h^0, \mathbf{b}_h, h, H_0$)
2:     Pre-smoothing: $\mathbf{u}_h = \text{smooth}^{v_1}(A_h, \mathbf{u}_h^0, \mathbf{b}_h)$
3:     Residual computation: $\mathbf{r}_h = \mathbf{b}_h - A_h\mathbf{u}_h$
4:     Coarse: $H = 2h,\ \mathbf{r}_H = I_h^H\mathbf{r}_h$
5:     if $H = H_0$ then
6:         Solve: $\mathbf{e}_H = (A_H)^{-1}\mathbf{r}_H$
7:     else
8:         Recursion: $\mathbf{e}_H = V\text{-cycle}(A_H, \mathbf{u}_H^0, \mathbf{r}_H, H, H_0)$
9:     end if
10:    Prolongation: $\mathbf{e}_h = I_H^h\mathbf{e}_H$
11:    Correction: $\mathbf{u}_h := \mathbf{u}_h + \mathbf{e}_h$
12:    Post-smoothing: $\mathbf{u}_h = \text{smooth}^{v_2}(A_h, \mathbf{u}_h, \mathbf{b}_h)$
13:    return $\mathbf{u}_h$
14: end procedure
```

1.5. Preconditioning for the Helmholtz Problem

The inherent indefiniteness of the operator presents significant challenges in achieving rapid convergence for Krylov subspace solvers applied to discretized Helmholtz equations. Preconditioning has consequently emerged as a fundamental approach to mitigate these computational difficulties. Preconditioning for the Helmholtz problem is crucial because unpreconditioned Krylov-based iterative methods become computationally inefficient, requiring over 1,000 iterations for a 2D problem and more than 10,000 iterations for a similar 3D problem [48]. Various preconditioning techniques have been developed and refined specifically to address the computational challenges associated with the Helmholtz equation.

Initial approaches focused on incomplete factorization techniques, such as incomplete Cholesky (IC) and incomplete LU (ILU) factorizations, which exposed significant fill-in problems that compromised matrix sparsity, particularly at high wavenumbers [49]. Gander and Nataf [50] have proposed an alternative approach utilizing an analytical ILU factorization method. However, the Analytical Incomplete LU (AILU) preconditioner exhibits a significant limitation in its numerical performance, as its effectiveness is restricted to constant wavenumber problems, with a tendency to diverge for non-constant wavenumber scenarios.

A pivotal advancement emerged with the development of operator-focused preconditioners. Bayliss *et al.* [51] initially proposed using the discretized Laplace operator as a preconditioner by setting the wavenumber to zero. Laird and Giles [52] improved this method by introducing a positive real shift to the operator. These preconditioners work well for medium wavenumbers, but numerical results indicate a significant increase in iterations for large wavenumbers. In 2004, Erlangga *et al.* [53] proposed an incorporation of a complex shift to the Laplace preconditioner for the Helmholtz problem, namely the Complex Shifted Laplace Preconditioner (CSLP), which has been widely used for solving Helmholtz problems. The CSLP preconditioner is a notable focus of this dissertation, distinguished by its effectiveness and straightforward implementation.

1.5.1. COMPLEX SHIFTED LAPLACE PRECONDITIONER (CSLP)

Recall that the Helmholtz operator A can be expressed through the discrete Laplacian operator Δ and the $n \times n$ identity matrix I as

$$A = -\Delta - k^2 I, \quad A \in \mathbb{C}^{n\times n}. \tag{1.80}$$

The CSLP Preconditioner is defined by

$$M = -\Delta - \left(\beta_1 + \mathrm{i}\beta_2\right) k^2 I, \quad M \in \mathbb{C}^{n\times n}, \ \beta_1, \beta_2 \in [0,1] \tag{1.81}$$

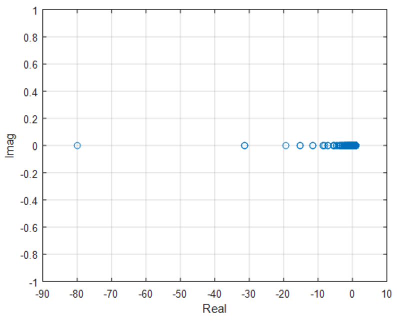
where i denotes the imaginary unit, and β_1 and β_2 are real numbers, which implies M is complex. The discretization of the CSLP operator typically employs identical boundary conditions as those specified in the original problem. The development of CSLP represents a significant advancement in preconditioning technique for the Helmholtz equation solvers. The CSLP-preconditioned Krylov solver shows linear iteration growth with increasing wavenumber.

Moreover, the emergence of the complex shift has facilitated a computationally feasible solution. The complex shift introduces damping and transforms the system into a configuration that is more conducive to approximate inversion through multigrid method [54]. Subsequently, ILU factorization [55] and algebraic multigrid techniques [56, 57] have been successfully utilized to invert the CSLP preconditioner. The polynomial fixed-point iteration method is also employed to resolve shifted systems, demonstrating robust convergence characteristics even when the magnitude of the shift is relatively minimal [58].

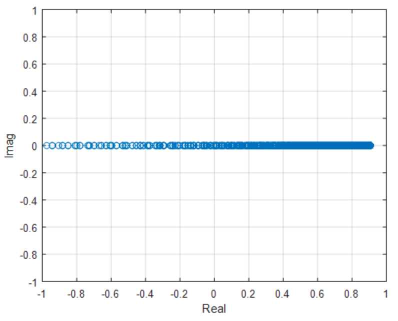
SPECTRAL PROPERTIES OF A PRECONDITIONED MATRIX

We will examine the spectral properties of the CSLP preconditioned matrix using a 2D Helmholtz problem with Dirichlet boundary conditions as an illustrative example.

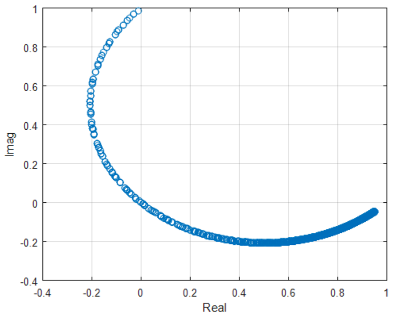
Figure 1.7 presents the spectrum of the preconditioned matrices $M^{-1}A$ for $(\beta_1,\beta_2) = (0,0)$ (Laplacian preconditioner), $(\beta_1,\beta_2) = (-1,0)$ (Laird preconditioner), $(\beta_1,\beta_2) = (0,1)$, $(\beta_1,\beta_2) = (1,1)$, $(\beta_1,\beta_2) = (1,0.5)$ and $(\beta_1,\beta_2) = (1,0.3)$. For the Laplacian preconditioner, as shown in Figure 1.7a, the eigenvalues are real, but there exist isolated values with large magnitude. For the Laird preconditioner, Figure 1.7b exhibits real eigenvalues bounded by -1 and 1. Once a complex shift is embedded, the spectrum of the preconditioned matrices becomes a curve in the complex plane. Whereas the real part of the preconditioner is zero (i.e. $(\beta_1,\beta_2) = (0,1)$), the eigenvalues have both positive and negative real parts. If both the real and imaginary parts of the preconditioner are non-zero, the real parts of the eigenvalues vary between 0 and 1. The difference between Figures 1.7d to 1.7f is that the smaller β_2 is, the fewer eigenvalues are near the origin.



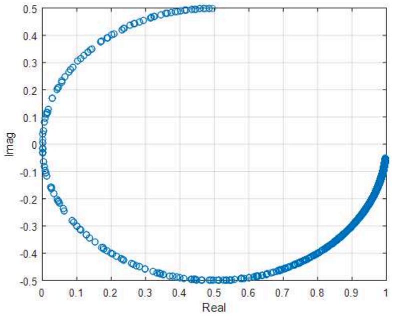
(a) $(\beta_1, \beta_2) = (0, 0)$



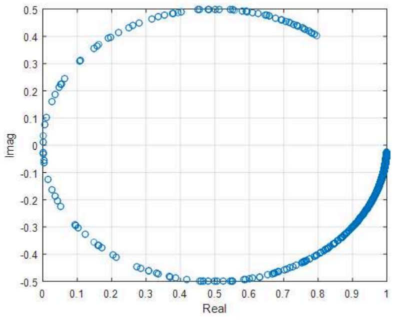
(b) $(\beta_1, \beta_2) = (-1, 0)$



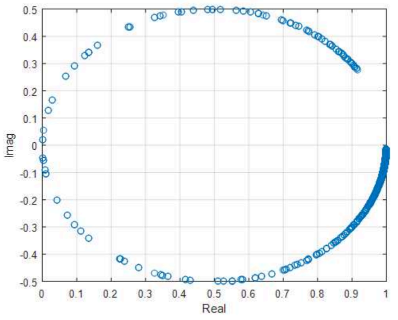
(c) $(\beta_1, \beta_2) = (0, 1)$



(d) $(\beta_1, \beta_2) = (1, 1)$



(e) $(\beta_1, \beta_2) = (1, 0.5)$



(f) $(\beta_1, \beta_2) = (1, 0.3)$

Figure 1.7: Two-dimensional spectrum of CSLP preconditioned matrix $M^{-1}A$ with $k = 40$, $(kh = 0.625)$ for different values of (β_1, β_2).

Optimal Shift

Several shift parameter values for β_1 and β_2 have been explored, constrained to the interval $[0,1]$. For $\beta_1 = 1$ and $\beta_2 \in (0,1]$, the eigenvalues of the preconditioned matrix cluster tightly in a circular pattern that improves the convergence of Krylov subspace methods [53].

van Gijzen *et al.* [59] have analyzed the optimal complex shift parameter for the shifted-Laplace preconditioner in combination with GMRES under the assumption of exact preconditioning operations, which offers valuable guidance for shift parameter selection for Krylov methods and approximate preconditioners. To achieve wavenumber-independent convergence, the change of the shift must remain $\mathcal{O}(k)$ and the preconditioner must be inverted exactly [60]; otherwise, the small eigenvalues of the preconditioned matrix will approach zero as the wavenumber increases.

Precisely inverting the preconditioner is often computationally expensive, so a single multigrid iteration is commonly used to obtain an approximate inverse. However, as the complex shift approaches zero, the multigrid method becomes inefficient, and the computational complexity of preconditioner approximation increases significantly. It has been demonstrated that employing multigrid techniques to approximate the preconditioner efficiently necessitates maintaining a relatively large complex shift, specifically $\mathcal{O}(k^2)$ [61]. Thus, choosing an appropriate complex shift is crucial for computational efficiency.

As an illustration of the issue at hand, we consider a 2D model problem with wavenumber $k = 40$. The problem is resolved on a grid of 10 grid points per wavelength ($kh = 0.625$). The linear system is preconditioned by the CLSP, where $\beta_1 = 1$ and β_2 varies from 0 to 1. The preconditioners are approximated by using a V-cycle. For the V-cycle multigrid method, 1 step Damped Jacobi method is used as both pre- and post- smoother. The full weight restriction and the bilinear interpolation described above are adopted. As for the coarse-grid operator, we use Galerkin coarsening. Figure 1.8 shows the number of GMRES iterations to reach a tolerance of 10^{-7} with respect to the preconditioned residual. In essence, the selection of shifts exhibits sensitivity, and multigrid approximation demonstrates improved efficiency with increased imaginary shift values. However, this leads to slower global convergence due to the preconditioner becoming a less accurate approximation of the original matrix.

Need of Projection

Although the CSLP demonstrates significant acceleration, its effectiveness diminishes for the Helmholtz problem at higher wavenumbers, as the small eigenvalues in the spectrum of the preconditioned matrix tend toward zero rather than maintaining their desired clustering around the point $(1,0)$ in the complex plane.

Figure 1.9 demonstrates the underlying phenomenon. As the value of k increases, the smaller eigenvalues exhibit a progressive migration toward the origin. In the 2D case, when k reaches 80, the eigenvalue clustering near the origin becomes notably pronounced. This clustering effect intensifies substantially as k approaches larger values. Thus, the small eigenvalues require careful consideration. Deflation is a technique frequently employed to eliminate specific portions of the spectrum and

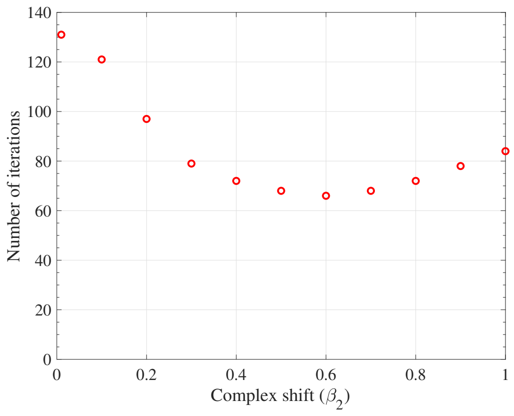
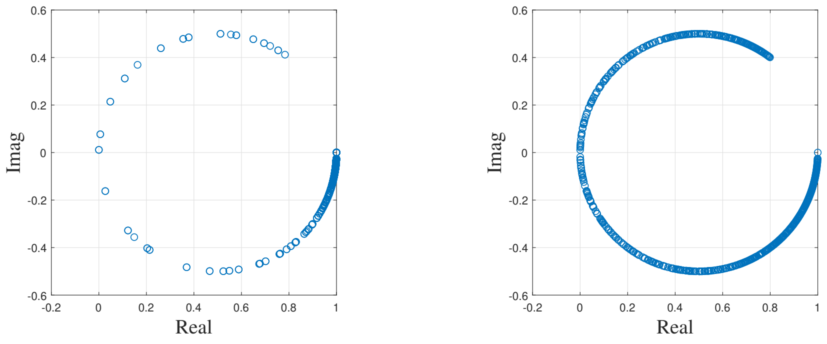


Figure 1.8: The number of GMRES iterations varies with the complex shift β_2 for $k = 40$. The preconditioners are approximately inverted by a multigrid V-cycle.



(a) $k = 20$ (b) $k = 80$

Figure 1.9: Two-dimensional spectrum of CSLP preconditioned matrix $M^{-1}_{(1,0.5)}A$ for different values of the wavenumber with $kh = 0.625$.

prevent undesirable eigenvalues from participating in the Krylov subspace iterations.

1.5.2. DEFLATION

The convergence rates of Krylov subspace methods for solving the Helmholtz equation are often significantly hindered by the presence of small or near-zero eigenvalues in the coefficient matrix. Deflation techniques have emerged as a powerful strategy to mitigate the adverse effects of these troublesome eigenvalues by altering the spectrum of the matrix in a favorable way.

Initially proposed independently by Nicolaides [62] and Dostál [63] as a means to accelerate the standard conjugate gradient method for SPD systems, the deflation method has been further investigated by Mansfield [64], Kolotilina [65], Vuik *et al.* [66] and Saad *et al.* [67]. In iterative systems, eigenvalue deflation can be achieved through two primary approaches: employing a projection preconditioner P to modify the spectrum [68], or enhancing the Krylov subspace by incorporating approximate eigenvectors associated with eigenvalues that impede convergence [69]. Given the computational expense of eigenvector construction, the projection preconditioner method is usually preferred for eliminating the effects of unwanted eigenvalues on the Krylov subspace [70].

The deflation method for addressing the Helmholtz problem was initially introduced by Erlangga and Nabben [71]. Subsequent research [72–74] led to the development of more computationally efficient variants of this approach. In a recent development, Dwarka and Vuik [75] introduced higher-order approximation schemes to construct deflation vectors. Due to the alignment of the near-zero eigenvalues of the fine-grid and coarse-grid operators, this advanced two-level deflation method exhibits convergence that is nearly independent of the wavenumber. The authors further extend the two-level deflation method to a multilevel deflation method [76]. By using higher-order deflation vectors, they show that up to the level where the coarse-grid linear systems remain indefinite, the near-zero eigenvalues of these coarse-grid operators remain aligned with the near-zero eigenvalues of the fine-grid operator. Combining this with the well-known CSLP preconditioner, they obtain a scalable solver for highly indefinite linear systems.

DEFLATION AS PRECONDITIONER

Consider the general linear system

$$\tilde{A}\mathbf{u} = \tilde{\mathbf{b}}, \quad \tilde{A} \in \mathbb{R}^{n\times n} \tag{1.82}$$

where $\tilde{A}$ is a symmetric coefficient matrix. In the context of preconditioning techniques for the Helmholtz problem, the operator $\tilde{A}$ may represent either the Helmholtz operator A_h or the coefficient matrix A_h preconditioned by CSLP, denoted as $M_h^{-1}A_h$. The projection preconditioner $P \in \mathbb{C}^{n\times n}$ can be defined as

$$P = I_{n\times n} - \tilde{A}Q, \quad \text{where } Q = ZE^{-1}Z^T, \; E = Z^T\tilde{A}Z \tag{1.83}$$

$$\tilde{A} \in \mathbb{C}^{n\times n}, \; Z \in \mathbb{R}^{n\times m}, \; m < n \tag{1.84}$$

where Z is introduced as a deflation matrix, which is supposed to be full-rank. Its m columns function as the deflation vectors. $E \in \mathbb{C}^{m\times m}$ is called the Galerkin or coarse matrix and $Q \in \mathbb{C}^{n\times n}$ is a correction matrix.

Choice of the Deflation Vectors

The choice of the deflation vectors will directly determine the character of the deflation preconditioner. Originally, a good choice of the deflation vectors is the eigenvectors of the coefficient matrix $\tilde{A}$. Suppose the spectrum of $\tilde{A}$ is given by

$$\Phi(\tilde{A}) = \{\lambda_i\}, \quad \text{where } 1 \leq i \leq n, \quad \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \tag{1.85}$$

and the corresponding eigenvector of λ_i is v_i. If we take the deflation matrix $Z = [v_1\ \mathbf{v_2}\ \ldots\ v_r]\,(r < n)$, then the eigenvalues of the deflated operator become

$$\Phi(P\tilde{A}) = \{0\ \ldots\ 0\ \lambda_{r+1}\ \ldots\ \lambda_n\} \tag{1.86}$$

In this way, the small eigenvalues are simply shifted to 0 by incorporating the deflation operator P. However, we usually do not know the eigenvectors of the matrix and it is expensive to compute them. Thus, we should consider approximating the eigenvectors corresponding to the unfavorable eigenvalues.

Another concern about the deflation vectors is the sparsity of the deflation matrix. A dense deflation matrix will lead to a dense coarse matrix. If we need to project a large number of eigenvalues, it is usually costly to store all the vectors and compute the inversion of a dense coarse matrix. Thus, it is preferable to choose sparse deflation vectors so as to obtain a sparse but full-rank deflation matrix.

The establishment of a comprehensive theoretical framework for optimal deflation vector determination remains an open challenge across diverse applications. The selection of appropriate deflation vectors is largely context-dependent, influenced by both problem-specific requirements and available data. The scientific literature presents several methodological approaches for deflation vector determination, encompassing approximated eigenvectors [66, 77, 78], solution and recycling techniques [79, 80], subdomain-based methods [81], and multigrid deflation vectors [71, 82]. Each approach offers distinct advantages for specific applications.

Originating from the observation that the multigrid inter-grid operators highlight the small frequencies and reserve them on the coarser level, multigrid inter-grid operators usually serve as deflation matrices in Helmholtz problem preconditioning. This approach allows the projection preconditioner to function similarly to coarse-grid correction in standard multigrid methods.

Variants for the Helmholtz Problem

Different variants of the deflation preconditioner can be derived using either the CSLP preconditioned Helmholtz operator or its unpreconditioned form. These variants are distinguished by how the operator E is constructed.

First, Erlangga and Nabben [71] proposed to deflate the CSLP-preconditioned Helmholtz operator, as the eigenvalues of $M_{h,(\beta_1,\beta_2)}^{-1} A_h$ begin to shift to the origin as the wavenumber increases. These small eigenvalues can be projected to zero by the preconditioner given by

$$\tilde{P}_h = I_h - \tilde{A}_h\tilde{Q}_h, \quad \tilde{Q}_h = I_{2h}^h \tilde{A}_{2h}^{-1} I_h^{2h}, \quad \tilde{A}_{2h} = I_h^{2h}\tilde{A}_h I_{2h}^h \tag{1.87}$$

where $\tilde{A}_h = M_{h,(\beta_1,\beta_2)}^{-1}A_h$. I_h^{2h} and I_{2h}^h are the restriction and interpolation operators defined in Equations (1.70) and (1.73), respectively. One can notice that the coarse operator $\tilde{A}_{2h}$ needs to be inverted. Since the idea of deflation is to project the small eigenvalues to zero, the exact inversion of $\tilde{A}_{2h}$ is necessary. However, this exact inversion becomes computationally impractical for large problems, especially given that the deflation preconditioner is sensitive to coarse-grid approximations. In the absence of an accurate approximation, the projection step can introduce numerous new eigenvalues close to zero, leading to adverse effects. To address this limitation, Erlangga and Nabben [83] extended the method to handle systems with nonsymmetric matrices. Instead of projecting the small eigenvalues to zero, they deflated the smallest eigenvalues to match the maximum eigenvalue. Erlangga and Nabben [83] demonstrated that the modified projection method was less sensitive to approximations of the coarse-grid system, allowing for the utilization of multilevel projected Krylov subspace iterations. Therefore, one can choose to deflate toward the largest eigenvalues of the preconditioned matrix by adding a term. The so-called Two-Level Krylov method (TLKM) preconditioner [74] reads as

$$\tilde{P}_{h,\gamma} = \tilde{P}_h + \gamma\tilde{Q}_h = \left(I_h - \tilde{A}_h\tilde{Q}_h\right) + \gamma\tilde{Q}_h \tag{1.88}$$

where γ is a constant, typically set to 1 as the CSLP preconditioner results in eigenvalues of the preconditioned matrix being bounded by 1 in modulus. The preconditioned linear system to be solved is

$$\tilde{P}_{h,\gamma}\tilde{A}_h u_h = \tilde{P}_{h,\gamma}\tilde{b}_h \tag{1.89}$$

where $\tilde{b}_h = M_{h,(\beta_1,\beta_2)}^{-1}b_h$. Compared to the original Helmholtz system, the TLKM preconditioner is equivalent to

$$P_{h,TLKM} = \tilde{P}_{h,\gamma}M_{h,(\beta_1,\beta_2)}^{-1} \tag{1.90}$$

One can observe that TLKM needs an application of $M_{h,(\beta_1,\beta_2)}^{-1}$ on the fine grid for every coarse grid iteration. This makes it too expensive to use.

Alternatively, one can deflate the Helmholtz operator, that is $\tilde{A}_h = A_h$ in Equation (1.87). The preconditioner becomes

$$P_h = I_h - A_h Q_h, \quad Q_h = I_{2h}^h A_{2h}^{-1} I_h^{2h}, \quad A_{2h} = I_h^{2h} A_h I_{2h}^h \tag{1.91}$$

$$P_{h,\gamma} = P_h + \gamma Q_h = (I_h - A_h Q_h) + \gamma Q_h \tag{1.92}$$

Combined with the standard preconditioner CSLP, a robust two-level preconditioned method, Adapted Deflation Variant 1 (A-DEF1) [74] reads as

$$P_{h,A-DEF1} = M_{h,(\beta_1,\beta_2)}^{-1}P_{h,\gamma} + Q_h \tag{1.93}$$

The preconditioned linear system to be solved becomes

$$P_{h,A-DEF1}A_h u_h = P_{h,A-DEF1}b_h. \tag{1.94}$$

Note that $P_{h,A-DEF1}A_h$ is nonsingular, so Equation (1.94) has a unique solution.

The groundbreaking research conducted by Dwarka and Vuik [75] demonstrates that misalignment between near-kernel eigenmodes of fine-grid and coarse-grid operators causes near-zero eigenvalues in deflation preconditioners. This misalignment stems from inadequate interpolation of grid functions at high wavenumbers. They have developed a novel method to quantify these effects through projection error analysis and proposed to employ a higher-order approximation scheme for deflation vector construction, instead of using the restriction and interpolation operators defined in Equations (1.70) and (1.73). The implementation of Equations (1.91) and (1.92) with a higher-order deflation vector will be referred to as Adapted Preconditioned Deflation (APD), and APD(ε) with the projection error minimizer ε. The interpolation and restriction operators for a one-dimensional grid function are as follows

$$\left[I_{2h}^{h}u^{2h}\right]_i = \begin{cases} \frac{1}{8}\left(u^{2h}_{(i-1)/2} + 6u^{2h}_{(i+1)/2} + u^{2h}_{(i+3)/2}\right) & \text{if } i \text{ is odd} \\ \frac{1}{2}\left(u^{2h}_{i/2} + u^{2h}_{(i+2)/2}\right) & \text{if } i \text{ is even} \end{cases} \tag{1.95}$$

for $i = 2, \cdots, n-1$ and

$$\left[I_{h}^{2h}u^{h}\right]_i = \frac{1}{8}\left(u^{h}_{2i-3} + 4u^{h}_{2i-2} + 6u^{h}_{2i-1} + 4u^{h}_{2i} + u^{h}_{2i+1}\right) \tag{1.96}$$

for $i = 2, \cdots, \frac{n-1}{2}$. Given it achieved wavenumber-independent convergence at large wavenumbers, APD will serve as the basis of this dissertation.

1.5.3. DOMAIN DECOMPOSITION METHODS (DDM)

A promising branch is the use of Domain Decomposition Methods (DDM) as preconditioning techniques [84].

The domain decomposition method was primarily developed to solve PDEs in complicated domains. The method evolved from the early Schwarz methods, which were initially conceived as an iterative algorithm for solving the Poisson equation defined over a union of two regular geometries. Subsequently, researchers developed a class of Schwarz methods that implemented more effective transmission conditions at subdomain interfaces compared to traditional Dirichlet conditions. A significant advancement in this field occurred when Lions [85] introduced Robin interface conditions as an alternative to the conventional Dirichlet interface conditions. These advanced approaches became known as Optimized Schwarz Methods (OSM). Several interface conditions such as absorbing boundary conditions and optimized interface conditions were derived [86].

The indefiniteness of the Helmholtz operator will become an obstacle when applying the Schwarz method to the Helmholtz problem. The domain decomposition method for solving the Helmholtz problem was first studied in [87]. There are two strategies to avoid the indefiniteness of the Helmholtz operator. One is domain decomposition with sufficiently small subdomains to make the minimal eigenvalues larger than k^2, such as [88, 89]. But this strategy will lead to too many subdomains.

Another strategy involves regularization of subdomain problems with optimized interface boundary conditions. The optimized Schwarz methods for the Helmholtz equation based on a good approximation of transmission boundary conditions has led to many variants [90–97], such as absorbing transmission conditions or perfectly matched layers for fast convergence and high-order transmission conditions. Several state-of-art algorithms, including the source transfer method [98, 99], the method based on single layer potentials [100], and the method of polarized traces [101, 102], can all be formulated in the context of the optimized Schwarz methods.

In addition to carefully designed transmission conditions, problem-specific coarse spaces constitute a fundamental component of the DDM strategy. Notable developments include the DtN and GenEO spectral coarse spaces [103], which utilize selected modes from local eigenvalue problems specifically tailored to the Helmholtz equation.

DDM has been widely used to develop efficient preconditioners and parallel solution methods for Helmholtz problems. While this method lies beyond the scope of the present dissertation, we refer the reader to [104–106] and references therein for comprehensive surveys.

1.6. PARALLEL COMPUTING

The numerical solution of the Helmholtz equation at high frequencies presents significant computational challenges that often exceed the capabilities of sequential computing. As the wavenumber increases, the required spatial resolution grows quadratically in two dimensions and cubically in three dimensions, leading to linear systems with billions of unknowns. Furthermore, the iterative solvers discussed in previous sections, although theoretically efficient, demand substantial computational resources, even with sophisticated preconditioners like the CSLP or deflation techniques. These computational demands necessitate the adoption of parallel computing strategies to achieve practical solution times for real-world applications. Parallel computing offers a promising solution by distributing both the computational workload and memory requirements across multiple processing units, enabling the solution of previously intractable problems. However, high-performance iterative methods must be designed to exploit parallelism effectively. For example, preconditioning techniques, such as the multigrid-based CSLP and deflation methods, must be adapted for parallel execution to maintain convergence rates without introducing prohibitive overhead. This section presents an overview of parallel computing concepts, architectures, and methodologies essential in developing and analyzing high-performance iterative methods for the Helmholtz equation.

1.6.1. TAXONOMY OF PARALLEL COMPUTING PARADIGMS

The foundation of parallel computing lies in different paradigms that determine how computational tasks are organized and executed. Flynn's taxonomy provides a systematic classification of these paradigms based on instruction and data streams [107].

Single Instruction Multiple Data (SIMD)

In the SIMD model, multiple processing units execute the same instruction simultaneously on different data elements. This model is effective for problems with a high degree of data parallelism, such as vector and matrix operations prevalent in numerical simulations. Modern processors, including Graphics Processing Units (GPUs), utilize SIMD architectures to accelerate computations by exploiting data-level parallelism.

Multiple Instruction Multiple Data (MIMD)

The MIMD model allows each processor to execute different instructions on different data independently. This flexibility accommodates a wide range of applications with varying computational patterns. MIMD architectures are the most common in general-purpose parallel computers, including multi-core CPUs and distributed computing systems.

Two additional categories exist: SISD (Single Instruction Single Data) and MISD (Multiple Instruction Single Data). SISD represents traditional non-parallel processing in single-processor computers, while MISD exists theoretically but has no practical applications.

1.6.2. Parallel Architectures

Parallel computing architectures define how processors are interconnected and how they access memory. The main architectures are shared memory, distributed memory, and hierarchical (hybrid) systems.

Shared memory architectures provide all processors direct access to a common memory space, facilitating rapid data sharing but potentially suffering from memory contention and cache coherency issues. The Uniform Memory Access (UMA) model, where all processors have equal access time to all memory locations, simplifies programming but may face scalability limitations.

Distributed memory architectures, in contrast, assign each processor its own local memory, requiring explicit communication through message passing. While this approach eliminates memory contention issues and offers better scalability, it introduces communication overhead and requires careful data distribution strategies.

Modern supercomputers often employ hierarchical (hybrid) architectures, combining aspects of both shared and distributed memory systems. These systems typically consist of multiple compute nodes connected via high-speed networks, with each node containing multiple cores sharing local memory. Programming such systems effectively often requires hybrid programming models that utilize both shared memory and message passing paradigms.

1.6.3. Parallelism

Parallel programming begins with identifying an algorithm's inherent parallelism. Different forms of parallelism induce different parallelization methods.

At the finest granularity, multiple functional units within modern processors enable instruction-level parallelism, executing multiple operations simultaneously when data dependencies permit. Pipelining further enhances throughput by overlapping different stages of instruction execution, crucial for maintaining processor efficiency.

Vector processing extends this concept to data-level parallelism, performing the same operation on multiple data elements simultaneously. Modern processors support this through SIMD instructions, while GPUs take this approach to extreme scales with thousands of parallel execution units. These capabilities are particularly relevant for the regular, structured computations common in finite difference methods for the Helmholtz equation.

Multiprocessing enables parallel execution through both threads and processes, allowing programs to utilize multiple cores or processors simultaneously. While distributed computing extends this parallelism across multiple machines, it requires careful management of communication and data distribution. In scientific computing, data parallelism is particularly important, as it allows multiple processors to independently process different portions of large datasets. This approach, commonly known as SPMD (Single Program Multiple Data), typically involves running identical code across processors with separate instruction pointers, distinguishing it from SIMD parallelism. SPMD represents the primary parallelization strategy in scientific computing on MIMD architectures.

1.6.4. PROGRAMMING MODELS AND SOFTWARE FRAMEWORKS

The implementation of parallel algorithms relies on various programming models and software frameworks, each designed for specific architectural paradigms.

MESSAGE PASSING INTERFACE (MPI)

MPI is a standardized, language-independent communication protocol used to program parallel computers [108]. It provides a rich set of functions for point-to-point and collective communication, enabling processes running on distributed memory systems to exchange data efficiently. MPI is widely used in scientific computing due to its portability and scalability.

OPEN MULTI-PROCESSING (OPENMP)

OpenMP is an application programming interface that supports shared memory multiprocessing programming in C, C++, and Fortran [109]. It uses compiler directives to parallelize code, allowing developers to introduce parallelism incrementally. OpenMP is particularly effective for loop-level parallelism and is easy to implement on shared memory systems.

Hybrid MPI+OpenMP programming combines both models to exploit multiple levels of parallelism. This approach has become increasingly important for modern hierarchical computing systems, allowing for efficient utilization of both inter-node and intra-node parallelism [110].

In addition, for hardware accelerators, particularly Graphics Processing Units (GPUs), CUDA and OpenCL provide programming models that exploit massive parallelism for suitable computational kernels. However, effective GPU implementation requires careful attention to memory access patterns and data transfer optimization.

1.6.5. Parallel Scalability and Performance

Understanding the scalability of parallel algorithms is crucial for evaluating their effectiveness and predicting performance at scale. The theoretical foundations of parallel scalability are captured by two fundamental laws. Amdahl's Law provides a theoretical upper bound on the speedup achievable through parallelization when dealing with a fixed problem size [111]. It emphasizes the limiting effect of sequential portions of an algorithm, expressing the maximum achievable speedup $S(N_p)$ with N_p processors as:

$$S(N_p) = \frac{1}{(1-\alpha)/N_p + \alpha} \tag{1.97}$$

where $\alpha \in [0,1]$ represents the fraction of the program that cannot be parallelized. This law highlights the importance of minimizing sequential bottlenecks in parallel programs.

Gustafson-Barsis' Law offers an alternative perspective by considering scenarios where the problem size scales with the available computational resources [112]. It suggests that by increasing the workload proportionally with the number of processors, linear speedup is achievable:

$$S(N_p) = N_p - \alpha(N_p - 1) \tag{1.98}$$

This law better reflects many practical applications where larger computational resources typically enable solving larger problems.

In practice, for initial performance evaluation and system comparison, the parallel performance can be characterized through two primary scaling metrics. *Strong scaling* measures how the solution time varies with the number of processors for a fixed total problem size. This metric is particularly relevant when rapid solution of a specific problem is required. *Weak scaling*, conversely, measures performance when the problem size increases proportionally with the number of processors, maintaining a constant workload per processor.

Parallel efficiency is defined as the ratio of speedup to the number of processors:

$$E(N_p) = \frac{S(N_p)}{N_p} \tag{1.99}$$

It provides a normalized measure of how effectively additional computational resources are utilized. High efficiency (close to 1) indicates good resource utilization, while lower values suggest diminishing returns from parallelization.

The *roofline model* provides a valuable analytical framework for understanding the performance limits of algorithms on modern computing architectures [113]. This model characterizes computational performance through the relationship between arithmetic intensity (I_a), which is defined as the number of operations per byte of memory traffic, and attainable performance, typically measured in FLOPS (floating point operations per second). Mathematically, the model expresses the attainable performance P as:

$$P = \min\left(\beta I_a, \pi\right) \tag{1.100}$$

where π represents the peak computational performance of the system, I_a denotes the arithmetic intensity of the algorithm, and β represents the system's memory bandwidth limit (byte per second). The resulting performance bound forms a distinctive roofline-shaped curve when plotted on a log-log scale. The ridge point, where the diagonal and horizontal lines intersect, marks the saturation of peak performance. This point represents the minimum arithmetic intensity ($I_a = \pi/\beta$) required to achieve peak performance. A kernel or application with arithmetic intensity I_a is considered memory-bound when $I_a < \pi/\beta$, limited by memory bandwidth. Conversely, when $I_a > \pi/\beta$, the computation is compute-bound, limited by peak computational performance.

1.6.6. CHALLENGES IN PARALLEL COMPUTING

While parallel computing offers significant performance benefits, it introduces challenges that must be addressed to achieve efficient computations.

Communication overhead represents a fundamental challenge, particularly in distributed memory systems. For iterative solvers applied to the Helmholtz equation, frequent boundary data exchange between subdomains can become a significant performance bottleneck. The communication-to-computation ratio often increases with the number of processors, potentially limiting scalability.

Load balancing presents another crucial challenge, especially for heterogeneous computing environments or problems with varying computational intensity across the domain.

Memory hierarchy management becomes increasingly complex in parallel systems. The presence of multiple cache levels, non-uniform memory access (NUMA) effects, and the need to maintain cache coherency can significantly impact performance. For large-scale problems, the memory access pattern can become as important as the computational complexity in determining overall performance.

Synchronization requirements and algorithmic dependencies can limit the achievable parallelism. In iterative methods for the Helmholtz equation, global operations such as inner products and norm calculations require synchronization across all processors, potentially creating bottlenecks.

Software complexity and maintainability present practical challenges in parallel implementation. The development of parallel numerical algorithms requires careful consideration of data structures, communication patterns, and synchronization mechanisms. The resulting code must remain maintainable and adaptable while achieving high performance across different parallel architectures.

Besides, factors such as memory bandwidth limitations and network latency can limit the scalability of parallel applications. One must consider these limitations to ensure that performance gains are realized as the number of processors increases.

These challenges underscore the importance of careful algorithm design and implementation in parallel scientific computing. Success in addressing these challenges often requires a combination of mathematical insight, algorithmic innovation, and detailed understanding of hardware characteristics.

Given these challenges, this dissertation contributes to developments of high-performance iterative solvers for solving the Helmholtz equation.

Parallel Solvers for the Helmholtz Equation

Efforts are underway to develop parallel scalable Helmholtz solvers. While domain decomposition methods offer a natural framework for parallel implementation [114, 115], various parallel programming paradigms have been explored, from MPI-based implementations for distributed memory systems to GPU-accelerated solutions. Riyanti *et al.* [116] presented a parallel multigrid-based CSLP preconditioner using MPI, demonstrating its effectiveness on heterogeneous problems with scalability up to 25 processors. Knibbe *et al.* [44] further introduced GPUs to accelerate Bi-CGSTAB as well as IDR(s) preconditioned by CSLP. Gordon and Gordon [117, 118] applied the block-parallel CARP-CG algorithm to the Helmholtz equation, achieving substantial reductions in residuals and showcasing applicability in both homogeneous and heterogeneous media. Calandra *et al.* [119, 120] proposed a geometric two-grid preconditioner for Helmholtz problems, which exhibits strong scaling in massively parallel setups. Ortega *et al.* [121] developed a parallel solution for the 3D Helmholtz equation, leveraging CUDA for computationally intensive operations and MPI for inter-processor communication. Recently, Bao and Feng [122] provided an MPI-based parallel algorithm for solving Helmholtz equations using sixth-order hybrid compact finite-difference schemes.

While the combination of advanced preconditioning strategies and parallel computing frameworks has led to significant advancements in computational efficiency and robustness, further research is necessary to investigate the integration of parallel CSLP and deflation techniques to further enhance the performance of Helmholtz solvers in practical applications.

1.7. Research Objectives and Contributions

The development of efficient solution methods for the Helmholtz equation represents a critical challenge at the intersection of computational mathematics and practical applications, which require higher frequencies, larger domains, and faster solutions. This dissertation addresses the growing gap between computational requirements of modern applications and the capabilities of current solution methods.

Our research focuses on developing high-performance iterative methods that can effectively utilize modern computing architectures while maintaining numerical efficiency. This work is driven by several critical requirements in contemporary scientific computing. The increasing complexity of industrial applications demands solution methods that can scale to higher frequencies and handle larger problems with sufficient computational efficiency. Furthermore, modern computing systems necessitate algorithms that can scale effectively across numerous processors while efficiently managing memory resources.

Current solution strategies often fall short in meeting these requirements simultaneously. While direct solvers provide robust solutions, their memory requirements become prohibitive for large-scale problems. The development of efficient iterative solvers, particularly those leveraging modern high-performance computing architectures, represents a critical research direction. Present advances in preconditioning techniques, including CSLP [53, 54] and deflation methods [72], particularly APD using high-order deflation vectors [75, 76], suggest promising pathways for improvement. However, their implementation, which explicitly store system matrices, face similar memory constraints despite their computational advantages. These limitations have motivated the exploration of matrix-free approaches, significantly reducing memory requirements while maintaining computational efficiency. It has been observed that the finite difference discretization allows all operators encountered in previous sections, including the Helmholtz operator, CSLP operator, inter-grid transfer operators, and coarse-grid operators, to be expressed through compact stencil notation. This stencil-based representation naturally facilitates matrix-free implementation, as all matrix-vector operations can be performed through direct stencil computations without storing explicit matrices. Moreover, the effective implementation of APD in parallel computing environments, especially for large-scale heterogeneous problems, remains an active area of research. The challenge is further compounded by the need to implement these methods in a matrix-free manner while maintaining their numerical properties and parallel efficiency.

Matrix-free methods for solving Helmholtz equations are important. The solution of Helmholtz problems requires exceptionally large linear systems due to wavenumber-dependent convergence and pollution error. Traditional matrix-based approaches face significant memory limitations when addressing industrial-scale problems, particularly for heterogeneous cases. Actually, as for heterogeneous Helmholtz problems, matrix-freeness is not trivial due to the variable diagonal elements. To our knowledge, many preconditioners, including the multigrid-based CSLP, are analyzed based on the Galerkin coarsening approach with explicit construction of the matrices. Moreover, the matrix-free implementation of the deflation methods presents additional complexity due to the requirements of Galerkin coarsening operations. For example, [123] required explicit matrix assembly and storage, restricting their implementation to grid sizes of 2501×751 using two computational cores. In contrast, our framework successfully processes larger problems (grid size 2945×961) on a single core. Similarly, the approach by [120], designed for massively parallel computing, also constructs explicit matrix and stores in CSR format. Furthermore, the high-order deflation method proposed by [75] achieves wavenumber-independent convergence through explicit matrix-based Galerkin coarsening.

This dissertation aims to develop robust, scalable parallel numerical methods for large-scale Helmholtz problems, focusing on three primary objectives: (i) implementing matrix-free parallel preconditioned Krylov methods and multigrid methods that eliminate the need for explicit matrix storage while maintaining computational efficiency, (ii) achieving wavenumber-independent convergence in parallel computing environments by leveraging the advanced preconditioning techniques including CSLP and APD through matrix-free operations, and (iii) developing a scalable parallel computing framework optimized for modern hierarchical architectures.

Our work makes several significant contributions to numerical methods and scientific computing. The first is a comprehensive matrix-free parallel implementation framework for the multigrid-based preconditioner, demonstrating exceptional efficiency in both two and three-dimensional settings. The second contribution introduces novel parallel higher-order deflation-based preconditioning techniques, featuring efficient matrix-free implementations and a Galerkin-based coarse-grid rediscretization scheme that achieves wavenumber-independent convergence.The third advancement is a scalable multilevel deflation preconditioning method, complemented by a hybrid MPI+OpenMP parallelization framework that addresses extreme-scale computational challenges.

1.8. Dissertation Outline

This dissertation develops and analyzes advanced numerical methods for the Helmholtz equation through seven chapters.

This chapter establishes the mathematical foundation, computational challenges, and current state-of-the-art in numerical methods and parallel computing approaches.

Chapter 2 presents our matrix-free parallel CSLP-preconditioned Krylov methods for two-dimensional problems, providing detailed performance analysis through achieved bandwidth, dot-product benchmarks and profiling. Chapter 3 extends this framework to three-dimensional heterogeneous problems, incorporating comprehensive parallel scalability analysis and performance optimization studies.

Chapter 4 introduces a novel parallel two-level deflation preconditioning method, presenting their matrix-free implementation and wavenumber-independent convergence. Chapter 5 advances this concept to multiple levels, exploring various configurations of multilevel deflation preconditioning for heterogeneous time-harmonic wave problems. Chapter 6 culminates in a robust parallel solver for extreme-scale three-dimensional Helmholtz problems, demonstrating its effectiveness on complex geological models.

Chapter 7 concludes by summarizing this dissertation and suggesting future research directions.

This progression demonstrates a comprehensive approach to developing and implementing high-performance iterative methods for the Helmholtz equation, emphasizing practical applicability and computational efficiency in large-scale parallel computing environments.

2

Matrix-free parallel CSLP-preconditioned iterative solvers in Two Dimensions

A matrix-free parallel iterative solver for the Helmholtz equation related to applications in seismic problems and its parallel performance is studied. We apply Krylov subspace methods, GMRES, Bi-CGSTAB and IDR(s), to solve the linear system obtained from a second-order finite difference discretization. The CSLP is employed to improve the convergence of Krylov solvers. The preconditioner is approximately inverted by multigrid iterations. For parallel computing, the global domain is partitioned blockwise. The standard MPI library is employed for data communication. 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 model problems show that the matrix-free parallel solution method has satisfactory parallel performance.

Parts of this chapter have been published in Scientific Computing in Electrical Engineering, Springer Nature Switzerland, 2024, pp. 61-68 [124].

This chapter focuses on the parallelization of Krylov methods, such as Generalized minimal residual method (GMRES), preconditioned by the multigrid-based Complex Shifted Laplace Preconditioner (CSLP) for Helmholtz equation. The CSLP preconditioner represents an early and widely adopted method where the number of iterations scales linearly within a certain range of the wavenumber. This parallel solution method is different from the earlier variants as given in [116, 123]. Kononov *et al.* [123] mainly parallelized the sequential program based on the data-parallel concept. Their idea was to decompose the matrix and the vector components. It results in a row-wise domain decomposition. In contrast, this work mainly starts with a block-wise domain decomposition and implements it in a matrix-free way. The former is a parallel computation based on data and blocks of the sequential program, while here it is parallelism based on domain decomposition. This idea is more flexible and allows us to implement large-scale parallel computing more flexibly. It is the basis for scalable parallel computing. Preliminary numerical experiments on model problems show that the matrix-free parallel solution method exhibits good parallel performance. Its weak scaling performance allows larger problems to be solved in similar time by increasing tasks proportionally, enabling higher resolution to minimize Helmholtz pollution errors.

The rest of this chapter is organized as follows. Section 2.1 describes the mathematical model that we will discuss. All numerical methods we use are described in Section 2.2, and Section 2.3 will illustrate the parallel implementation. The numerical performance is explored in Section 2.4, where we also evaluate our method extended to different iterative solvers. Section 2.5 contains the conclusion.

2.1. MATHEMATICAL MODEL

In this chapter, We will consider the 2D Helmholtz equation on a rectangular domain Ω with boundary $\Gamma = \partial\Omega$ supplied with Dirichlet boundary conditions (1.5) or first-order Sommerfeld boundary conditions (1.8).

In this chapter, we examine four 2D model problems of increasing complexity. The first is a two-dimensional homogeneous problem with Dirichlet boundary conditions and a known analytical solution, serving as a baseline validation case. The second problem introduces a point source at the domain center, considered with both Dirichlet (MP-2a) and Sommerfeld radiation (MP-2b) boundary conditions. To evaluate performance on heterogeneous media, we consider two problems from geophysical applications: the wedge problem, featuring three distinct velocity layers, and the Marmousi problem, a complex benchmark with 158 horizontal layers that represents realistic geological structures. Each problem is formulated as a Helmholtz equation with appropriate boundary conditions and source terms.

2.2. NUMERICAL METHODS

Following the finite difference discretization framework introduced in Chapter 1, we employ a second-order scheme on uniform grids, resulting in a sparse, indefinite linear system

$$A_h\mathbf{u}_h = (-\Delta_h - k^2 I_h)\mathbf{u}_h = \mathbf{b}_h \tag{2.1}$$

where the discrete Helmholtz operator and boundary conditions can be represented through compact stencil notation.

To solve the resulting large-scale linear systems, we employ preconditioned Krylov subspace methods, including full GMRES, GMRES(m), Bi-CGSTAB, and IDR(s). These iterative methods are accelerated using the CSLP preconditioner,

$$M_h = -\Delta_h - \left(\beta_1 + \beta_2 \mathrm{i}\right) k^2 I_h \tag{2.2}$$

where $\beta_1 = 1$ and $\beta_2 = 0.5$. The CSLP preconditioner itself is efficiently approximated using the standard geometric multigrid method. A multigrid method involves several components that need a careful design to achieve excellent convergence. In this chapter, a damped Jacobi smoother with relaxation $\omega = 0.8$ is used. The so-called full weighting restriction operator and the bilinear interpolation operator are employed for the inter-grid transfer operations. The coarse-grid operator M_{2h} is constructed by re-discretizing on the coarse mesh in the same way as the operator M_h is obtained on the fine mesh. The classical multigrid V-cycle is performed. Instead of solving the coarsest-grid problem directly, we will solve it by full GMRES.

2.3. PARALLEL IMPLEMENTATION

To implement parallel computing, the standard MPI library is employed for data communications among the processors. Therefore, the design of MPI topology will be the basis. Further, the domain partition and the data structure within the processors will determine the implementation of the matrix-vector multiplication and dot product. Unless noted otherwise, we are mainly concerned with a regular rectangular computational domain in this section.

2.3.1. MPI TOPOLOGY

During the MPI setup, the first step is to determine the total number of processors (denotes as np) and the number of processors in each direction. In two-dimensional cases, the number of processors in the x- and y- direction is denoted as $npx0$ and $npy0$ respectively.

To tell different processors whether their corresponding subdomains contain physical boundaries or interface boundaries, we define npx and npy to describe the position in the x- and y- directions for every processor, as shown by Figure 2.1. For example, npx is in the range of $[0, npx0-1]$. When npx is equal to 0, the process needs to deal with the left physical boundary. When npx equals $npx0-1$, the process needs to handle the right-hand physical boundary. It is similar in the y- direction.

When acquiring np processors and creating a parallel computing environment using MPI, each processor will own its ID rank (0 to $np-1$). We assign each subdomain to each processor according to the x-line lexicographic order. Besides, we also need to decide the ID rank of adjacent processors (physically in the computational domain) for data transfer. For example, in Figure 2.1, for a processor with ID = 7,

$npx = 3$ and $npy = 1$ correspondingly, the ID of the adjacent processor in the north is $YP = (npy+1)\times npx0 + npx = 11$. If the boundary condition in the x- direction is not periodic, then the ID of the adjacent processor in the east is $XP = NULL$. If it is a periodic boundary condition, then $XP = npy \times npx0 + ((npx+1) - npx0) = 4$. This way, each processor knows with which processor it should synchronize the interface boundary data.

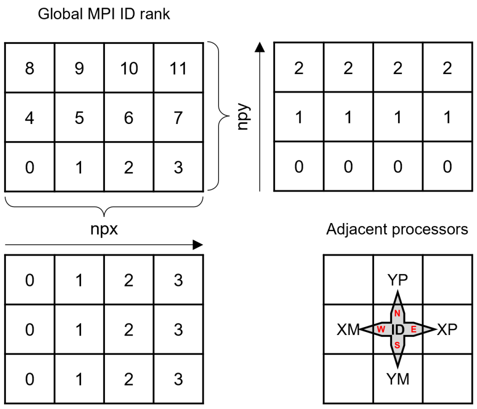


Figure 2.1: MPI setup for npx0 × npy0 = 4×3

2.3.2. Domain Partition and Data Structure

Based on the MPI setup, we can partition the computational domain blockwise and allocate the variables to the corresponding processor. In domain partitioning, we choose to carry out the partition between two grid points, *i.e.* along the red dotted line shown in Figure 2.2. Therefore, the boundary points of adjacent subdomains are adjacent grid points in the global grids. Several (denoted by lap) layers of overlapping grid points are introduced outward at each interface boundary to represent the adjacent grid points, that is the red grid points of the subdomain on the right of Figure 2.2. In our method, the grid unknowns are stored as an array based on the grid ordering (i, j) instead of a column vector based on x-line lexicographic ordering. Alternatively, the partition can be carried out along the grid points, the boundary points of adjacent subdomains are the same grid points in the global grids. In contrast, the former is more economical and avoids the operations of averaging, but it may bring inconvenience to the implementation of the multigrid method.

For example in Figure 2.2, we divide the number of grid points as evenly as possible along x- and y- directions. The number of grid points in x- and y- directions within each subdomain is denoted as nx and ny respectively. The grid-ordering arrays $u(i, j)$ are assigned to 3×3 different processors. To store and use the data from adjacent processors, the arrays are extended based on the subdomain grid structure. For second-order finite-difference discretization, the variable lap only needs to be 1. Then, the indices range of the array $\mathbf{u}$ becomes $(1-lap : nx+lap, 1-lap : ny+lap)$.

Within a certain subdomain, the operations and array updates are limited to the range $(1:nx, 1:ny)$. The data $u(i,j)$ for $i=1$ and nx, as well as $j=1$ and ny 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 of interface grid points.

In addition, the global index number of the first grid point in a subdomain is recorded as parameters, for example, i_offset and j_offset in Figure 2.2. This is mainly used when collecting global variables and implementing a multigrid method based on the global grid.

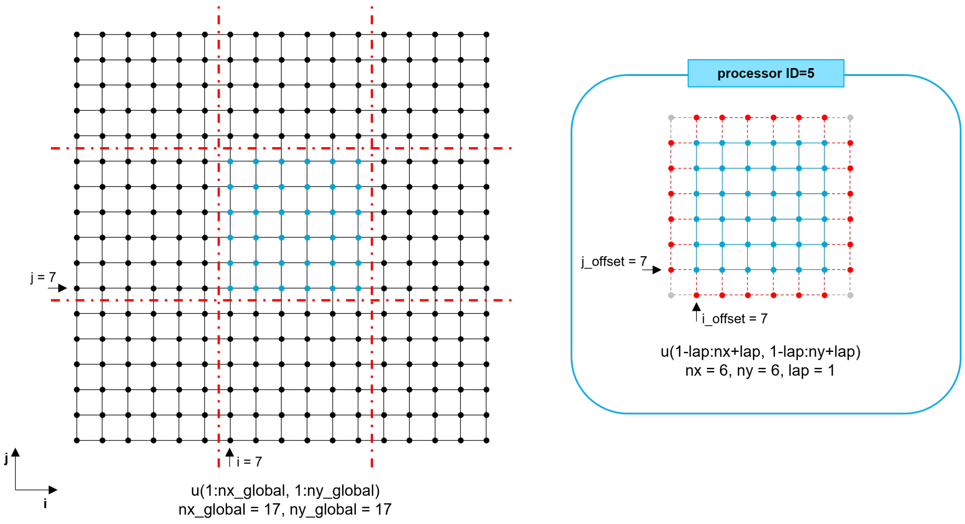

j = 7
i = 7
j
i
u(1:nx_global, 1:ny_global)
nx_global = 17, ny_global = 17
processor ID=5
j_offset = 7
i_offset = 7
u(1-lap:nx+lap, 1-lap:ny+lap)
nx = 6, ny = 6, lap = 1

Figure 2.2: Two-dimensional domain partition for np = 3×3 and the data structure

2.3.3. Matrix-free Matrix-Vector Multiplication

Consider a two-dimensional problem discretized using the second-order central finite difference method, resulting in N unknowns. The resulting system is solved by the CSLP-preconditioned Krylov subspace method with the matrices assembled. Apart from the variables vectors, we need extra memory to store the sparse matrix A_h with $5N$ non-zero elements, M_h with $5N$ non-zero elements, M_{2h} with $\frac{9N}{4}$, inter-grid transfer operator Z with $\frac{9N}{4}$, etc. To minimize memory limitations and solve real-world large-scale problems, we implement the preconditioned Krylov subspace methods in a matrix-free way instead of constructing the coefficient matrices explicitly.

In our method, instead of constructing the coefficient matrices explicitly, we implement the matrix-vector multiplication in a matrix-free way. Note that the Helmholtz operator as well as the CSLP has a similar stencil which only needs the data of the four adjacent grid points in the current iteration step.

Considering any grid point (i,j) $(1\le i\le nx,\ 1\le j\le ny)$, define ap, aw, ae, as and an as the multipliers of $u(i,j)$, $u(i-1,j)$, $u(i+1,j)$, $u(i,j-1)$ and $u(i,j+1)$, respectively.

When physical boundary conditions are encountered, it is only necessary to set the multiplier corresponding to meaningless grid points to zero. For example, if $u(i,j)$ is a left boundary grid point, aw is zero. For Dirichlet boundary conditions, we can simply set $ap = 1$, $aw = ae = as = an = 0$.

The Helmholtz operator can be implemented according to Equation (1.50). We have

$$ap = \frac{4 - k^2h^2}{h^2} \qquad aw = ae = as = an = -\frac{1}{h^2} \tag{2.3}$$

For the CSLP operator, according to Equation (2.2), we will have

$$ap = \frac{4 - \left(\beta_1 + \beta_2 \mathrm{i}\right) k^2 h^2}{h^2} \qquad aw = ae = as = an = -\frac{1}{h^2} \tag{2.4}$$

Thus, computing $v = A_h u$ or $y = M_h x$ can be conducted in a matrix-free way by Algorithm 8.

Algorithm 8 Matrix-free $\mathbf{v_h} = A_h \mathbf{u_h}$.

```
1: Input array $\mathbf{u_h}$;
2: Initialize the coefficient based on the stencil:
3:     $ap = 4 - k^2h^2$, $aw = ae = as = an = -1$ ;
4: Internal grid points ($i = 2 \cdots nx - 1,\ j = 2 \cdots ny - 1$):
5:     $v_h(i,j) = ap \cdot u_h(i,j) + aw \cdot u_h(i-1,j) + ae \cdot u_h(i+1,j) + as \cdot u_h(i,j-1) + an \cdot u_h(i,j+1)$;
6: Boundary grid points ($i = 1, nx,\ j = 1, ny$):
7:     adjust $ap$, $aw$, $ae$, $as$ and $an$ and compute $v_h(i,j)$;
8: Return $\mathbf{v_h}/h^2$.
```

2.3.4. PARALLEL MULTIGRID ITERATION BASED ON GLOBAL GRID

In this section, we will consider the parallel implementation of the multigrid iteration based on the original global grid.

First, we customize a data type called `grid`. This type includes three extended grid variables u, rhs and res representing unknowns, right-hand sides and residuals respectively, as well as nx_global, nx, i_offset and other grid parameters mentioned above. Then, according to the relationship between the fine grid and coarse grid, the parameters of the coarse grid are determined by the grid parameters of the fine one. For example, point (i_c, j_c) in the coarse grid corresponds to point $(2i_c - 1, 2j_c - 1)$ in the fine grid. The restriction, as well as interpolation of the grid variables, can be implemented according to the stencils based on the index correspondence between the coarse and fine grid. Finally, the coarse grid problem can be solved by GMRES similarly. These operations are all implemented in a matrix-free way.

For a V-cycle, after reaching a manually predefined the coarsest grid size, for example each domain must contain at least 2×2 grid points, the coarsen operation will stop and employ a full GMRES solver to solver the coarse problem parallelly. In the following, the predefined coarsest grid size is denoted by $nx_{coarsest} \times ny_{coarsest}$. Alternatively, the solver can be set to switch from a parallel to a sequential mode after reaching a certain multigrid coarse level. It will be a future research direction. This work only focuses on parallelism.

2.4. Numerical Experiments

The program is developed in Fortran 90 and 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. The Numerical experiments were conducted on a Linux cluster. Computational node "nw-node7" utilizes an Intel Core i7-10700 CPU with 8 physical cores (16 logical threads via hyperthreading), 2.9 GHz base frequency (4.8 GHz turbo), and 16 MB shared L3 cache. Node "nw-node11" employs an Intel Xeon Gold 6152 processor featuring 22 physical cores (44 logical threads with hyperthreading), 2.1 GHz base frequency (3.7 GHz turbo), and 30.25 MB L3 cache. Unless otherwise specified, the latter will be the default computing platform for this chapter.

In the numerical experiments, unless mentioned, the following convergence criterion of the preconditioned GMRES algorithm is used.

$$\frac{\left\|M^{-1}\mathbf{b}_h - M_h^{-1}A_h\mathbf{u}_h^k\right\|_2}{\left\|M_h^{-1}\mathbf{b}_h\right\|_2} \leq 10^{-6} \tag{2.5}$$

The number of iterations required is denoted by #Iter. The stopping criterion for the coarse grid preconditioner solver is 10^{-8}. In order to illustrate the accuracy of the numerical approximation and compare the difference between sequential and parallel results, we denote

$$\text{Abs. err.} = \left\|\mathbf{u}_{exact} - \mathbf{u}_h^k\right\|_\infty \tag{2.6}$$

where $\mathbf{u}^k$ is the numerical solution after k iterations. u_{exact} is the analytical solution.

This section will mainly illustrate the numerical performance of our solver for the model problems on different computational nodes. The speedup and parallel efficiency are used to estimate the parallel performance so far. The WallClockTime for the preconditioned GMRES solver to reach the stopping criterion is denoted as t_w. The speedup S_p is defined by

$$S_p = \frac{t_s}{t_p} \tag{2.7}$$

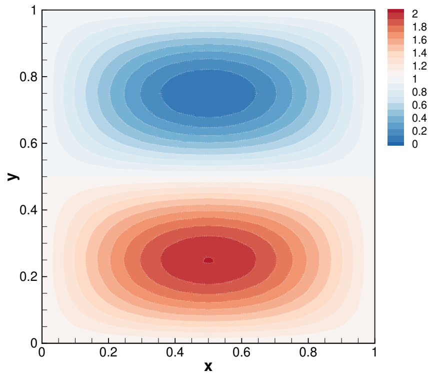
where t_s and t_p are the WallClockTime for sequential and parallel computation, respectively. The parallel efficiency E_p is given by

$$E_P = \frac{S_p}{np} \times 100\% = \frac{t_s}{t_p \cdot np} \times 100\% \tag{2.8}$$

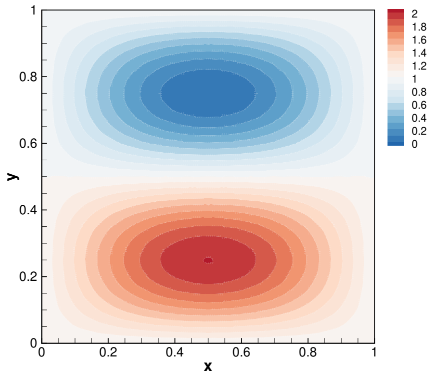
where np is the number of processors.

2.4.1. Closed-off Problem

Figure 2.3 shows the intuitive numerical approximation for the closed-off problem at $k = 100$, $kh = 0.625$. It can be seen that the results obtained by parallel computing are consistent with the sequential results.



(a) Real part of the sequentially computed solution



(b) Real part of the 4-processor parallel computing solution

Figure 2.3: Numerical solutions for the closed-off problem at $k = 100$, $kh = 0.625$

Parallel Efficiency

Maintaining $kh = 0.625$, we solve the closed-off problem for various wavenumbers until reaching convergence. Table 2.1 reveals diminished parallel efficiency when the subdomain grid sizes are reduced, primarily attributable to the substantial communication overhead incurred during processing. It can be roughly estimated from the table that the grid size of each subdomain must be at least 20×20 to ensure that the parallel efficiency is not less than 70%. Besides, when k is 320 and the number of iterative steps is as large as 1115, the parallel efficiency of our method is still up to 80%. One can also observe that the number of required iterations exhibits a significant increase as wavenumber k increases.

Weak Scalability

As shown in Table 2.2, we test the numerical performance of parallel CSLP preconditioned GMRES for different grid sizes as $k = 100$ is fixed. The speedups indicate that the parallel efficiencies are more than 80% for different test cases. Furthermore, as highlighted in Table 2.2, the wall-clock time (t_w) remains relatively constant when the number of processors and grid size increase proportionally, demonstrating the weak scalability characteristics of our parallel algorithm.

Profiling

To further demonstrate the performance of our parallel Krylov solver preconditioned by multigrid-based CSLP, we conducted comprehensive profiling using `gprof` to identify computational bottlenecks and resource utilization patterns.

The profiling results for sequentially computed and parallel computing with 9 processors are shown in Tables 2.3 and 2.4, respectively. We can find that the dot-

Table 2.1: Parallel performance of CSLP preconditioned GMRES for closed-off problem with different wavenumber k on nw-node11. $kh = 0.625$, CSLP approximated by V-cycle, $nx_{coarsest} \times ny_{coarsest} = 9 \times 9$

k	nx	np	#Iter	t_w(s)	Abs. err.	S_p	E_p
40	65	1	66	0.08	2.250E-05	1.00	-
		4	66	0.04	2.250E-05	2.17	54.24
60	97	1	135	0.47	1.711E-05	1.00	-
		9	135	0.11	1.593E-05	4.23	47.00
80	129	1	224	1.84	1.899E-04	1.00	-
		4	224	0.56	1.630E-04	3.27	81.81
		16	224	0.30	1.795E-04	6.14	38.41
120	193	1	532	22.57	6.306E-04	1.00	-
		4	531	6.05	6.767E-04	3.73	93.21
		9	531	2.99	6.747E-04	7.56	84.01
160	257	1	789	90.85	4.176E-03	1.00	-
		4	790	21.77	4.159E-03	4.17	104.31
		16	790	7.37	4.159E-03	12.32	77.03
180	289	1	855	132.17	7.552E-04	1.00	-
		4	855	32.15	7.540E-04	4.11	102.79
		9	855	16.21	7.539E-04	8.15	90.58
240	385	1	983	310.33	1.249E-03	1.00	-
		4	983	78.25	1.249E-03	3.97	99.14
		9	983	35.37	1.249E-03	8.77	97.48
		16	983	24.70	1.249E-03	12.56	78.52
320	513	1	1115	723.18	4.329E-03	1.00	-
		4	1115	181.64	4.329E-03	3.98	99.54
		16	1115	52.40	4.329E-03	13.80	86.26

Table 2.2: Numerical performance of parallel CSLP preconditioned GMRES on nw-node11 for different grid sizes for $k = 100$. CSLP approximated by V-cycle, $nx_{coarsest} \times ny_{coarsest} = 9 \times 9$.

grid size	kh	np	#Iter	Abs. err.	t_w(s)	S_p
161×161	0.625	1	343	3.14351E-03	**6.81**	1.00
		4	344	2.93436E-03	1.89	3.59
321×321	0.3125	1	308	6.56389E-03	25.11	1.00
		4	308	6.56555E-03	**6.20**	4.05
481×481	0.2083	1	377	6.23648E-05	82.07	1.00
		4	378	5.45449E-05	21.10	3.89
		9	377	7.71755E-05	**9.52**	8.62
641×641	0.15625	1	328	8.12076E-04	113.06	1.00
		4	328	8.12861E-04	29.76	3.80
		9	328	8.12081E-04	14.04	8.05
		16	328	8.11990E-04	**8.76**	12.91

products as well as the vector operations *axpy* in the Arnodi iteration of GMRES take up most of the time. In the case of parallel computing with 9 processors, the percentage does not change significantly, and the communication functions do not stand out as the main time-consuming function.

Table 2.3: Part of flat profile results for solving the closed-off problem with $k = 128$, $nx \times ny = 513 \times 513$, $np = 1$

% time[1]	cumulative seconds[2]	self seconds[3]	calls[4]	Procedure[5]
42.77	174.01	174.01	485	Arnoldi_
42.27	345.99	171.98	117855	dot_prod_
5.65	368.96	22.97	12561	cslp_op_
2.66	379.79	10.83	5844	damp_jacobi_smoother_
1.75	386.89	7.10	5844	diagcslp_op_
1.39	392.54	5.65	487	helmholtz2d_
0.75	395.60	3.06	487	V_cycle

[1] The percentage of the total running time of the program used by this function.
[2] A running sum of the number of seconds accounted for by this function and those listed above it.
[3] The number of seconds accounted for by this function alone.
[4] The number of times this function was invoked.
[5] The name of the function.

Table 2.4: Part of flat profile results for solving the closed-off problem with $k = 128$, $nx \times ny = 513 \times 513$, $np = 9$

%	cumulative	self		
time	seconds	seconds	calls	Procedure
44.94	293.64	293.64	4365	Arnoldi_
43.98	581.05	287.41	1060695	dot_prod_
3.3	602.64	21.59	113067	cslp_op_
2.43	618.52	15.88	52596	damp_jacobi_smoother_
0.95	624.74	6.22	52596	diagcslp_op_
0.79	629.93	5.19	4383	helmholtz2d_
0.73	639.84	4.79	4383	V_cycle

Dot-Product Benchmark

According to the profiling, the dot-product operation constitutes the most computationally intensive component. To evaluate computational performance, in this section, we focus on analyzing the bandwidth achieved by our parallel implementation of dot-product operation. The achieved bandwidth is calculated by dividing the total memory accessed by the execution time.

In our numerical framework, variables from a two-dimensional physical domain, denoted as $u(nx, ny)$ and $v(nx, ny)$, are represented as two-dimensional arrays, and their dot-product is subsequently computed. Additionally, we examine three alternative implementations of the dot-product operation. The investigation encompasses four distinct methodologies for implementing the dot-product calculation as shown in Algorithms 9 to 12.

The first two implementations explore loop-level optimizations, where Algorithm 9 maintains the 2D array structure while leveraging loop unrolling to reduce loop overhead and improve instruction-level parallelism. Algorithm 10 takes this optimization further by treating the arrays as continuous memory blocks, potentially improving cache utilization through better memory access patterns. Algorithm 11 represents a more abstract approach that relies on built-in array operations. Algorithm 12 strikes a balance between optimization and code clarity by combining a simple loop structure with optimized intrinsic functions, making it particularly effective when the compiler can efficiently optimize the built-in dot-product function.

Analysis of the dot product benchmark on **nw-node7** shows contrasting performance patterns. As shown in Tables 2.5 and 2.6, for small arrays, the implementation achieves good speedup and scalable bandwidth, as the data fits within the CPU cache, which offers significantly higher bandwidth than main memory. However, with large arrays that exceed cache capacity, performance deteriorates because the processor of nw-node7 has limited memory architecture with a maximum bandwidth of $36\,\mathrm{GB\,s^{-1}}$. In these cases, adding cores provides no speedup since they must share this fixed bandwidth limit.

Performance evaluations of the dot-product benchmark conducted on **nw-node11** are presented in Table 2.7. The analysis demonstrates that even with large-scale

Algorithm 9 Dot-product: Loop-based 2D arrays

```
Require: 2D Arrays $u(nx,ny), v(nx,ny)$
 1: for $j = 1$ to $ny$ do
 2:     for $i = 1$ to $nx$ step 4 do
 3:         $dp0 \leftarrow dp0 + u(i,j) \times v(i,j)$
 4:         $dp1 \leftarrow dp1 + u(i+1,j) \times v(i+1,j)$
 5:         $dp2 \leftarrow dp2 + u(i+2,j) \times v(i+2,j)$
 6:         $dp3 \leftarrow dp3 + u(i+3,j) \times v(i+3,j)$
 7:     end for
 8: end for
 9: $dot\_prod\_local \leftarrow dp0 + dp1 + dp2 + dp3$
10: MPI_Allreduce($dot\_prod\_local$, $dot\_prod\_global$, ..., MPI_SUM, ...)
```

Algorithm 10 Dot-product: Loop-based 1D arrays

```
Require: 2D Arrays $u(nx,ny), v(nx,ny)$
 1: for $i = 1$ to $nx \times ny$ step 4 do
 2:     $dp0 \leftarrow dp0 + u(i) \times v(i)$
 3:     $dp1 \leftarrow dp1 + u(i+1) \times v(i+1)$
 4:     $dp2 \leftarrow dp2 + u(i+2) \times v(i+2)$
 5:     $dp3 \leftarrow dp3 + u(i+3) \times v(i+3)$
 6: end for
 7: $dot\_prod\_local \leftarrow dp0 + dp1 + dp2 + dp3$
 8: MPI_Allreduce($dot\_prod\_local$, $dot\_prod\_global$, ..., MPI_SUM, ...)
```

Algorithm 11 Dot-product: Original 2D array-based

```
Require: 2D Arrays $u(nx,ny), v(nx,ny)$
 1: $dot\_prod\_local \leftarrow$ sum$(u(1:nx,1:ny) \times v(1:nx,1:ny))$
 2: MPI_Allreduce($dot\_prod\_local$, $dot\_prod\_global$, ..., MPI_SUM, ...)
```

Algorithm 12 Dot-product: Loop intrinsic dot-product function

```
Require: 2D Arrays $u(nx,ny), v(nx,ny)$
 1: for $j = 1$ to $ny$ do
 2:     $dot\_prod\_local \leftarrow dot\_prod\_local +$ dot_product$(u(1:nx,j), v(1:nx,j))$
 3: end for
 4: MPI_Allreduce($dot\_prod\_local$, $dot\_prod\_global$, ..., MPI_SUM, ...)
```

Table 2.5: Dot-product performance test on nw-node7 using two 512×512 arrays, repeated 2000 times, accessing 0.84 GB of memory total.

np	t_w(s)	S_p	Achieved bandwidth (GB s^{-1})
Loop-based 2D			
1	0.029	1.00	28.73
2	0.028	1.04	29.81
4	0.014	2.04	58.75
8	0.008	3.86	110.99
Loop-based 1D			
1	0.029	1.00	28.65
2	0.028	1.04	29.86
4	0.014	2.07	59.16
8	0.008	3.91	112.03
Original array-based			
1	0.045	1.00	18.86
2	0.023	1.95	36.86
4	0.012	3.85	72.56
8	0.006	7.18	135.41
Loop intrinsic dot_product			
1	0.042	1.00	19.83
2	0.023	1.86	36.80
4	0.011	3.70	73.48
8	0.006	6.95	137.87

Table 2.6: Dot-product performance test on nw-node7 using two 10000×10000 arrays, repeated 30 times, accessing 48.0 GB of memory total.

np	t_w(s)	S_p	Achieved bandwidth (GB s^{-1})
Loop-based 2D			
1	2.598	1.00	18.48
2	2.042	1.27	23.51
4	1.579	1.65	30.40
8	1.423	1.83	33.74
Loop-based 1D			
1	2.595	1.00	18.50
2	2.037	1.27	23.57
4	1.577	1.65	30.44
8	1.422	1.83	33.77
Original array-based			
1	2.980	1.00	16.11
2	1.914	1.56	25.08
4	1.535	1.94	31.28
8	1.371	2.17	35.02
Loop intrinsic dot_product			
1	2.971	1.00	16.16
2	1.913	1.55	25.09
4	1.535	1.94	31.27
8	1.370	2.17	35.03

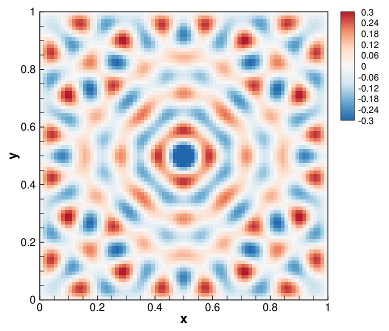
arrays of dimensions 10000×10000, the implementation maintains scalable bandwidth efficiency. The results suggest that the processing units on nw-node11 exhibit superior peak bandwidth capabilities compared to those on nw-node7. Through the dot-product benchmark analysis, we have confirmed that our implementation is indeed memory-bounded. Consequently, Algorithm 12 will be adopted as the standard version for implementation in our solver frameworks.

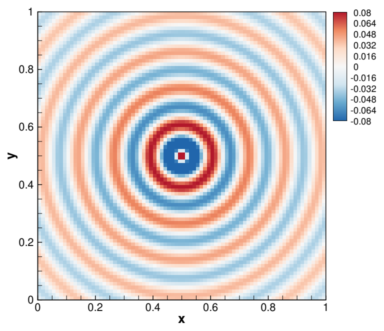
Table 2.7: Dot-product performance test on ne-node11 using two 10000×10000 arrays, repeated 30 times, accessing 48.0 GB of memory total.

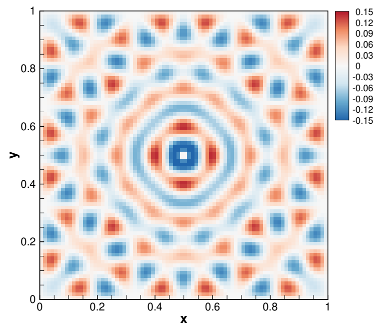
np	t_w(s)	S_p	Achieved bandwidth (GB s^{-1})
	Loop-based 2D		
1	4.401	1.00	10.91
2	2.318	1.90	20.71
4	1.181	3.73	40.65
8	0.630	6.99	76.20
	Loop-based 1D		
1	4.408	1.00	10.89
2	2.320	1.90	20.69
4	1.177	3.75	40.80
8	0.629	7.01	76.36
	Original array-based		
1	4.633	1.00	10.36
2	2.403	1.93	19.98
4	1.224	3.79	39.23
8	0.645	7.18	74.42
	Loop intrinsic dot_product		
1	4.698	1.00	10.22
2	2.427	1.94	19.78
4	1.240	3.79	38.72
8	0.650	7.23	73.87

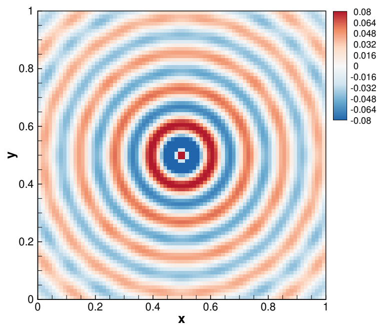
2.4.2. Constant-Wavenumber Problem with Point Source

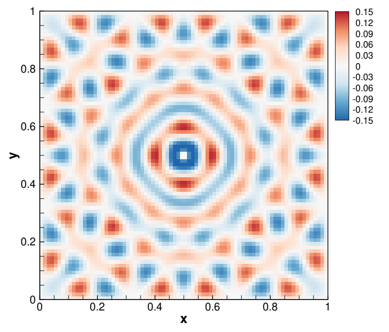
Figure 2.4 shows a qualitative comparison between the analytical solution and numerical approximation of MP-2a and MP-2b in Section 1.2.1 for $k = 50$. Our solver provides a reasonable approximation of the exact solution, independent of parallel partitioning. Nevertheless, minor discrepancies in wave amplitudes are observed, primarily attributable to discretization errors arising from the finite-difference representation of the Dirac delta function.


(a) MP-2a: Real part of the analytical solution


(b) MP-2b: Real part of the analytical solution


(c) MP-2a: Real part of the sequentially computed solutions


(d) MP-2b: Real part of the sequentially computed solutions


(e) MP-2a: Real part of the parallel computing solutions with 4 processors

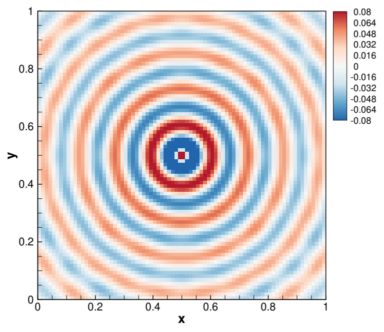
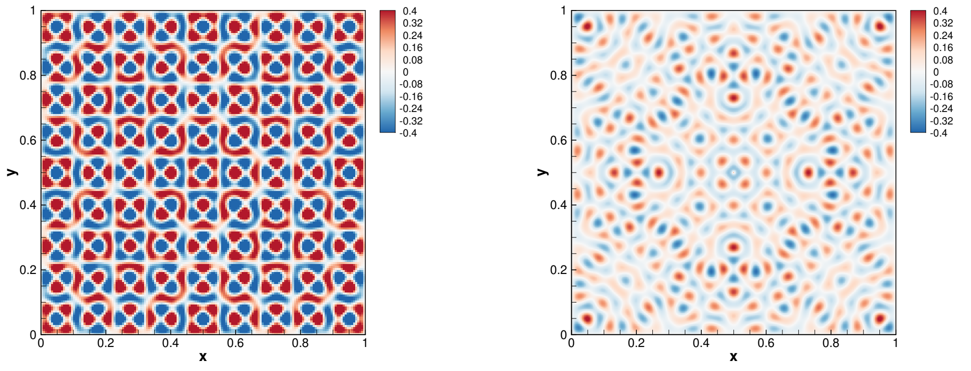

(f) MP-2b: Real part of the parallel computing solutions with 4 processors

Figure 2.4: Solutions for MP-2 at $k = 50$, $kh = 0.625$

Comparison of Boundary Conditions

To illustrate the pollution effect, we analyze the numerical approximation of MP-2a with wavenumber $k = 100$, as illustrated in Figure 2.5. The results in Figure 2.5a demonstrate that a resolution of 10 grid points per wavelength proves insufficient to produce an accurate solution. However, when the resolution is increased to 50 grid points per wavelength, as shown in Figs. 2.5b, both phase and amplitude of the numerical solution exhibit satisfactory accuracy. These findings confirm that for a given wavenumber, the pollution effect can be minimized by increasing the spatial resolution of the computational grid.

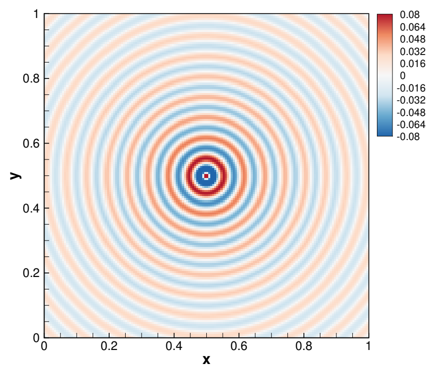
As for MP-2b in Figure 2.6, maintaining 10 grid points per wavelength is sufficient to effectively control both phase and amplitude discrepancies when $k = 100$. This behavior can be attributed to the attenuation effects inherent in the Sommerfeld boundary conditions, which reduce the severity of the pollution effect by preventing boundary reflections, thereby relaxing resolution requirements.



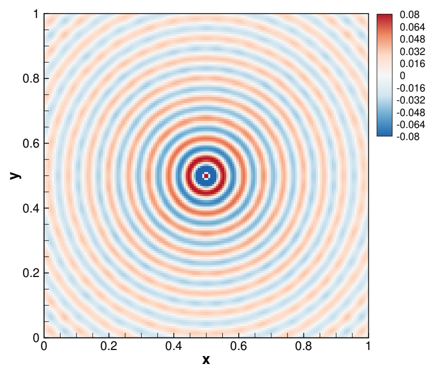
(a) $kh = 0.625$ (b) $kh = 0.125$

Figure 2.5: Solutions for MP-2a at $k = 100$

Figure 2.7 demonstrates that the CSLP-preconditioned GMRES algorithm exhibits superior convergence rates for MP-2b compared to MP-2a. This enhanced performance can be attributed to the distinct spectral properties of the preconditioned matrix $M_h^{-1}A_h$, as illustrated in Figure 2.8. The application of Sommerfeld boundary conditions results in eigenvalues tending to cluster more around the point (1, 0), whereas the Dirichlet boundary conditions yield a significant number of eigenvalues near zero. This spectral distribution pattern explains the relatively poor convergence behavior exhibited by MP-2a during intermediate iterations, as demonstrated in Figure 2.7a.

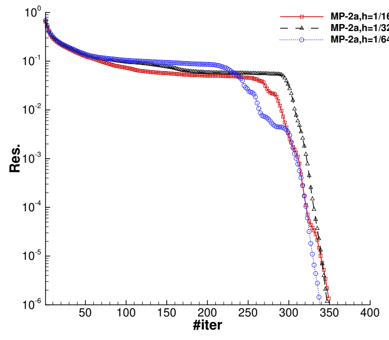


(a) MP-2b: Real part of the analytical solution

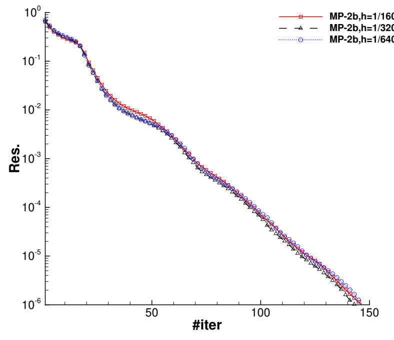


(b) MP-2b: Real part of the numerical approximation

Figure 2.6: Solutions for MP-2b at $k = 100$, $kh = 0.625$

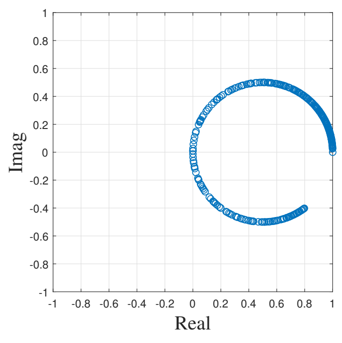


(a) MP-2a

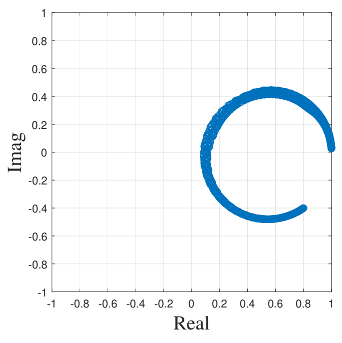


(b) MP-2b

Figure 2.7: The convergence behavior for MP-2 with different boundary conditions, $k = 100$.



(a) MP-2a



(b) MP-2b

Figure 2.8: Spectrum of $M_h^{-1} A_h$ for MP2 with $k = 100$, $kh = 0.625$.

Comparison of different Krylov methods

According to 2.4.1, memory and bandwidth requirements are an important factor affecting the parallel efficiency. The Arnoldi process in GMRES requires many dot product operations, requiring global communication in parallel computing. To restrict work and memory requirements, one may think of the GMRES(m) procedure, or other Krylov methods that use short recurrences. In fact, the matrix-free parallel implementation together with CSLP are not limited to the GMRES algorithm. All the ingredients can be directly generalized to GMRES(m) or other Krylov methods like Bi-CGSTAB and IDR(s).

To show the dominant operations of different methods, first we compare the profiling results of full GMRES, GMRES(m), Bi-CGSTAB, IDR(s) in the case of sequentially computed, as shown in Tables 2.8 to 2.11. The numerical solution employs GMRES(m) with a restart parameter of $m = 50$. For the IDR(s) algorithm, $s = 4$ is chosen because it is a good compromise between performance and storage [45]. It can be seen that in GMRES and GMRES(m), the Arnoldi process as well as the dot product accounts for the largest proportion of the running time. The second is the related operations of approximating the inverse of CSLP by the multigrid method, such as the smoother, which take up most of the time in Bi-CGSTAB and IDR(4). The dot product operations no longer play a significant role in Bi-CGSTAB and IDR(4). The proportion of matrix-vector multiplication becomes obvious instead.

Table 2.8: GMRES: part of flat profile results for MP-2b, $k = 100$, $nx \times ny = 161 \times 161$, $np = 1$

% time	cumulative seconds	self seconds	calls	Procedure
38.53	2.25	2.25	147	_prearnoldi
36.82	4.40	2.15	10878	_dot_prod
6.51	4.78	0.38	1490	_damp_jacobi_smoother
3.08	4.96	0.18	1776	_cslp_op_bc
2.74	5.12	0.16	149	_finestgrid_define
2.4	5.26	0.14	149	_v_cycle
2.23	5.39	0.13	745	_prolongation_en_correct
1.71	5.49	0.10	1	_pre_fullgmres
1.54	5.58	0.09	149	_helmholtz2d_bc
1.2	5.65	0.07	10348050	_cslp2d_stencils
1.03	5.71	0.06	149	_mg_fullgmres

Numerical experiments show that, for small wavenumber, there is little difference between Bi-CGSTAB, IDR(s), GMRES(m) and GMRES, since CSLP is a good preconditioner. To compare the parallel performance of these Krylov methods, we will consider MP-2b with $k = 400$ and $kh = 0.625$. Results are presented in Table 2.12, where “#Matvec” denotes the number of matrix-vector multiplications. Among the methods examined, GMRES demonstrated the highest computational cost. Nevertheless, GMRES exhibited superior parallel efficiency compared to other methods. Analysis of performance profiles indicates that increasing the ratio of matrix-vector multiplications to dot products does not inherently enhance parallel efficiency within

Table 2.9: GMRES(m): part of flat profile results for MP-2b, $k = 100$, $nx \times ny = 161 \times 161$, $np = 1$

% time	cumulative seconds	self seconds	calls	Procedure
25.74	1.05	1.05	200	_prearnoldi
24.76	2.06	1.01	5100	_dot_prod
15.69	2.70	0.64	2050	_damp_jacobi_smoother
4.66	2.89	0.19	2457	_cslp_op_bc
4.66	3.08	0.19	1025	_prolongation_en_correct
4.17	3.25	0.17	205	_finestgrid_define
3.92	3.41	0.16	205	_v_cycle
2.45	3.51	0.10	14237250	_cslp2d_stencils
2.45	3.61	0.10	1	_pre_restartgmres
1.96	3.69	0.08	205	_multigrid_based_cslp
1.72	3.76	0.07	210	_norm

Table 2.10: Bi-CGSTAB: part of flat profile results for MP-2b, $k = 100$, $nx \times ny = 161 \times 161$, $np = 1$

% time	cumulative seconds	self seconds	calls	Procedure
25	0.48	0.48	1730	_damp_jacobi_smoother
10.42	0.68	0.20	173	_v_cycle
9.9	0.87	0.19	1	_cidrs
6.77	1.00	0.13	865	_prolongation_en_correct
6.25	1.12	0.12	173	_finestgrid_define
5.73	1.23	0.11	2070	_cslp_op_bc
5.47	1.34	0.11	12014850	_cslp2d_stencils
4.69	1.43	0.09	175	_cmatvec
4.17	1.51	0.08	865	_coarsegrid_create
3.39	1.57	0.07	175	_helmholtz2d_bc
2.6	1.62	0.05	865	_restriction

Table 2.11: IDR(4): part of flat profile results for MP-2b, $k = 100$, $nx \times ny = 161 \times 161$, $np = 1$

% time	cumulative seconds	self seconds	calls	Procedure
16.43	0.34	0.34	1630	_damp_jacobi_smoother
15.46	0.66	0.32	1	_cidrs
9.66	0.86	0.20	163	_v_cycle
7.73	1.02	0.16	815	_prolongation_en_correct
6.76	1.16	0.14	164	_cp_dot
5.31	1.27	0.11	165	_cmatvec
5.07	1.38	0.11	11320350	_cslp2d_stencils
4.83	1.48	0.10	1951	_cslp_op_bc
4.83	1.58	0.10	815	_restriction
4.83	1.68	0.10	163	_finestgrid_define
4.59	1.77	0.10	165	_helmholtz2d_bc

this framework. IDR(4) and Bi-CGSTAB showed similar execution times, with IDR(4) demonstrating slightly superior parallel performance.

To explore the scalability, considering MP-2b with $k = 200$, we increase the number of processors correspondingly while refining the grid. For each method, the time consumed remains almost constant in Table 2.13. The results further demonstrate that our proposed matrix-free parallel solution method exhibits weak scalability, irrespective of the chosen Krylov subspace method.

Table 2.12: Parallel performance of different parallel CSLP-preconditioned Krylov methods for MP-2b $k = 400$ and $kh = 0.625$.

np	#Matvec	t_w(s)	S_p	E_p
		GMRES		
1	563	305.50	-	-
4	563	83.15	3.67	91.85
9	563	39.01	7.83	87.02
16	563	23.96	12.75	79.69
		GMRES(m)		
1	650	74.82	-	-
4	650	19.95	3.75	93.74
9	650	9.86	7.59	84.30
16	650	6.36	11.77	73.55
		Bi-CGSTAB		
1	632	39.73	-	-
4	629	10.81	3.68	91.90
9	628	5.80	6.85	76.11
16	631	3.76	10.57	66.07
		IDR(4)		
1	624	43.83	-	-
4	602	11.56	3.79	94.76
9	602	6.12	7.16	79.56
16	587	3.96	11.06	69.14

2.4.3. Wedge Model Problem

In this section, the two-dimensional wedge problem described in Section 1.2.1 is used to evaluate the performance of our parallel solution method for a simple heterogeneous medium.

Figure 2.9 shows the typical wave diffraction pattern of the wedge problem at $f = 80$ Hz. One can observe that the wavefront is significantly curved at the slow-to-fast interface. The wave is mainly reflected in that transition region.

Table 2.14 gives the required number of matrix-vector multiplications (denoted by #Matvec), CPU-time and relative speedup of different CSLP-preconditioned Krylov methods for the wedge problem. The source frequency is $f = 80$ Hz with grid size 769×1281, which indicates $kh \leq 0.26$ and guarantees more than 20 grid points per wavelength.

According to the results, among the different Krylov methods, GMRES has the least number of matrix-vector multiplications and the best parallel efficiency, but requires

Table 2.13: Parallel performance of different Krylov methods for MP-2b with $k = 200$ while refining the grid.

Grid size	np	#Matvec	t_w(s)
		GMRES	
321×321	4	285	5.27
481×481	9	310	6.79
641×641	16	283	6.80
		GMRES(m)	
321×321	4	350	2.47
481×481	9	350	2.76
641×641	16	350	3.31
		Bi-CGSTAB	
321×321	4	336	1.41
481×481	9	342	1.61
641×641	16	302	1.77
		IDR(4)	
321×321	4	321	1.52
481×481	9	321	1.73
641×641	16	289	1.9

the most CPU time. The time required for GMRES(m) is about 20% of the time for GMRES, while the time required for Bi-CGSTAB and IDR(4) is only 11% of the time for GMRES. This confirms that it is not suitable to use full GMRES if the number of iterations is large. For this case, the number of matrix-vector multiplications of IDR(4) is close to that of GMRES. And IDR(4) is the least time-consuming. Our matrix-free parallel CSLP-preconditioned method is still suitable for the common Krylov solvers for heterogeneous Helmholtz problems. Table 2.15 illustrates that it still leads to a satisfactory scalability if we increase the number of processors correspondingly while refining the grid.

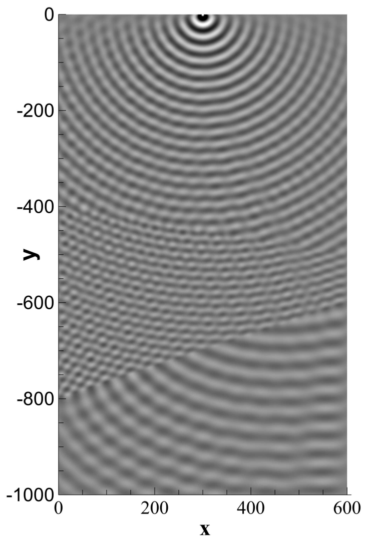


Figure 2.9: An example of the wave diffraction pattern of the wedge problem at $f = 80\,\text{Hz}$

2.4.4. Marmousi Model Problem

In this section, the so-called Marmousi model problem in Section 1.2.1 is considered for the industrial applications that usually involve a highly heterogeneous medium. Compared to the wedge-shaped velocity distribution, the Marmousi velocity distribution includes both discontinuous and much more complex velocity variations.

A typical wave diffraction pattern of the Marmousi problem at $f = 40\,\text{Hz}$ is shown Figure 2.10. Despite the rather complex velocity distribution, it is still possible to find an explanation for the wave diffraction pattern. We can observe that the wave propagates mainly in channels formed by a "fast" medium embedded in a "slow" medium.

Table 2.16 presents the required number of matrix-vector multiplications (denoted by #Matvec), CPU-time and relative speedup of different CSLP-preconditioned Krylov methods for the Marmousi problem. The source frequency is $f = 40\,\text{Hz}$ with a grid size of 2945×961, which indicates $kh \leq 0.54$ and guarantees more than 10 grid points per wavelength.

One can find that for such a setup of the Marmousi problem, a huge number of iterations are required to reduce the relative residual to 10^{-6}. Likewise, GMRES has the least number of matrix-vector multiplications but requires the most CPU time. The parallel efficiency is also reduced to 50%. IDR(4) and Bi-CGSTAB have similar performances. In contrast to Table 2.12, IDR(4) and Bi-CGSTAB exhibit higher

Table 2.14: MP-3: parallel performance of different parallel CSLP-preconditioned Krylov methods, $f = 80\,\text{Hz}$, grid size 769×1281. One iteration of the multigrid V-cycle is used to approximate the inverse of the preconditioner.

np	#Matvec	t_w(s)	S_p	E_p
		GMRES		
1	613	989.74	-	-
2	613	475.92	2.08	103.98
8	613	126.99	7.79	97.42
18	613	78.50	12.61	70.04
		GMRES(m)		
1	700	205.86	-	-
2	700	103.60	1.99	99.35
8	700	29.16	7.06	88.25
18	700	18.43	11.17	62.06
		Bi-CGSTAB		
1	722	111.31	-	-
2	713	56.12	1.98	99.17
8	739	19.31	5.76	72.05
18	745	12.91	8.62	47.91
		IDR(4)		
1	600	102.60	-	-
2	604	52.89	1.94	96.99
8	640	19.14	5.36	67.00
18	638	12.15	8.44	46.91

Table 2.15: MP-3: CPU time consumed by different parallel CSLP-preconditioned Krylov methods while refining the grid, $f = 40\,\text{Hz}$. One iteration of the multigrid V-cycle is used to approximate the inverse of the preconditioner. The number of matrix-vector multiplications is in parentheses. It should be noted that GMRES(m) with $m = 50$ converges in 7 iterations.

Grid size	np	GMRES	GMRES(m)	Bi-CGSTAB	IDR(4)
385 × 641	2	25.34 (278)	11.99 (350)	6.20 (321)	6.10 (282)
769 × 1281	8	31.92 (283)	14.95 (350)	7.98 (301)	7.41 (251)
1153 × 1921	18	68.14 (282)	27.12 (350)	14.51 (312)	13.48 (239)

parallel efficiency than GMRES. The results illustrate that the matrix-free parallel CSLP-preconditioned method also works for the highly heterogeneous Helmholtz problems.

The data presented in Table Table 2.17 fails to exhibit satisfactory weak scalability, primarily attributed to the substantial computational requirements of large grid sizes approaching the bandwidth limitations of the compute node, consequently leading to diminished parallel efficiency.

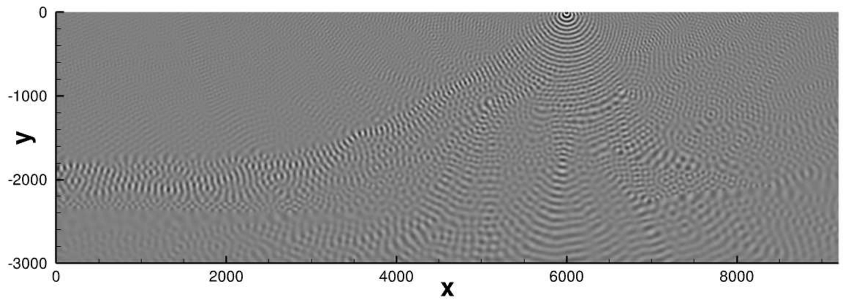


Figure 2.10: An example of the wave diffraction pattern of the Marmousi problem at 40 Hz

Table 2.16: MP-4: parallel performance of different parallel CSLP-preconditioned Krylov methods, $f = 40$ Hz, grid size 2945×961. One iteration of the multigrid V-cycle is used to approximate the inverse of the preconditioner.

np	#Matvec	t_w(s)	S_p	E_p
		GMRES		
1	2872	61705.58	-	-
3	2872	21892.21	2.82	93.95
12	2872	10309.02	5.99	49.88
		GMRES(m)		
1	3350	2958.46	-	-
3	3350	907.40	3.26	108.68
12	3350	374.10	7.91	65.90
		Bi-CGSTAB		
1	4124	2431.94	-	-
3	4435	712.34	3.41	113.80
12	4513	279.09	8.71	72.61
		IDR(4)		
1	4438	2881.58	-	-
3	4688	854.72	3.37	112.38
12	4484	334.59	8.61	71.77

Table 2.17: MP-4: CPU time consumed by different parallel CSLP-preconditioned Krylov methods while refining the grid, $f = 20\,\text{Hz}$. One iteration of the multigrid V-cycle is used to approximate the inverse of the preconditioner. The number of matrix-vector multiplications is in parentheses.

Grid size	np	GMRES	GMRES(m)	Bi-CGSTAB	IDR(4)
1473 × 481	3	543.50 (1106)	76.84 (1200)	45.04 (1090)	44.34 (973)
2945 × 961	12	1160.57 (1019)	135.91 (1200)	73.65 (1117)	77.86 (1054)

2.5. Conclusions

This chapter presents a systematic investigation of matrix-free parallel solution methods for the two-dimensional Helmholtz equation using preconditioned Krylov subspace methods with CSLP. It is closely related to our research objectives (i). Our numerical experiments validate the accuracy of the numerical solution through comparison with the analytical solution and demonstrate its robust performance across various model problems. Performance analysis reveals the memory-bound nature of computations, with the dot-product operations and multigrid-based preconditioner comprising the primary computational cost.

A key achievement is the weak scalability of our matrix parallel framework, maintaining consistent iteration counts as problem size increases, while enabling improved accuracy through grid refinement for medium wavenumbers. This property establishes the possible utility for pollution-free simulations. Furthermore, the flexibility of the framework in accommodating various Krylov methods enhances its adaptability to different problem characteristics and computational requirements.

The robust parallel performance observed across diverse model problems, ranging from homogeneous to heterogeneous media, validates the effectiveness of our matrix-free approach. This success in handling varying degrees of problem complexity suggests promising potential for broader applications. The insights gained regarding performance characteristics and scalability will be valuable for research in subsequent Chapters. Our current implementation solves 2D problems effectively; the extension to 3D heterogeneous high-frequency Helmholtz equations will be discussed in the next chapter.

3 Matrix-free parallel CSLP-preconditioned iterative solvers in Three Dimensions

The matrix-free parallel preconditioned Krylov subspace method is extended to the 3D heterogeneous Helmholtz equation. The CSLP is employed and 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.

Parts of this chapter have been published in Electronic Transactions on Numerical Analysis **59**, 270–294 (2023) [125].

Building upon the two-dimensional framework presented in Chapter 2, this chapter is interested in parallelizing Krylov subspace methods, such as GMRES, Bi-CGSTAB, and IDR(s), preconditioned by the multigrid-based CSLP for the 3D Helmholtz equation. 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. [116, 123] 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 chapter 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 chapter is organized as follows. Section 3.1 describes the mathematical model and discretization technique that we will discuss. The numerical methods are briefly described in Section 3.2. Section 3.3 describes the parallel implementation. The numerical performance is explored in Section 3.4. Finally, Section 3.5 contains our conclusions.

3.1. MATHEMATICAL MODELS

The mathematical foundation remains the heterogeneous Helmholtz equation in the frequency domain, now defined on a three-dimensional parallelepipedal domain

$$-\Delta u(x,y,z) - k(x,y,z)^2\, u(x,y,z) = b(x,y,z), \tag{3.1}$$

which is complemented by appropriate boundary conditions as detailed in Chapter 1.

To validate and evaluate our three-dimensional solver, we consider a progression of model problems with increasing complexity. The analytical closed-off problem with constant coefficients serves as a verification benchmark, followed by the wedge problem that introduces simple heterogeneity through distinct velocity layers. The culminating test case is the industry-standard SEG/EAGE salt model, which incorporates realistic geological features including complex salt dome structures within a large-scale computational domain. These model problems present distinct computational challenges: while the 3D closed-off problem enables direct accuracy assessment through comparison with analytical solutions, the 3D wedge and 3D SEG/EAGE salt models introduce the practical difficulties of heterogeneous media and complex geometries encountered in geophysical applications.

The discretization follows the second-order finite difference scheme on vertex-centered grids, maintaining at least ten grid points per wavelength to ensure solution accuracy. For detailed mathematical formulations and discretization schemes, we refer to the comprehensive treatment provided in Chapter 1. The resulting discrete systems retain the characteristics of being indefinite (and complex-valued), but with significantly larger dimensions compared to the 2D counterparts, thereby motivating our focus on efficient parallel solution strategies.

3.2. NUMERICAL METHOD

Building upon the foundation established in Chapter 1 and the 2D implementation discussed in Chapter 2, we employ preconditioned Krylov subspace methods to solve the discretized 3D Helmholtz problem. Our framework primarily utilizes GMRES, Bi-CGSTAB, and IDR(s) methods.

For preconditioning, we continue to employ the CSLP, with parameters $\beta_1 = 1$ and $\beta_2 = 0.5$ except where specified otherwise. The preconditioner is approximately inverted using a geometric multigrid method.

3.2.1. 3D MULTIGRID FOR THE PRECONDITIONER SOLVE

This section presents an overview of the key components comprising a 3D geometric multigrid method, which represents an extension of the two-dimensional framework discussed in Chapter 1.

SMOOTHERS

In this framework, we will mainly use the damped Jacobi smoother as it is easy to parallelize and has been shown effective in previous work [49].The impact of varying the relaxation parameter for the damped Jacobi smoother and using different smoothers on the convergence properties have already been extensively investigated by [49, 126]. 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. We will fix the relaxation parameter at $\omega = 0.8$, as this value was found to work well for our test problems.

RESTRICTION

Figure 3.1 shows part of a 3D fine grid with a coarse grid obtained by standard coarsening. Two sets of uniform grids with size h and $H = 2h$ are used to discretize a regular computational domain $\Omega = (0,1) \times (0,1) \times (0,1)$.

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

$$I_h^H = \frac{1}{64}\left[\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}_h^H \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H\right]. \tag{3.2}$$

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

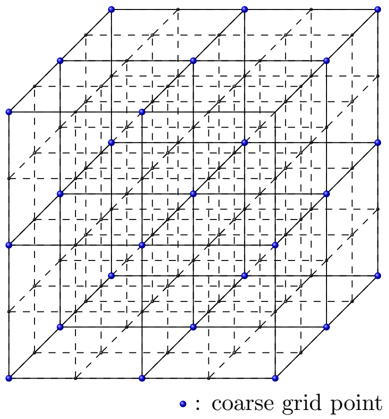


Figure 3.1: Vertex-centered 3D standard coarsening.

follows.

$$
\begin{aligned}
b^H_{i_1,i_2,i_3} = I^H_h r^h_{2i_1,2i_2,2i_3} = \frac{1}{64}\Big(& 8r^h_{2i_1,2i_2,2i_3} \\
&+4r^h_{2i_1-1,2i_2,2i_3} + 4r^h_{2i_1+1,2i_2,2i_3} + 4r^h_{2i_1,2i_2-1,2i_3} \\
&+4r^h_{2i_1,2i_2+1,2i_3} + 4r^h_{2i_1,2i_2,2i_3-1} + 4r^h_{2i_1,2i_2,2i_3+1} \\
&+2r^h_{2i_1-1,2i_2-1,2i_3} + 2r^h_{2i_1+1,2i_2-1,2i_3} + 2r^h_{2i_1-1,2i_2+1,2i_3} + 2r^h_{2i_1+1,2i_2+1,2i_3} \\
&+2r^h_{2i_1-1,2i_2,2i_3-1} + 2r^h_{2i_1+1,2i_2,2i_3-1} + 2r^h_{2i_1-1,2i_2,2i_3+1} + 2r^h_{2i_1+1,2i_2,2i_3+1} \\
&+2r^h_{2i_1,2i_2-1,2i_3-1} + 2r^h_{2i_1,2i_2+1,2i_3-1} + 2r^h_{2i_1,2i_2-1,2i_3+1} + 2r^h_{2i_1,2i_2+1,2i_3+1} \\
&+r^h_{2i_1-1,2i_2-1,2i_3-1} + r^h_{2i_1+1,2i_2-1,2i_3-1} + r^h_{2i_1-1,2i_2+1,2i_3-1} + r^h_{2i_1+1,2i_2+1,2i_3-1} \\
&+r^h_{2i_1-1,2i_2-1,2i_3+1} + r^h_{2i_1+1,2i_2-1,2i_3+1} + r^h_{2i_1-1,2i_2+1,2i_3+1} + r^h_{2i_1+1,2i_2+1,2i_3+1}\Big),
\end{aligned}
\tag{3.3}
$$

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

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

$$
I^h_H = \frac{1}{8}\left[\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}^h_H \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}^h_H \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}^h_H\right]. \tag{3.4}
$$

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

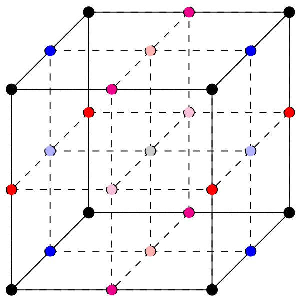


Figure 3.2: The allocation map of interpolation operator.

$$
\begin{aligned}
&I_H^h u_{i_1,i_2,i_3}^H = \\
&\begin{cases}
u_{i_1,i_2,i_3}^H, & \bullet \\
\frac{1}{2}\left(u_{i_1,i_2,i_3}^H + u_{i_1+1,i_2,i_3}^H\right), & \bullet \\
\frac{1}{2}\left(u_{i_1,i_2,i_3}^H + u_{i_1,i_2+1,i_3}^H\right), & \bullet \\
\frac{1}{2}\left(u_{i_1,i_2,i_3}^H + u_{i_1,i_2,i_3+1}^H\right), & \bullet \\
\frac{1}{4}\left(u_{i_1,i_2,i_3}^H + u_{i_1+1,i_2,i_3}^H + u_{i_1,i_2+1,i_3}^H + u_{i_1+1,i_2+1,i_3}^H\right), & \bullet \\
\frac{1}{4}\left(u_{i_1,i_2,i_3}^H + u_{i_1+1,i_2,i_3}^H + u_{i_1,i_2,i_3+1}^H + u_{i_1+1,i_2,i_3+1}^H\right), & \bullet \\
\frac{1}{4}\left(u_{i_1,i_2,i_3}^H + u_{i_1,i_2+1,i_3}^H + u_{i_1,i_2,i_3+1}^H + u_{i_1,i_2+1,i_3+1}^H\right), & \bullet \\
\frac{1}{8}\left(u_{i_1,i_2,i_3}^H + u_{i_1,i_2+1,i_3}^H + u_{i_1+1,i_2,i_3}^H\right. & \\
+\, u_{i_1+1,i_2+1,i_3}^H + u_{i_1,i_2,i_3+1}^H + u_{i_1,i_2+1,i_3+1}^H & \\
\left.+\, u_{i_1+1,i_2,i_3+1}^H + u_{i_1+1,i_2+1,i_3+1}^H\right), & \bullet
\end{cases}
\end{aligned}
\tag{3.5}
$$

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

COARSE-GRID OPERATOR

With a parallel implementation in mind, we keep using the re-discretization approach to obtain the coarse-grid operator. Boundary conditions of the preconditioner operator are set identically to the corresponding model problems.

3.2.2. 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 in Equation (1.81). One does not need to construct M explicitly since the result of Mx can be calculated by its corresponding stencil. Besides, the matrix-free restriction and prolongation operators in the multigrid method can be implemented according to Equation (3.3) and Equation (3.5), respectively.

3.3. 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. In addition, 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.

3.3.1. Parallel Setup

For the MPI setup, the first step is to determine the total number of processes (denoted by 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 $npx0$, $npy0$and $npz0$ respectively. As shown in Figure 3.3, the entire computational domain is partitioned into $3\times4\times5$ 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 npx, npy, and npz to describe the position in the x-, y- and z- directions for every process. For example, npx is in the range of $[0, npx0-1]$. When npx is equal to 0, the process needs to deal with the left physical boundary. When npx equals $npx0-1$, 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 3.5). 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 3.4).

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 based on the subdomain grid structure as shown in Figure 3.4. Thus,

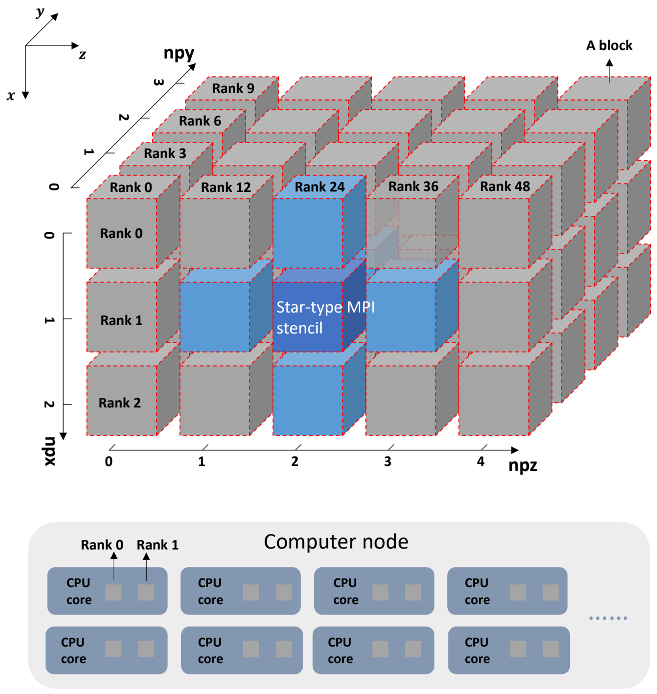


Figure 3.3: The schematic diagram of the MPI topology for npx0 × npy0 × npz0 = 3 × 4 × 5.

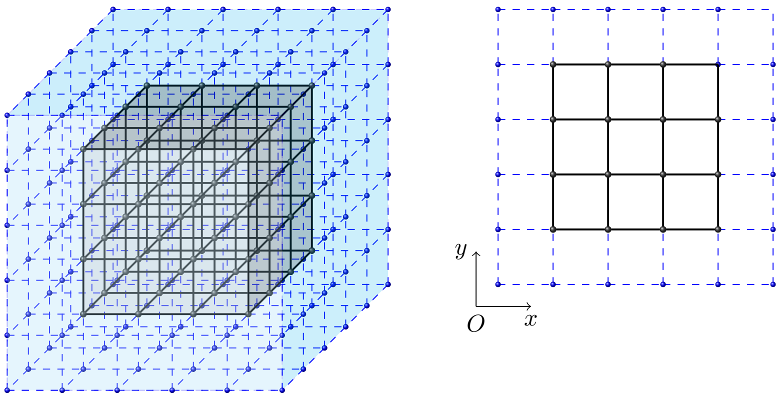


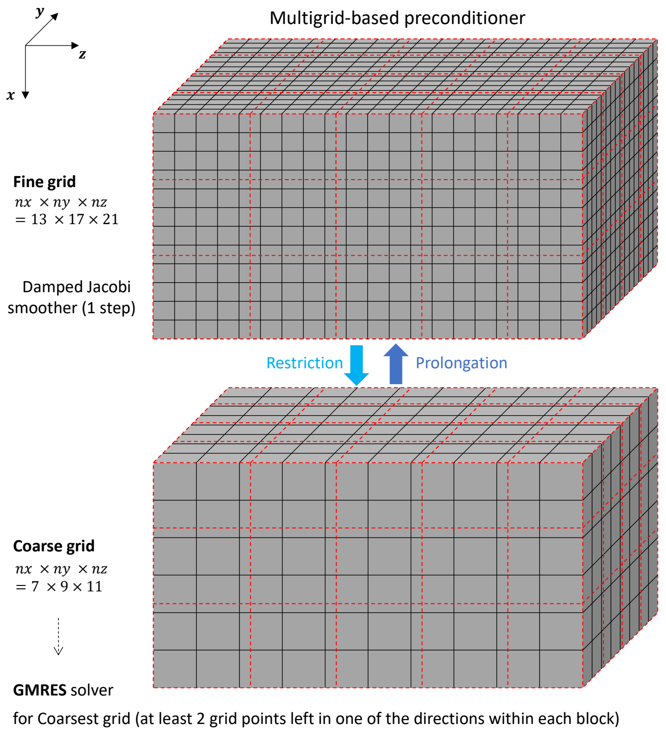
Figure 3.4: The schematic diagram of the local grid structure.

the index range of the array $\mathbf{u}$ becomes $(1-lap : nx+lap, 1-lap : ny+lap, 1-lap : nz+lap)$. 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.

3.3.2. 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 3.5, 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 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 each 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 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. This 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.



y
z
x
Multigrid-based preconditioner
Fine grid
$nx \times ny \times nz$
$= 13 \times 17 \times 21$
Damped Jacobi smoother (1 step)
Restriction
Prolongation
Coarse grid
$nx \times ny \times nz$
$= 7 \times 9 \times 11$
GMRES solver
for Coarsest grid (at least 2 grid points left in one of the directions within each block)

Figure 3.5: The schematic diagram of parallel multigrid based on global-grid coarsening. Grid size is chosen at random for demonstration purposes.

3.4. Numerical Experiments

The numerical experiments are primarily conducted on the Linux supercomputer DelftBlue [127]. 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 GB s^{-1} per socket, and a 100 Gbit s^{-1} 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 used in this section, the number of matrix-vector multiplications (denote 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.

$$\frac{\left\|M_h^{-1}\mathbf{b}_h - M_h^{-1}A_h\mathbf{u}_h^k\right\|_2}{\left\|M_h^{-1}\mathbf{b}_h\right\|_2} \leq 10^{-6}. \tag{3.6}$$

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

$$\frac{\left\|\mathbf{b}_h - A_h\mathbf{u}_h^k\right\|_2}{\|\mathbf{b}_h\|_2} \leq 10^{-6}. \tag{3.7}$$

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 1.4.2. 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 initial experiments, the tolerance should be 3 orders of magnitude smaller than that 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_w. The speedup S_p is defined by

$$S_p = \frac{t_r}{t_p}, \tag{3.8}$$

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

$$E_p = \frac{S_p}{np/np_r} = \frac{np_r \cdot t_r}{np \cdot t_p}, \tag{3.9}$$

where np_r and np is the number of processors for reference and parallel computations, respectively. 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's performance when the problem size and the number of processing elements increase proportionally. This allows us to assess our parallel solver's 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 at the end of the section.

3.4.1. Validation

We validate the numerical results by comparing numerical 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.

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 (1.27), we can estimate the magnitude of the error between the numerical and analytical solutions. Figure 3.6 shows the logarithmic error $\log_{10}(|u - u_{\text{analy}}|)$ of numerical results obtained by CSLP-preconditioned IDR(4). The left panel is the result obtained by sequential computing, and the right panel is the result 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 $[a_1 \;\; a_2 \;\; a_3 \;\; a_4] \cdot [b_1 \;\; b_2 \;\; b_3 \;\; b_4]'$ 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, regarding 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. But extensive numerical experiments indicate that these differences are within an acceptable error range and will not significantly affect the convergence and accuracy.

Model Problems with Non-constant Wavenumber

The so-called wedge problem is a typical model with non-constant wavenumbers. Figure 3.7 shows a reasonable wave diffraction pattern of the wedge problem at $f = 40\,\text{Hz}$ and $80\,\text{Hz}$, which are obtained by parallel CSLP-preconditioned IDR(4) solver. Note that the wavefront is significantly curved at the slow-to-fast interface.

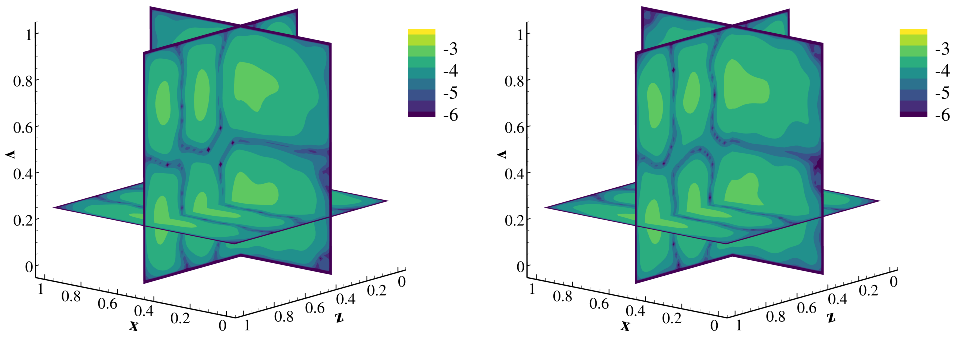


Figure 3.6: The logarithmic error $\log_{10}(|u - u_{\text{analy}}|)$ of numerical results obtained by CSLP-preconditioned IDR(4) with $np = 1$ (left) and $np = 3\times3\times3$ (right) for 3D Closed-off problem grid size $65\times65\times65$, $k = 40$.

The wave is mainly reflected in that transition region. It illustrates that the parallel framework also works for the case of non-constant wavenumbers.

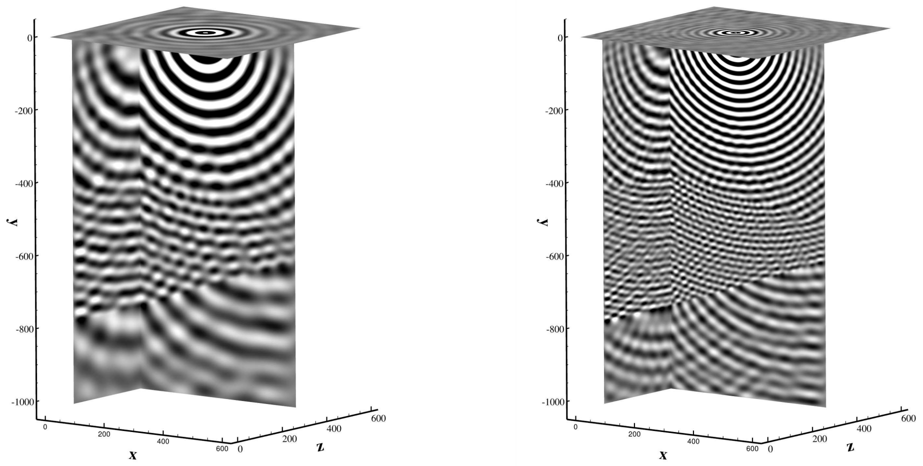


Figure 3.7: Real part of numerical solutions for 3D wedge Problem with grid size $193\times321\times193$ at $f = 40\,\text{Hz}$ (left), and $385\times641\times385$ at $80\,\text{Hz}$ (right).

For practical application, we further consider the 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 $641\times641\times193$, 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 $17\times17\times17$, 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 3.8 gives a reasonable wave diffraction pattern of this model at $f = 5\,\text{Hz}$. It is obtained by our parallel CSLP-preconditioned IDR(4) solver.

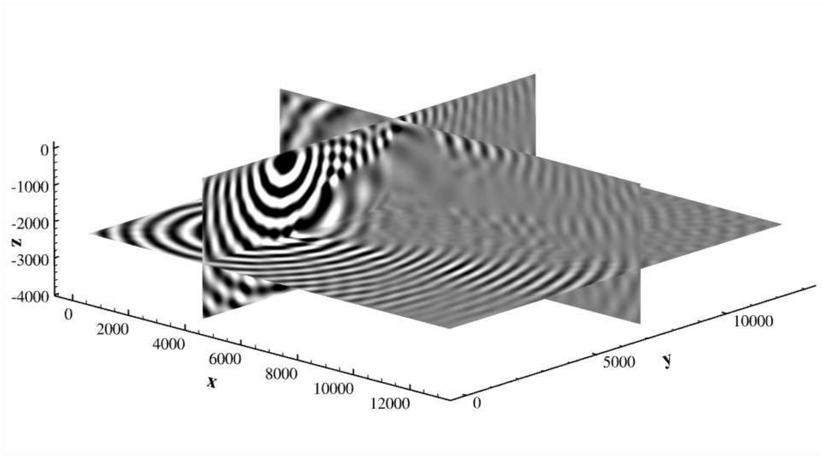


Figure 3.8: Real part of the solutions for 3D SEG/EAGE Salt Model with grid size $641 \times 641 \times 193$, at $f = 5\,\text{Hz}$.

KRYLOV METHODS

Various Krylov-based iterative methods can be implemented in our parallel framework; see Table 3.1. 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) are sensitive to the accumulation of rounding errors and the re-ordering of the linear system. Additionally, IDR(s) uses random vectors to initialize, 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.

MULTIGRID CYCLE TYPES

V-cycles follow a simple down-up pattern through grid levels (coarsening followed by refinement), while F-cycles enhance this approach by performing additional coarse-grid corrections during the refinement phase. F-cycles offer better convergence properties than V-cycles at a moderate increase in computational cost, making them an effective intermediate option between V-cycles and the more expensive W-cycles.

Table 3.1: 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
2×1×1	145	359	211
1×2×3	145	403	192
3×3×3	145	407	201

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 of the complex shift β_2 in CSLP on convergence is studied. As shown in Figure 3.9, 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 convergence results by using the F-cycle. Figure 3.9 confirms the introduction in Section 1.5.1, 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. For the 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 of 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 [128, §9].

When the complex shift is -1, as shown in Figure 3.10, 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 due to the additional levels of coarsening and interpolation, but the improved convergence can still make it a viable option in certain cases. It should be noted that IDR(4) exhibits convergence similar to GMRES in Figure 3.10 due to the use of right preconditioning for IDR(4) and left preconditioning for GMRES. This observation underscores the importance of preconditioning strategy selection in iterative solvers.

3.4.2. 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 3.2.

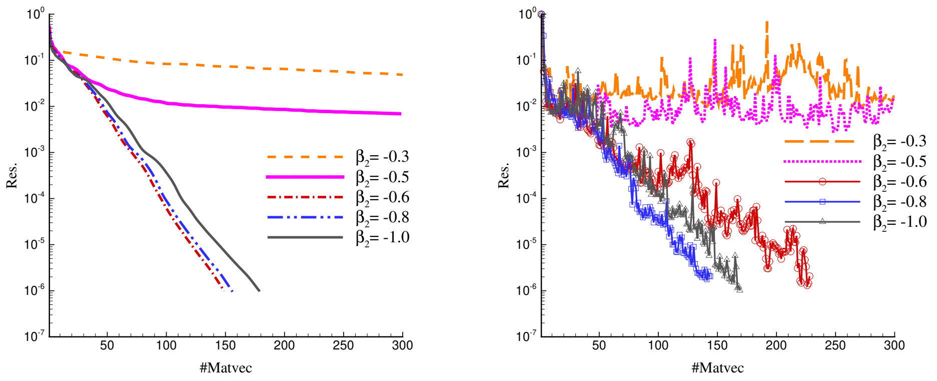


Figure 3.9: Convergence behavior of the parallel CSLP-preconditioned Krylov solvers, GMRES (left) and IDR(s) (right). One multigrid F-cycle is used for preconditioner solving. 3D wedge Problem with grid size $193 \times 321 \times 193$ at $f = 40\,\text{Hz}$ is solved. Different complex shifts of CSLP (β_2) are compared.

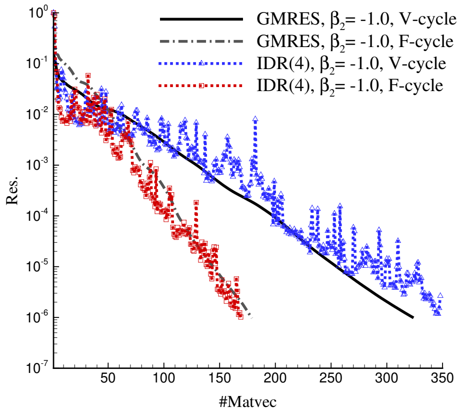


Figure 3.10: Convergence behavior of the parallel CSLP-preconditioned Krylov solver using one multigrid V- or F-cycle for preconditioning. The complex shift of CSLP is $\beta_2 = -1.0$. 3D wedge Problem with grid size $193 \times 321 \times 193$ at $f = 40\,\text{Hz}$ is solved.

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 Givens rotation matrix in GMRES becomes the bottleneck.

For IDR(4), when the number of processors and grid size increase 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 3.2: Weak scaling analysis of parallel CSLP-preconditioned GMRES and IDR(4) for 3D Closed-off problem, $k = 40$.

grid		GMRES		IDR(4)	
size	np	#Matvec	t_w(s)	#Matvec	t_w(s)
$129\times129\times129$	8	146	30.19	199	21.03
$193\times193\times193$	27	137	80.10	187	26.93
$257\times257\times257$	64	143	97.20	190	25.22

3.4.3. 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.

3D Closed-Off Problem

We initiate our analysis by comparing the strong scaling performance of our parallel framework with a PETSc [129, 130] 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 $129\times129\times129$ 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 preconditioner (CSLP). 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 3.3 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.

Table 3.3: Comparisons with PETSc implementation for one GMRES iteration. Parallel CSLP-preconditioned GMRES is used to solve the 3D Closed-off problem with grid size $129 \times 129 \times 129$, $k = 40$.

	PETSc			Present		
np	t_w(s)	S_p	E_p	t_w(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

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 3.4 and Table 3.5, 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 the growing number of processors, we can observe a moderate decrease in parallel efficiency. This is mainly due to increased communication, which leads to a decrease in the ratio of computation/communication.

Table 3.4: Performance of the parallel CSLP-preconditioned IDR(4) for the 3D wedge problem with grid size $193 \times 321 \times 193$ at $f = 40\,\text{Hz}$.

$npx \times npy \times npz$	#Matvec	t_w(s)	S_p	E_p
1 × 1 × 1	395	1033.11		
1 × 2 × 1	421	557.63	1.85	0.93
2 × 2 × 2	377	152.88	6.76	0.84

Table 3.5: Performance of the parallel CSLP-preconditioned IDR(4) for the 3D wedge problem with grid size $385 \times 641 \times 385$ at $f = 80\,\text{Hz}$.

$npx \times npy \times npz$	Nodes	#Matvec	t_w(s)	S_p	E_p
4×4×3	1	835	1313.82		
4×6×4	2	821	952.94	1.38	0.69
6×8×4	4	825	418.74	3.14	0.78
6×8×6	6	832	298.33	4.40	0.73

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 3.6 collects the number of matrix-vector multiplications and wall-clock time versus the number of compute nodes. The fact that we can solve 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.

Table 3.6: Performance of the parallel CSLP-preconditioned IDR(4) on DelftBlue for 3D SEG/EAGE Salt Model with grid size 641 × 641 × 193 at $f = 5\,\text{Hz}$.

$npx \times npy \times npz$	Nodes	#Matvec	t_w(s)	S_p	E_p
6×4×2	1	413	897.25		
6×8×2	2	423	510.56	1.76	0.88
6×8×4	4	423	298.86	3.00	0.75
9×8×4	6	404	203.31	4.41	0.74

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[1] (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 with 16 cores at 2.6 GHz, and 12×16G DDR4 2666 MHz ECC REG memory. The cluster is equipped with an Intel Omni-Path high-speed network with a transmission bandwidth of $100\,\text{GB}\,\text{s}^{-1}$ 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.

As shown in Table 3.7, 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.

Table 3.7: Performance of the parallel CSLP-preconditioned IDR(4) on Magic Cube 3 for 3D SEG/EAGE Salt Model with grid size 641 × 641 × 193 at $f = 5\,\text{Hz}$.

$npx \times npy \times npz$	Nodes	#Matvec	t_w(s)	S_p	E_p
4 × 4 × 2	1	405	505.14		
4 × 4 × 4	2	418	287.60	1.76	0.88
8 × 8 × 2	4	390	155.64	3.25	0.81

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

[1]Magic Cube 3: https://www.ssc.net.cn/en/resource-hardware.html

demonstrate the solver's potential for solving challenging 3D large-scale heterogeneous Helmholtz problems with limited memory and time consumption.

3.4.4. 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)[2]. 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 [131] for profiling and tracing, Scalasca [132] for extended analyses, and CUBE [133] for presentation.

Table 3.8 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. 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 the precondition, matrix-vector multiplication, and dot-product operations, as shown in Tables 3.9, 3.10, and 3.11, respectively. It can be seen that parallel matrix-vector multiplication can maintain fairly good efficiency. The precondition component, exhibiting similar behavior to the whole framework in Table 3.8, can be considered as the main factor that affects the overall efficiency. The results in Table 3.11 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 can be implemented in future work.

[2]POP CoE: https://www.pop-coe.eu

Table 3.8: 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	1	2	6
np	48	96	288
Parallel Efficiency (PE)[1]	0.83	0.78	0.32
Load Balance (LB) [2]	0.91	0.82	0.78
Communication Efficiency (CommE)[3]	0.92	0.96	0.41
Serialisation Efficiency (SerE) [4]	0.92	0.96	0.42
Transfer Efficiency (TE) [5]	0.99	1.00	0.97

[1] *Parallel efficiency* is the ratio of mean computation time to total runtime of all processes. PE = Load Balance × Communication Efficiency.

[2] *Load balance* is the mean/maximum ratio of computation time outside of MPI

[3] *Communication efficiency* is the maximum across all processes of the ratio between useful computation time and total runtime. CommE = Serialisation Efficiency × Transfer Efficiency.

[4] *Serialisation efficiency* is estimated from idle time within communications where no data is transferred.

[5] *Transfer efficiency* relates to essential time spent in data transfers.

Table 3.9: Execution efficiency of the precondition part of parallel IDR(4) on different numbers of compute nodes of DelftBlue

Compute nodes	1	2	6
np	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 3.10: Execution efficiency of the matrix-vector multiplications of parallel IDR(4) on different numbers of compute nodes of DelftBlue

Compute nodes	1	2	6
np	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 3.11: Execution efficiency of dot-product operations of parallel IDR(4) on different numbers of compute nodes of DelftBlue

Compute nodes	1	2	6
np	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

3.5. CONCLUSIONS

We have presented a comprehensive investigation of a matrix-free parallel framework for solving large-scale 3D Helmholtz problems in heterogeneous media in accordance with our objectives. The framework, built upon CSLP-preconditioned Krylov subspace methods, demonstrates significant advantages in both computational efficiency and practical applicability. Through extensive numerical experiments, we have validated the accuracy of the numerical approximations, while establishing its robust performance across various test cases of increasing complexity. Additionally, a POP performance analysis has been provided to address the bottleneck of this framework.

In conjunction with Chapter 2, our implementation makes several key contributions to the field of numerical methods for wave problems. First, the matrix-free parallelization of the CSLP preconditioner provides an effective alternative to traditional matrix-based approaches, achieving good convergence properties while maintaining modest memory requirements. Second, the framework's demonstrated weak and strong scaling properties establish its viability for large-scale applications, particularly important for minimizing pollution errors through refined grid resolution. Third, the flexibility in accommodating various Krylov subspace methods and multigrid techniques makes it a versatile tool for different problem characteristics.

The performance analysis reveals that our framework achieves substantial parallel efficiency on large-scale architectures, making it particularly suitable for practical applications in geophysics and other fields requiring the solution of heterogeneous

Helmholtz problems. The successful application to the SEG/EAGE salt model, a benchmark problem of significant complexity, demonstrates the capability of this framework to handle realistic scenarios encountered in industrial applications.

The subsequent chapters will focus on the investigation of advanced preconditioning techniques, specifically the APD method, to enhance convergence properties for high-frequency regimes, and the exploration of optimization strategies to further improve parallel efficiency.

4
Matrix-free Parallel Two-Level Deflation in Two Dimensions

A matrix-free parallel two-level deflation method combined with CSLP for 2D heterogeneous Helmholtz problems is proposed. Motivated by the limitations imposed by excessive computational time and memory constraints when employing a sequential solver with assembled matrices, we parallelize the two-level deflation method without constructing any matrices. Our approach utilizes preconditioned Krylov subspace methods and approximates the CSLP preconditioner with a parallel geometric multigrid V-cycle. For the two-level deflation, standard inter-grid deflation vectors and further high-order deflation vectors are considered. As another main contribution, the matrix-free Galerkin coarsening approach and a novel re-discretization scheme as well as high-order finite-difference schemes on the coarse grid are studied to obtain wavenumber-independent convergence. The optimal settings for an efficient coarse-grid problem solver are investigated. Numerical experiments of model problems show that the wavenumber independence has been obtained for medium wavenumbers. The matrix-free parallel framework shows satisfactory weak and strong parallel scalability.

Parts of this chapter have been published in Journal of Computational Physics **514**, 113264 (2024) [134].

Based on the parallel framework of the CSLP-preconditioned Krylov subspace methods built in the previous chapters, we propose a matrix-free parallel implementation of the two-level-deflation preconditioner for the two-dimensional Helmholtz problems. Besides matrix-free parallelization of each component of the deflation technique, as one of our main contributions, we extensively studied and compared several matrix-free implementations of coarse-grid operators. To compare the scalability of our matrix-free parallel solver with the benchmark results in [75], which used an exact solver on the coarse-grid system, we first solve the coarse-grid system close to machine precision using a default tolerance of 10^{-12}. This allows us to study the impact of the coarse-grid operators on the convergence of the two-level deflation preconditioner while minimizing the potential influence of an inexact coarse-system solver. Upon establishing the convergence behavior, we then focus on finding an efficient and practical setting for the coarse-system solver to optimize the performance of the proposed method. We show that the coarse-grid re-discretization schemes derived from the Galerkin coarsening approach yield convergence properties that are nearly independent of the wavenumber. Furthermore, we investigate the weak and strong scaling properties of the present parallel solution method in a massively parallel setting.

It is important to note that the present chapter focuses on the Helmholtz equation discretized on regular 2D finite difference grids in quadrilateral domains. The extension of the method to more complex geometries and unstructured grids requires further investigation. However, the 2D method presented in this chapter serves as a foundation and proof of concept for the proposed parallel matrix-free implementation of the two-level deflation preconditioner. In principle, the key ideas and techniques can be generalized to 3D problems, although such extensions are non-trivial and require significant additional work.

The chapter is organized as follows. The mathematical models and numerical methods are presented in Section 4.1. Section 4.2 introduces the parallel framework and matrix-free operators in detail. The experimental results will be discussed in Section 4.3, and the conclusions follow in Section 4.4.

4.1. Mathematical Models and Numerical Methods

This chapter extends our investigation to incorporating deflation techniques within the preconditioned Krylov framework for the two-dimensional Helmholtz equation on a rectangular domain Ω.

$$-\Delta u(x,y) - k(x,y)^2 u(x,y) = b(x,y). \tag{4.1}$$

The mathematical formulation builds upon previous chapters, where we consider the Helmholtz equation with both Dirichlet and first-order Sommerfeld radiation boundary conditions in heterogeneous media.

Our numerical experiments encompass three model problems of increasing complexity: a constant-wavenumber case for validation, the wedge problem introducing simple layered heterogeneity, and the industry-standard Marmousi problem that represent realistic geological structures. These problems are discretized using second-order finite differences on vertex-centered grids, maintaining at least ten grid points per wavelength to ensure solution accuracy.

The solution methodology combines the matrix-free parallel CSLP, previously discussed in Chapter 2, with a two-level deflation strategy. The CSLP parameters are set to $\beta_1 = 1$ and $\beta_2 = 0.5$, with the preconditioner approximately inverted using a geometric multigrid method. The two-level deflation reads as

$$P_h = M^{-1}_{h,(\beta_1,\beta_2)}(I_h - A_h Q_h) + \gamma Q_h, \quad \text{where } Q_h = I^h_{2h} A^{-1}_{2h} I^{2h}_h, \; A_{2h} = I^{2h}_h A_h I^h_{2h} \tag{4.2}$$

Throughout our numerical experiments, we set $\gamma = 1$ unless otherwise specified. The deflation matrix I^h_{2h} can be constructed using either bilinear interpolation, resulting in the A-DEF1 variant, or through the higher-order interpolation scheme introduced by [75], leading to the APD variant. Although this work establishes a comprehensive two-level deflation framework applicable to both variants, particular emphasis is placed on the matrix-free parallel implementation and performance analysis of the APD formulation.

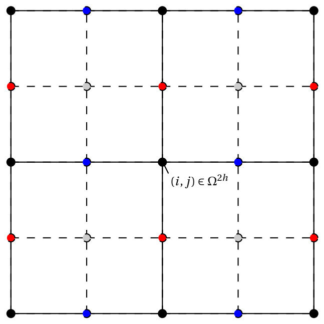
Since we aim for a wavenumber-independent convergence, we use the deflation preconditioner to enhance the convergence of GMRES-type methods. Besides the full GMRES method, the GCR method will also be considered as it inherently allows for variable preconditioning.

4.2. Matrix-Free Parallel Implementation

We will employ the Matrix-Free Parallel framework from Chapter 2. The parallel implementation is based on the MPI Cartesian topology, which partitions the computational domain blockwise. The parallel multigrid iteration is implemented on the global grid, with coarse grid parameters determined by the corresponding fine grid parameters using the index correspondence between coarse and fine grids. As shown in Figure 4.1, point (i, j) in the coarse grid corresponds to point $(2i-1, 2j-1)$ in the fine grid. The coarsening process stops when a predefined coarsest grid size is reached, and the coarsest grid problem is solved in parallel using GMRES. The coarsest grid size is chosen to ensure that each subdomain contains a sufficient number of grid points for efficient parallel computation. For the matrix-free implementation, in addition to the Helmholtz and CSLP operators described in Chapter 2, particular attention must be paid to the higher-order deflation vectors and coarse-grid operators in two-level deflation.

4.2.1. Higher-order Deflation Vectors

Restriction and interpolation operations are performed using stencils, such as Equations (1.72) and (1.74), based on the index correspondence between coarse and fine grids, enabling a matrix-free implementation. For two-dimensional cases, as shown in Figure 4.1, the higher-order interpolation can be implemented as follows.



● ● ● ○ : fine grid points $\in \Omega^h$
● : coarse grid points $\in \Omega^{2h}$

Figure 4.1: The allocation map of interpolation operator

$$
I_{2h}^{h} u_{i_c,j_c}^{2h} =
\begin{cases}
\frac{1}{64}\Big(6u_{i_c-1,j_c}^{2h} + 36u_{i_c,j_c}^{2h} + 6u_{i_c+1,j_c}^{2h} \\
\quad + u_{i_c-1,j_c-1}^{2h} + 6u_{i_c,j_c-1}^{2h} + u_{i_c+1,j_c-1}^{2h} \\
\quad + u_{i_c-1,j_c+1}^{2h} + 6u_{i_c,j_c+1}^{2h} + u_{i_c+1,j_c+1}^{2h}\Big), & \bullet \\
\frac{1}{16}\Big(u_{i_c,j_c-1}^{2h} + 6u_{i_c,j_c}^{2h} + u_{i_c,j_c+1}^{2h} \\
\quad + u_{i_c+1,j_c-1}^{2h} + 6u_{i_c+1,j_c}^{2h} + u_{i_c+1,j_c+1}^{2h}\Big), & \bullet \\
\frac{1}{16}\Big(u_{i_c-1,j_c}^{2h} + 6u_{i_c,j_c}^{2h} + u_{i_c+1,j_c}^{2h} \\
\quad + u_{i_c-1,j_c+1}^{2h} + 6u_{i_c,j_c+1}^{2h} + u_{i_c+1,j_c+1}^{2h}\Big), & \bullet \\
\frac{1}{4}\Big(u_{i_c,j_c}^{2h} + u_{i_c,j_c+1}^{2h} + u_{i_c+1,j_c}^{2h} + u_{i_c+1,j_c+1}^{2h}\Big), & \circ
\end{cases}
\tag{4.3}
$$

where $(i_c, j_c) \in \Omega^{2h}$. The higher-order interpolation can then be represented by the following stencil notation

$$
[I_{2h}^{h}] = \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_{2h}^{h} \tag{4.4}
$$

The higher-order restriction can be implemented in a matrix-free way as follows.

$$
\begin{aligned}
&I_h^{2h} u_{2i_c-1,2j_c-1}^h = \\
&\frac{1}{64}\Big[u_{2i_c-3,2j_c+1}^h + 4u_{2i_c-2,2j_c+1}^h + 6u_{2i_c-1,2j_c+1}^h + 4u_{2i_c,2j_c+1}^h + u_{2i_c+1,2j_c+1}^h \\
&+ 4u_{2i_c-3,2j_c}^h + 16u_{2i_c-2,2j_c}^h + 24u_{2i_c-1,2j_c}^h + 16u_{2i_c,2j_c}^h + 4u_{2i_c+1,2j_c}^h \\
&+ 6u_{2i_c-3,2j_c-1}^h + 24u_{2i_c-2,2j_c-1}^h + 36u_{2i_c-1,2j_c-1}^h + 24u_{2i_c,2j_c-1}^h + 6u_{2i_c+1,2j_c-1}^h \\
&+ 4u_{2i_c-3,2j_c-2}^h + 16u_{2i_c-2,2j_c-2}^h + 24u_{2i_c-1,2j_c-2}^h + 16u_{2i_c,2j_c-2}^h + 4u_{2i_c+1,2j_c-2}^h \\
&+ u_{2i_c-3,2j_c-3}^h + 4u_{2i_c-2,2j_c-3}^h + 6u_{2i_c-1,2j_c-3}^h + 4u_{2i_c,2j_c-3}^h + u_{2i_c+1,2j_c-3}^h\Big]
\end{aligned}
\tag{4.5}
$$

where $(i_c, j_c) \in \Omega^{2h}$. The higher-order restriction can then be represented by the following stencil notation

$$
[I_h^{2h}] = \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_h^{2h}
\tag{4.6}
$$

4.2.2. Matrix-Free Coarse-Grid Operator

The deflation operator can be applied in a matrix-free manner by leveraging the matrix-free application of its individual components, namely the inter-grid transfer operators and the CSLP preconditioner. The main matrix-vector multiplication of the coarse-grid problem, $y = A_{2h}x$, where x and y denote arbitrary vectors on the coarse grid, is a key operation in the deflation process. Generally, the coarse-grid operator A_{2h} can be built by Galerkin coarsening. However, in this way, one needs to construct the matrices and perform matrix-matrix multiplications, which do not fit our matrix-free framework.

In this section, we discuss various possibilities for implementing the coarse-grid operator in a matrix-free way. As one of the main contributions of this chapter, we propose a novel finite-difference scheme called "Re-discretized scheme from Galerkin coarsening (ReD-Glk)" for the re-discretization of the coarse-grid problem. The ReD-Glk method approximates the results of the Galerkin coarsening approach while overcoming its limitations within the matrix-free framework. We also discuss other approaches, including the straightforward matrix-free Galerkin coarsening operator (str-Glk), the re-discretization approach employing the standard explicit second-order and fourth-order schemes, and a fourth-order compact finite-difference scheme.

Re-discretized Scheme from Galerkin Coarsening (ReD-Glk)

Combining the idea of the Galerkin coarsening and re-discretization approach, we propose to use the stencils from the results of the Galerkin coarsening operator as the finite-difference scheme to perform re-discretization on the coarse grid. In terms of the interior grid points, since the inter-grid operations and the Helmholtz operator can be represented by the so-called stencil notations, it is possible to express the result of $I_h^{2h}A_hI_{2h}^h$ by using a stencil notation.

Remark 4.2.1 (Stencil Operations). *Let $V_h \cong \mathbb{C}^{N^2}$, $V_{th} \cong \mathbb{C}^{N_1^2}$, and $V_{sh} \cong \mathbb{C}^{N_2^2}$ be function spaces on grids with spacing h, th, and sh, respectively, where $t = 2^k$, $k \in \mathbb{Z}$, $s = 2^n$, $n \in \mathbb{N}_0$.*

Assume $A : V_h \to V_{th}$ is a linear operator represented by a stencil,

$$[A] = \left\{a_{\mathbf{p}}\right\}_{\mathbf{p}\in\mathcal{N}_A} \tag{4.7}$$

so that

$$(Au)(\mathbf{i}) = \sum_{\mathbf{p}\in\mathcal{N}_A} a_{\mathbf{p}}\, u(\mathbf{i}+\mathbf{p}), \tag{4.8}$$

where $\mathcal{N}_A \subset \mathbb{Z}^2$ is finite, centre-symmetric, with $a_{-\mathbf{p}} = a_{\mathbf{p}}$. Define the width of the stencil $w_A = \max_{\mathbf{p}\in\mathcal{N}_A} \|\mathbf{p}\|_\infty$.

Similarly, let $B : V_{sh} \to V_h$ be given by

$$[B] = \left\{b_{\mathbf{q}}\right\}_{\mathbf{q}\in\mathcal{N}_B}, \tag{4.9}$$

where $\mathcal{N}_B \subset \mathbb{Z}^2$ is also center-symmetric, with $b_{\mathbf{q}} = b_{-\mathbf{q}}$. The width $w_B = \max_{\mathbf{q}\in\mathcal{N}_B} \|\mathbf{q}\|_\infty$.

*For the composite operator $C = AB : V_{sh} \to V_{th}$, the stencil $[C] = \left\{c_{\mathbf{r}}\right\}_{\mathbf{r}\in\mathcal{N}_C}$ can be obtained by doing a **convolution** between $[A]$ and $[B]$, i.e.,*

$$c_{\mathbf{r}} = ([A] * [B])\big|_{stride\ s} = \sum_{\mathbf{r}\in\mathcal{N}_C} a_{\mathbf{p}}\, b_{s\,\mathbf{r}-\mathbf{p}}. \tag{4.10}$$

where s is the stride of convolution. The width w_C of the composite stencil satisfies $w_C = \left\lfloor \frac{w_A + w_B}{s} \right\rfloor$, where $\lfloor\cdot\rfloor$ is the floor operation.

For A being the Helmholtz operator, we have $\mathcal{N}_A = \{(0,0), (\pm1,0), (0,\pm1)\}$. For B being the restriction operator in Equation (4.6), $\mathcal{N}_B = [-2,2]^2 \cap \mathbb{Z}^2$. Then, stride $s = 2$, and $\mathcal{N}_C = [-1,1]^2 \cap \mathbb{Z}^2$.

We define a diagonal matrix, denoted by $\mathcal{K}(k_{i,j}^2)_h$, characterized by spatially varying wavenumbers as its constituent elements. Recall that the discrete Helmholtz operator A_h can be obtained by subtracting $\mathcal{K}(k_{i,j}^2)_h$ from the Laplacian operator $-\Delta_h$, *i.e.*

$$A_h = -\Delta_h - \mathcal{K}\left(k_{i,j}^2\right)_h \tag{4.11}$$

Laplace Operator If we only consider the internal grid points, using the higher-order deflation vectors I_h^{2h} and I_{2h}^h, we have

$$[I_h^{2h}] * [-\Delta_h] * [I_{2h}^h] =$$
$$\frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_h^{2h} * \frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_h * \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_{2h}^h \tag{4.12}$$

$$\Rightarrow [-\Delta_{2h}] = \frac{1}{(2h)^2}\cdot\frac{1}{256}\begin{bmatrix} -3 & -44 & -98 & -44 & -3 \\ -44 & -112 & 56 & -112 & -44 \\ -98 & 56 & 980 & 56 & -98 \\ -44 & -112 & 56 & -112 & -44 \\ -3 & -44 & -98 & -44 & -3 \end{bmatrix}_{2h} \tag{4.13}$$

Taylor series expansion analysis shows that the finite difference scheme based on the stencil Equation (4.13) is a second-order approximation of 2D Laplacian, that is,

$$-\Delta_{2h}u_{2h} = 4\left[-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \left(\frac{13}{48}\frac{\partial^4 u}{\partial x^4} + \frac{1}{2}\frac{\partial^4 u}{\partial x^2 \partial y^2} + \frac{13}{48}\frac{\partial^4 u}{\partial y^4}\right)(2h)^2\right] + \mathcal{O}(h^4) \tag{4.14}$$

Note that the higher-order deflation vectors enlarge the size of the stencil but maintain the accuracy of finite-difference discretization.

Constant Wavenumber For the diagonal wavenumber matrix, a similar derivation can be made

$$[I_h^{2h}] * [\mathcal{K}(k_{i,j}^2)_h] * [I_{2h}^h] =$$

$$\frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_h^{2h} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & k(i,j)^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_h * \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}_{2h}^h \tag{4.15}$$

where $(i,j) \in \Omega^h$. Note that, for non-constant wavenumber, the result will be a 5×5 stencil, of which each element contains the wavenumber on up to 25 fine grid points. This is complicated to implement and leads to extra work.

In order to obtain a simple stencil, suppose the wavenumber is a constant k, we will have

$$[\mathcal{K}(k^2)_{2h}] = \frac{k^2}{64^2}\begin{bmatrix} 1 & 28 & 70 & 28 & 1 \\ 28 & 784 & 1960 & 784 & 28 \\ 70 & 1960 & 4900 & 1960 & 70 \\ 28 & 784 & 1960 & 784 & 28 \\ 1 & 28 & 70 & 28 & 1 \end{bmatrix}_{2h} \tag{4.16}$$

Non-Constant Wavenumber The stencil of the wavenumber operator needs to be slightly modified for non-constant wavenumber cases. For each node of the stencil, the wavenumber on the corresponding coarse grid point is used. Then the stencil of the coarse-grid operator from Galerkin coarsening is obtained by

$$[A_{2h}] = [-\Delta_{2h}] - [\mathcal{K}(k_{i_c,j_c}^2)_{2h}] \tag{4.17}$$

Since the computational stencil is extended to 5×5, it is important to note a drawback of using a wider stencil. The memory access pattern for wider stencils becomes less efficient compared to 3×3 stencils, and additional communication and computation overhead will be required. While the wider stencils result in some performance decreases, our approach leverages the benefits of the matrix framework and scalable outer iterations to compensate for these potential drawbacks. The close-to wavenumber independent convergence presented in our work demonstrates that the overall performance is not significantly affected by the use of wider stencils, and the potential improvements in convergence outweigh the additional overhead.

Boundary Conditions Since they are 5×5 stencils, two grid points on the boundary should be considered. We can make use of the ghost grid point. Specifically, we can calculate the value of a ghost point using the method described in Equation (1.44) for Sommerfeld radiation boundary conditions and Equation (1.46) for Dirichlet boundary conditions.

Then, the aforementioned 5×5 stencils will be applicable on the second grid point of the boundary. It should be noted that one can set the wavenumber of the ghost point to zero without making any approximation, which is needed by Equation (4.16). The first point of the boundary can be discretized using the second-order method. However, this simplification on the boundary may result in an extra number of iterations although it may keep the wavenumber-independent convergence, which will be reflected in the numerical results in the next section.

Re-Discretization Using Standard Finite-Difference Schemes (ReD)

An alternative is to obtain the operator A_{2h} by re-discretizing the Helmholtz operator on the coarse grid. Although it leads to the consequence that P_h is no longer a projection, this will not break the convergence in practical application. The natural way to re-discretize is to use the same discretization method as on the fine grid, that is, use the second-order finite difference scheme to discretize the Laplace operator, and then subtract the diagonal matrix of wavenumbers. In this way, we will get the same computational stencil as Equation (1.50) (denoted as **ReD-$\mathcal{O}$2**).

In addition, inspired by high-order deflation vectors [75], one can consider discretizing the Laplace operator using a fourth-order finite difference scheme (**ReD-$\mathcal{O}$4**) for the internal grid points (x_i, y_j), i.e.,

$$\frac{-u_{i-2,j}+16u_{i-1,j}-30u_{i,j}+16u_{i+1,j}-u_{i+2,j}}{12h^2}=\frac{\partial^2 u}{\partial x^2}\left(x_i,y_j\right)+\mathcal{O}(h^4) \tag{4.18}$$

where $u_{i,j}$ denotes the value of function $u(x,y)$ on the grid point (x_i,y_j) (and similar for y-derivative). The complexity of the computational stencils obtained by using the re-discretization approach will be less than that of stencils from Galerkin coarsening. For example, if a fourth-order finite difference scheme is used, the stencil for the coarse-grid operator becomes

$$[A_{2h}]=\frac{1}{12\cdot(2h)^2}\begin{bmatrix} 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -16 & 0 & 0\\ 1 & -16 & 60-k_{2h}^2(i_c,j_c)(2h)^2 & -16 & 1\\ 0 & 0 & -16 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\end{bmatrix}_{2h} \tag{4.19}$$

where $(i_c, j_c) \in \Omega^{2h}$.

As for the boundary, since the stencil is extended to 5×5, more boundary grid points need to be considered. In this chapter, the stencils for the boundary grid points will be changed to that of the second-order re-discretization method.

Additional experiments show that a higher-order re-discretization scheme for the boundaries will not make any improvement in the convergence compared to the second-order method.

Instead of using the explicit central finite-difference schemes, which do not fully use the entire template and the wavenumber information of adjacent grid points, one may think of a compact finite-difference approximation of the Helmholtz operator that has the following 3×3 stencil at grid point (i_c, j_c):

$$[A_{2h}] = \frac{1}{(2h)^2}\begin{bmatrix} a_c & a_s & a_c \\ a_s & a_0 & a_s \\ a_c & a_s & a_c \end{bmatrix} - \begin{bmatrix} b_c & b_s & b_c \\ b_s & b_0 & b_s \\ b_c & b_s & b_c \end{bmatrix} k^2 \tag{4.20}$$

where the coefficients a and b are non-zeros.

The fourth-order finite difference scheme introduced by Singer and Turkel [22] has a similar form as Equation (4.20). From [22], we have

$$a_0 = \frac{10}{3}, \quad a_s = -\frac{2}{3}, \quad a_c = -\frac{1}{6} \tag{4.21}$$

$$b_0 = \frac{2}{3} + \frac{\gamma}{36}, \quad b_s = \frac{1}{12} - \frac{\gamma}{72}, \quad b_c = \frac{\gamma}{144} \tag{4.22}$$

with γ an arbitrary constant. Here we will use $\gamma = 1$. We denote this approach as **ReD-cmp$\mathcal{O}$4**.

As for the boundary, we can use the fourth-order approximations of the Sommerfeld radiation boundary condition given by Erlangga and Turkel [135]. For instance, suppose $u_{0,j}$ is a ghost point on the left of $u_{1,j}$, it can be approximated by

$$u_{0,j} = 2ikh\left(1 - \frac{k^2h^2}{6}\right)u_{1,j} + u_{2,j} \tag{4.23}$$

The relationship between numerical dispersion and order of accuracy plays a crucial role in the performance of solvers for the Helmholtz equation. Turkel *et al.* [23] proposed a sixth-order compact scheme for a non-constant wavenumber and showed the connection between the order of accuracy and numerical dispersion. Therefore, higher-order accuracy is advantageous in terms of efficiency, as it allows for maintaining accuracy with fewer grid points for larger wave numbers. Chen *et al.* [27] investigated minimal dispersion when using the CSLP preconditioner and found that, for certain step sizes and large wavenumbers, a second-order scheme with minimal numerical dispersion may compete with a sixth-order scheme in terms of accuracy. Stolk [29] demonstrated that matching the phase slowness errors between the fine and coarse grid discretization is essential for efficient multigrid convergence. A careful design of the discretization scheme (4.20) and exploring the impact of matching phase slowness errors between fine and coarse grids on the overall convergence of our method could provide valuable insights and potentially lead to further improvements in performance. This avenue of research merits exploration in future studies.

Straightforward Galerkin Coarsening Operator (str-Glk)

The Galerkin coarsening operation can be computed in a matrix-free manner by performing the interpolation, fine-grid Helmholtz operation, and restriction sequentially, as shown in Equation (4.24).

$$\text{Interpolation: } v_{i1} = I_{2h}^{h} v_i \tag{4.24a}$$

$$\text{Fine-grid operator: } v_{i2} = A_h v_{i1} \tag{4.24b}$$

$$\text{Restriction: } \hat{v}_i = I_{h}^{2h} v_{i2} \tag{4.24c}$$

This approach is essentially equivalent to the matrix-based Galerkin coarsening method presented in [75], but it is implemented here in a matrix-free framework. However, it should be noted that this method requires the application of the fine-grid Helmholtz operator A_h at every Krylov iteration on the coarse level, which can be computationally expensive.

4.3. Experimental Results

The numerical experiments are conducted on the Linux supercomputer DelftBlue [127]. In the numerical experiments, the convergence test is based on the l_2-norm of the residual. Unless mentioned otherwise, the preconditioned relative residual is reduced to 10^{-6} by the deflated GMRES algorithm. Since it is a two-level method, it may still be expensive to solve the coarse-grid problem directly. The (CSLP-preconditioned) GMRES is employed for approximating the inverse of the coarse-grid operator. The (preconditioned) relative residuals on the coarse level are reduced to a certain tolerance. It is important to note that GMRES has the disadvantage of increasing the cost of each iteration as the algorithm proceeds. This can be a concern when a large number of iterations is required, as observed in some of our numerical experiments. We have explored the use of alternative solvers, such as Bi-CGSTAB, for the coarse-grid system. However, our experiments have shown that Bi-CGSTAB often requires significantly more iterations than GMRES to converge and may even diverge for model problems with large wavenumbers. Thus, a full GMRES solver is employed for the inner iterations to ensure stability and to enable a fair comparison with the exact coarse-grid solver. Moreover, to study the convergence properties of the deflation-preconditioned Krylov-subspace method for solving the Helmholtz problems, we aimed to obtain the approximation of A_{2h}^{-1} in the deflation process as accurately as possible. Therefore, we set the tolerance for the coarse-grid solver to 10^{-12}. Furthermore, we will explore the optimal tolerance for the coarse-grid solver in our study.

The inverse of the CSLP preconditioner is approximated by a multigrid V-cycle. According to [124], the stopping criterion for the coarsest grid preconditioner solver should be 2-3 orders of magnitude smaller than the stopping criterion for the outer iteration. Thus, a distinct full-GMRES is used to reduce the relative residuals to 10^{-8} on the predefined coarsest level of the V-cycle.

This section will mainly illustrate the numerical performance of our solver for the model problems on DelftBlue. The number of iterations (denoted as #iter) will be the main metric to estimate the convergence. The speedup and parallel efficiency are used to study the parallel performance. The wall-clock time for the preconditioned Krylov solver to reach the stopping criterion is denoted as t_w. The speedup S_p in Equation (3.8) and the parallel efficiency E_p in Equation (3.9) serve as key performance metrics.

4.3.1. Complexity and Spectral Analysis of the Coarse-Grid Operators

Recalling from the previous section, there are the following possible coarse-grid operators in matrix-free:

- **str-Glk**: Straightforward Galerkin coarsening approach
- **ReD-$\mathcal{O}$2**: Re-discretization using the second-order finite-difference scheme
- **ReD-$\mathcal{O}$4**: Re-discretization using the fourth-order finite-difference scheme
- **ReD-cmp$\mathcal{O}$4**: Re-discretization using a scheme derived from the fourth-order compact finite difference of the Helmholtz equation
- **ReD-Glk**: Re-discretization using a scheme derived from the Galerkin coarsening approach (A ghost point is included for the second boundary grid point and ReD-$\mathcal{O}$2 is only for the first boundary grid point)

To give a preliminary comparison of the above methods for obtaining coarse-grid operators, we conduct a rough complexity analysis to quantify the FLOPs for performing the coarse-grid operator $y = A_{2h}x$ by different methods.

Remark 4.3.1 (FLOPs of sparse matrix-vector multiplication)**.** *Given $x \in \mathbb{R}^p$ and sparse $A \in \mathbb{R}^{p\times p}$, the number of non-zero elements (nnz) of each row is q, then the upper bound of total FLOPs for Ax is $2pq$*

Suppose the number of fine grid points is N and that of the coarse grid is M, if the matrices are constructed explicitly, we will have $A_h \in \mathbb{R}^{N\times N}$, $A_{2h} \in \mathbb{R}^{M\times M}$, $I_h^{2h} \in \mathbb{R}^{M\times N}$ and $I_{2h}^h \in \mathbb{R}^{N\times M}$. In Table 4.1, we list the nnz of the relevant matrices and the approximate FLOPs of $A_{2h}x$ for different methods. Since $4M \approx N$, the total FLOPs of the straightforward Galerkin coarsening operator is around $162M$. To verify the results of the complexity analysis, we also test the elapsed CPU time for performing the coarse-grid operators derived from the Galerkin coarsening approach. We observe that the ratios between the elapsed CPU time are fairly consistent with the estimated FLOPs. From the complexity analysis of the coarse-grid operator, without considering the overall convergence characteristics, the re-discretization approach seems to be preferable in the frame of matrix-free implementation.

To better understand the behavior of the proposed re-discretization approaches, we have performed a numerical spectral analysis for the coarse-grid operators obtained by different re-discretization approaches as well as the fine-grid operator. For problem MP-1a with wavenumber $k = 50$ and $kh = 0.625$, we numerically solve for the eigenvalues of the coarse-grid operators in ascending order and find the index j_{min}, where

Table 4.1: Upper bound on the number of non-zero elements of each relevant matrix and the approximate FLOPs as well as the elapsed CPU time for performing the coarse-grid operator 100 times. The coarse grid size is 7681×7681.

Methods	str-Glk			ReD-$\mathscr{O}$2	ReD-$\mathscr{O}$4	ReD-cmp$\mathscr{O}$4	ReD-Glk
Operators	I_h^{2h}	A_h	I_{2h}^h	A_{2h}	A_{2h}	A_{2h}	A_{2h}
Max. nnz per row	25	5	9	5	9	9	25
FLOPs	$50M + 10N + 18N$			$10M$	$18M$	$18M$	$50M$
CPU time	587.95			29.49	42.30	43.62	187.37

the magnitude of eigenvalues $|\lambda^{j_{min}}(A)|$ is the smallest. As shown in Table 4.2, one can compare the index of the near-zero eigenvalues of the coarse-grid operators with the fine-grid operator to investigate their alignment. The results demonstrate that the proposed coarsening approaches effectively align the near-singular eigenmodes of the fine-grid and coarse-grid operators. This alignment is crucial for preventing the reappearance of near-zero eigenvalues and ensuring the effectiveness of the two-level deflation preconditioner.

Table 4.2: The index of the near-zero eigenvalues for the fine- and coarse-grid operators. MP-1a with wavenumber $k = 50$ and $kh = 0.625$.

	A_h	A_{2h}				
Methods	FD-$\mathscr{O}$2	str-Glk	ReD-$\mathscr{O}$2	ReD-$\mathscr{O}$4	ReD-cmp$\mathscr{O}$4	ReD-Glk
$j_{\min}$	186, 187	186, 187	206, 207	188, 189	184, 185	186, 187

4.3.2. Scalable Convergence

As we introduced, in the context of solving the Helmholtz problem using iterative solvers, the number of iterations required increases significantly as the wavenumber increases. A recent study by Dwarka and Vuik [75] has shown promising results in achieving convergence properties that are independent of the wavenumber using high-order deflation preconditioning methods. In their work, the Galerkin coarsening approach is employed to derive the coarse-grid operator. In this subsection, we further explore various scenarios utilizing the re-discretization method for the coarse-grid operator to investigate which approach can achieve wavenumber-independent convergence. By examining different cases and comparing the outcomes, we aim to provide insights into the effectiveness of these methods.

Low-Order Deflation

To observe the possible impact on the convergence behavior of using the re-discretization approach, we first compare the convergence behavior of low-order deflation using the straightforward Galerkin coarsening and second-order re-discretization (ReD-$\mathscr{O}$2) approaches. As mentioned, regardless of the computational cost, we can obtain the coarse-grid operator as eq. (4.24) in the frame of the Galerkin coarsening approach. Tables 4.3 and 4.4 present the number of outer iterations and wall-clock time for solving MP-1a using A-DEF1 preconditioned GMRES with varying kh, comparing straightforward Galerkin coarsening and ReD-$\mathcal{O}2$ approaches. One can find the number of outer iterations is consistent with the results in [72] when the straightforward Galerkin coarsening approach is used. In parentheses is the approximate number of iterations required to solve the coarse-grid problem (CGP) once by (preconditioned) GMRES. The coarse-grid problem, inheriting properties from the original Helmholtz problem, suffers from slow convergence without preconditioning. Applying the CSLP preconditioner to the coarse-grid solver accelerates convergence at the coarse level, as evidenced by a reduction in inner iterations, while maintaining the same number of outer iterations. The wall-clock time reflects that for large wavenumber problems, one can save significant computational cost by employing the CSLP preconditioner to solve the coarse-grid system. The idea of applying a recursive strategy, where another two-level deflation is employed on the coarse level, is highly promising and aligns with our ongoing research. The scope of this chapter is limited to the development and benchmarking of the parallel matrix-free two-level deflation approach, which provides a solid foundation for our work on multilevel extensions.

Table 4.3: The number of iterations and wall-clock time t_w required to solve MP-1a using low-order A-DEF1 preconditioned GMRES for different k and kh. The coarse-grid problem (CGP) obtained by the **str-Glk approach** is solved by full-GMRES (GMRES) and CSLP preconditioned GMRES (CSLP-GMRES). In parentheses is the number of iterations required to solve the coarse grid problem once.

			(GMRES)		(CSLP-GMRES)	
Grid size	k	kh	#iter (coarse)	$t_w(s)$	#iter (coarse)	$t_w(s)$
33 × 33	20	0.625	8 (55)	0.03	8 (54)	0.14
65 × 65	40	0.625	16 (257)	1.14	16 (247)	2.09
129 × 129	80	0.625	39 (1058)	187.07	44 (903)	191.16
65 × 65	20	0.3125	6 (131)	0.14	6 (74)	0.24
129 × 129	40	0.3125	9 (527)	11.61	9 (256)	3.54
257 × 257	80	0.3125	18 (2217)	1285.90	19 (918)	325.04

Table 4.4: The number of iterations and wall-clock time t_w required to solve MP-1a using low-order A-DEF1 preconditioned GMRES for different k and kh. The coarse-grid problem (CGP) obtained by **ReD-$\mathcal{O}$2 approach** is solved by full-GMRES (GMRES) and CSLP preconditioned GMRES (CSLP-GMRES). In parentheses is the number of iterations required to solve the coarse grid problem once.

			(GMRES)		(CSLP-GMRES)	
Grid size	k	kh	#iter (coarse)	$t_w(s)$	#iter (coarse)	$t_w(s)$
33 × 33	20	0.625	21 (82)	0.06	21 (61)	0.33
65 × 65	40	0.625	40 (328)	4.64	41 (251)	5.33
129 × 129	80	0.625	97 (1609)	1343.97	97 (1125)	767.11
65 × 65	20	0.3125	17 (123)	0.32	17 (59)	0.41
129 × 129	40	0.3125	20 (660)	31.37	20 (209)	5.53
257 × 257	80	0.3125	25 (2645)	2037.38	29 (840)	327.73

Table 4.5: The number of iterations and wall-clock time t_w required to solve the 2D wedge problem using low-order A-DEF1 preconditioned GMRES for different frequencies f and grid size. The coarse-grid problem (CGP) obtained by str-Glk or ReD-$\mathcal{O}2$ approach is solved by CSLP preconditioned GMRES. In parentheses is the number of iterations required to solve the coarse grid problem once.

			str-Glk		ReD-$\mathcal{O}2$	
Grid size	f	kh	#iter (coarse)	$t_w(s)$	#iter (coarse)	$t_w(s)$
73 × 121	10	0.35	8 (112)	2.91	24 (104)	5.93
145 × 241	20	0.35	9 (278)	19.15	31 (242)	43.88
289 × 481	40	0.35	12 (629)	348.25	34 (547)	642.49
577 × 961	80	0.35	19 (1448)	13079.7	41 (1220)	15674.43
145 × 241	10	0.175	7 (108)	4.57	25 (107)	12.28
289 × 481	20	0.175	7 (257)	54.35	31 (245)	162.96
577 × 961	40	0.175	8 (568)	829.33	33 (533)	2307.76

As for the second-order re-discretization approach, Table 4.4 shows that the second-order re-discretized A_{2h} leads to an increase in the outer iterations. Table 4.5 confirms that our matrix-free two-level deflation methods also work for the case with a non-constant wavenumber and exhibit similar convergence behavior to the constant-wavenumber case.

A noteworthy observation from our results is the relationship between kh and the number of outer iterations when the wavenumber is fixed. The straightforward Galerkin coarsening approach (str-Glk), which is essentially equivalent to the matrix-based method, aligns with the established trend of fewer iterations for smaller kh values, as corroborated by Sheikh *et al.* [72]. When the second-order re-discretization approach (ReD-$\mathcal{O}2$) is employed, it consistently applies to the constant wavenumber problem, as shown in Table 4.4. However, for the wedge problem with frequency $f = 10$, 20, 40 Hz in Table 4.5, a smaller kh does not lead to fewer outer iterations. Here we mainly aim to demonstrate the applicability of our matrix-free method to lower-order deflation and the convergence behavior versus the wavenumber, exploring the effects of kh through theoretical analysis and broader numerical experiments to reach a strong conclusion can be a direction for future research.

Higher-Order Deflation

To get scalable convergence with respect to the wavenumber, this section considers using the higher-order deflation vectors, *i.e.* APD.

Table 4.6 shows the number of iterations required to solve the constant wavenumber problem (MP-1b) using APD-preconditioned GMRES. We found that when the str-Glk approach is used for the coarse grid problem, the number of iterations required shows the wavenumber independence for the case with the wavenumber up to 10^3.

When the coarse-grid operator is obtained by the ReD-$\mathcal{O}2$, it requires more outer iterations, which increase significantly with the increase of the wavenumber.

Obtaining the coarse-grid operator by ReD-$\mathcal{O}4$ results in a slightly lower number of outer iterations compared to ReD-$\mathcal{O}2$. Nevertheless, it remains dependent on the wavenumber

As ReD-cmp$\mathcal{O}$4 has the same order of accuracy as ReD-$\mathcal{O}$4, little difference in the number of outer iterations is observed. The compact scheme (ReD-cmp$\mathcal{O}$4) offers advantages over the standard scheme (ReD-$\mathcal{O}$4) in terms of memory access and computational efficiency due to its compact stencil. The compact scheme significantly reduces the number of iterations required to solve the coarse grid problem, which is a notable benefit considering that the inner solver accounts for a significant portion of the computational cost.

As for using ReD-Glk, we can obtain close to wavenumber independence. The number of outer iterations for using ReD-Glk is slightly more than that of str-Glk. This is due to the simplified treatment for the discretization of the boundary conditions mentioned in Section 4.2.2 as the operation on the internal grid points will be consistent with the Galerkin coarsening approach for the constant-wavenumber model.

Table 4.6: The number of iterations required to solve MP-1b using APD-preconditioned GMRES for different k and kh. The coarse-grid operator is obtained by str-Glk, ReD-$\mathcal{O}$2, ReD-$\mathcal{O}$4, ReD-cmp$\mathcal{O}$4 and ReD-Glk, respectively. The coarse-grid problem is solved by CSLP preconditioned GMRES. In parentheses is the number of iterations required to solve the coarse grid problem once.

Grid size	k	kh	str-Glk	ReD-$\mathcal{O}$2	ReD-$\mathcal{O}$4	ReD-cmp$\mathcal{O}$4	ReD-Glk
65 × 65	40	0.625	7 (126)	20 (98)	17 (106)	19 (91)	9 (128)
129 × 129	80	0.625	7 (236)	30 (305)	18 (298)	20 (185)	9 (251)
257 × 257	160	0.625	7 (495)	87 (731)	19 (650)	23 (362)	9 (585)
513 × 513	320	0.625	7 (1030)	319 (1539)	23 (1330)	28 (690)	10(1276)
1025×1025	640	0.625	8 (2089)	1099 (3193)	34 (2662)	44 (1328)	11 (2590)
2049×2049	1280	0.625	9 (4230)	3417 (6750)	79 (5315)	109 (2660)	13 (5266)
129 × 129	40	0.3125	5 (142)	18 (76)	18 (81)	18 (76)	7 (144)
257 × 257	80	0.3125	5 (228)	19 (205)	18 (212)	18 (188)	7 (231)
513 × 513	160	0.3125	5 (411)	21 (438)	18 (458)	19 (395)	7 (432)
1025 × 1025	320	0.3125	5 (809)	28 (885)	20 (909)	20 (775)	6 (859)
2049 × 2049	640	0.3125	5 (1630)	53 (1722)	23 (1763)	24 (1506)	6 (1690)

Table 4.7: The number of iterations required to solve the 2D wedge problem using APD-preconditioned GMRES for different frequencies f and grid size. The coarse-grid operator is obtained by str-Glk, ReD-$\mathcal{O}$2, ReD-$\mathcal{O}$4, ReD-cmp$\mathcal{O}$4 and ReD-Glk, respectively. The coarse-grid problem is solved by CSLP preconditioned GMRES. In parentheses is the number of iterations required to solve the coarse grid problem once.

Grid size	f	kh	str-Glk	ReD-$\mathcal{O}$2	ReD-$\mathcal{O}$4	ReD-cmp$\mathcal{O}$4	ReD-Glk
73 × 121	10	0.349	7 (145)	22 (104)	22 (108)	22 (99)	9 (138)
145 × 241	20	0.349	6 (301)	28 (244)	27 (243)	28 (228)	9 (303)
289 × 481	40	0.349	6 (580)	31 (535)	29 (519)	30 (491)	9 (585)
577 × 961	80	0.349	6 (1206)	37 (1200)	30 (1175)	31 (1069)	9 (1255)
1153 × 1921	160	0.349	6 (2799)	58 (2826)	34 (2721)	35 (2353)	8 (3032)

For the non-constant wavenumber problem, the results for solving the so-called wedge problem and the heterogeneous Marmousi problem are given in Table 4.7 and Table 4.8, respectively. The convergence behaviors exhibited by using different coarse-grid operators are consistent with solving the constant wavenumber problem. For heterogeneous model problems, one can find that ReD-cmp$\mathcal{O}4$ and ReD-$\mathcal{O}4$ exhibit comparable performance. Thus, we will only present the results for ReD-$\mathcal{O}4$ in the subsequent numerical experiments. It is worth mentioning that the stencil eq. (4.16) in ReD-Glk is obtained based on the constant wavenumber assumption. By applying the stencil to the coarse-grid problem, where heterogeneity from the original problem is preserved through restriction, the near-zero eigenvalues of the coarse-grid operator maintain their positions with small shift. Consequently, this approach demonstrates wavenumber independence even when addressing non-constant wavenumber problems. This is consistent with the work of Dwarka and Vuik [75].

In terms of time consumption, see the wall-clock time required to solve the Marmousi problem in Table 4.8, we found that ReD-Glk stands out as the most efficient choice among the re-discretization approaches and is comparable to that of the str-Glk method. Considering the superior complexity of the ReD-Glk method in terms of performing the coarse-grid operator one time, it will be cheaper than the str-Glk method when generalized to the potential multilevel deflation method. Thus, we consider ReD-Glk as the optimal coarse-grid re-discretization scheme. In subsequent numerical experiments, unless otherwise specified, ReD-Glk will be the default method.

Table 4.8: The number of iterations and wall-clock time required to solve the 2D Marmousi problem using APD-preconditioned GMRES for different frequencies f and grid size ($kh = 0.5236$) with 12 processors. The coarse-grid operator is obtained by str-Glk, ReD-$\mathcal{O}2$, ReD-$\mathcal{O}4$, ReD-cmp$\mathcal{O}4$ and ReD-Glk, respectively. The coarse-grid problem is solved by CSLP preconditioned GMRES. In parentheses is the number of iterations required to solve the coarse grid problem once.

Grid size		737×241	1473×481	2945×961
f(Hz)		10	20	40
	#iter	7	7	7
str-Glk	(coarse)	(775)	(1858)	(5380)
	t_w (s)	120.67	1039.96	27496.67
	#iter	38	71	>200[1]
ReD-$\mathcal{O}2$	(coarse)	(748)	(1988)	(>5700)
	t_w (s)	615.54	9606.14	>432000
	#iter	30	34	50
ReD-$\mathcal{O}4$	(coarse)	(762)	(1947)	(5484)
	t_w (s)	402.79	4386.57	173650.06
	#iter	32	37	60
ReD-cmp$\mathcal{O}4$	(coarse)	(762)	(1947)	(5484)
	t_w (s)	316.53	5128.28	175379.27
	#iter	10	10	11
ReD-Glk	(coarse)	(802)	(1923)	(5592)
	t_w (s)	164.79	1376.40	46008.03

[1] “>" indicates it does not converge to the specified tolerance within the allowed maximum wall-clock time.

4.3.3. Efficient Coarse-Grid Solver

It appears that using the ReD-Glk format for coarse-grid re-discretization leads to near wavenumber-independent convergence. Despite this, a large number of iterations is still needed for the coarse-grid problem, especially for large wavenumbers. This is partially due to the fact that we are currently employing a two-level deflation method. Furthermore, as mentioned earlier, a strict tolerance is used for the convergence test of the coarse-grid solver. Consequently, this section explores the range of possible values for the tolerance of the coarse-grid solver.

Using higher-order deflation vectors and ReD-Glk for re-discretization of the coarse-grid system, the present APD-preconditioned GMRES is utilized to solve the linear system ($A_h u_h = b_h$), reducing the preconditioned relative residual to 10^{-6}. The stopping criteria for the coarse-grid iterative solver ($A_{2h} v_2 = v_1$) ranged from 10^{-1} to 10^{-13}. Table 4.9 shows that, for various model problems, the number of outer iterations required remains constant for all convergence criteria smaller than 10^{-2} on the coarse level, which is in line with the results in [82].

Table 4.9: The number of APD-preconditioned GMRES iterations required to solve corresponding model problems when using different stopping criteria for the coarse-grid iterative solver

Model Problems	10^{-1}	10^{-3}	10^{-5}	10^{-7}	10^{-9}	10^{-11}
MP-1b, $k = 320$, 513×513	14	10	10	10	10	10
MP-1b, $k = 320$, 1025×1025	11	7	6	6	6	6
Wedge, $f = 40\,\text{Hz}$, 289×481	12	9	9	9	9	9
Marmousi, $f = 20\,\text{Hz}$, 1473×481	13	10	10	10	10	10

For example, in the scenario of the Marmousi problem ($f = 20\,\text{Hz}$, grid size 1473×481), Figure 4.2 displays the change in iterations required to solve a single coarse-grid problem with the CSLP-preconditioned GMRES solver, for various tolerances, along with the total wall-clock time and the relative residual of the final solution. Note that the number of iterations for the CSLP-preconditioned GMRES solver to solve a single coarse-grid problem increases linearly as the size of the tolerance decreases. The wall-clock time elapsed increases even faster as the tolerance becomes more stringent. Once the tolerance exceeds 10^{-5}, the relative residual of the final solution increases, while it remains constant for the tolerance less than 10^{-6}. This pattern holds for larger grid sizes.

The findings suggest that, when the present APD-preconditioned GMRES is used, maintaining the tolerance of the coarse-grid solver at the same order of magnitude as that of the outer iterations enables a consistent number of outer iterations and constant relative residual of the final solution while minimizing the total wall-clock time. It is worth noting that for the GMRES algorithm, we employ left preconditioning and use the preconditioned relative residual as the criterion for convergence. Additional numerical experiments have been conducted to investigate the use of right preconditioning with GMRES, using the unpreconditioned relative residual as the convergence criterion. The results also demonstrate a consistent number of outer iterations across various tolerances for the coarse-grid solver, with only one more iteration compared

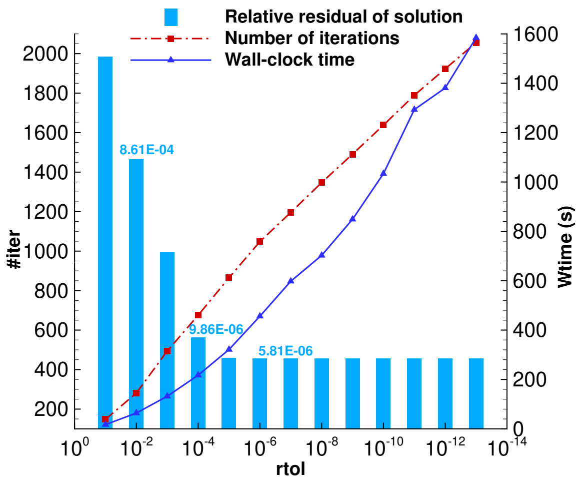


Figure 4.2: Tolerance of the coarse-grid solver for Marmousi problem with $f = 20\,\text{Hz}$ and grid size 1473×481. The present APD-preconditioned GMRES is employed for outer iteration.

to the results shown in Table 4.9, which strictly reduces the final relative residual to below 10^{-6}. Nevertheless, we can reach the same conclusion regarding the impact of the coarse-grid solver tolerance on the outer iterations and the relative residual of the solution. We attribute this to that the standard GMRES algorithm for outer iteration expects a constant preconditioner. However, if the tolerance of the coarse-grid solver increases beyond a certain threshold, it leads to variable preconditioning.

The results presented in Figure 4.2 indicate that solving the coarse-grid problem for the Marmousi model with grid size 1473×481 and frequency $f = 20\,\text{Hz}$ requires over 1000 GMRES iterations to achieve a relative residual of 10^{-6}.

To investigate the possibility of using a larger tolerance for the coarse-grid solver and further reduce the solution time, we will proceed with the outer iteration using the GCR algorithm, which allows for a variable preconditioner. While making this transition, all other aspects of the deflation method remain unchanged, except for the use of the GCR algorithm with right preconditioning.

For the Marmousi model with grid size 1473×481 and frequency $f = 20\,\text{Hz}$, the number of outer GCR iterations remains constant at 11, when varying the tolerances of the coarse-grid solver from 10^{-2} to 10^{-12}. If the tolerance of the coarse-grid solver is set to 10^{-1}, it requires 12 outer iterations. The impact of the coarse-grid solver tolerance on the number of iterations required to solve the coarse-grid problem, the total wall-clock time, and the relative residual of the final solution is illustrated in Figure 4.3. The variation in the number of iterations for solving the coarse-grid problem and the total wall-clock time is similar to that shown in Figure 4.2. Similarly, when the tolerance of the coarse-grid solver is smaller than the tolerance of the outer iterations, the residual of the solution remains unchanged. However, when

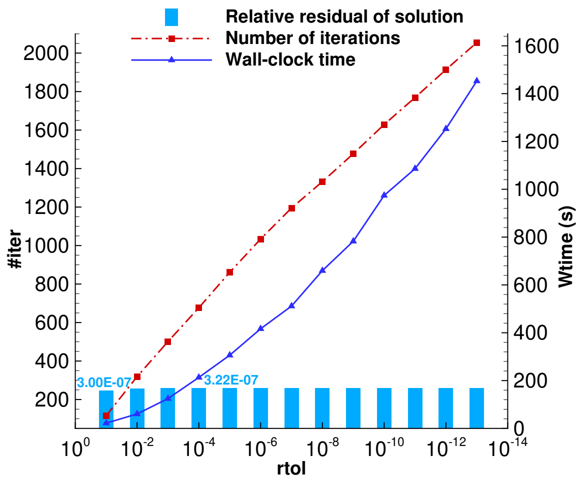


Figure 4.3: Tolerance of the coarse-grid solver for Marmousi problem with $f = 20\,\text{Hz}$ and grid size 1473×481. The present APD-preconditioned GCR is employed for outer iteration.

the tolerance of the coarse-grid solver exceeds the tolerance of the outer iterations, the residual of the solution remains within the tolerance instead of increasing, as observed in the standard GMRES method. The same behavior is observed in the case of the non-constant wavenumber model problems. Thus, using the GCR algorithm as the outer iteration method allows us to set the tolerance of the coarse-grid solver to a relatively large value 10^{-1}, while still maintaining the desired accuracy of the final solution. For this MP-1b model problem, if the APD-preconditioned GMRES is employed and the tolerance for the coarse-grid solver is set to 10^{-6}, the total wall-clock time is 145.41 s. If the APD-preconditioned GCR is used and the tolerance for the coarse-grid solver is set to 10^{-1}, although one extra outer iteration is required, the total wall-clock time is 9.48 s. The total computation time is reduced by around 95% of the time required when using a tolerance of 10^{-6} for the coarse-grid solver.

Table 4.10 presents the number of iterations and wall-clock time required to solve the 2D Marmousi problem using the APD-preconditioned GCR method with different coarse-grid operators. The coarse-grid problem is solved using the CSLP-preconditioned GMRES method with a tolerance of 10^{-1}. In comparison to Table 4.8, there is a notable decrease in the number of iterations required for the coarse-grid problem and the total computation time when using a tolerance of 10^{-1}. The proposed ReD-Glk remains the top choice among the re-discretization approaches and now requires a wall-clock time similar to the str-Glk method.

Additional numerical experiments have provided further evidence that, regardless of whether the standard GMRES algorithm is employed with left or right preconditioning as the outer iteration algorithm, if the tolerance of the coarse-grid solver is set larger than that of the outer iterations, the residual of the final solution will increase.

Table 4.10: The number of iterations and wall-clock time required to solve the 2D Marmousi problem using APD-preconditioned GCR for different frequencies f and grid size ($kh = 0.5236$) with 12 processors. The coarse-grid problem is solved by CSLP preconditioned GMRES with a tolerance 10^{-1}. The coarse-grid operator is obtained by str-Glk, ReD-$\mathcal{O}$2, ReD-$\mathcal{O}$4 and ReD-Glk, respectively. In parentheses is the number of iterations required to solve the coarse grid problem once.

Grid size		737×241	1473×481	2945×961
f(Hz)		10	20	40
	#iter	8	9	9
str-Glk	(coarse)	(78)	(205)	(627)
	t_w (s)	2.47	14.00	282.82
	#iter	40	71	233
ReD-$\mathcal{O}$2	(coarse)	(34)	(171)	(373)
	t_w (s)	10.52	98.45	5359.75
	#iter	33	35	41
ReD-$\mathcal{O}$4	(coarse)	(27)	(70)	(162)
	t_w (s)	5.68	28.47	356.37
	#iter	11	12	12
ReD-Glk	(coarse)	(63)	(116)	(489)
	t_w (s)	3.25	14.62	255.38

However, when utilizing the APD-preconditioned Flexible GMRES algorithm, which also allows for variable preconditioning, similar to the APD-preconditioned GCR approach, even with a relatively larger tolerance for the coarse-grid solver, the accuracy of the final solution can still be maintained within the specified tolerance range.

In summary, while solving the coarse-grid problem exactly is expected to provide the best convergence in the deflation method [136], the modified projection method employed in this study, as proposed by Erlangga and Nabben [83], is less sensitive to approximations of the coarse-grid system. Within a certain range of tolerances, the outer iterations do not change significantly. However, using a less strict tolerance of 10^{-1} does result in 2-3 extra outer iterations. Despite this, the computational savings achieved by solving the coarse-grid system with a looser tolerance outweigh the cost of the additional outer iterations. In the subsequent parallel performance study of the parallel preconditioned GMRES and GCR, we will set the tolerance for the coarse-grid problem solver to 10^{-6} and 10^{-1}, respectively.

To further reduce the number of iterations required to solve the coarse-grid problem, different complex shift values (β_2) for the CSLP preconditioner are investigated. Figure 4.4 illustrates the impact of the complex shift of the CSLP preconditioner for solving the coarse-grid system. The results demonstrate that a shift larger than 0.5 is satisfactory, with a larger shift generally leading to a decrease in the number of iterations required to solve the coarse-grid system. This is because one multigrid V-cycle is employed to approximate the CSLP preconditioner. This observation aligns with the analysis presented in [54]. It provides a valuable insight that a shift of 1.0 for the CSLP preconditioner will save some iterations for solving the coarse-grid problem when we use the proposed matrix-free parallel two-level deflation method to solve more complex problems with larger wavenumber.

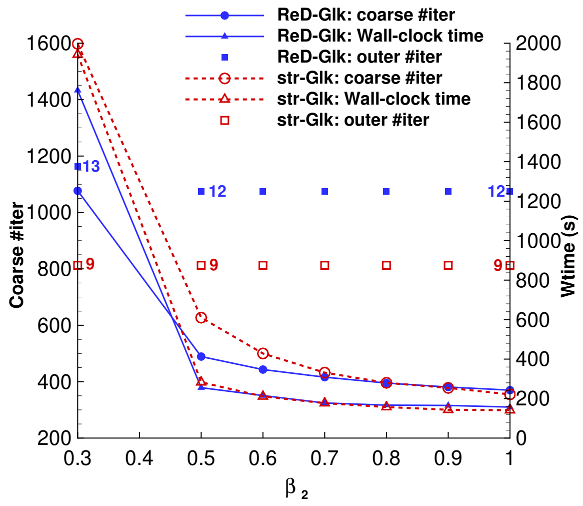


Figure 4.4: Effect of the complex shift β_2. Marmousi problem with $f = 40$ Hz and grid size 2945×961. The present APD-preconditioned GCR is employed for outer iteration.

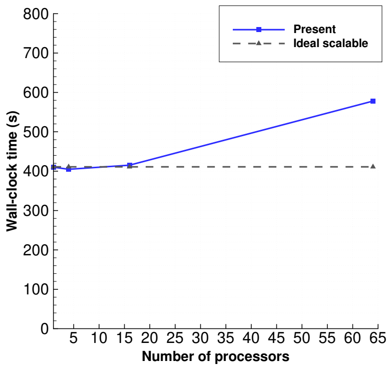
4.3.4. Parallel Performance

In this section, we conduct a comprehensive evaluation of the parallel performance of our solver, focusing on weak scaling and strong scaling. Weak scaling refers to the solver's performance when the problem size and the number of processing elements increase proportionally. This allows us to assess our parallel solver's ability to solve large-scale heterogeneous Helmholtz problems with minimized pollution error by using smaller step size h. On the other hand, strong scaling measures the capability of our parallel solution method to solve a fixed problem size more quickly by adding more resources. Through these scaling assessments, we aim to understand the effectiveness of our solver in handling various problem sizes and resource allocations, providing crucial insights into its optimal operational parameters and potential scalability limits.

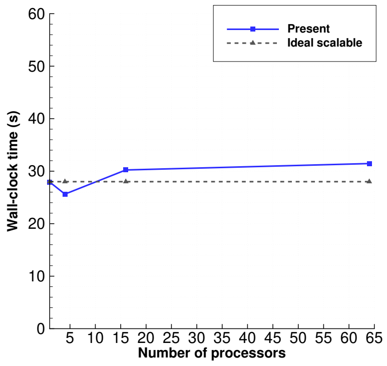
Weak Scaling

Figure 4.5 shows the results for the weak scalability test solving the MP-1b model problem with $k = 160$ from 1, 4, 16, up to 64 processes. The problem size was refined from 1025×1025, 2049×2049, 4097×4097, up to 8193×8193, ensuring that each process handled a grid size of approximately 1024×1024. One can find that, for the same problem and with the same number of processes, the APD-preconditioned GCR method with a coarse-grid solver tolerance of 10^{-1} exhibits a significantly reduced computational time compared to the APD-preconditioned GMRES method with a coarse-grid solver tolerance of 10^{-6}.

In Figure 4.5, it appears that as the grid size and the processes increase proportionally, the required wall-clock time does not remain perfectly constant but tends to increase slightly. This behavior may be attributed to the additional communication time required for larger problems and more processes. Additionally, the limited bandwidth also contributes to some increase in the wall-clock time as the problem size grows larger. The same trend holds for model problems with non-constant wavenumbers, as shown in Table 4.11. The present weak scalability meets our requirements for minimizing pollution error by grid refinement within a certain range.



(a) APD-preconditioned GMRES



(b) APD-preconditioned GCR

Figure 4.5: Weak scaling for MP-1b with $k = 160$ and a grid size of 1024×1024 per process.

Table 4.11: Weak scaling for model problems with non-constant wavenumber. The present APD-preconditioned GMRES/GCR is employed for outer iteration.

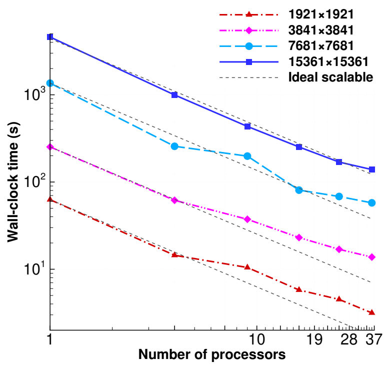
		GMRES		GCR	
grid size	np	#iter	t_w (s)	#iter	t_w (s)
		Wedge, $f = 40\,\text{Hz}$			
577 × 961	6	9 (251)	53.41	10 (46)	4.86
1153 × 1921	24	10 (259)	68.59	10 (43)	5.75
		Marmousi, $f = 10\,\text{Hz}$			
737 × 241	3	10 (414)	103.92	11 (63)	10.55
1473 × 481	12	9 (378)	111.20	10 (58)	12.08
2945 × 961	48	9 (383)	156.45	10 (58)	17.72

Strong Scaling

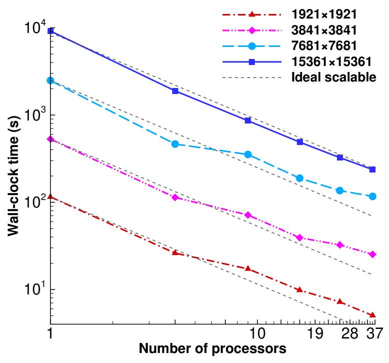
We are also interested in the strong scalability properties of the present parallel deflation method for the Helmholtz problems. First of all, numerical experiments show that the number of external iterations required is found to be independent of the number of processes, which is a favorable property of our solution method.

We consider the MP-1b problem with a wavenumber of $k = 100$ and $k = 200$ on a growing number of processes and grid size. Figure 4.6 presents the wall-clock time versus the number of processes. The numerical experiment conducted on close to 48 processes exhibits a moderate decrease in terms of parallel efficiency. This can be partly attributed to the increased amount of communication, resulting in a significant decrease in the computation/communication ratio. If we increase the grid size, an improvement in parallel efficiency can be observed. This is because the number of ghost-grid layers used for communication remains constant, the amount of data to be communicated doubles when the number of grid points doubles in each direction. In contrast, the total number of grid points increases fourfold, resulting in a larger ratio of computational and communication time. For larger problems, a slight decrease in parallel efficiency when approaching 48 processes can be explained by the limited memory access bandwidth of a single compute node.

Furthermore, we also observe that the computational time increases nearly proportionally to the grid size, approximately by a factor of 4-5. However, the additional computational time required for data communication becomes increasingly evident. From Figure 4.6, one can find that similar strong scalability holds for a larger wavenumber.



(a) $k = 100$



(b) $k = 200$

Figure 4.6: Strong scaling of APD-preconditioned GCR for MP-1b with various grid sizes.

The strong scalability property of the present solver on multiple compute nodes is also investigated. Figure 4.7 illustrates the time required to solve the wedge model problem with a frequency of $f = 40\,\text{Hz}$ on at most 6 compute nodes (a total of 384 processes). For a grid size of 2305×3841, the numerical experiment performed on more than 96 processes exhibits a decrease in parallel efficiency. However, as the grid size increases to 4609×7681 and 9217×15361, which consequently increases the computation/communication ratio, the parallel efficiency significantly improves, with speedup even surpassing the ideal case, i.e., superlinear speedup. When parallel computing tasks can fully utilize the caches on multiple compute nodes, data access speeds are faster, thus enhancing computational efficiency. For solving the wedge problem with a higher frequency of $f = 100\,\text{Hz}$, as depicted in Figure 4.7, we observe similar patterns to $f = 40\,\text{Hz}$. Despite the constant number of outer iterations, the number of iterations required to solve the coarse-grid problem increases from around 45 to around 110. Consequently, the computation time also increases by a factor of 2-3. In terms of parallel efficiency, we observe similar behavior.

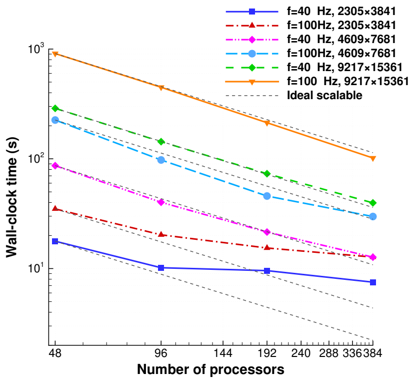


Figure 4.7: Strong scaling of APD-preconditioned GCR for wedge problem with $f = 40\,\text{Hz}$ and $f = 100\,\text{Hz}$.

4.4. Conclusions

A matrix-free parallel two-level deflation preconditioner for solving two-dimensional Helmholtz problems in both homogeneous and heterogeneous media is proposed in this chapter.

By leveraging the matrix-free parallel framework and geometric multigrid-based CSLP built in previous chapters, we provide a matrix-free parallelization method for the general two-level deflation preconditioner for the Helmholtz equation discretized with finite differences. Numerical experiments demonstrate that, compared with the Galerkin coarsening approach, the method of using re-discretization to obtain the coarse-grid operator of deflation will slow down convergence to some extent.

To enhance the convergence properties, the higher-order approximation scheme proposed by Dwarka and Vuik [75] is employed to construct the deflation vectors. The study presents higher-order deflation vectors in two dimensions and their matrix-free parallel implementation. Various methods for implementing matrix-free coarse-grid operators are proposed and compared. The numerical experiments demonstrate the effectiveness of the proposed coarse-grid re-discretization scheme based on the Galerkin coarsening approach, which achieves wavenumber-independent convergence for both constant and non-constant wave number model problems in accordance with objectives (i) and (ii) in Section 1.7.

Furthermore, we have studied in detail the tolerance setting of the coarse grid solver when the deflation preconditioned GMRES type methods are employed to solve the

Helmholtz problems. An optimal tolerance for the coarse grid solver can effectively reduce the solution time.

Finally, the performance of the present parallel GCR solver preconditioned by higher-order deflation is studied, including both weak and strong scalability. Using a matrix-free approach to reduce memory requirements and weak scalability allows us to minimize pollution errors by refining the grid. Strong scalability allows us to solve high-frequency heterogeneous Helmholtz problems faster and more effectively in parallel. The presented two-level preconditioner serves as a benchmark for the development of a matrix-free parallel multilevel deflation method in the next Chapter to achieve objective (iii).

5 Matrix-Free Parallel Scalable Multilevel Deflation Preconditioning

This chapter presents an integration and extension of our previous developments, combining the matrix-free parallel framework with an advanced multilevel extension of the two-level deflation method. Higher-order deflation vectors and re-discretization schemes derived from the Galerkin coarsening approach are employed for a matrix-free parallel implementation. We suggest a robust and efficient configuration of the matrix-free multilevel deflation method, which yields a close to wavenumber-independent convergence and good time efficiency. Numerical experiments demonstrate the effectiveness of our approach for increasingly complex model problems. The matrix-free implementation of the preconditioned Krylov subspace methods reduces memory consumption, and the parallel framework exhibits satisfactory parallel performance and weak parallel scalability. This work represents a significant step towards developing efficient, scalable, and parallel multilevel deflation preconditioning methods for large-scale real-world applications in wave propagation.

Parts of this chapter have been published in Journal of Scientific Computing **102**, 47 (2025) [137].

Although conventional multigrid methods using standard smoothing and coarse grid corrections fail for the Helmholtz equation, their high efficiency in solving positive definite problems has motivated research into developing robust multilevel approaches for this equation [76, 138–140]. In this chapter, we explore methods to extend the wavenumber-independent convergence from the two-level to a multilevel setting based on the work in previous chapters. While previous works have established the theoretical foundations for multilevel deflation methods [72, 73, 76, 82], this chapter's focus is on a parallel scalable implementation of multilevel higher-order deflation for practical large-scale applications. We aim to perform comprehensive numerical experiments to validate the theoretical predictions and demonstrate the method's effectiveness in large-scale scenarios, where parallel implementation challenges often exceed idealized theoretical assumptions. This chapter presents significant innovations in solving large-scale Helmholtz problems. We develop novel re-discretization schemes for multilevel hierarchies, ensuring effective approximation of Galerkin coarsening operators across all levels while maintaining the matrix-free parallel framework. Through comprehensive numerical experiments, we establish appropriate parameters for robust convergence across different problem scales. Furthermore, we introduce an optimal tolerance for coarse-level iterations, a previously unexplored but essential component for achieving wavenumber-independent convergence in practical multilevel deflation methods. These innovations culminate in a highly efficient parallel framework that demonstrates both wavenumber-independent convergence and excellent scaling properties in massively parallel environments, as validated by extensive numerical experiments.

The mathematical model and discretization schemes employed in this chapter are the same as those those presented in Chapter 4. We consider the two-dimensional Helmholtz equation with both Dirichlet and Sommerfeld radiation boundary conditions in heterogeneous media.

This chapter is organized as follows. We present the matrix-free parallel variant of the multilevel deflated Krylov method in Section 5.1. Section 5.2 presents an optimally tuned configuration of the matrix-free parallel multilevel deflation method. Finally, we present numerical results to evaluate parallel performance in Section 5.3. Section 5.4 contains our conclusions.

5.1. Multilevel Deflation

When using the two-level higher-order deflation method in practical large-scale applications, solving the coarse-grid system remains expensive, whether solved exactly [76], or approximately by CSLP-preconditioned Krylov methods [134]. In accordance with the multigrid method, as the coarse grid system has similar properties as the original Helmholtz operator, one can obtain a multilevel framework by applying the two-level cycle recursively, as shown in Algorithm 13. The flexible subspace Krylov method FGMRES preconditioned by two-level deflation is applied recursively on subsequent coarse-grid systems E. Compared to the multilevel deflation proposed in [76], a few remarks are noted here.

First, Algorithm 13 does not include the process of determining the corresponding

Algorithm 13 Recursive two-level deflated FGMRES: `TLADP-FGMRES(A, b)`

```
1: Determine the current level $l$ and dimension $m$ of the Krylov subspace
2: Initialize $u_0$, compute $r_0 = b - Au_0$, $\beta = ||r_0||$, $v_1 = r_0/\beta$;
3: Define $\bar{H}_m \in \mathbb{C}^{(m+1)\times m}$ and initialize to zero
4: for $j = 1,2,...,m$ or until convergence do
5:     $\hat{v}_j = Z^T v_j$                                   ▷ Restriction
6:     if $l+1 == l_{max}$ then                   ▷ Predefined coarsest level $l_{max}$
7:         $\tilde{v} \approx E^{-1}\hat{v}$                     ▷ Approximated by CSLP-FGMRES
8:     else
9:         $l = l+1$
10:        $\tilde{v}$ =TLADP-FGMRES(E, $\hat{v}$)      ▷ Apply two-level deflation recursively
11:    end if
12:    $t = Z\tilde{v}$                                      ▷ Interpolation
13:    $s = At$
14:    $\tilde{r} = v_j - s$
15:    $r \approx M^{-1}\tilde{r}$              ▷ CSLP, by multigrid method or Krylov iterations
16:    $x_j = r + t$
17:    $w = Ax_j$
18:    for $i := 1,2,...,j$ do
19:        $h_{i,j} = (w, v_i)$
20:        $w = w - h_{i,j} v_i$
21:    end for
22:    $h_{j+1,j} := ||w||_2$, $v_{j+1} = w/h_{j+1,j}$
23: end for
24: $X_m = [x_1,...,x_m]$, $\bar{H}_m = \{h_{i,j}\}_{1\le i\le j+1, 1\le j\le m}$
25: $u_m = u_0 + X_m y_m$ where $y_m = \arg\min_y ||\beta e_1 - \bar{H}_m y||$
26: Return $u_m$
```

coarser-grid system E and CSLP preconditioner M on the current level l. This will be elaborated on in the next subsection.

Second, for efficient parallelization, we employ a GMRES method to solve the coarsest grid problem approximately rather than a direct solver. This method is preconditioned by CSLP, which is defined according to the coarsest grid operator. We will numerically investigate the necessary accuracy (or the number of iterations) for solving the coarsest grid problem in the next section.

Third, the multilevel deflation method requires approximating the inverse of CSLP on each level. In [76], this is accomplished using several Krylov subspace iterations (*e.g.*, Bi-CGSTAB) on all levels. The authors set the maximum number of iterations to $C_{it}(N^l)^{\frac{1}{4}}$, where C_{it} is a constant and N^l denotes the problem size on level l. This strategy allows the benefits of using a small shift, resulting in a preconditioner similar to the original Helmholtz operator that retains the ability to shift indefiniteness at certain levels. However, the maximum number of iterations is positively correlated with the grid size on each level, indicating that larger grid sizes require more iterations. Considering the large-scale applications, utilizing Krylov subspace iterations on the first level (finest grid) or the second level may become computationally intensive. Therefore, we propose employing a multigrid cycle to approximate CSLP on the first or second level. Several Krylov subspace iterations can then be applied on the coarser levels. However, in the case of multigrid-based CSLP, ensuring a sufficiently large complex shift is essential. In addition to setting the maximum number of iterations, a relative tolerance as stopping criteria for the iterations is also established in this chapter. This allows iterations to cease once the maximum number of iterations or the tolerance is reached.

Fourth, as shown in Algorithm 13, the number of deflated FGMRES iterations is specified by m. The cycle type of the multilevel deflation technique is determined by the number of iterations of the deflated FGMRES on each coarse level, except for the finest level. If only one iteration is allowed on the coarser levels, a V-cycle is obtained, which is similar to the V-cycle structure of multigrid when $\gamma = 1$. Correspondingly, two iterations on the coarser levels will result in a W-cycle. According to the multigrid method, the W-cycle may offer faster convergence than V-cycle but at the expense of computational efficiency. For increasingly complex model problems, striking a balance between optimal convergence and computational efficiency in the selection of m, hence determining the necessary accuracy (or the number of iterations) for the coarser levels, will be a focal point of this study.

Fifth, in Algorithm 13, all involved matrix-vector multiplications (lines 5, 7, 10, 12, 13, 15, 17) are expressed to denote the outcome of these operations. In our implementation, we compute and return the result of matrix-vector multiplication based on input variables through a linear combination. No explicit construction of any matrices takes place in our approach.

5.1.1. Matrix-Free Parallel Implementation

Matrix-free implementations offer a compelling alternative to standard sparse matrix data formats in large-scale computational scenarios. Besides the reduced memory consumption, matrix-free methods exhibit performance advantages and can potentially outperform matrix-vector multiplications with stored matrices [141]. These improvements enable the solution of larger-scale Helmholtz problems previously constrained by memory limitations, while also enhancing the applicability of modern data-driven methods [142].

Performance Analysis Using Roofline Model

This section presents a performance analysis of matrix-vector multiplication operations $\mathbf{v} = A\mathbf{u}$ comparing our matrix-free implementation with traditional CSR matrix-based approaches. The analysis focuses specifically on the Helmholtz operator with variable wavenumber, using a five-point stencil discretization. We consider the matrix-vector multiplication as it constitutes the primary computational kernel in preconditioned Krylov subspace methods, typically accounting for the majority of computational time. Our analysis employs the roofline model [113], a performance model that bounds computational kernel performance based on peak computational performance and memory bandwidth limitations. As matrix-vector multiplication is typically memory-bound, we focus on arithmetic intensity (I), defined as the ratio of floating-point operations (FLOPs) to memory accesses:

$$I = \frac{\text{Total FLOPs}}{\text{Total Bytes Accessed}}. \tag{5.1}$$

Matrix-Free Implementation The matrix-free implementation directly applies the five-point stencil operation for the discrete Laplacian operator combined with the wavenumber term. For variable wavenumber $\mathbf{k}$, the implementation of matrix-vector multiplication can be expressed in the following computational kernel (in Fortran):

```fortran
! Pre-computed stencil coefficients
ap = -4.d0/h**2
as = aw = ae = an = 1.d0/h**2
do j = 1, ny
    do i = 1, nx
        v(i,j) = as * u(i,j-1)                  & ! South neighbor
                + aw * u(i-1,j)                 & ! West neighbor
                + (ap - k(i,j)**2) * u(i,j) & ! Center point
                + ae * u(i+1,j)                 & ! East neighbor
                + an * u(i,j+1)                   ! North neighbor
    end do
end do
```

In analyzing the memory access patterns, we consider the memory access per grid point operation. The implementation requires reading of five vector elements (center and four neighbors), each consuming 8 bytes in double precision. Additionally, we need to access one wavenumber value (8 bytes) and write one result value (8 bytes). Therefore, the total memory access per point is bounded by 56 bytes.

The computational intensity involves five multiplications for the stencil coefficients, four additions for combining the stencil components, and two additional operations (one square operation and one subtraction) for the wavenumber term. This results in 11 floating-point operations per grid point.

Consequently, the arithmetic intensity for the matrix-free implementation is:

$$I_{MF} \geq \frac{11}{56} \approx 0.1964 \text{ FLOPs/byte} \tag{5.2}$$

CSR Matrix-Based Implementation The compressed sparse row (CSR) format represents the sparse matrix A using three arrays: values (`A%value`), column indices (`A%col_indices`), and row pointers (`A%row_ptr`) [36]. The CSR format implementation is structured as follows:

```
do i = 1, A%nrow
    v(i) = 0.d0
    do j = A%row_ptr(i), A%row_ptr(i+1)-1
        v(i) = v(i) + A%values(j) * u(A%col_indices(j))
    end do
end do
```

The memory access pattern for CSR implementation is more complex. For each non-zero element, we must read the matrix value (8 bytes), the column index (4 bytes), and access the corresponding vector element (8 bytes, assuming the vector is too large to fit into the cache). Additional memory operations include accessing row pointers (4 bytes per row) and reading/writing the result vector (16 bytes per row). For our five-point stencil case, with five nonzero elements per row, the total memory access is bounded by 120 bytes per row.

The computation for each non-zero element requires one multiplication and one addition, resulting in 10 total FLOPs per row. Thus, the arithmetic intensity for the CSR implementation is:

$$I_{CSR} \geq \frac{10}{120} \approx 0.0833 \text{ FLOPs/byte} \tag{5.3}$$

Based on this theoretical analysis, we expect the matrix-free implementation to outperform the CSR matrix-based implementation by approximately a factor of 2.35.

Numerical Validation To validate our theoretical analysis, we conducted extensive performance measurements comparing both implementations. For each grid size, we performed 100 consecutive matrix-vector multiplications to obtain statistically stable performance measurements. The performance metrics are reported in billions of floating-point operations per second (GFLOPs/s), calculated using the theoretical operation count for each implementation.

Table 5.1 presents the experimental results, which strongly support our theoretical analysis. The matrix-free implementation consistently achieves superior performance, with the advantage becoming more pronounced as the problem size increases. For larger problem sizes ($N > 10^6$), we observe performance improvements approaching a factor of 3, exceeding our theoretical prediction of 2.35. This enhanced performance can be attributed to memory hierarchy effects. The matrix-free implementation

Table 5.1: Performance comparison of matrix-free and CSR matrix-based implementations for matrix-vector multiplications.

Problem size N	Matrix-free (GFLOPs/s)	CSR-Matrix (GFLOPs/s)	Performance Ratio
289	4.5918	2.5298	1.82
1,089	5.1946	2.8848	1.80
4,225	6.1535	2.8094	2.19
16,641	5.0961	2.8327	1.80
66,049	6.1740	2.6128	2.36
263,169	6.1361	2.8208	2.18
1,050,625	6.2861	2.3977	2.62
4,198,401	5.9456	1.9992	2.97
16,785,409	5.5556	1.9200	2.89
67,125,249	5.4626	1.8979	2.88
268,468,225	5.4626	1.8958	2.88

exhibits superior cache utilization, particularly for large-scale problems where the memory access patterns of the CSR format become increasingly inefficient.

The characteristics of these implementations have significant implications for parallel computing performance. The matrix-free implementation's regular memory access patterns facilitate better parallel efficiency through predictable memory access and reduced NUMA (Non-Uniform Memory Access) effects. Furthermore, in distributed memory systems, the matrix-free approach minimizes communication overhead, requiring only ghost point exchanges along subdomain boundaries. These parallel computing advantages, combined with the superior cache utilization observed in our sequential tests, suggest even more pronounced performance benefits in parallel computing environments, particularly for large-scale problems on distributed memory systems.

OPERATORS IN MULTILEVEL DEFLATION

This section details the matrix-free implementation of operators in the multilevel deflation method. Matrix-free matrix-vector multiplication is implemented using stencil notation. The computational stencils for both the finest-level and second-level operators (Helmholtz and CSLP preconditioner) and grid-transfer operators (higher-order interpolation and restriction) are detailed in [134]. To enable true multilevel deflation, we extend the matrix-free implementation to coarser levels. We denote the Helmholtz operators as $A_{2^{l-1}h}$ and the CSLP operators as $M_{2^{l-1}h}$, where l is the level number ($l = 1$ represents the finest grid). Starting from the second-level grid, we want to find the computational stencils for A_{4h} so that it is a good approximation to the Galerkin coarsening operator $Z^T A_{2h} Z$. Following [134], we decompose the Helmholtz operator into Laplace and wavenumber operators (assuming a constant wavenumber). By applying Galerkin coarsening operations to their stencils, we obtain the following stencils of the Laplace and wavenumber operators for interior points on the third-level coarse grid:

$$[-\Delta_{4h}] = \frac{1}{4096}\cdot\frac{1}{1024}\cdot\frac{1}{h^2}\cdot \begin{bmatrix} -3 & -534 & -5773 & -11956 & -5773 & -534 & -3 \\ -534 & -32844 & -207370 & -354088 & -207370 & -32844 & -534 \\ -5773 & -207370 & -294371 & 384244 & -294371 & -207370 & -5773 \\ -11956 & -354088 & 384244 & 2945488 & 384244 & -354088 & -11956 \\ -5773 & -207370 & -294371 & 384244 & -294371 & -207370 & -5773 \\ -534 & -32844 & -207370 & -354088 & -207370 & -32844 & -534 \\ -3 & -534 & -5773 & -11956 & -5773 & -534 & -3 \end{bmatrix}_{4h},$$

$$[\mathcal{K}(k^2)_{4h}] = \frac{1}{4096}\cdot\frac{1}{4096}\cdot k^2\cdot \begin{bmatrix} 1 & 322 & 3823 & 8092 & 3823 & 322 & 1 \\ 322 & 103684 & 1231006 & 2605624 & 1231006 & 103684 & 322 \\ 3823 & 1231006 & 14615329 & 30935716 & 14615329 & 1231006 & 3823 \\ 8092 & 2605624 & 30935716 & 65480464 & 30935716 & 2605624 & 8092 \\ 3823 & 1231006 & 14615329 & 30935716 & 14615329 & 1231006 & 3823 \\ 322 & 103684 & 1231006 & 2605624 & 1231006 & 103684 & 322 \\ 1 & 322 & 3823 & 8092 & 3823 & 322 & 1 \end{bmatrix}_{4h}.$$

Using these stencils, the Helmholtz operator and CSLP operator on the third level can be obtained according to their definitions.

Continuing this process iteratively, we can obtain stencils for the Helmholtz operator on coarser levels. It should be noted that starting from the third level, the size of the computation stencils will remain at 7×7. Specific stencils for the fourth to sixth levels can be found in Appendix A.

Boundary

Introducing an accurate boundary scheme for the aforementioned 7×7 computational stencils remains an open problem. In this chapter, we present a simple yet effective approach, involving the introduction of a ghost point outside the physical boundaries, as depicted in Equations (1.44) and (1.46). We apply standard second-order finite-difference discretization to points on the physical boundary. For points near the boundary, we set additional grid points beyond the ghost point to zero. It is important to note that the wavenumbers of the ghost points are also required. For Dirichlet boundary conditions, we determine the wavenumbers of the ghost points similar to Equation (1.46). In other cases, the wavenumbers of the ghost points are uniformly set to zero. This zero-padding approach is motivated by the observation that the coefficients outside the 3×3 kernel become small, and thus, the influence of these points on the overall solution is expected to be minimal. By setting these points to zero, we aim to simplify the computation while maintaining the accuracy of the solution.

To develop a parallel scalable iterative solver, the matrix-free multilevel deflated Krylov subspace methods are implemented within the parallel framework presented in previous chapters.

5.2. CONFIGURATION

Before presenting the performance analysis of the matrix-free multilevel deflation method, we systematically tune the essential components of the algorithm to achieve an optimal balance between computational efficiency and numerical robustness. This section establishes the precise configuration that ensures wavenumber-independent convergence while minimizing computational overhead for complex numerical applications. The outer FGMRES iterations start with a zero initial guess and terminate when the relative residual in Euclidean norm reaches 10^{-6}. Note that all presented results in this section are obtained from sequential computations. In our notation, Ln represents the n-th level in the multigrid hierarchy, where L1 corresponds to the finest level.

5.2.1. TOLERANCE FOR SOLVING THE COARSEST PROBLEM

In this subsection, we explore the tolerance considerations for solving the coarsest problem. For better comparison and fewer other influencing factors, we perform a V-cycle three-level deflation approach to solve the constant wavenumber model problem with Sommerfeld radiation boundary conditions. The finest level represents the first level, and the third level corresponds to the coarsest problem, which will be addressed using GMRES preconditioned with CSLP. To ensure an accurate inverse of CSLP on each level, we employ Bi-CGSTAB iterations, reducing the relative residual to 10^{-8}, assuming that the optimal tolerance is unknown. Since the Krylov subspace method instead of the multigrid method is employed to solve the CSLP here, a small complex shift can be used. Here we will use $\beta_2 = 0.1$. The model problem with wavenumber $k = 200$ is solved with two kinds of resolution, that is $kh = 0.3125$ and 0.625, respectively. As analyzed in [76], the third level will remain indefinite for $kh = 0.3125$, while it becomes negative definite for $kh = 0.625$.

Table 5.2 illustrates the impact of varying tolerances for solving the coarsest problem on the convergence behavior. Specifically, it presents the number of outer iterations required to reduce the relative residual to 10^{-6} and the corresponding number of iterations needed to solve the coarsest problem once.

Table 5.2: The impact of varying tolerances for solving the coarsest problem on the number of outer iterations. In parentheses is the number of iterations required to solve the corresponding coarsest problem once.

kh	10^0	10^{-1}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
0.3125	12 (1)	5 (8)	5 (17)	5 (31)	5 (44)	5 (58)
0.625	16 (1)	31 (32)	33 (67)	33 (133)	33 (203)	33 (256)

From Table 5.2, we observe varying accuracy requirements for solving the coarsest grid problem corresponding to different values of kh. In the case of $kh = 0.3125$, a relative tolerance of 10^{-1} is necessary for maintaining the convergence of the outer iterations. Conversely, for $kh = 0.625$, a single iteration of the coarsest grid solver is sufficient. A strict tolerance in solving the coarsest grid even leads to more outer iterations.

We attribute this phenomenon to the nature of the coarsest grid system, whether it is indefinite or negative definite. According to [76], the third level remains indefinite for $kh = 0.3125$, while it becomes negative definite for 0.625. If the coarsest grid system is indefinite, the relative tolerance of the iterative solver should be ensured at 10^{-1} or smaller. On the contrary, if the coarsest grid system is negative definite, one iteration is adequate. However, it leads to more outer iterations compared to the former case.

To further validate this conclusion, we note the following numerical observations (not presented in tables or figures): using the model problem with Dirichlet boundary conditions at $kh = 0.625$, a tolerance of 10^0 results in outer iterations of 27, while 10^{-1} leads to 32. This brief numerical observation supports our finding that only one iteration suffices when the system becomes negative definite on the coarsest grid. It should be noted that, for the sake of uniformity, if a tolerance of 10^0 appears in this chapter, it means that only one iteration is performed.

5.2.2. Tolerance for Solving CSLP

In this section, we explore the accuracy requirements for the approximate inverse of CSLP on each grid level. Consistent with the solver settings from the previous subsection, we perform one iteration on the coarsest level for $kh = 0.625$ and set the tolerance for the coarsest grid solver to 10^{-1} for $kh = 0.3125$. We vary the tolerance for the Bi-CGSTAB solver used in the approximate CSLP solution. The results are presented in Table 5.3 and Table 5.4, where "Ln Bi-CGSTAB" denotes the number of Bi-CGSTAB iterations needed to achieve the corresponding tolerance on the n-th level. The phrases "Outer FGMRES" and "Coarsest FGMRES" denote the number of FGMRES iterations required for the outer solvers and the coarsest problem solver, respectively.

Table 5.3: Tolerance study for CSLP approximation for $kh = 0.625$.

	10^{-1}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
Outer FGMRES	18	16	16	16	16
Coarsest FGMRES	1	1	1	1	1
L1 Bi-CGSTAB	69	171	408	666	780
L2 Bi-CGSTAB	15	39	84	119	177
L3 Bi-CGSTAB	29	89	174	370	544

From Table 5.3 and Table 5.4, it is observed that whether the coarsest grid system remains indefinite or becomes negative definite, setting a tolerance stricter than 10^{-2} for the approximate inverse of CSLP does not necessarily result in a further reduction in the number of outer iterations. For cases with tolerances of 10^{-1} and 10^{-2}, while the number of outer iterations is reduced by $1-2$ with a tolerance of

Table 5.4: Tolerance study for CSLP approximation for $kh = 0.3125$.

	10^{-1}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
Outer FGMRES	6	5	5	5	5
Coarsest FGMRES	8	8	8	8	8
L1 Bi-CGSTAB	22	338	1149	1771	2344
L2 Bi-CGSTAB	10	44	131	251	300
L3 Bi-CGSTAB	34	48	114	198	272

10^{-2}, achieving this tolerance requires several times more iterations, particularly on the first and second levels, where computations are expensive. We choose to set the tolerance for solving the CSLP to 10^{-1}, striking a balance between achieving sufficient accuracy in the solution and minimizing the overall computational cost. Given that the scaling behavior of CSLP is well-established in the literature [49, 53], we limit our analysis to a single configuration and wavenumber, as this adequately demonstrates the effectiveness of our chosen tolerance.

For a tolerance on the order of 10^{-1}, where the required number of iterations is not substantial, we choose to use the more stable GMRES solver for better approximations to the inverse of CSLP. Furthermore, according to [76], while setting the tolerance at 10^{-1}, we limit the maximum number of iterations to $6(N^l)^{\frac{1}{4}}$, where N^l denotes the size of the problem on level l. This allows the iterations to cease once the maximum number of iterations or the tolerance level is reached.

5.2.3. ON WAVENUMBER INDEPENDENT CONVERGENCE

To achieve a robust multilevel deflation method, we expand our investigation of convergence to deeper levels, larger wavenumbers, and more complex model problems.

FOR CONSTANT WAVENUMBER PROBLEM

We employ the V-cycle multilevel deflation method with coarsening to different levels to solve the constant wavenumber model problem with increasing wavenumber k. As mentioned, GMRES instead of Bi-CGSTAB is used to solve the CSLP with complex shift $\beta_2 = 0.1$. The relative tolerance for solving CSLP is set to 10^{-1}. For the coarsest problem, one iteration is performed if it is negative definite; otherwise, CSLP-preconditioned GMRES iterations are employed to reduce the relative residual to 10^{-1}. The tolerance for the outer FGMRES iterations is 10^{-6}. We consider the scenario with $kh = 0.3125$, indicating that from the fourth level onward, the linear system becomes negative definite.

From Table 5.5, we observe that for the multilevel deflation method, if the coarsest grid system remains indefinite, it exhibits convergence behavior that is close to wavenumber independent, corresponding to the three-level deflation method in the table. However, if the coarsest grid system becomes negative definite, as shown by the four-level deflation method in the table, convergence results can still be achieved, but the number of outer iterations starts to increase with the wavenumber. We also find that continuing to deeper levels, as demonstrated by the five-level deflation method

Table 5.5: The number of outer iterations required for the constant wavenumber model problems with increasing wavenumber k by the multilevel deflation combined with CSLP with complex shift $\beta_2 = 0.1$

k	Multilevel Deflation		
	Three-level	Four-level	Five-level
100	6	9	8
200	6	13	12
400	7	20	20
800	7	37	37

in the table, does not lead to an increase in the number of outer iterations compared to the four-level deflation. While theoretically we could continue to deeper levels until the coarsest problem becomes small enough for direct solving, this approach is less favorable in massively parallel computing environments due to the increased communication costs and potential load imbalance.

Table 5.6: The number of outer iterations required for the constant wavenumber model problems with increasing wavenumber k by the multilevel deflation combined with CSLP with complex shift $\beta_2 = k^{-1}$

k	Multilevel Deflation		
	Three-level	Four-level	Five-level
100	6	6	6
200	6	7	7
400	6	8	8
800	7	9	9

As mentioned above, the Krylov subspace iterations for the CSLP allow the benefits of using a small shift, resulting in a preconditioner similar to the original Helmholtz operator that retains the ability to shift indefiniteness at certain levels. Similar to [76], one can use the inverse of the wavenumber k as the shift ($\beta_2 = k^{-1}$). As observed in Section 5.2.2, having a tolerance of 10^{-1} to approximate the inverse of CSLP leads to an increase in the number of outer iterations. For a small complex shift, the residual cannot be reduced to 10^{-1} within the maximum number of iterations given. Using the more stable GMRES often provides a relatively accurate approximation compared to the Bi-CGSTAB method. This is one of the reasons why GMRES is employed for approximating the inverse of CSLP on the coarse grid levels in this study.

As shown in Table 5.6, with complex shift $\beta_2 = k^{-1}$, the close-to wavenumber independent convergence is obtained even for the multilevel deflation methods where the coarsest grid problems become negative definite. Hereafter, we denote this configuration, which is mainly a parallel matrix-free implementation of the V-cycle multilevel deflation method proposed in [76], as **MADP-v1** (Matrix-free multilevel Adapted Deflation Preconditioning).

FOR NON-CONSTANT WAVENUMBER PROBLEM

In this section, we apply MADP-v1 to non-constant wavenumber problems. Note that for the model problems described in Section 1.2.1, due to the use of a computational domain based on actual physical dimensions rather than being scaled to a unit length, we use a complex shift $\beta_2 = (k_{\text{dim}})^{-1}_{\text{max}}$, where k_{dim} is the so-called dimensionless wavenumber, defined as

$$k_{\text{dim}} = \sqrt{\left(\frac{2\pi f}{c}\right)^2 L_x L_y},$$

with L_x and L_y denoting the lengths of the computational domain in the x and y directions, respectively.

In Table 5.7, we give the results for the wedge problem with $kh = 0.349$, indicating that the linear systems become negative definite from the fourth-level coarse grid onward. We find that the latter case requires more outer iterations and significantly more CPU time. Upon further observation of the solving process, it is observed that coarsening to negative definite levels requires a higher number of GMRES iterations to approximate the CSLP compared to the scenario of coarsening to indefinite levels. In cases where the coarsening remains on indefinite levels, the tolerance of 10^{-1} is achieved within the maximum number of iterations. However, in cases where the coarsening goes to negative definite levels, the number of iterations reaches the maximum specified value without achieving the same tolerance. For example, consider the wedge problem with a grid size of 1153×1921 and $f = 160\,\text{Hz}$. On the first and second levels, the four-level deflation requires 232 and 164 GMRES iterations to approximate the CSLP per outer iteration, respectively, whereas the three-level deflation only requires 73 and 49 GMRES iterations.

Table 5.8 reports the number of iterations required and the time elapsed for the Marmousi problem with $kh = 0.54$. In this scenario, the linear systems become negative definite from the third-level grid onward. Despite being coarsened to deeper negative definite levels, the number of outer iterations remains constant and the computational time is comparable. However, one can find that, for such a highly heterogeneous model problem, the number of outer iterations starts increasing with the frequency. This is consistent with the results in [76].

Table 5.7: Number of iterations required and time elapsed for the wedge problem with $kh = 0.349$ for the largest wavenumber k. In parentheses are the number of iterations to solve the coarsest grid system. L1 and L2 represent the number of GMRES iterations required to approximate the inverse of CSLP on the first and second level, respectively.

		Three-level MADP-v1				Four-level MADP-v1			
f (Hz)	Grid size	Outer #iter	#iter for CSLP		CPU	Outer #iter	#iter for CSLP		CPU
		(Coarsest)	L1	L2	time (s)	(Coarsest)	L1	L2	time (s)
20	145×241	7 (2)	20	51	3.78	8 (1)	48	59	7.00
40	289×481	7 (3)	25	59	20.14	9 (1)	116	83	103.31
80	577×961	8 (4)	38	61	195.14	11 (1)	164	116	907.00
160	1153×1921	8 (4)	73	49	1060.50	13 (1)	232	164	5101.73

Table 5.8: Number of iterations required and time elapsed for the Marmousi problem with $kh = 0.54$ for the largest wavenumber k.

f (Hz)	Grid size	Three-level MADP-v1				Five-level MADP-v1			
		Outer	#iter for CSLP		CPU	Outer	#iter for CSLP		CPU
		#iter	L1	L2	time (s)	#iter	L1	L2	time (s)
10	737×241	12	124	88	133.91	12	124	88	165.43
20	1473×481	16	175	124	1560.03	16	175	124	1743.97
40	2945×961	23	247	175	21557.94	23	247	175	21233.10

In summary, the variant MADP-v1, based on the configuration proposed in [76], utilizes a V-cycle type and allows the combination of CSLP with a smaller complex shift $\beta_2 = (k_{\text{dim}})^{-1}_{\text{max}}$. For constant wavenumber problems, MADP-v1 achieves near wavenumber-independent convergence. However, for non-constant wavenumber model problems, as the wavenumber increases, the number of outer iterations will increase gradually, and the cost of using Krylov subspace methods to solve CSLP will become more noticeable.

For practical applications, where the wavenumber is usually non-constant within the domain, and also for better scalability, the coarsening in this multilevel deflation method should not be limited only to indefinite levels. Therefore, we will pay more attention to the common occurrence of coarsening to negative definite levels. The case of coarsening to indefinite levels will serve as a reference for the case of coarsening to negative definite levels. We aim to achieve at least similar convergence and computational efficiency in the case of coarsening to negative definite levels.

On the Tolerance for Coarse Levels

Sheikh *et al.* [72] stated that using $n2$, $n3$, and $n4$ iterations on the second, third, and fourth levels, respectively, can accelerate convergence, and their results indicate that larger $n2$ leads to better convergence for larger wavenumber. Additionally, Dwarka and Vuik [76] demonstrates that employing a W-cycle instead of a V-cycle for constructing the multilevel hierarchy results in a reduced number of iterations across reported frequencies. In this chapter, we attribute this to the accuracy of solving on the second, third, and fourth levels.

In contrast to the previous configuration of a single iteration on each coarser level, we introduced distinct tolerances for iterations on the second, third, and fourth levels, exploring their impact on outer iterations and CPU time. The numerical experiments utilized the Marmousi model problem with a grid size of 1473×961 and frequencies of 10 Hz and 20 Hz, corresponding to $kh = 0.27$ and 0.54, respectively. As we mentioned, for $kh = 0.27$, the linear systems of the second and third levels remain indefinite, while that of the fourth level becomes negative definite. For $kh = 0.54$, the third and fourth levels become negative definite.

It is evident in Figures 5.1 and 5.2 that the number of outer iterations is correlated with the accuracy of solving the second-level grid system. Overall, a higher number of outer iterations usually corresponds to increased CPU time. However, we also observe

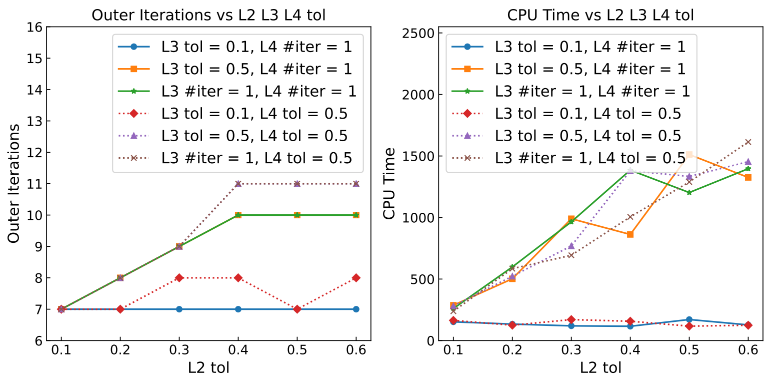


Figure 5.1: Outer iterations and CPU time vary from different tolerances on the second (L2), third (L3), and fourth (L4) levels. Five-level deflation for Marmousi problem, grid size 1473×481, $f = 10\,\text{Hz}$

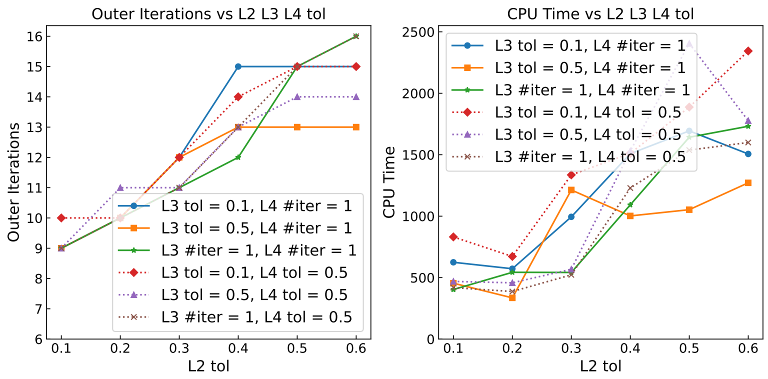


Figure 5.2: Outer iterations and CPU time vary from different tolerances on the second (L2), third (L3), and fourth (L4) levels. Five-level deflation for Marmousi problem, grid size 1473×481, $f = 20\,\text{Hz}$

that the minimum CPU time does not necessarily align with the minimum number of outer iterations, as depicted in Figure 5.2. This suggests that sacrificing a few extra outer iterations may result in computational time savings. A balance between the number of outer iterations and computational time needs to be identified.

For $kh = 0.27$, it is best to set a tolerance of 10^{-1} for the second and third levels and perform one iteration for coarser levels. Conversely, for the case of $kh = 0.54$, the recommended tolerances for the second and third levels are 2×10^{-1} and 5×10^{-1}, respectively. From extensive numerical experiments across various grid sizes and multilevel deflation methods, it is observed that the optimal setting of the tolerance for solving the second, third, and fourth-level grid systems, corresponding to the minimum CPU time and the fewest outer iterations, may vary. However, on a comprehensive scale, a robust and acceptable configuration is to set a tolerance for solving the second-level grid system to 10^{-1}, while performing only one iteration on the other coarser grid levels. Let us denote this configuration as **MADP-v2**.

We applied the MADP-v2 to solve the wedge and Marmousi problems, respectively. The results are presented in Tables 5.9 and 5.10. Compared to the corresponding results in Tables 5.7 and 5.8, MADP-v2 results in reduced computation time and a lower number of iterations across all reported frequencies, showcasing a closer-to-wavenumber-independent behavior. In addition to reduced outer iterations, we also observe that, when setting the tolerance to 10^{-1} for the second-level coarse grid system, the number of iterations required to solve CSLP on the first-level grid is significantly reduced. Comparing Table 5.9 with Table 5.7 (three-level deflation), in the case of coarsening to negative definite levels, MADP-v2 achieves similar convergence and computational efficiency.

Table 5.9: Number of outer FGMRES-iterations for the wedge problem with $kh = 0.349$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

f (Hz)	Grid size	Four-level MADP-v2			
		Outer #iter	#iter for CSLP		CPU
		(L2 #iter)	L1	L2	time (s)
20	145×241	6 (2)	6	59	4.69
40	289×481	6 (2)	8	83	19.26
80	577×961	7 (2)	7	116	148.05
160	1153×1921	7 (3)	15	164	1113.86

5.2.4. COMBINED WITH MULTIGRID-BASED CSLP

Further observation reveals that, in cases where coarsening reaches negative definite levels, a significant portion of the computational time is still dedicated to approximating the inverse of CSLP on the first and second levels. Moreover, these iterations on the second level typically reach the specified maximum number of iterations $6(N^l)^{\frac{1}{4}}$ rather than achieving a tolerance of 10^{-1}. The use of GMRES iterations for solving CSLP on the first and second levels consumes a substantial amount of time, since

Table 5.10: Number of outer FGMRES-iterations for the Marmousi problem with $kh = 0.54$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

f (Hz)	Grid size	Three-level MADP-v2				Five-level MADP-v2			
		Outer #iter	#iter for CSLP		CPU	Outer #iter	#iter for CSLP		CPU
		(L2 #iter)	L1	L2	time (s)	(L2 #iter)	L1	L2	time (s)
10	737×241	8 (3)	23	88	58.01	8 (3)	23	88	48.29
20	1473×481	9 (3)	25	124	422.33	9 (3)	20	124	364.21
40	2945×961	10 (3)	39	175	5267.53	10 (3)	39	175	4106.46

the scale of the first- and second-level grid systems is large.

Instead of employing the GMRES or Bi-CGSTAB methods, we can utilize the multigrid method to approximate the inverse of CSLP on the first and second levels. On coarser levels, GMRES iterations are still used to approximate the inverse of CSLP. However, as known, the multigrid method requires that the complex shift should not be too small. Consequently, we cannot use $\beta_2 = (k_{\text{dim}})^{-1}_{\text{max}}$ as the complex shift for CSLP. Therefore, in the multilevel deflation methods combined with the multigrid-based CSLP, a complex shift of $\beta_2 = 0.5$ will be consistently utilized. (Additional numerical experiments have demonstrated that $\beta_2 = 0.5$ is a superior choice among other smaller complex shifts.)

Except for the choice of the complex shift for CSLP and the method used to solve CSLP on the first- and second-level coarse grid systems, the remaining settings are mostly inherited from MADP-v2. Specifically, a tolerance of 10^{-1} is set for solving the second-level grid system, and only one iteration is performed on other coarser levels. This modified configuration is denoted as **MADP-v3**. As it combines with multigrid-based CSLP on the first and second levels, this variant can be considered as an extension of the two-level deflation method proposed in [134].

The number of iterations and computation time required for solving the wedge and Marmousi problems using MADP-v3 are presented in Tables 5.11 and 5.12, respectively. We observe that compared to MADP-v2 (as shown in Tables 5.9 and 5.10), while the number of outer iterations has increased, it exhibits nearly wavenumber-independent convergence, with computation time three times faster. Moreover, compared to the two-level deflation method proposed in [134], the current multilevel deflation method ensures a similar number of outer iterations while significantly reducing computation time.

Similarly to the last section, the optimal tolerance setting is studied for the second, third and fourth levels, as shown in Figures 5.3 and 5.4. From the figures, it can be observed that performing only one iteration on the fourth level (L4) or setting a tolerance of 0.5 has little impact on the outer iterations but introduces additional computational costs. For this reason, we can keep one iteration on the fourth level. In comparison to performing only one iteration on the third level (L3), setting a smaller tolerance on L3 helps slow down the increase in the number of outer iterations but leads to more computational costs. From the perspective of computation time, performing only one iteration on L3 remains the optimal choice. With the

Table 5.11: Number of outer FGMRES-iterations for the wedge problem with $kh = 0.349$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

		Four-level MADP-v3		Five-level MADP-v3	
f (Hz)	Grid size	Outer #iter (L2 #iter)	CPU time (s)	Outer #iter (L2 #iter)	CPU time (s)
20	145×241	10 (6)	1.73	10 (6)	1.83
40	289×481	10 (10)	8.08	10 (10)	8.87
80	577×961	10 (17)	48.05	10 (18)	64.54
160	1153×1921	11 (34)	356.76	11 (34)	367.53
320	2305×3841	11 (66)	3458.14	11 (64)	3065.03

Table 5.12: Number of outer FGMRES-iterations for the Marmousi problem with $kh = 0.54$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

		Two-level Deflation [134]		Three-level MADP-v3		Five-level MADP-v3	
f (Hz)	Grid size	Outer #iter	CPU time (s)	Outer #iter (L2 #iter)	CPU time (s)	Outer #iter (L2 #iter)	CPU time (s)
10	737×241	11	23.15	11 (17)	16.53	11 (13)	18.57
20	1473×481	11	224.21	11 (30)	110.79	11 (24)	108.03
40	2945×961	12	4354.83	13 (69)	1220.61	13 (50)	1084.42

incorporation of multigrid-based CSLP, the number of outer iterations increases as the tolerance of the second level (L2) increases, while the computation time shows a trend of decreasing first and then increasing. If one wants to minimize the computation time, choosing the tolerance of the second level as 0.3 while still performing only one iteration on other coarser levels can be the optimal option. Let us denote this configuration as **MADP**.

Tables 5.13 and 5.14 present the number of iterations required and computation time to solve the wedge and Marmousi problems using MADP. Compared with Tables 5.11 and 5.12, it can be seen that although a few extra outer iterations are consumed, reduced computation time is obtained.

Therefore, we regard the variant **MADP**, which can balance both convergence and computational efficiency, as an optimal configuration of the present matrix-free multilevel deflation method. This variant employs a tolerance of 0.3 for solving the second-level grid system and performs only one iteration on other coarser levels. A multigrid V-cycle is used to solve the CSLP on the first- and second-level grid systems. On coarser levels, several GMRES iterations approximate the inverse of CSLP with a tolerance of 10^{-1}. In the subsequent sections, we will use this variant for numerical experiments.

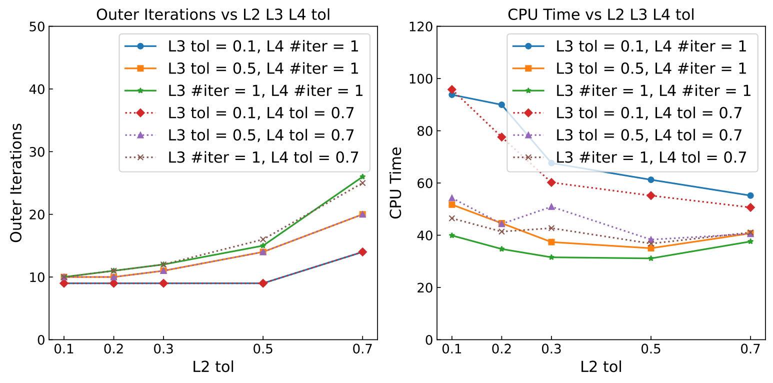


Figure 5.3: Outer iterations and CPU time vary from different tolerances on the second (L2), third (L3), and fourth (L4) levels. Five-level deflation combined with multigrid-based CSLP on first and second levels. Marmousi problem, grid size 1473×481, $f = 10\,\text{Hz}$

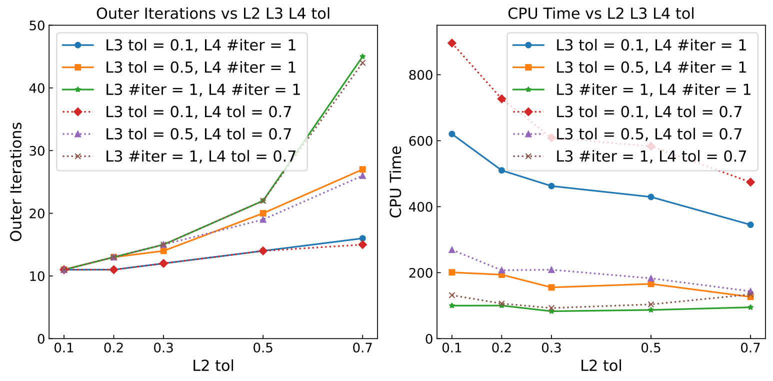


Figure 5.4: Outer iterations and CPU time vary from different tolerances (tol) on the second (L2), third (L3), and fourth (L4) levels. Five-level deflation combined with multigrid-based CSLP on first and second levels. Marmousi problem, grid size 1473×481, $f = 20\,\text{Hz}$

Table 5.13: Number of outer FGMRES-iterations for the wedge problem with $kh = 0.349$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

		Four-level MADP		Five-level MADP	
f (Hz)	Grid size	Outer #iter (L2 #iter)	CPU time (s)	Outer #iter (L2 #iter)	CPU time (s)
20	145×241	11 (4)	1.51	11 (4)	1.55
40	289×481	12 (6)	6.34	12 (6)	6.42
80	577×961	13 (10)	39.62	13 (10)	42.21
160	1153×1921	15 (17)	410.27	14 (18)	400.64
320	2305×3841	16 (33)	2748.70	16 (32)	2762.84

Table 5.14: Number of outer FGMRES-iterations for the Marmousi problem with $kh = 0.54$ for the largest wavenumber k. In parentheses are the number of iterations to solve the second-level grid system.

		Three-level MADP		Five-level MADP	
f (Hz)	Grid size	Outer #iter (L2 #iter)	CPU time (s)	Outer #iter (L2 #iter)	CPU time (s)
10	737×241	14 (10)	12.42	13 (7)	12.67
20	1473×481	16 (19)	93.08	15 (15)	84.06
40	2945×961	19 (43)	929.62	18 (29)	816.38

5.2.5. Complexity Analysis

We next analyze the complexity of the present multilevel deflation method in relation to the problem size or to the frequency, equivalently. In this numerical experiment, the wedge model problem is solved using **five-level MADP**. The grid resolution, *i.e.* kh, is kept fixed to a specific value, while the frequency is growing from 10 Hz to 160 Hz for $kh = 0.349$ or even 640 Hz for $kh = 0.1745$, respectively. The case of $f = 640$ Hz leads to a linear system with approximately 142 million unknowns. The number of outer iterations and the number of iterations on the second level are reported in Table 5.15. Similarly to the results in Table 5.13, the number of outer iterations is rather moderate and is found to grow slightly with respect to frequency.

Figure 5.5 shows the evolution of computational time versus problem size for $kh = 0.349$ and $kh = 0.1745$. As the grid size increases, the computational time of the present matrix-free multilevel deflation method shows a similar trend to the matrix-based version proposed by [76]. However, with single-core sequential computing, the present method can handle grid sizes much larger than those achievable in [76]. If N represents the total number of unknowns, it has been observed that the computational time follows a behavior of $\mathcal{O}(N)$ for small grid sizes and asymptotically approaches $\mathcal{O}(N^{1.4})$. This is comparable to the geometric two-grid preconditioner in [120]. The reason for this behavior is that, as the frequency increases, the number of iterations required on the second level almost increases linearly with frequency, as shown in Table 5.15.

One can think about continuing a similar approach, setting a tolerance of 10^{-1}

Table 5.15: Number of outer FGMRES-iterations for the wedge problem with $kh = 0.1745$. In parentheses are the number of iterations to solve the second-level grid system.

Grid size	#unknowns	f (Hz)	Outer #iter (L2 #iter)	CPU time (s)
145× 241	34945	10	10 (2)	1.13
289× 481	139009	20	11 (3)	4.14
577× 961	554497	40	12 (4)	21.64
1153× 1921	2214913	80	12 (7)	127.47
2305× 3841	8853505	160	13 (13)	1003.71
4609× 7681	35401729	320	14 (27)	7678.83
9217× 15361	141582337	640	17 (47)	53481.69

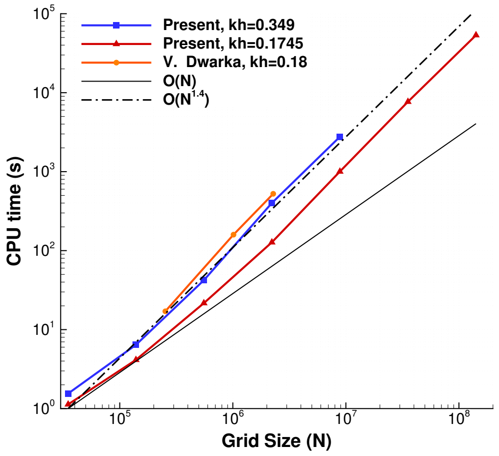


Figure 5.5: Evolution of computational time versus problem size. wedge model problem. The data in orange is extracted from [75].

on the third level, ensuring that the number of iterations on the second level is independent of the wavenumber, and so forth. This is feasible but only limited to indefinite levels. Setting tolerance on negative definite levels, *i.e.*, performing more than one iteration, may lead to a significant increase in outer iterations and computational time, consistent with the conclusion in Section 5.2.1. For instance, considering the wedge model problem with $kh = 0.1745$, where the fourth-level grid system remains indefinite, turning negative definite onwards the fifth level. We can extend MADP-v3 by setting a tolerance of 10^{-1} for iterations on the third and fourth levels instead of performing one iteration. Table 5.16 provides the required number of outer iterations and the number of iterations on the second, third, and fourth levels. We can observe that the number of outer iterations and iterations on the second and third levels are almost independent of the wavenumber, while the number of iterations on the fourth level gradually increases with the wavenumber. However, as shown in Figure 5.6, the computation time required is more than MADP. Together with the case of $kh = 0.087$ in the figure, it can be observed that there are subtle differences in the growth trend compared to that of MADP, possibly approaching $\mathcal{O}(N^{1.3})$. While it is beneficial to set a tolerance for coarser levels for problems with smaller kh, whether the present multilevel deflation method can be closer to $\mathcal{O}(N)$ remains an open problem that requires further study. To complement this study, it would be interesting to perform the same complexity analysis for more levels and three-dimensional cases. These are left to a future line of research. Additionally, in the numerical experiments on current two-dimensional problems, it does not lead to a significant reduction in computation time. Therefore, we consider MADP to still be the optimal choice.

Table 5.16: Number of outer iterations and the number of iterations on the second, third, and fourth levels when a tolerance of 10^{-1} is set on these levels. Wedge problem with $kh = 0.1745$.

Grid size	f (Hz)	Outer #iter	L2 #iter	L3 #iter	L4 #iter
289× 481	20	10	3	2	16
577× 961	40	10	3	2	20
1153× 1921	80	10	3	2	30
2305× 3841	160	10	3	2	46
4609× 7681	320	9	3	2	75

5.3. Parallel Performance

In this section, we aim to present both weak scalability and strong scalability. Through this analysis, our goal is to offer insight into the suitability of the present multilevel deflation method for practical large-scale applications in the context of heterogeneous time-harmonic wave problems.

The **parallel six-level MADP** preconditioned FGMRES is used as the default approach in this section to solve model problems. All numerical experiments are carried out on the Linux supercomputer DelftBlue [127].

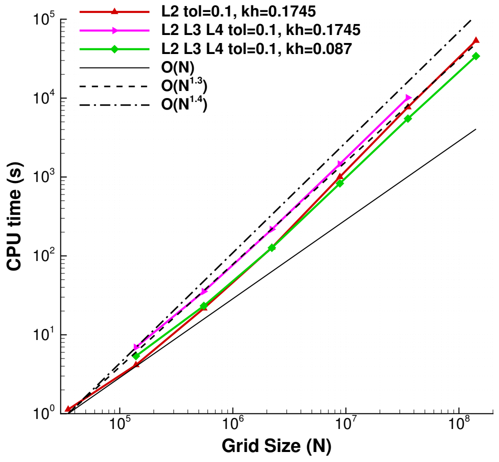


Figure 5.6: Evolution of computational time versus problem size. Wedge model problem.

5.3.1. Weak Scalability

To assess the weak scalability of the proposed matrix-free parallel multilevel deflation preconditioning method, we keep the wavenumber or frequency unchanged and solve the model problems across varying problem sizes but maintain a fixed workload per processor. The computational times for different problem sizes and the corresponding number of processors are summarized in Tables 5.17 and 5.18. As the grid undergoes refinement while maintaining a constant wavenumber, the parameter kh gradually decreases. In the context of deflation preconditioning, it has been documented that a smaller kh leads to a reduction in the number of outer iterations [72]. As kh continues to diminish, the number of outer iterations tends to stabilize. Additionally, the advantages of one or two less iterations may be counteracted by the overhead of data communication. Consequently, we observe that the computational time initially decreases due to the reduced number of outer iterations, and then remains almost constant, even as the grid size expands to tens of millions with over a thousand parallel computing cores.

This behavior is highly commendable, as it allows for the efficient resolution of large linear systems within a reasonable computational timeframe on a parallel distributed memory machine. It is important to emphasize that the advantages of the suggested approach should be considered in the context of minimizing pollution error by grid refinement for real-world application of Helmholtz problems.

5.3.2. Strong Scalability

We are also interested in the strong scalability properties of the present parallel multilevel deflation preconditioning method for the Helmholtz problems. In this

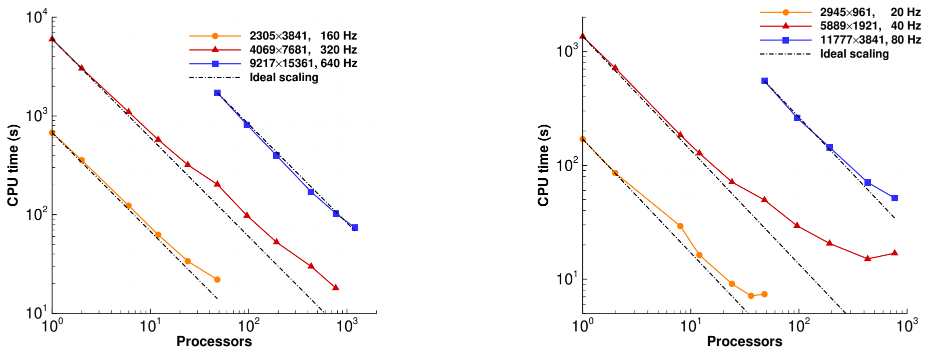
Table 5.17: Weak scaling for the model problem with constant wavenumber.

Grid size	#unknowns	np	#iter	CPU time (s)
		$k = 400$		
641×641	410881	1	16	49.68
1281×1281	1640961	4	13	21.63
2561×2651	6558721	16	12	16.13
5121×5121	26224641	64	11	21.66
		$k = 1600$		
2561×2561	6558721	16	20	168.26
5121×5121	26224641	64	14	100.84
10241×10241	104878081	256	13	79.69
20481×20481	419471361	1024	13	93.62

Table 5.18: Weak scaling for the wedge model problem with $f = 320\,\text{Hz}$.

Grid size	#unknowns	np	#iter	CPU time (s)
2305×3841	8853505	48	16	69.75
4609×7681	35401729	192	14	53.20
9217×15361	141582337	768	14	67.03

section, we perform numerical experiments on the nonconstant-wavenumber model problems with fixed problem sizes while varying the number of processors. First of all, the numerical experiments show that the number of iterations required is found to be independent of the number of processors used for parallel computing, which is a favorable property of our multilevel deflation method.



(a) Wedge problem (b) Marmousi problem

Figure 5.7: Strong scaling of the parallel multilevel APD-FGMRES for the non-constant wavenumber model problems with various grid sizes and frequencies.

Figure 5.7 plots the computational time versus the number of processors. The figures show a decrease in parallel efficiency as the number of processors increases, particularly for the Marmousi model. The analysis suggests that maintaining a minimum of around one million unknowns per processor ensures a parallel efficiency of 60% or higher. If there are fewer than 50 thousand unknowns per processor, the ratio of computation load to data communication may significantly decrease, leading to poor parallel efficiency. However, increasing the grid size for the same model problem can result in improved parallel efficiency. This is because, while the number of ghost-grid layers used for communication remains constant, the amount of data to be communicated doubles when the number of grid points doubles in each direction. Meanwhile, the total number of grid points increases fourfold, resulting in a larger ratio of computation load to data communication and better parallel efficiency. Overall, as demonstrated in solving the wedge problem with a grid size of 9217×15361 and a frequency of $f = 640\,\text{Hz}$, the current matrix-free multilevel deflation approach can effectively solve complex Helmholtz problems with grid sizes up to tens of millions. It demonstrates strong parallel scalability, maintaining efficiency across more than a thousand processors.

5.4. CONCLUSIONS

In this chapter, we present an advanced matrix-free parallel scalable multilevel deflation preconditioning method for solving the Helmholtz equation in heterogeneous time-harmonic wave problems, benchmarked on large-scale real-world models. Building on recent advancements in higher-order deflation preconditioning, our approach extends these techniques to a parallel implementation. The incorporation of the deflation technique with CSLP, along with higher-order deflation vectors and re-discretization schemes derived from the Galerkin coarsening approach, forms a comprehensive setup for matrix-free parallel implementation. The proposed re-discretized finite-difference schemes at each coarse level contribute to a convergence behavior similar to that of the matrix-based deflation method.

We have explored different configurations of the multilevel deflation method, conducting research and comparing various variants. We note that the performance of different cycle types of the present multilevel deflation method is impacted by whether the coarsest-level system is negative definite. We suggest that for the multilevel deflation method coarsening to negative definite levels, ensuring a certain accuracy in iterations on the second-level grid is crucial to maintain a consistent number of outer iterations. Based on the complexities revealed during our study, we propose a robust and efficient variant, MADP. This variant employs the following settings: a tolerance of 0.3 for solving the second-level grid system, with only one iteration performed on other coarser levels; a multigrid V-cycle to solve CSLP on the first- and second-level grid systems; several GMRES iterations to approximate the inverse of CSLP on coarser levels, using a tolerance of 10^{-1}. It addresses the challenges posed by negative definite coarsest-level systems and does not lead to worse complexities.

Our numerical experiments illustrate the effectiveness of the matrix-free parallel multilevel deflation preconditioner, demonstrating convergence properties that are nearly independent of the wavenumber. The reduction in memory consumption achieved through matrix-free implementation, along with satisfactory weak and strong parallel scalability, emphasizes the practical applicability of our approach for large-scale real-world applications in wave propagation, meeting our objectives.

6

Robust Parallel Multilevel Deflation Solver: A Tool for 3D Full-Waveform Inversion

A robust and scalable parallel solver is presented for large-scale 3D Helmholtz problems arising in seismic wave modeling. We introduce an advanced multilevel deflation method with a matrix-free implementation that employs re-discretized schemes to approximate the Galerkin coarsening operator, achieving close-to wavenumber-independent convergence. The solver implements a hybrid MPI+OpenMP parallelization framework that optimizes both computational efficiency and memory utilization, enabling the solution of unprecedented problem sizes. Our comprehensive validation encompasses constant wavenumber problems, the SEG/EAGE Salt model, and the challenging GO_3D_OBS subduction zone model. The method demonstrates excellent parallel scalability up to thousands of CPU cores while maintaining consistent convergence rates across varying frequencies and problem sizes. Proper selection of components within the deflation preconditioner framework significantly enhances convergence behavior, particularly for complex geological structures. Notable achievements include successful resolution of the GO_3D_OBS model with approximately 3.8 billion degrees of freedom at 5 Hz, demonstrating excellent parallel scalability. The solver's robust performance across diverse geological settings, coupled with its excellent scaling properties, establishes it as a powerful forward modeling engine for practical applications such as full-waveform inversion (FWI).

Parts of this chapter have been submitted to GEOPHYSICS (2024).

So far, we have developed a series of increasingly sophisticated preconditioning techniques in the matrix-free framework [124, 125]. Initially, we implemented a parallel two-level deflation preconditioning [134] that demonstrated wavenumber-independent convergence and excellent parallel scalability. Building upon this foundation, we proposed a robust matrix-free parallel scalable multilevel deflation algorithm [137] that further enhanced computational efficiency. However, the application of high-order deflation algorithms to large-scale realistic three-dimensional heterogeneous problems remained largely unexplored.

In this chapter, we present a significant extension of our previous work [134, 137] on matrix-free parallel multilevel deflation preconditioning for 2D time-harmonic wave problems to address large-scale 3D scenarios, particularly as a forward engine for practical applications like FWI. This chapter presents three significant contributions. First, we extend our parallel multilevel deflation preconditioning from 2D to 3D settings, with particular emphasis on developing re-discretization schemes for coarse levels in 3D that effectively approximate the Galerkin coarsening operator. This extension maintains the method's wavenumber-independent convergence. Second, we implement a hybrid MPI/OpenMP parallelization strategy that significantly enhances the capability of our solution method. This hybrid approach not only mitigates memory constraints but also efficiently utilizes CPUs within compute nodes, enabling better computational resource utilization. Third, through optimization of solver components, we demonstrate the capability to efficiently solve the GO_3D_OBS model with up to 3.8 billion degrees of freedom - to our knowledge, the first time this realistic crustal model has been solved at such scale - which can serve as a benchmark for extreme-scale wave propagation problems. The effectiveness of our approach is demonstrated through extensive numerical experiments, showing excellent scalability up to thousands of computing cores.

6.1. PROBLEM DESCRIPTION

The 3D Helmholtz equation defined over a parallelepipedal domain with first-order Sommerfeld radiation boundary conditions is our primary mathematical model.

$$-\Delta u(x,y,z) - k(x,y,z)^2 u(x,y,z) = b(x,y,z), \ \in \Omega \subset \mathbb{R}^3, \tag{6.1}$$

While the governing equations maintain their fundamental structure in the transition to three dimensions, the increased computational complexity and memory requirements necessitate careful consideration of implementation strategies. Our numerical investigation encompasses three model problems of increasing complexity.

First, the 3D constant-wavenumber problem with point source in a unit cube domain, as introduced in Section 1.2.2, provides a valuable verification tool through its analytical solution.

Second, the industry-standard SEG/EAGE Salt model in Section 1.2.2 presents a realistic heterogeneous scenario to demonstrate the effectiveness and scalability of our extended 3D matrix-free parallel multilevel deflation preconditioning method. We consider a computational domain with dimensions of 12.8 km × 12.8 km × 3.84 km, utilizing grid sizes 641 × 641 × 193 and 1281 × 1281 × 385. The latter yields around

632 million degrees of freedom, representing a computationally challenging problem that is well-suited for assessing the efficacy of our parallel implementation. The simulations span frequencies from 5 Hz to 15 Hz.

Finally, we consider the GO_3D_OBS model in Section 1.2.2 to evaluate the performance of our parallel multilevel deflation approach in a realistic extreme-scale geophysical context. For our numerical experiments, we select a target region within the GO_3D_OBS model with dimensions of $20\,\text{km} \times 100\,\text{km} \times 29.6\,\text{km}$. To discretize this target, we employ a grid interval of 50 m and 25 m for simulations of frequency 3 Hz and 5 Hz, respectively. The latter results in a finite-difference grid of dimensions $801 \times 4001 \times 1185$, yielding approximately 3.8 billion degrees of freedom. The GO_3D_OBS model is particularly noteworthy for its significantly larger domain compared to conventional benchmark models like the SEG/EAGE Salt model. By employing this model, we aim to demonstrate the applicability and efficiency of our method in scenarios that closely resemble contemporary geophysical exploration challenges. This approach allows us to assess not only the computational performance of our algorithm but also its potential impact on practical seismic imaging applications.

6.2. Numerical Methods

Our method builds upon the multilevel deflation preconditioned Krylov subspace method and its MADP-v3 variant proposed in [137]. The algorithm employs flexible GMRES(5) (FGMRES(5)) with a tolerance of 10^{-1} on the second-level grid system, while performing single FGMRES iteration on coarser levels. The CSLP is solved using a multigrid V-cycle on the first- and second-level grid systems, with several GMRES iterations (tolerance 10^{-1}) approximating the inverse of CSLP on coarser levels. The primary modification from [137] is the adoption of restarted FGMRES(5) instead of full FGMRES for both outer iterations and coarse-grid solutions.This adaptation is motivated by memory constraints, as full FGMRES requires substantial memory that becomes a limiting factor for the large-scale three-dimensional problems addressed in this work. Furthermore, this study will briefly investigate the impact of the parameter γ in Equation (4.2), which shifts the deflated eigenvalues from the origin to the value γ.

6.2.1. Matrix-free Parallel Implementation

The parallel implementation employs a hybrid strategy where the three-dimensional domain is first partitioned among MPI processes, followed by thread-level parallelization within each subdomain. In Chapter 3, we have described the details of the parallelization of geometric multigrid methods based on blockwise domain partitioning, which is based on standard MPI (Message Passing Interface). To further enhance computational efficiency, we have augmented the implementation with OpenMP directives for shared-memory parallelization of all computational kernels related to matrix-vector products, restriction, and interpolation operations. Specifically, the matrix-vector multiplication kernels are optimized using collapse clauses for nested loops. Thread-private variables ensure safe concurrent execution of stencil computations, while workshare constructs are utilized for vectorized operations.

Matrix-free implementations offer a compelling alternative to standard sparse matrix data formats in large-scale computational scenarios. Besides the reduced memory consumption, matrix-free methods exhibit performance advantages and can potentially outperform matrix-vector multiplications with stored matrices [141, 142]. These improvements enable the solution of larger-scale Helmholtz problems previously constrained by memory limitations. We observe that for matrices admitting a seven-point stencil representation, matrix-vector multiplications can be efficiently computed through the direct combination of stencil coefficients with their corresponding grid values, thereby circumventing the necessity of explicit matrix storage. While the stencil formulations for the Helmholtz and CSLP operators on the finest grid have been established in the preceding section, we shall now extend our discussion to encompass the stencil representations of the deflation vectors and the corresponding operators on coarser grid levels.

Higher-order Interpolation Operator

To facilitate the transfer of operators between grid levels, we employ a higher-order interpolation operator $Z_{3\mathrm{D}}$. We construct Z_{3D} using a tensor product of one-dimensional interpolation weights derived from a Bézier polynomial scheme [75]. The one-dimensional interpolation weights are given by

$$w = \frac{1}{8}[1, 4, 6, 4, 1] \tag{6.2}$$

The three-dimensional operator Z_{3D} is then formed by the outer product of these weights. It results in a $5 \times 5 \times 5$ stencil where each element is computed by the product of binomial coefficients as follow

$$Z_{3D}(i, j, k) = \frac{1}{8^3}\binom{4}{i-1}\binom{4}{j-1}\binom{4}{k-1} \tag{6.3}$$

For illustrative purposes, a slice of the stencil of Z_{3D} is presented as follow

$$[Z_{3D}(:, :, 3)] = \frac{1}{8^3} \times \begin{bmatrix} 6 & 24 & 36 & 24 & 6 \\ 24 & 96 & 144 & 96 & 24 \\ 36 & 144 & 216 & 144 & 36 \\ 24 & 96 & 144 & 96 & 24 \\ 6 & 24 & 36 & 24 & 6 \end{bmatrix} \tag{6.4}$$

Helmholtz and CSLP Operator on Coarse Levels

We denote the Helmholtz operators as $A_{2^{l-1}h}$ and the CSLP operators as $M_{2^{l-1}h}$, where l is the level number ($l = 1$ represents the finest grid). The coarse-grid operators are theoretically obtained by the Galerkin coarsening procedure. To avoid explicitly constructing the matrices and performing matrix-matrix multiplications, we aim to find computational stencils for A_{2h} so that it is a good approximation to the Galerkin coarsening operator $Z_{3D}^{\top} A_h Z_{3D}$.

In practical computational implementations, especially for large-scale problems, the storage and multiplication of full sparse matrices can be computationally expensive and memory-intensive. To address this issue, we employ a matrix-free approach for the implementation of the Helmholtz and CSLP operators on coarse levels, which relies on the use of precomputed stencils rather than explicitly storing the full operator matrices.

Similar to [134], we split the Helmholtz operator into the Laplace operator and the wavenumber term. The core idea is to extract the local stencils of the coarse-grid operators through small-scale matrix-matrix multiplications performed in advance. It should be noted that this computation is confined to the pre-processing stage. The actual solver operates independently, using only the precomputed stencil data. Specifically, we compute

$$[\Delta_{2h}] = \mathrm{Row}_c\left(\tilde{Z}_{3D}^{\top}\tilde{\Delta}_h\tilde{Z}_{3D}\right) \tag{6.5}$$

where $\tilde{\Delta}_h \in \mathbb{R}^{17^3\times 17^3}$ is constructed from the Laplacian stencil $[\Delta_h]$, and $\tilde{Z}_{3D} \in \mathbb{R}^{17^3\times 9^3}$ is constructed from the interpolation stencil $[Z_{3D}]$. The $\mathrm{Row}_c(\cdot)$ denotes the extraction of the row corresponding to the central grid point after the Galerkin projection. This row contains all non-zero entries that define the stencil for the operator Δ_{2h}. The process is recursively applied to obtain stencils for $\Delta_{4h}, \Delta_{8h}, \Delta_{16h}, \cdots$. It is noted that from the third level onward, the size of the stencils remains fixed at $7\times 7\times 7$ [76].

As for the wavenumber operator, a key step is to first assume that the wavenumber is locally constant k. Then, the same procedure is applied to the wavenumber operators $\mathcal{K}(k^2)_h, \mathcal{K}(k^2)_{2h}, \cdots$ to obtain the stencils $\left[\mathcal{K}(k^2_{i,j,k})_{2h}\right], \left[\mathcal{K}(k^2_{i,j,k})_{4h}\right], \cdots$, respectively.

Specific stencils for the second to fifth levels can be found in Appendix B.

6.3. NUMERICAL RESULTS

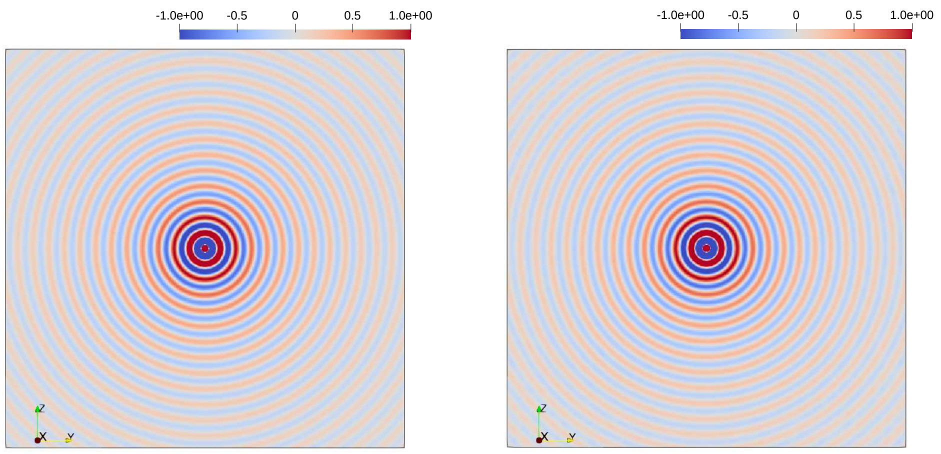
This section presents a systematic evaluation of our matrix-free parallel multilevel deflation method, demonstrating its effectiveness across increasingly challenging scenarios for 3D Helmholtz problems. Our evaluation strategy progresses through three carefully selected test cases, each addressing specific aspects of the method's capabilities. We begin with constant wavenumber problems, where analytical solutions enable rigorous validation of numerical accuracy and convergence properties. The investigation then advances to the more challenging SEG/EAGE Salt model, featuring heterogeneous velocity structures that allow comprehensive assessment of both convergence behavior and parallel performance in realistic settings. Finally, we demonstrate the method's capability to tackle extreme-scale problems through the GO_3D_OBS model, successfully solving systems with unprecedented size of 3.8 billion degrees of freedom. Throughout these experiments, we systematically evaluate three key aspects: solution accuracy, convergence characteristics, and parallel scalability. Special attention is given to the optimization strategies of the multilevel deflation method for extreme-scale applications, revealing how carefully designed algorithmic adjustments enable efficient solution of these large-scale problems.

All numerical experiments are carried out on the Linux supercomputer DelftBlue [143], which operates on the Red Hat Enterprise Linux 8 operating system. Each compute node is furnished with two Intel Xeon E5-6448Y CPUs featuring 32 cores at 2.1 GHz, 250 GB of RAM, a memory bandwidth of 132 GB s^{-1} per socket, and a 100 Gbit s^{-1} InfiniBand card. The present solver is developed in Fortran 90 and is compiled using GNU Fortran 8.5.0 with the compiler options `-O3` for optimization purposes. The Open MPI library (v4.1.1) is used for data communication. For the outer FGMRES iterations, a zero initial guess is selected and it is regarded as convergent when the L$_2$-norm of the residual normalized by the L$_2$-norm of the right-hand side vector satisfies the following relation

$$\frac{||b_h - A_h u_h||_2}{||b_h||_2} \leq 10^{-6}. \tag{6.6}$$

6.3.1. Constant Wavenumber Problem with Point Source

To validate our numerical approach, we first examine the constant wavenumber problem as a fundamental benchmark case. This provides a well-defined reference point for assessing solution accuracy against analytical solutions. We solve this problem for three wavenumbers: $k = 80$ on 129^3 and 257^3 grids, using 2^3 cores; $k = 160$ on 257^3 and 513^3 grids, using 4^3 cores; $k = 320$ on 513^3 and 1025^2 grids, using 8^3 cores. Figure 6.1 compares the analytical and numerical wavefields at plane $x = 0.5$. The numerical approximation shows good agreement with the analytical solution, demonstrating the basic validity of our approach.



(a) Real part of the analytical solution at plane $x = 0.5$

(b) Real part of the numerical approximation at plane $x = 0.5$

Figure 6.1: The constant wavenumber model problem with wavenumber $k = 160$

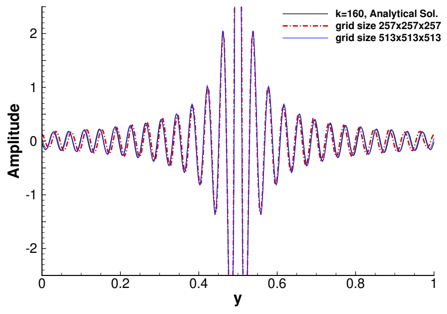
To provide a more quantitative comparison, we define the relative L_2-error between analytical solution and numerical approximation as

$$err. = \frac{||u_{anal.} - u_{num.}||_2}{||u_{anal.}||_2}. \tag{6.7}$$

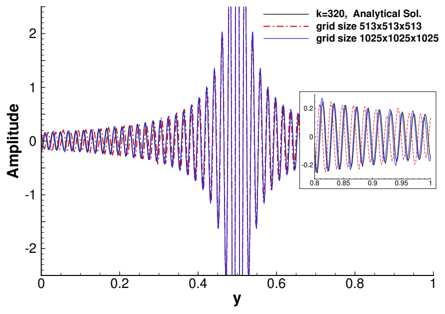
It should be noted that we exclude from the summation the source grid point itself to avoid bias introduced by the point source implementation. Table 6.1 presents the error comparison across various wavenumbers and grid resolutions. Figure 6.2 also visualizes the amplitude in a line across the source along the x-profile.

Table 6.1: Comparison of errors between analytical solution and numerical approximation for constant wavenumber problems

Grid size	k	kh	$err.$
129×129×129	80	0.625	0.293
257×257×257	80	0.3125	0.098
257×257×257	160	0.625	0.555
513×513×513	160	0.3125	0.152
513×513×513	320	0.625	1.017
1025×1025×1025	320	0.3125	0.277



(a) $k = 160$



(b) $k = 320$

Figure 6.2: Comparison between analytical solution and numerical approximation along a x profile cross cutting the source position.

The numerical results demonstrate that using 10 grid points per wavelength ($kh = 0.625$) yields good agreement with the analytical solution in the near-source region (see Figure 6.2). However, the accuracy deteriorates with increasing distance from the source due to accumulated numerical dispersion effects. This effect becomes more pronounced with larger wavenumber $k = 320$. This error can be mitigated by using higher-order, low-dispersion finite difference schemes [23, 27, 29, 31], eigenvalue-based deflation [12], or by increasing the grid resolution. As shown in Figure 6.2, increasing the grid size to $513 \times 513 \times 513$ (20 grid points per wavelength) significantly improves the alignment between numerical approximations and analytical solutions. This result underscores the potential of our scalable parallel solver to minimize pollution error through finer grid resolution, leveraging high-performance computing resources, as we have demonstrated in [125, 134]. Since the current research prioritizes the efficiency of our matrix-free parallel deflation method for large-scale problems, extending the framework to a higher-order, low-dispersion finite difference scheme remains a topic for future study.

Understanding the relative performance of the deflation methods in 3D settings is crucial for practical applications. Thus, we present a comparative analysis of three related matrix-free parallel preconditioners, including the CSLP [125], Two-Level Deflation (TL-ADP) [134], and Three-Level Deflation (3L-ADP), to show their effectiveness across scenarios with different grid sizes. Table 6.2 presents the number of outer iterations for various grid sizes and wavenumbers. To compare the two-level deflation and multilevel deflation which exhibit similar outer iterations, we also report the CPU time required for the sequential computing.

Table 6.2: Comparison of preconditioners for constant wavenumber problems

		CSLP(V)+GMRES		TL-ADP+FGMRES		3L-ADP+FGMRES	
grid size	k	#iter	CPU time (s)	#iter	CPU time (s)	#iter	CPU time (s)
129×129×129	40	85	76	12	140	12	225
257×257×257	80	172	1775	12	1659	12	1954
513×513×513	160	343	62716	12	28441	12	24681

Analysis of the results presented in Table 6.2 reveals several significant findings regarding convergence behavior and computational efficiency. First, both the higher-order deflation methods significantly outperform the CSLP in terms of the number of iterations required for convergence. This improvement is consistent across all grid sizes and wavenumbers tested. The deflation methods (TL-ADP and 3L-ADP) achieve close-to wavenumber-independent convergence, which is in alignment with our results and theory [75, 76, 134]. The number of iterations remains constant at 12 for all grid sizes and wavenumbers, while the CSLP method shows a substantial increase in iterations as the problem size grows. This wavenumber-independent behavior is a crucial advantage for large-scale problems with high wavenumbers.

Regarding CPU time efficiency, a different pattern emerges. For smaller grid sizes (129×129×129), the CSLP method initially shows lower CPU times compared to the deflation methods. This is likely due to the additional costs within a single iteration associated with the deflation techniques. However, as the grid size increases to 513×513×513, both deflation methods demonstrate superior performance in terms of CPU time. Notably, for the grid size 513×513×513, the two-level method achieves a speedup of approximately 2.2 compared to CSLP, while the multilevel (three-level) method further improves performance with a speedup of 2.5. This suggests that the multilevel approach becomes more efficient as the problem size increases, which is consistent with the results in [76]. The improved performance of the multilevel method for larger problems is particularly promising for tackling even more extensive 3D simulations.

These findings demonstrate the efficacy of deflation-based preconditioners in attaining wavenumber-independent convergence for problems with constant wavenumber.

The multilevel deflation approach, in particular, shows great promise for large-scale problems, as it maintains the low iteration count while improving CPU time efficiency for the largest grid size tested.

6.3.2. 3D SEG/EAGE SALT MODEL PROBLEM

To further evaluate the effectiveness of our multilevel deflation approach, we conducted experiments solving the 3D SEG/EAGE Salt model problem, which presents a more challenging, heterogeneous scenario compared to the constant wavenumber problem discussed earlier. Figure 6.3 presents the wavefields computed in the SEG/EAGE salt model at 15 Hz. One can observe that wave propagation patterns align with complex velocity structures. The SEG/EAGE salt model results show strong reflections at the salt body boundaries, with wave fronts curving at these interfaces.

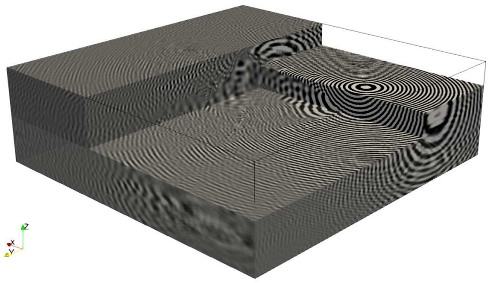


Figure 6.3: The wavefields computed in the SEG/EAGE salt model at 15 Hz

WAVENUMBER-INDEPENDENT CONVERGENCE

Table 6.3 compares the sequential computing time for two-level, three-level, and four-level deflation on a grid size of $641\times641\times193$ at $f = 5$ Hz.

Table 6.3: Comparison of deflation levels for the SEG/EAGE Salt model problem with a grid size of $641\times641\times193$, $f = 5$ Hz

Deflation	#iter	CPU time (s)
Two-level	16	20877
Three-level	17	19768
Four-level	17	14660

The results in Table 6.3 demonstrate that multilevel deflation outperforms two-level deflation for this large-scale heterogeneous problem. While the number of iterations remains relatively constant across all deflation levels, there is a significant reduction in CPU time as we increase the number of levels. The four-level deflation method, in particular, shows a substantial improvement, reducing the CPU time by approximately 30% compared to the two-level approach.

We examined the performance of the four-level deflation method across different frequencies. Table 6.4 presents the number of outer FGMRES iterations required to solve the SEG/EAGE Salt model problem with increasing frequencies and corresponding grid sizes.

Table 6.4: Four-level deflation performance for SEG/EAGE Salt model at various frequencies

Grid size	f (Hz)	#iter
641×641×193	5	17
641×641×193	7.5	17
1281×1281×385	10	18
1281×1281×385	15	16

The results in Table 6.4 provide strong evidence for the wavenumber-independent convergence of our multilevel deflation method. As the frequency increases from 5 Hz to 15 Hz, the number of iterations remains remarkably stable, ranging from 16 to 18. This consistency in iteration count further validates the effectiveness of our approach with respect to the wavenumber-independent convergence [75, 76].

It is worth noting that the grid size 1281×1281×385 represents a substantial increase in the total number of unknowns. Despite this growth in problem size, the multilevel deflation method maintains its efficiency, demonstrating its scalability and robustness.

These results for the SEG/EAGE Salt model complement our findings for constant wavenumber problems. They provide compelling evidence that the multilevel deflation approach is not only effective for simpler, homogeneous problems but also excels in handling complex, large-scale heterogeneous scenarios.

Parallel Performance

After establishing the effectiveness of our multilevel deflation approach, we now turn our attention to its parallel performance. We focus on strong scalability, examining how the method performs as we increase the number of cores while keeping the problem size constant. To increase the scalability to even larger problems, in this work, we extend our parallel framework from the pure MPI parallelization to a hybrid MPI + OpenMP parallelization that significantly mitigates the limitations due to available memory per CPU on the computing nodes.

MPI Only Parallelization We consider the 3D SEG/EAGE Salt model problem with a grid size of 641×641×193 at a fixed frequency of 5 Hz. We evaluate the performance of two-level, three-level, and four-level deflation methods as we increase the number of cores (MPI ranks). Across all levels of parallelization, the number of outer iterations remains consistent with sequential results in Table 6.3 for each method, demonstrating the numerical robustness of our approach under parallel decomposition.

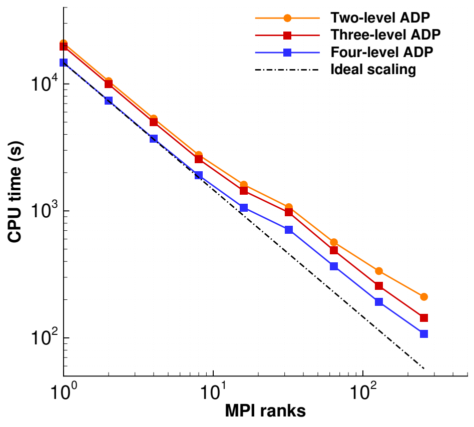


Figure 6.4: Strong scaling of two-level, three-level, and four-level deflation methods for SEG/EAGE Salt model, $641\times641\times193$, $f = 5\,\text{Hz}$

Figure 6.4 presents the computational time versus the number of MPI ranks used. These results provide compelling evidence that the multilevel method not only scales better but also requires less absolute CPU time across all core counts, demonstrating its superiority in a distributed computing environment. Our matrix-free parallel framework exhibits near-perfect scaling for the higher-order multilevel deflation methods at reasonable core counts. As the number of cores increases, we observe a moderate decrease in scaled parallel efficiency. This decline can be attributed to the inter-compute-node communications and a decrease in the computation-to-communication ratio [125, 134].

We also tested the scaling behavior for a larger grid size of $1281 \times 1281 \times 385$ and $f = 10\,\text{Hz}$. As evident from Figure 6.5, our method exhibits excellent scaling for over 1000 cores with this larger problem size. It underscores the potential of our parallel implementation to address computationally intensive wave propagation simulations in a real-world, large-scale application.

Hybrid MPI and OpenMP Parallelization Although our pure MPI implementation exhibits excellent scalability, it encounters memory constraints, especially when addressing problems of substantial size. As previously noted, each compute node has a finite amount of memory (theoretically 256 GB). For exceptionally large problems, this can lead to memory exhaustion, limiting the size of problems that can be tackled effectively.

A common strategy to address this issue is to distribute fewer MPI ranks (cores) per compute node, allowing more memory to be assigned to each core within a node. However, this approach can be inefficient if we leave some CPUs within a compute node idle. To optimize resource utilization, we have implemented a hybrid MPI+OpenMP approach, using OpenMP directives to parallelize the nested loops in

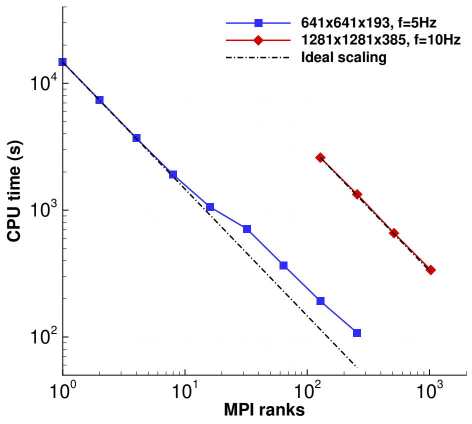


Figure 6.5: Strong scaling of the four-level deflation method for SEG/EAGE Salt model with increasing grid sizes and frequencies

our matrix-free matrix-vector multiplication routines.

To determine the optimal configuration for this hybrid approach, we conducted tests on a single compute node with 64 CPUs, exploring various combinations of MPI ranks and OpenMP threads. Table 6.5 presents the results of these tests, comparing the performance of different MPI+OpenMP configurations against the pure MPI implementation.

Table 6.5: Performance comparison of MPI-only vs. Hybrid MPI+OpenMP implementations on a 64-CPU compute node

	MPI only	MPI+OpenMP	
MPI ranks	CPU time (s)	Threads	CPU time (s)
1	14660.09	64	1073.25
2	7381.27	32	647.19
4	3701.84	16	496.95
8	1902.68	8	415.08
16	1059.68	4	374.40
32	712.67	2	362.51
64	367.56	1	360.97

The results in Table 6.5 confirm the general observation that MPI typically provides better parallel scaling than OpenMP due to forced data locality. In our specific implementation, where OpenMP parallelization is limited to the matrix-free matrix-vector multiplication routines, we find that a configuration of 32 MPI ranks with 2 threads per rank achieves performance comparable to that of 64 MPI ranks. This configuration allows for 8 GB of memory per core, effectively doubling the available memory per core without significant performance degradation. An alternative but still effective choice is 16 MPI ranks with 4 threads per rank, which allows for 16 GB of memory per core. The constant CPU time across these configurations demonstrates that the performance of our implementation remains stable regardless of whether 2 or 4 threads are used per MPI rank.

We conducted strong scaling tests of the hybrid MPI+OpenMP approach solving the 3D SEG/EAGE Salt model problem with a grid size of $1281 \times 1281 \times 385$ and $f = 10\,\text{Hz}$. Based on our previous findings, we employed a configuration of 32 MPI ranks with 2 threads per rank, which balances performance and memory utilization effectively.

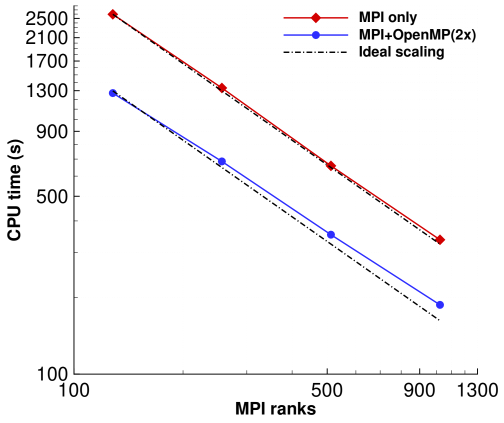


Figure 6.6: Strong scaling of pure MPI implementation and hybrid MPI+OpenMP implementation with 2 threads per rank. Four-level deflation method is used to solve the SEG/EAGE Salt model with a grid size of $1281 \times 1281 \times 385$ and $f = 10\,\text{Hz}$

Figure 6.6 illustrates the strong scaling results of both our pure MPI implementation and the hybrid MPI+OpenMP implementation. The results demonstrate the superior performance of our hybrid implementation. With 2 threads per rank, the CPU time is consistently about half of that required by the pure MPI implementation with the same number of MPI ranks. This performance gain is particularly noteworthy as it comes without sacrificing scalability; in fact, the hybrid implementation exhibits near-perfect scaling up to 2048 CPUs. This impressive scalability can be attributed to efficient load balancing between MPI processes and OpenMP threads.

This hybrid MPI+OpenMP implementation enhances our ability to tackle larger problem sizes. By reducing the number of MPI ranks and utilizing OpenMP threads, we can effectively double the available memory per MPI process without sacrificing computational efficiency. This capability is crucial for simulating wave propagation in very large or highly detailed geological models, where memory constraints often pose significant challenges.

6.3.3. GO_3D_OBS Model Problem

Having demonstrated the effectiveness of our multilevel deflation method for the SEG/EAGE Salt model, we now turn to the more challenging GO_3D_OBS model, which presents a significantly more complex velocity structure representative of a subduction zone environment. This model serves as an excellent test case for evaluating our method's capability to handle realistic crustal-scale wave propagation problems while also revealing important computational challenges that need to be addressed.

Figure 6.7 shows the wavefields computed in the GO_3D_OBS model at 3 Hz. The wavefields reveal intricate wave propagation patterns through the subduction zone structure. The ability of our method to capture these complex wave behaviors in heterogeneous media further showcases its applicability to realistic geophysical scenarios.

Performance tests with a five-level deflation method on a $401 \times 2001 \times 593$ grid at 3 Hz demonstrates promising yet challenging results. Figure 6.8 illustrates the strong scaling results of our hybrid MPI+OpenMP implementation with 2 threads per rank. While parallel efficiency remains satisfactory up to moderate core counts, we observe a notable decrease at 576 MPI ranks, attributable to the reduced grid size per MPI rank and increased data communication overhead.

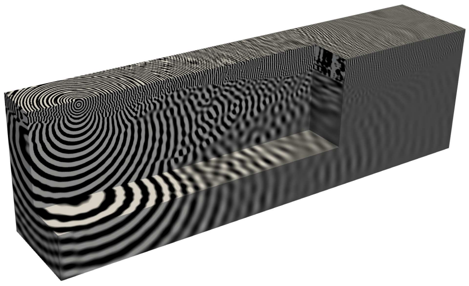


Figure 6.7: The wavefields computed in GO_3D_OBS model at 3 Hz

Comparative analysis reveals that solving the GO_3D_OBS model requires significantly more computational effort than the SEG/EAGE Salt model, despite the latter having a larger grid size ($1281 \times 1281 \times 385$ at 10 Hz). The GO_3D_OBS model requires 29 outer iterations for convergence. More critically, we observe that iterations on the second level consistently reach the predefined maximum limit (100) rather than achieving the desired tolerance of 10^{-1}. This convergence challenge for the coarse-level problems becomes even more pronounced when scaling to the grid size $801 \times 4001 \times 1185$. For instance, it requires over 300 iterations to reach the tolerance of 10^{-1} on the second level, which is computationally prohibitive for practical applications.

These observations suggest that the complexity of the GO_3D_OBS model leads to a larger cluster of near-zero eigenvalues in the resulting linear system, a challenge

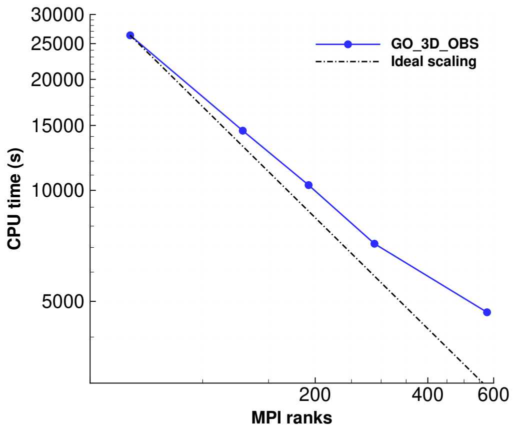


Figure 6.8: Strong scaling of hybrid MPI+OpenMP implementation with 2 threads per rank. Five-level deflation method is used to solve the GO_3D_OBS model with a grid size of $401 \times 2001 \times 593$ at 3 Hz

that becomes increasingly pronounced with larger grid sizes in three dimensions [72, 73, 76]. The flexible configuration of our multilevel deflation method enables the adjustment of specific components to effectively handle such extreme-scale problems while maintaining reasonable convergence rates.

We suggest two complementary strategies that leverage the flexibility of our multilevel deflation method. These strategies are specifically designed to handle the larger cluster of near-zero eigenvalues characteristic of complex heterogeneous problems at extreme scales. The first strategy involves implementing a more accurate approximation of the multigrid-based CSLP, particularly on the first and second levels where convergence issues are most pronounced. As investigated in [125], employing either two V-cycles or an F-cycle to approximate the CSLP can significantly accelerate convergence.

The second strategy focuses on optimizing the deflation preconditioner through a proper choice of the parameter γ in the deflation preconditioner defined as (4.2). The term γQ performs a spectral transformation whereby zero eigenvalues in the spectrum of $P_{\text{ADP}}A|_{\gamma=0}$ are mapped to γ in the spectrum of $P_{\text{ADP}}A|_{\gamma\neq0}$ [73, 82]. By selecting an appropriate value for γ, we can maintain reasonable convergence rates while significantly reducing the number of iterations required on the second level, thereby decreasing the overall computational time. This adjustment is particularly effective for the GO_3D_OBS model, where the second-level iterations were previously reaching the maximum limit without achieving the desired tolerance.

We have investigated the impact of the γ parameter on solver performance using a grid size of $401 \times 2001 \times 593$ at 3 Hz with 576 MPI ranks and 2 threads per rank. As shown in Table 6.6, the choice of γ significantly influences both convergence behavior and computational efficiency. With $\gamma = 1.5$, we achieve optimal performance, reducing the number of outer iterations from 29 to 25 and decreasing the average second-level iterations from 97 to 73, resulting in a substantial reduction in CPU time.

Table 6.6: Impact of γ parameter on solver performance for the GO_3D_OBS model ($401 \times 2001 \times 593$ grid points, 3 Hz). Results show the number of outer FGMRES iterations (#iter), average iterations per outer iteration on second level (L2 #iter), and total CPU time.

γ	#iter	L2 #iter	CPU time (s)
1.0	29	97	5114.29
1.5	25	73	3124.98
2.0	27	74	3401.24
2.5	28	85	3831.82
3.0	29	92	4312.34

Table 6.7: Five-level deflation performance for the GO_3D_OBS model at various frequencies, using 576 MPI ranks with 2 threads per rank

Grid size	f (Hz)	γ	#iter	CPU time (s)
401×2001×593	2.5	1.5	23	2589
401×2001×593	3.0	1.5	25	3124
801×4001×1185	4.0	3.0	29	22098
801×4001×1185	5.0	3.0	34	22006

Building upon these established optimizing strategies, we proceed to demonstrate the capability of our method to resolve the GO_3D_OBS model at an unprecedented scale. Table 6.7 presents the number of outer FGMRES iterations required to solve the GO_3D_OBS model with increasing frequencies and corresponding grid sizes. Using two V-cycles for approximating CSLP and $\gamma = 3$, our method successfully handles the challenging case with grid size $801 \times 4001 \times 1185$ at 5 Hz, representing approximately 3.8 billion degrees of freedom. The solution converges within 34 outer iterations, though second-level iterations do not reach a 10^{-1} tolerance within 100 iterations, demonstrating the effectiveness of our parameter tuning strategy in maintaining reasonable convergence rates even at extreme scales.

The parallel performance results further validate the practical viability of our approach. Using 576 MPI ranks with 2 threads per rank, the solution time is 22006 seconds, while doubling the computational resources to 1152 MPI ranks reduces this to 12607 seconds. This translates to an impressive 87% parallel efficiency when scaling from 1152 to 2304 CPU cores, confirming the method's strong scaling capabilities for extreme-scale problems.

With at least 12 grid points per wavelength and a convergence tolerance of 10^{-6}, the resulting solution serves as a benchmark for high-resolution 3D seismic modeling in complex geological settings. Figure 6.9 illustrates the solution quality through multiple perspectives: the velocity profile reveals the complex structural features of the subduction zone, while the wavefield cross-sections at different depths (7.5 km, 12.5 km, and 19.5 km) demonstrate the method's ability to capture wave propagation through varying geological features, from the accretionary wedge through the subduction megathrust to the Moho discontinuity.

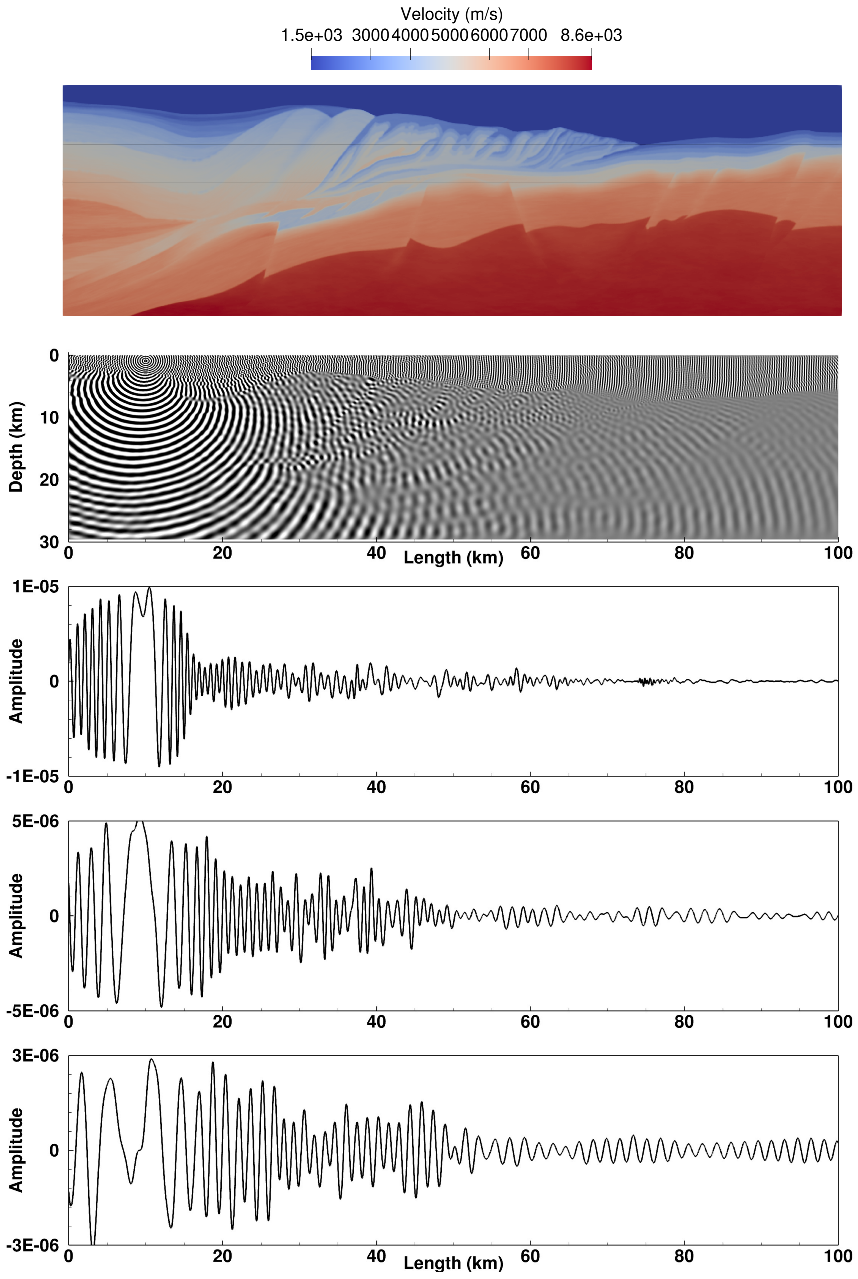


Figure 6.9: Solution for the GO_3D_OBS model ($401 \times 2001 \times 593$ grid points, 3 Hz). From top to bottom: P-wave velocity distribution at $x = 10$ km, showing key geological features; Computed wavefield at $x = 10$ km; Wavefield profiles along three horizontal lines crossing the source position at different depths: accretionary wedge ($z = 7.5$ km), subduction megathrust ($z = 12.5$ km), and Moho discontinuity ($z = 19.5$ km). Solid lines in the velocity profile indicate profile locations.

These comprehensive numerical results establish that our multilevel deflation method, enhanced by proper parameter selection and efficient parallel implementation, successfully addresses the challenges posed by extreme-scale Helmholtz problems in realistic geological settings. The combination of robust convergence properties, excellent parallel scalability, and high solution accuracy positions our method as a practical tool for large-scale seismic modeling applications.

6.4. CONCLUSIONS

In this chapter, we have presented a robust and scalable parallel solver for large-scale three-dimensional Helmholtz problems based on a high-order multilevel deflation method. The key innovations of our approach include a matrix-free implementation utilizing re-discretized schemes for the Galerkin coarsening operator, and a hybrid MPI+OpenMP parallelization framework that effectively addresses both computational and memory challenges in extreme-scale scenarios. Through these developments and demonstrated results, we have successfully realized all objectives outlined in Section 1.7.

Our comprehensive numerical experiments, progressing from constant wavenumber problems to the complex SEG/EAGE Salt and GO_3D_OBS models, demonstrate several significant achievements. First, the solver exhibits close-to wavenumber-independent convergence across varying frequencies and problem sizes, a critical property for seismic applications. Second, by selecting an appropriate shift term for the deflation preconditioner, we successfully addressed the challenges posed by enlarged clusters of near-zero eigenvalues characteristic of complex heterogeneous problems. Third, our hybrid parallelization approach achieves excellent scalability while optimizing memory utilization, as evidenced by the 86% parallel efficiency when scaling from 1152 to 2304 CPU cores in solving the GO_3D_OBS model with approximately 3.8 billion degrees of freedom.

The successful resolution of the GO_3D_OBS subduction zone model, representing a state-of-the-art problem at unprecedented scale in seismic modeling, validates our method's practical viability. Our solver's combination of robust convergence properties, excellent parallel scalability, and ability to handle realistic geological complexity positions it as a powerful forward modeling engine for FWI and other seismic imaging applications.

Building upon these achievements, several promising directions emerge for future research. The development of more sophisticated coarse-grid re-discretization strategies may lead to improved convergence rates and computational efficiency. The framework established in this work could be extended to handle even larger-scale problems through GPU acceleration. Furthermore, theoretical investigation of the shift-term parameter selection could provide rigorous foundations for parameter optimization, enhancing the solver's robustness across diverse geological settings.

7
Conclusions and Outlook

7.1. Conclusions

This dissertation has addressed fundamental challenges in solving large-scale Helmholtz problems through the development of high-performance iterative methods. Our research has focused on creating efficient solution strategies that combine robust numerical properties with practical applicability for modern computing architectures. The work has resulted in significant advances in matrix-free implementations, parallel computing frameworks, and preconditioning techniques for time-harmonic wave problems described by the Helmholtz equation.

The investigation began with the development of a matrix-free parallel framework for CSLP-preconditioned Krylov subspace methods, initially demonstrated in two dimensions and subsequently extended to three-dimensional problems. This framework established the foundation for efficient memory utilization and parallel efficiency while maintaining computational accuracy and convergence properties. The successful implementation for both homogeneous and heterogeneous media demonstrated the versatility and practical utility of the framework.

Building upon this foundation, we introduced novel approaches to two-level deflation preconditioning, incorporating higher-order deflation vectors and innovative matrix-free implementations of coarse-grid operators. A significant accomplishment was the formulation of novel re-discretization methods founded on Galerkin coarsening principles, coupled with a thorough investigation of coarse-grid solution precision requirements. This advancement enabled convergence independent of wavenumber while maintaining both matrix-free computational advantages and computational efficiency. The extension to multilevel deflation, whose components have been systematically optimized through extensive numerical experiments to achieve close-to wavenumber-independent convergence, significantly augments the capability of this method in addressing large-scale computational problems.

This research finally resulted in a robust parallel multilevel deflation solver, implemented through a hybrid MPI+OpenMP framework. This solver successfully addressed the computational challenges of three-dimensional heterogeneous Helmholtz problems, demonstrating excellent parallel scalability and memory efficiency. The successful application to the GO_3D_OBS subduction zone model, with approximately 3.8 billion degrees of freedom, validates the practical viability of this approach for extreme-scale problems.

Several key theoretical and practical contributions emerge from this work. First, the development of matrix-free implementations for CSLP and deflation preconditioning has addressed the real large-scale problems efficiently by reduced memory requirements. Second, the achievement of near-wavenumber-independent convergence through carefully designed parallel deflation strategies represents a substantial advancement in solving high-frequency wave problems. Third, the hybrid parallelization framework has demonstrated exceptional scalability, achieving 86% parallel efficiency on thousands of CPU cores. This research has particular significance for applications like seismic imaging, where the ability to handle large-scale heterogeneous problems efficiently is crucial. Through these achievements, we have successfully realized all objectives outlined in Section 1.7.

Through a combination of mathematical rigor, algorithmic innovation, and careful parallel implementation, this research achieves efficient, scalable solutions to large-scale Helmholtz problems, enhancing the scope of reliable and fast forward modeling in wave propagation. The results point toward a future in which deeper integration of preconditioning techniques and high-performance computing can push the limits of solving the high-frequency Helmholtz problems. The methods outlined here provide a strong platform for any subsequent work in academic or industrial environments where high-performance wave modeling are paramount. The methodologies and frameworks developed in this dissertation contribute to the broader field of scientific computing, demonstrating how careful algorithm design, combined with modern computing architectures, can address previously intractable problems. As computational demands continue to grow, these approaches provide a possible direction for future developments in high-performance computational methods for problems with similar challenges.

7.2. OUTLOOK

The research presented in this dissertation opens several promising avenues for future investigation, spanning theoretical developments, computational enhancements, and extended applications. From a numerical methodology perspective, the extension to high-order finite difference schemes with minimized dispersion represents a natural evolution of our approach. Such developments would further enhance the capability to handle high-frequency wave propagation while maintaining the advantages of our matrix-free parallel framework.

A crucial area for theoretical advancement lies in the development of improved re-discretization schemes for the coarse-grid operator in higher-order deflation methods. The challenge is to design compact stencils that preserve the alignment of the near-zero eigenvalues with the fine-grid operator while enabling better parallel efficiency. This mathematical investigation would strengthen the theoretical foundations of our approach while yielding practical performance benefits.

The optimization of our computational framework presents another significant research direction. The incorporation of GPU acceleration into our hybrid parallelization strategy could substantially enhance computational performance. This advancement would require careful algorithm adaptation to leverage the unique characteristics of GPU architectures while preserving the numerical properties of our methods.

A theoretical investigation into the optimal selection of shift parameters for deflation preconditioning merits particular attention. Establishing rigorous mathematical foundations for parameter choice would enhance the robustness of our methods across diverse problem settings. This analysis could lead to adaptive parameter selection strategies that automatically optimize performance based on problem characteristics.

Furthermore, the investigation of advanced domain decomposition strategies and enhanced multilevel deflation methods could enable a more efficient solution for extreme-scale systems.

Last but not least, the extension of our methodology to broader applications in wave propagation presents compelling opportunities. The framework developed here could be adapted to address elastic wave equations, Maxwell equations, and frequency-domain seismic imaging in the context of FWI, potentially revolutionizing approaches to these challenging problems.

These future research directions collectively aim to advance both the theoretical understanding and practical applicability of high-performance iterative methods for wave propagation problems. The combination of mathematical rigor, algorithmic innovation, and careful parallel implementation, and practical applicability will remain central to these developments, continuing the trajectory established in this dissertation.

Bibliography

[1] H. Helmholtz. ‘Theorie der Luftschwingungen in Röhren mit offenen Enden.’ In: *Journal für die reine und angewandte Mathematik* 1860.57 (1860), pp. 1–72. DOI: doi:10.1515/crll.1860.57.1.

[2] R. Green. ‘The seismic refraction method - a review’. In: *Geoexploration* 12.4 (1974), pp. 259–284. DOI: 10.1016/0016-7142(74)90015-5.

[3] H. Gohe and T. Wedekind. ‘Der ultraschall in der medizin’. In: *Klinische Wochenschrift* 19 (1940), pp. 25–29. DOI: 10.1007/BF01772492.

[4] A. Bayliss, C. Goldstein and E. Turkel. ‘The Numerical Solution of the Helmholtz Equation for Wave Propagation Problems in Underwater Acoustics’. In: *Computers & Mathematics with Applications* 11.7-8 (1985), pp. 655–665. DOI: 10.1016/0898-1221(85)90162-2.

[5] R. G. Pratt. ‘Seismic Waveform Inversion in the Frequency Domain, Part 1: Theory and Verification in a Physical Scale Model’. In: *GEOPHYSICS* 64.3 (1999), pp. 888–901. ISSN: 0016-8033. DOI: 10.1190/1.1444597.

[6] L. Sirgue and R. G. Pratt. ‘Efficient Waveform Inversion and Imaging: A Strategy for Selecting Temporal Frequencies’. In: *GEOPHYSICS* 69.1 (2004), pp. 231–248. ISSN: 0016-8033. DOI: 10.1190/1.1649391.

[7] J. Virieux and S. Operto. ‘An Overview of Full-Waveform Inversion in Exploration Geophysics’. In: *GEOPHYSICS* 74.6 (2009), WCC1–WCC26. DOI: 10.1190/1.3238367.

[8] A. Stanziola, S. R. Arridge, B. T. Cox and B. E. Treeby. ‘A Helmholtz Equation Solver Using Unsupervised Learning: Application to Transcranial Ultrasound’. In: *Journal of Computational Physics* 441 (2021), p. 110430. DOI: 10.1016/j.jcp.2021.110430.

[9] P.-H. Tournier, M. Bonazzoli, V. Dolean, F. Rapetti, F. Hecht, F. Nataf, I. Aliferis, I. El Kanfoud, C. Migliaccio, M. De Buhan, M. Darbas, S. Semenov and C. Pichot. ‘Numerical Modeling and High-Speed Parallel Computing: New Perspectives on Tomographic Microwave Imaging for Brain Stroke Detection and Monitoring’. In: *IEEE Antennas and Propagation Magazine* 59.5 (2017), pp. 98–110. DOI: 10.1109/MAP.2017.2731199.

[10] M. Lax, W. H. Louisell and W. B. McKnight. ‘From Maxwell to paraxial wave optics’. In: *Physical Review A* 11.4 (1975), pp. 1365–1370. DOI: 10.1103/PhysRevA.11.1365.

[11] A. Bulygin, I. Meglinski and Y. Kistenev. 'Non-Paraxial Effects in the Laser Beams Sharply Focused to Skin Revealed by Unidirectional Helmholtz Equation Approximation'. In: *Photonics* 10.8 (2023), p. 907. DOI: 10.3390/photonics10080907.

[12] V. Dwarka and C. Vuik. 'Pollution and accuracy of solutions of the Helmholtz equation: A novel perspective from the eigenvalues'. In: *Journal of Computational and Applied Mathematics* 395 (2021), p. 113549.

[13] R. Plessix and W. Mulder. 'Separation-of-variables as a preconditioner for an iterative Helmholtz solver'. In: *Applied Numerical Mathematics* 44.3 (2003), pp. 385–400.

[14] R. J. Versteeg and G. Grau, eds. *The Marmousi experience*. Proceedings of the 1990 EAEG workshop on Practical Aspects of Seismic Data Inversion, Eur. Ass. Expl. Geophys, 1991.

[15] D. Colton and R. Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer, 1994. DOI: 10.1007/978-1-4614-4942-3.

[16] 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).

[17] A. Górszczyk and S. Operto. 'GO_3D_OBS: the multi-parameter benchmark geomodel for seismic imaging method assessment and next-generation 3D survey design (version 1.0)'. In: *Geoscientific Model Development* 14.3 (2021), pp. 1773–1799. DOI: 10.5194/gmd-14-1773-2021.

[18] K. J. Marfurt. 'Accuracy of Finite-difference and Finite-element Modeling of the Scalar and Elastic Wave Equations'. In: *GEOPHYSICS* 49.5 (1984), pp. 533–549. DOI: 10.1190/1.1441689.

[19] A. Bayliss, C. I. Goldstein and E. Turkel. 'On Accuracy Conditions for the Numerical Computation of Waves'. In: *Journal of Computational Physics* 59.3 (1985), pp. 396–404. DOI: 10.1016/0021-9991(85)90119-6.

[20] I. M. Babuska and S. A. Sauter. 'Is the Pollution Effect of the FEM Avoidable for the Helmholtz Equation Considering High Wave Numbers?' In: *SIAM Journal on Numerical Analysis* 34.6 (1997), pp. 2392–2423. DOI: 10.1137/S0036142994269186.

[21] K. Wang and Y. S. Wong. 'Pollution-Free Finite Difference Schemes for Non-Homogeneous Helmholtz Equation'. In: *International Journal of Numerical Analysis & Modeling* 11.4 (2014).

[22] I. Singer and E. Turkel. 'High-Order Finite Difference Methods for the Helmholtz Equation'. In: *Computer Methods in Applied Mechanics and Engineering* 163.1-4 (1998), pp. 343–358. DOI: 10.1016/S0045-7825(98)00023-1.

[23] E. Turkel, D. Gordon, R. Gordon and S. Tsynkov. 'Compact 2D and 3D sixth order schemes for the Helmholtz equation with variable wave number'. In: *Journal of Computational Physics* 232.1 (2013), pp. 272–287. DOI: 10.1016/j.jcp.2012.08.016.

[24] T. Wu and R. Xu. 'An Optimal Compact Sixth-Order Finite Difference Scheme for the Helmholtz Equation'. In: *Computers & Mathematics with Applications* 75.7 (2018), pp. 2520–2537. DOI: 10.1016/j.camwa.2017.12.023.

[25] R. Itzá Balam and M. Uh Zapata. 'A New Eighth-Order Implicit Finite Difference Method to Solve the Three-Dimensional Helmholtz Equation'. In: *Computers & Mathematics with Applications* 80.5 (Sept. 2020), pp. 1176–1200. DOI: 10.1016/j.camwa.2020.06.011.

[26] C.-H. Jo, C. Shin and J. H. Suh. 'An optimal 9-point, finite-difference, frequency-space, 2-D scalar wave extrapolator'. In: *GEOPHYSICS* 61.2 (1996), pp. 529–537. DOI: 10.1190/1.1443979.

[27] Z. Chen, D. Cheng and T. Wu. 'A dispersion minimizing finite difference scheme and preconditioned solver for the 3D Helmholtz equation'. In: *Journal of Computational Physics* 231.24 (2012), pp. 8152–8175. DOI: 10.1016/j.jcp.2012.07.048.

[28] Z. Wu and T. Alkhalifah. 'A Highly Accurate Finite-Difference Method with Minimum Dispersion Error for Solving the Helmholtz Equation'. In: *Journal of Computational Physics* 365 (2018), pp. 350–361. DOI: 10.1016/j.jcp.2018.03.046.

[29] C. C. Stolk. 'A dispersion minimizing scheme for the 3-D Helmholtz equation based on ray theory'. In: *Journal of Computational Physics* 314 (2016), pp. 618–646. DOI: 10.1016/j.jcp.2016.03.023.

[30] P.-H. Cocquet, M. J. Gander and X. Xiang. 'A Finite Difference Method with Optimized Dispersion Correction for the Helmholtz Equation'. In: *Domain Decomposition Methods in Science and Engineering XXIV*. Ed. by P. E. Bjørstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman and O. B. Widlund. Cham: Springer International Publishing, 2018, pp. 205–213. DOI: 10.1007/978-3-319-93873-8_18.

[31] H. S. Aghamiry, A. Gholami, L. Combe and S. Operto. 'Accurate 3D Frequency-Domain Seismic Wave Modeling with the Wavelength-Adaptive 27-Point Finite-Difference Stencil: A Tool for Full-Waveform Inversion'. In: *GEOPHYSICS* 87.3 (May 2022), R305–R324. DOI: 10.1190/geo2021-0606.1.

[32] 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*. Ed. by I. G. Graham, T. Y. Hou, O. Lakkis and R. Scheichl. Vol. 83. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 325–363. DOI: 10.1007/978-3-642-22061-6_10.

[33] F. Sourbier, A. Haidar, L. Giraud, H. Ben-Hadj-Ali, S. Operto and J. Virieux. 'Three-dimensional Parallel Frequency-domain Visco-acoustic Wave Modelling Based on a Hybrid Direct/Iterative Solver'. In: *Geophysical Prospecting* 59.5 (2011), pp. 834–856. DOI: 10.1111/j.1365-2478.2011.00966.x.

[34] S. Operto, J. Virieux, P. Amestoy, J.-Y. L'Excellent, L. Giraud and H. B. H. Ali. '3D Finite-Difference Frequency-Domain Modeling of Visco-Acoustic Wave Propagation Using a Massively Parallel Direct Solver: A Feasibility Study'. In: *GEOPHYSICS* 72.5 (2007), SM195–SM211. DOI: 10.1190/1.2759835.

[35] M. R. Hestenes and E. Stiefel. 'Methods of conjugate gradients for solving linear systems'. In: *Journal of Reasearch of the National Bureau of Standards* 49.6 (1952).

[36] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[37] C. C. Paige and M. A. Saunders. 'Solution of Sparse Indefinite Systems of Linear Equations'. In: *SIAM Journal on Numerical Analysis* 12.4 (1975), pp. 617–629. DOI: 10.1137/0712047.

[38] R. Fletcher. 'Conjugate Gradient Methods for Indefinite Systems'. In: *Numerical Analysis*. Ed. by G. A. Watson. Berlin, Heidelberg: Springer, 1976, pp. 73–89. DOI: 10.1007/BFb0080116.

[39] H. A. van der Vorst. 'Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems'. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. DOI: 10.1137/0913035.

[40] S. C. Eisenstat, H. C. Elman and M. H. Schultz. 'Variational Iterative Methods for Nonsymmetric Systems of Linear Equations'. In: *SIAM Journal on Numerical Analysis* 20.2 (1983), pp. 345–357. DOI: 10.1137/0720023.

[41] Y. Saad and M. H. Schultz. 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems'. In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869. DOI: 10.1137/0907058.

[42] H. A. Van der Vorst and C. Vuik. 'GMRESR: A Family of Nested GMRES Methods'. In: *Numerical Linear Algebra with Applications* 1.4 (1994), pp. 369–386. DOI: 10.1002/nla.1680010404.

[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'. In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 1035–1062. DOI: 10.1137/070685804.

[44] H. Knibbe, C. W. Oosterlee and C. Vuik. 'GPU implementation of a Helmholtz Krylov solver preconditioned by a shifted Laplace multigrid method'. In: *Journal of Computational and Applied Mathematics* 236.3 (2011), pp. 281–293. DOI: 10.1016/j.cam.2011.07.021.

[45] M. B. Van Gijzen and P. Sonneveld. 'Algorithm 913: An Elegant IDR(s) Variant That Efficiently Exploits Biorthogonality Properties'. In: *ACM Transactions on Mathematical Software (TOMS)* 38.1 (2011), 5:1–5:19. DOI: 10.1145/2049662.2049667.

[46] J. W. Pearson and J. Pestana. 'Preconditioners for Krylov Subspace Methods: An Overview'. In: *GAMM-Mitteilungen* 43.4 (2020), e202000015. DOI: 10.1002/gamm.202000015.

[47] A. Greenbaum, V. Pták and Z. Strakos. 'Any Nonincreasing Convergence Curve Is Possible for GMRES'. In: *SIAM Journal on Matrix Analysis and Applications* 17.3 (1996), pp. 465–469. ISSN: 0895-4798. DOI: 10.1137/S0895479894275030.

[48] R.-E. Plessix. 'A Helmholtz Iterative Solver for 3D Seismic-Imaging Problems'. In: *GEOPHYSICS* 72.5 (2007), SM185–SM194. DOI: 10.1190/1.2738849.

[49] Y. A. Erlangga. 'A robust and efficient iterative method for the numerical solution of the Helmholtz equation'. PhD Thesis. Delft University of Technology, 2005. URL: http://resolver.tudelft.nl/uuid:af9be715-6ebf-4fc1-b948-ebd9d2c4167b.

[50] M. Gander and F. Nataf. 'AILU for Helmholtz Problems: A New Preconditioner Based on an Analytic Factorization'. In: *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* 331.3 (2000), pp. 261–266. DOI: 10.1016/S0764-4442(00)01632-3.

[51] A. Bayliss, C. I. Goldstein and E. Turkel. 'An iterative method for the Helmholtz equation'. In: *Journal of Computational Physics* 49.3 (1983), pp. 443–457. DOI: 10.1016/0021-9991(83)90139-0.

[52] A. Laird and M. Giles. *Preconditioned iterative solution of the 2D Helmholtz equation*. Tech. rep. 2002.

[53] Y. A. Erlangga, C. Vuik and C. W. Oosterlee. 'On a class of preconditioners for solving the Helmholtz equation'. In: *Applied Numerical Mathematics* 50.3-4 (2004), pp. 409–425. DOI: 10.1016/j.apnum.2004.01.009.

[54] Y. A. Erlangga, C. W. Oosterlee and C. Vuik. 'A novel multigrid based preconditioner for heterogeneous Helmholtz problems'. In: *SIAM Journal on Scientific Computing* 27.4 (2006), pp. 1471–1492. DOI: 10.1137/040615195.

[55] D. Osei-Kuffuor and Y. Saad. 'Preconditioning Helmholtz Linear Systems'. In: *Applied Numerical Mathematics*. Special Issue: NUMAN 2008 60.4 (Apr. 2010), pp. 420–431. DOI: 10.1016/j.apnum.2009.09.003.

[56] T. Airaksinen, E. Heikkola, A. Pennanen and J. Toivanen. 'An Algebraic Multigrid Based Shifted-Laplacian Preconditioner for the Helmholtz Equation'. In: *Journal of Computational Physics* 226.1 (2007), pp. 1196–1210. DOI: 10.1016/j.jcp.2007.05.013.

[57] M. Bollhöfer, M. J. Grote and O. Schenk. 'Algebraic Multilevel Preconditioner for the Helmholtz Equation in Heterogeneous Media'. In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3781–3805. DOI: 10.1137/080725702.

[58] X. Liu, Y. Xi, Y. Saad and M. V. De Hoop. 'Solving the Three-Dimensional High-Frequency Helmholtz Equation Using Contour Integration and Polynomial Preconditioning'. In: *SIAM Journal on Matrix Analysis and Applications* 41.1 (2020), pp. 58–82. DOI: 10.1137/18M1228128.

[59] M. B. van Gijzen, Y. A. Erlangga and C. Vuik. 'Spectral Analysis of the Discrete Helmholtz Operator Preconditioned with a Shifted Laplacian'. In: *SIAM Journal on Scientific Computing* 29.5 (2007), pp. 1942–1958. DOI: 10.1137/060661491.

[60] 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?' In: *Numerische Mathematik* 131.3 (2015), pp. 567–614. DOI: 10.1007/s00211-015-0700-2.

[61] P.-H. Cocquet and M. J. Gander. 'How Large a Shift Is Needed in the Shifted Helmholtz Preconditioner for Its Effective Inversion by Multigrid?' In: *SIAM Journal on Scientific Computing* 39.2 (2017), A438–A478. DOI: 10.1137/15M102085X.

[62] R. A. Nicolaides. 'Deflation of conjugate gradients with applications to boundary value problems'. In: *SIAM Journal on Numerical Analysis* 24.2 (1987), pp. 355–365. DOI: 10.1137/0724027.

[63] Z. Dostál. 'Conjugate gradient method with preconditioning by projector'. In: *International Journal of Computer Mathematics* 23.3-4 (1988), pp. 315–323. DOI: 10.1080/00207168808803625.

[64] L. Mansfield. 'Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers'. In: *SIAM journal on scientific and statistical computing* 12.6 (1991), pp. 1314–1323. DOI: 10.1137/0912071.

[65] L. Y. Kolotilina. 'Twofold deflation preconditioning of linear algebraic systems. I. Theory'. In: *Journal of Mathematical Sciences* 89.6 (1998), pp. 1652–1689. DOI: 10.1007/BF02355371.

[66] C. Vuik, A. Segal and J. Meijerink. 'An Efficient Preconditioned CG Method for the Solution of a Class of Layered Problems with Extreme Contrasts in the Coefficients'. In: *Journal of Computational Physics* 152.1 (1999), pp. 385–403. DOI: 10.1006/jcph.1999.6255.

[67] Y. Saad, M. Yeung, J. Erhel and F. Guyomarc'h. 'A deflated version of the conjugate gradient algorithm'. In: *SIAM Journal on Scientific Computing* 21.5 (2000), pp. 1909–1926. DOI: 10.1137/S1064829598339761.

[68] J. Frank and C. Vuik. 'On the Construction of Deflation-Based Preconditioners'. In: *SIAM Journal on Scientific Computing* 23.2 (2001), pp. 442–462. DOI: 10.1137/S1064827500373231.

[69] R. B. Morgan. 'GMRES with Deflated Restarting'. In: *SIAM Journal on Scientific Computing* 24.1 (2002), pp. 20–37. DOI: 10.1137/S1064827599364659.

[70] J. M. Tang. 'Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems'. PhD Thesis. Delft University of Technology, 2008. URL: https://resolver.tudelft.nl/uuid:e8c5f63b-ee7d-4a59-90da-a8025f5f88b0.

[71] Y. A. Erlangga and R. Nabben. 'On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian'. In: *Electronic Transactions on Numerical Analysis* 31.403-424 (2008), p. 3.

[72] A. H. Sheikh, D. Lahaye and C. Vuik. ‘On the Convergence of Shifted Laplace Preconditioner Combined with Multilevel Deflation’. In: *Numerical Linear Algebra with Applications* 20.4 (2013), pp. 645–662. DOI: 10.1002/nla.1882.

[73] A. Sheikh, D. Lahaye, L. Garcia Ramos, R. Nabben and C. Vuik. ‘Accelerating the Shifted Laplace Preconditioner for the Helmholtz Equation by Multilevel Deflation’. In: *Journal of Computational Physics* 322 (2016), pp. 473–490. DOI: 10.1016/j.jcp.2016.06.025.

[74] A. H. Sheikh. ‘Development of the Helmholtz Solver based on a Shifted Laplace Preconditioner and a Multigrid Deflation technique’. PhD Thesis. Delft University of Technology, 2014. DOI: 10.4233/uuid:1020f418-b488-4435-81ee-2b4f6a5024e1.

[75] V. Dwarka and C. Vuik. ‘Scalable convergence using two-level deflation preconditioning for the Helmholtz equation’. In: *SIAM Journal on Scientific Computing* 42.2 (2020), A901–A928. DOI: 10.1137/18M1192093.

[76] V. Dwarka and C. Vuik. ‘Scalable Multi-Level Deflation Preconditioning for Highly Indefinite Time-Harmonic Waves’. In: *Journal of Computational Physics* 469 (2022), p. 111327. DOI: 10.1016/j.jcp.2022.111327.

[77] K. Burrage, J. Erhel, B. Pohl and A. Williams. ‘A Deflation Technique for Linear Systems of Equations’. In: *SIAM Journal on Scientific Computing* 19.4 (1998), pp. 1245–1260. DOI: 10.1137/S1064827595294721.

[78] A. Chapman and Y. Saad. ‘Deflated and Augmented Krylov Subspace Techniques’. In: *Numerical Linear Algebra with Applications* 4.1 (1997), pp. 43–66. DOI: 10.1002/(SICI)1099-1506(199701/02)4:1<43::AID-NLA99>3.0.CO;2-Z.

[79] M. Clemens, M. Wilke, R. Schuhmann and T. Weiland. ‘Subspace Projection Extrapolation Scheme for Transient Field Simulations’. In: *IEEE Transactions on Magnetics* 40.2 (2004), pp. 934–937. DOI: 10.1109/TMAG.2004.824583.

[80] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson and S. Maiti. ‘Recycling Krylov Subspaces for Sequences of Linear Systems’. In: *SIAM Journal on Scientific Computing* 28.5 (2006), pp. 1651–1674. DOI: 10.1137/040607277.

[81] C. Vuik, A. Segal, L. el Yaakoubi and E. Dufour. ‘A Comparison of Various Deflation Vectors Applied to Elliptic Problems with Discontinuous Coefficients’. In: *Applied Numerical Mathematics.* Developments and Trends in Iterative Methods for Large Systems of Equations - in Memorium Rudiger Weiss 41.1 (2002), pp. 219–233. DOI: 10.1016/S0168-9274(01)00118-0.

[82] J. M. Tang, R. Nabben, C. Vuik and Y. A. Erlangga. ‘Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods’. In: *Journal of scientific computing* 39.3 (2009), pp. 340–370. DOI: 10.1007/s10915-009-9272-6.

[83] Y. A. Erlangga and R. Nabben. ‘Multilevel projection-based nested Krylov iteration for boundary value problems’. In: *SIAM J. Sci. Comput.* 30.3 (2008), pp. 1572–1595. DOI: 10.1137/070684550.

[84] V. Dolean, P. Jolivet and F. Nataf. *An introduction to domain decomposition methods*. Algorithms, theory, and parallel implementation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015. ISBN: 978-1-611974-05-8. URL: http://dx.doi.org/10.1137/1.9781611974065.ch1.

[85] P.-L. Lions. 'On the Schwarz alternating method. I'. In: *First international symposium on domain decomposition methods for partial differential equations*. Vol. 1. Paris, France. 1988, p. 42.

[86] M. J. Gander. 'Optimized schwarz methods'. In: *SIAM Journal on Numerical Analysis* 44.2 (2006), pp. 699–731. DOI: 10.1137/S0036142903425409.

[87] B. Després. 'Méthodes de décomposition de domaine pour la propagation d'ondes en régime harmonique. Le théorème de Borg pour l'équation de Hill vectorielle'. PhD Thesis. Paris 9, 1991.

[88] X.-C. Cai and O. B. Widlund. 'Domain decomposition algorithms for indefinite elliptic problems'. In: *SIAM Journal on Scientific and Statistical Computing* 13.1 (1992), pp. 243–258. DOI: 10.1137/0913013.

[89] J. Li and X. Tu. 'Convergence analysis of a balancing domain decomposition method for solving a class of indefinite linear systems'. In: *Numerical Linear Algebra with Applications* 16.9 (2009), pp. 745–773. DOI: 10.1002/nla.639.

[90] L. C. McInnes, R. F. Susan-Resiga and D. E. Keyes. 'Additive Schwarz Methods with Nonreflecting Boundary Conditions for the Parallel Computation of Helmholtz'. In: *Domain Decomposition Methods 10* 218 (1998), p. 325.

[91] J. Douglas Jr and D. B. Meade. 'Second-order transmission conditions for the Helmholtz equation'. In: *Ninth International Conference on Domain Decomposition Methods*. Citeseer. 1998, pp. 434–440.

[92] A. Toselli. 'Overlapping methods with perfectly matched layers for the solution of the Helmholtz equation'. In: *Eleventh International Conference on Domain Decomposition Methods*. Citeseer. 1999, pp. 551–558.

[93] F. Collino, S. Ghanemi and P. Joly. 'Domain decomposition method for harmonic wave propagation: a general presentation'. In: *Computer methods in applied mechanics and engineering* 184.2-4 (2000), pp. 171–211. DOI: 10.1016/S0045-7825(99)00228-5.

[94] M. J. Gander, F. Magoules and F. Nataf. 'Optimized Schwarz methods without overlap for the Helmholtz equation'. In: *SIAM Journal on Scientific Computing* 24.1 (2002), pp. 38–60. DOI: 10.1137/S1064827501387012.

[95] 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*. Ed. by O. B. Widlund and D. E. Keyes. Berlin, Heidelberg: Springer, 2007, pp. 205–212. DOI: 10.1007/978-3-540-34469-8_21.

[96] Y. Boubendir, X. Antoine and C. Geuzaine. 'A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation'. In: *Journal of Computational Physics* 231.2 (2012), pp. 262–280. DOI: 10.1016/j.jcp.2011.08.007.

[97] M. J. Gander and H. Zhang. 'Optimized Schwarz methods with overlap for the Helmholtz equation'. In: *SIAM Journal on Scientific Computing* 38.5 (2016), A3195–A3219. DOI: 10.1137/15M1021659.

[98] Z. Chen and X. Xiang. 'A source transfer domain decomposition method for Helmholtz equations in unbounded domain'. In: *SIAM Journal on Numerical Analysis* 51.4 (2013), pp. 2331–2356. DOI: 10.1137/130917144.

[99] W. Leng and L. Ju. 'An additive overlapping domain decomposition method for the Helmholtz equation'. In: *SIAM Journal on Scientific Computing* 41.2 (2019), A1252–A1277. DOI: 10.1137/18M1196170.

[100] C. C. Stolk. 'A rapidly converging domain decomposition method for the Helmholtz equation'. In: *Journal of Computational Physics* 241 (2013), pp. 240–252. DOI: 10.1016/j.jcp.2013.01.039.

[101] L. Zepeda-Núnez and L. Demanet. 'The method of polarized traces for the 2D Helmholtz equation'. In: *Journal of Computational Physics* 308 (2016), pp. 347–388. DOI: 10.1016/j.jcp.2015.11.040.

[102] L. Zepeda-Núnez, R. J. Hewett and L. Demanet. 'Preconditioning the 2D Helmholtz Equation with Polarized Traces'. In: *SEG Technical Program Expanded Abstracts 2014*. Denver, Colorado: Society of Exploration Geophysicists, Aug. 2014, pp. 3465–3470. DOI: 10.1190/segam2014-1275.1.

[103] N. Bootland, V. Dolean, P. Jolivet and P.-H. Tournier. 'A Comparison of Coarse Spaces for Helmholtz Problems in the High Frequency Regime'. In: *Comput. Math. Appl.* 98 (2021), pp. 239–253. DOI: 10.1016/j.camwa.2021.07.011.

[104] B. Engquist and L. Ying. 'Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers'. In: *Multiscale Model. Simul.* 9.2 (2011), pp. 686–710. DOI: 10.1137/100804644.

[105] 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'. In: *Siam Review* 61.1 (2019), pp. 3–76. DOI: 10.1137/16M109781X.

[106] E. Treister and R. Yovel. 'A hybrid shifted Laplacian multigrid and domain decomposition preconditioner for the elastic Helmholtz equations'. In: *Journal of Computational Physics* 497 (2024), p. 112622. DOI: 10.1016/j.jcp.2023.112622.

[107] M. J. Flynn. 'Some computer organizations and their effectiveness'. In: *IEEE transactions on computers* 100.9 (1972), pp. 948–960. DOI: 10.1109/TC.1972.5009071.

[108] MPI Forum. *MPI: A Message-Passing Interface Standard*. Technical Report. USA: University of Tennessee, 1994.

[109] L. Dagum and R. Menon. ‘OpenMP: An Industry Standard API for Shared-Memory Programming’. In: *IEEE Computational Science and Engineering* 5.1 (1998), pp. 46–55. DOI: 10.1109/99.660313.

[110] R. Rabenseifner, G. Hager and G. Jost. ‘Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes’. In: *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. 2009, pp. 427–436. DOI: 10.1109/PDP.2009.43.

[111] G. M. Amdahl. ‘Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities’. In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS ’67 (Spring). New York, NY, USA: Association for Computing Machinery, 1967, pp. 483–485. DOI: 10.1145/1465482.1465560.

[112] J. L. Gustafson. ‘Reevaluating Amdahl’s Law’. In: *Commun. ACM* 31.5 (1988), pp. 532–533. DOI: 10.1145/42411.42415.

[113] S. Williams, A. Waterman and D. Patterson. ‘Roofline: An Insightful Visual Performance Model for Multicore Architectures’. In: *Communications of the ACM* 52.4 (2009), pp. 65–76. DOI: 10.1145/1498765.1498785.

[114] M. Taus, L. Zepeda-Núñez, R. J. Hewett and L. Demanet. ‘L-Sweeps: A scalable, parallel preconditioner for the high-frequency Helmholtz equation’. In: *Journal of Computational Physics* 420 (2020), p. 109706. DOI: 10.1016/j.jcp.2020.109706.

[115] P.-H. Tournier, P. Jolivet, V. Dolean, H. S. Aghamiry, S. Operto and 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’. In: *GEOPHYSICS* (2022), pp. 1–84. DOI: 10.1190/geo2021-0702.1.

[116] 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’. In: *Journal of Computational Physics* 224.1 (2007), pp. 431–448. DOI: 10.1016/j.jcp.2007.03.033.

[117] D. Gordon and R. Gordon. ‘Robust and highly scalable parallel solution of the Helmholtz equation with large wave numbers’. In: *Journal of Computational and Applied Mathematics* 237.1 (2013), pp. 182–196. DOI: 10.1016/j.cam.2012.07.024.

[118] D. Gordon and R. Gordon. ‘Parallel Solution of High Frequency Helmholtz Equations Using High Order Finite Difference Schemes’. In: *Applied Mathematics and Computation* 218.21 (2012), pp. 10737–10754. DOI: 10.1016/j.amc.2012.04.052.

[119] 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’. In: *Numerical Linear Algebra with Applications* 20.4 (2013), pp. 663–688. DOI: 10.1002/nla.1860.

[120] 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*. Springer, 2017, pp. 141–155. ISBN: 978-3-319-28832-1. DOI: 10.1007/978-3-319-28832-1_6.

[121] G. Ortega, J. Lobera, I. García, M. Pilar Arroyo and E. M. Garzón. ‘Parallel Resolution of the 3D Helmholtz Equation Based on Multi-graphics Processing Unit Clusters’. In: *Concurrency and Computation: Practice and Experience* 27.13 (2015), pp. 3205–3219. DOI: 10.1002/cpe.3212.

[122] T. Bao and X. Feng. ‘A Parallel High-Order Accuracy Algorithm for the Helmholtz Equations’. In: *International Journal of Computer Mathematics* 101.1 (Jan. 2024), pp. 56–94. DOI: 10.1080/00207160.2024.2305143.

[123] 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’. In: *Computing and Visualization in Science* 11.3 (2007), pp. 139–146. DOI: 10.1007/s00791-007-0069-6.

[124] J. Chen, V. Dwarka and C. Vuik. ‘Matrix-Free Parallel Preconditioned Iterative Solvers for the 2D Helmholtz Equation Discretized with Finite Differences’. In: *Scientific Computing in Electrical Engineering*. Springer Nature Switzerland, 2024, pp. 61–68. DOI: 10.1007/978-3-031-54517-7_7.

[125] J. Chen, V. Dwarka and C. Vuik. ‘A matrix-free parallel solution method for the three-dimensional heterogeneous Helmholtz equation’. In: *Electronic Transactions on Numerical Analysis* 59 (2023), pp. 270–294. DOI: 10.1553/etna_vol59s270.

[126] L. R. Hocking and C. Greif. ‘Optimal Complex Relaxation Parameters in Multigrid for Complex-Shifted Linear Systems’. In: *SIAM Journal on Matrix Analysis and Applications* 42.2 (2021), pp. 475–502. DOI: 10.1137/20M1342161.

[127] D. (DHPC). *DelftBlue Supercomputer (Phase 1)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1. 2022.

[128] V. Dwarka. ‘Iterative methods for time-harmonic waves: towards accuracy and scalability’. PhD Thesis. Delft University of Technology, 2022. DOI: 10.4233/uuid:4d26359a-89ad-4a5a-9fac-cd8e9e59c743.

[129] 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. rep. ANL-21/39 - Revision 3.19. Argonne National Laboratory, 2023.

[130] 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*. Ed. by E. Arge, A. M. Bruaset and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202.

[131] 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. Oleynik, 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*. Ed. by H. Brunst, M. S. Müller, W. E. Nagel and M. M. Resch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–91. DOI: 10.1007/978-3-642-31476-6_7.

[132] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker and B. Mohr. 'The Scalasca performance toolset architecture'. In: *Concurrency and Computation: Practice and Experience* 22.6 (2010), pp. 702–719. DOI: https://doi.org/10.1002/cpe.1556.

[133] 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*. Ed. by H. Mix, C. Niethammer, H. Zhou, W. E. Nagel and M. M. Resch. Cham: Springer International Publishing, 2021, pp. 237–259. DOI: 10.1007/978-3-030-66057-4_13.

[134] J. Chen, V. Dwarka and C. Vuik. 'A Matrix-free parallel two-level deflation preconditioner for two-dimensional heterogeneous Helmholtz problems'. In: *Journal of Computational Physics* 514 (2024), p. 113264. DOI: 10.1016/j.jcp.2024.113264.

[135] Y. Erlangga and E. Turkel. 'Iterative schemes for high order compact discretizations to the exterior Helmholtz equation'. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 46.3 (2012), pp. 647–660. DOI: 10.1051/m2an/2011063.

[136] R. Nabben and C. Vuik. 'A Comparison of Deflation and the Balancing Preconditioner'. In: *SIAM Journal on Scientific Computing* 27.5 (2006), pp. 1742–1759. DOI: 10.1137/040608246.

[137] J. Chen, V. Dwarka and C. Vuik. 'Matrix-Free Parallel Scalable Multilevel Deflation Preconditioning for Heterogeneous Time-Harmonic Wave Problems'. In: *Journal of Scientific Computing* 102.2 (2025), p. 47. DOI: 10.1007/s10915-024-02786-w.

[138] H. C. Elman, O. G. Ernst and D. P. O'Leary. 'A Multigrid Method Enhanced by Krylov Subspace Iteration for Discrete Helmholtz Equations'. In: *SIAM Journal on Scientific Computing* 23.4 (2001), pp. 1291–1315. DOI: 10.1137/S1064827501357190.

[139] S. Kim and S. Kim. 'Multigrid simulation for high-frequency solutions of the Helmholtz problem in heterogeneous media'. In: *SIAM Journal on Scientific Computing* 24.2 (2002), pp. 684–701. DOI: 10.1137/S1064827501385426.

[140] P. Lu and X. Xu. 'A robust multilevel preconditioner based on a domain decomposition method for the Helmholtz equation'. In: *Journal of Scientific Computing* 81 (2019), pp. 291–311. DOI: 10.1007/s10915-019-01015-z.

[141] D. Drzisga, U. Rüde and B. Wohlmuth. ‘Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids’. In: *SIAM Journal on Scientific Computing* 42.6 (2020), B1429–B1461. DOI: 10.1137/19M1267891.

[142] D. Drzisga, T. Köppl and B. Wohlmuth. ‘A Semi Matrix-Free Twogrid Preconditioner for the Helmholtz Equation with Near Optimal Shifts’. In: *Journal of Scientific Computing* 95.3 (2023), p. 82. DOI: 10.1007/s10915-023-02195-5.

[143] D. (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

A
2D Re-Discretization Schemes for Coarse Levels

The 2D stencils of the Laplace and wavenumber operators for interior points on the fourth-, fifth- and sixth-level coarse grid introduced in Section 5.1.1 are as follows:

$$A_{8h} = \frac{1}{4096}\cdot\frac{1}{1024}\cdot\frac{1}{1024}\cdot\frac{1}{h^2}\cdot$$

$$\begin{bmatrix} -10395 & -887166 & -7871637 & -15491748 & -7871637 & -887166 & -10395 \\ -887166 & -39105612 & -215169378 & -348459432 & -215169378 & -39105612 & -887166 \\ -7871637 & -215169378 & -265120059 & 413761124 & -265120059 & -215169378 & -7871637 \\ -15491748 & -348459432 & 413761124 & 2809129936 & 413761124 & -348459432 & -15491748 \\ -7871637 & -215169378 & -265120059 & 413761124 & -265120059 & -215169378 & -7871637 \\ -887166 & -39105612 & -215169378 & -348459432 & -215169378 & -39105612 & -887166 \\ -10395 & -887166 & -7871637 & -15491748 & -7871637 & -887166 & -10395 \end{bmatrix},$$

$$K_{8h} = \frac{1}{4096}\cdot\frac{1}{4096}\cdot\frac{1}{1024}\cdot k^2\cdot$$

$$\begin{bmatrix} 27225 & 3939210 & 40768695 & 83544780 & 40768695 & 3939210 & 27225 \\ 3939210 & 569967876 & 5898859542 & 12088170168 & 5898859542 & 569967876 & 3939210 \\ 40768695 & 5898859542 & 61050008889 & 125106029556 & 61050008889 & 5898859542 & 40768695 \\ 83544780 & 12088170168 & 125106029556 & 256372094224 & 125106029556 & 12088170168 & 83544780 \\ 40768695 & 5898859542 & 61050008889 & 125106029556 & 61050008889 & 5898859542 & 40768695 \\ 3939210 & 569967876 & 5898859542 & 12088170168 & 5898859542 & 569967876 & 3939210 \\ 27225 & 3939210 & 40768695 & 83544780 & 40768695 & 3939210 & 27225 \end{bmatrix},$$

$$A_{16h} = \frac{1}{4096}\cdot\frac{1}{1024}\cdot\frac{1}{1024}\cdot\frac{1}{1024}\cdot\frac{1}{h^2}\cdot$$

$$\begin{bmatrix} -13491387 & -1011388446 & -8590720245 & -16705596516 & -8590720245 & -1011388446 & -13491387 \\ -1011388446 & -41427399756 & -220811304386 & -353095695272 & -220811304386 & -41427399756 & -1011388446 \\ -8590720245 & -220811304386 & -262703195227 & 427978620452 & -262703195227 & -220811304386 & -8590720245 \\ -16705596516 & -353095695272 & 427978620452 & 2827174335440 & 427978620452 & -353095695272 & -16705596516 \\ -8590720245 & -220811304386 & -262703195227 & 427978620452 & -262703195227 & -220811304386 & -8590720245 \\ -1011388446 & -41427399756 & -220811304386 & -353095695272 & -220811304386 & -41427399756 & -1011388446 \\ -13491387 & -1011388446 & -8590720245 & -16705596516 & -8590720245 & -1011388446 & -13491387 \end{bmatrix},$$

$$K_{16h} = \frac{1}{4096} \cdot \frac{1}{4096} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot k^2 \cdot$$

$$\begin{bmatrix} 158684409 & 19907719338 & 199630340247 & 405976871820 & 199630340247 & 19907719338 & 158684409 \\ 19907719338 & 2497518765316 & 25044582577654 & 50931743533240 & 25044582577654 & 2497518765316 & 19907719338 \\ 199630340247 & 25044582577654 & 251141703197401 & 510732601675060 & 251141703197401 & 25044582577654 & 199630340247 \\ 405976871820 & 50931743533240 & 510732601675060 & 1038647851363600 & 510732601675060 & 50931743533240 & 405976871820 \\ 199630340247 & 25044582577654 & 251141703197401 & 510732601675060 & 251141703197401 & 25044582577654 & 199630340247 \\ 19907719338 & 2497518765316 & 25044582577654 & 50931743533240 & 25044582577654 & 2497518765316 & 19907719338 \\ 158684409 & 19907719338 & 199630340247 & 405976871820 & 199630340247 & 19907719338 & 158684409 \end{bmatrix},$$

$$A_{32h} = \frac{1}{4096} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot \frac{1}{h^2} \cdot$$

$$\begin{bmatrix} -14618265915 & -1063059274398 & -8934400311925 & -17322732417892 & -8934400311925 & -1063059274398 & -14618265915 \\ -1063059274398 & -42774103061580 & -226217899925314 & -360608941958056 & -226217899925314 & -42774103061580 & -1063059274398 \\ -8934400311925 & -226217899925314 & -266795319274715 & 439293870677284 & -266795319274715 & -226217899925314 & -8934400311925 \\ -17322732417892 & -360608941958056 & 439293870677284 & 2882610253296592 & 439293870677284 & -360608941958056 & -17322732417892 \\ -8934400311925 & -226217899925314 & -266795319274715 & 439293870677284 & -266795319274715 & -226217899925314 & -8934400311925 \\ -1063059274398 & -42774103061580 & -226217899925314 & -360608941958056 & -226217899925314 & -42774103061580 & -1063059274398 \\ -14618265915 & -1063059274398 & -8934400311925 & -17322732417892 & -8934400311925 & -1063059274398 & -14618265915 \end{bmatrix},$$

$$K_{32h} = \frac{1}{4096} \cdot \frac{1}{4096} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot \frac{1}{1024} \cdot k^2 \cdot$$

$$\begin{bmatrix} 706549519225 & 85722084590890 & 852977249303575 & 1731387418334860 & 852977249303575 & 85722084590890 & 706549519225 \\ 85722084590890 & 10400227566028036 & 103487421517331808 & 210060490730926272 & 103487421517331824 & 10400227566028036 & 85722084590890 \\ 852977249303575 & 103487421517331840 & 1029751161146567936 & 2090206046973178368 & 1029751161146568192 & 103487421517331792 & 852977249303575 \\ 1731387418334860 & 210060490730926208 & 2090206046973178624 & 4242735025361522688 & 2090206046973178624 & 210060490730926208 & 1731387418334860 \\ 852977249303575 & 103487421517331840 & 1029751161146568064 & 2090206046973178624 & 1029751161146567936 & 103487421517331792 & 852977249303575 \\ 85722084590890 & 10400227566028036 & 103487421517331808 & 210060490730926208 & 103487421517331808 & 10400227566028036 & 85722084590890 \\ 706549519225 & 85722084590890 & 852977249303575 & 1731387418334860 & 852977249303575 & 85722084590890 & 706549519225 \end{bmatrix}.$$

B

3D Re-discretization Scheme for Coarse Levels

As introduced in Section 6.2.1, we present the computational stencils corresponding to the discretized Laplace operators $\Delta_{2^{l-1}h}$ and associated wavenumber terms $\mathcal{K}(k^2_{i,j,k})_{2^{l-1}h}$ for interior nodes within the corresponding coarse-grid (l-th) level in 3D scenarios.

Suppose the presented stencils are $7\times7\times7$ arrays centered around the grid point $(i,j,k) = (0,0,0)$. Let $a_{i,j,k}$ denote the stencil coefficient at position (i,j,k) relative to the central point. As the stencil exhibits radial symmetry, we can define

$$a_{i,j,k} = f(r), \quad r = \sqrt{i^2 + j^2 + k^2}, \quad i,j,k = -3,-2,\cdots,3.$$

where $f(r)$ is a function mapping the distance r to the corresponding coefficient value. The distances r will take on discrete values consistent with the stencil grid spacing. Table B.1 and B.2 present the coefficients corresponding to each discrete value of r.

Table B.1: Computational stencils for the discretized Laplace operators $\Delta_{2^{l-1}h}$

r^2	Δ_{2h} $(h^{-2})^\star$	Δ_{4h} (h^{-2})	Δ_{8h} (h^{-2})	Δ_{16h} (h^{-2})	Δ_{32h} (h^{-2})
0	1.5701293945E+0	2.0810596962E+0	3.7899080641E+0	7.4089777374E+0	1.4734333085E+1
1	2.6916503906E-1	5.0871183397E-1	9.8862184825E-1	1.9621172110E+0	3.9166032206E+0
2	-3.5888671875E-2	1.6178222490E-2	6.2374914024E-2	1.3819040572E-1	2.8291157951E-1
3	-7.1777343750E-2	-9.8258635233E-2	-1.7454480404E-1	-3.3852933590E-1	-6.7184014760E-1
4	-9.7198486328E-2	-1.3917780761E-1	-2.5384774652E-1	-4.9578511425E-1	-9.8565558173E-1
5	-4.4006347656E-2	-8.4633644903E-2	-1.6446169783E-1	-3.2622637693E-1	-6.5107007270E-1
6	-1.9653320313E-2	-4.8904286290E-2	-1.0006117458E-1	-2.0095027797E-1	-4.0227853400E-1
8	-3.0975341797E-3	-1.4371655416E-2	-3.2364034736E-2	-6.6438746697E-2	-1.3371917032E-1
9^*	-ğ	-5.5457470007E-3	-1.3522009341E-2	-2.8220807272E-2	-5.7025303896E-2
9	-1.3122558594E-3	-7.5410612626E-3	-1.7706937597E-2	-3.6712135717E-2	-7.4070560816E-2
10	-	-2.6786775561E-3	-6.8790650430E-3	-1.4534058718E-2	-2.9457773736E-2
11	-	-1.2932208192E-3	-3.4937743183E-3	-7.4699173828E-3	-1.5184488379E-2
12	-6.8664550781E-5	-9.2338607647E-4	-2.4876297845E-3	-5.3237002139E-3	-1.0824901025E-2
13	-	-2.4811155163E-4	-7.7852953697E-4	-1.7196473811E-3	-3.5232869013E-3
14	-	-1.1955149239E-4	-3.9313800738E-4	-8.7782070955E-4	-1.8032962903E-3
17	-	-1.0964577086E-5	-4.3354500566E-5	-1.0079189293E-4	-2.0909404149E-4
18	-	-1.4024553820E-6	-9.2153822173E-6	-2.3193307852E-5	-4.8995829836E-5
19	-	-6.6982465796E-7	-4.5883901079E-6	-1.1661643338E-5	-2.4694663026E-5
22	-	-5.9197191149E-8	-4.8044680234E-7	-1.2685731350E-6	-2.7114431323E-6
27	-	-2.6193447411E-10	-4.5701442630E-9	-1.3819593422E-8	-3.0492200157E-8

* All Δ values are scaled by h^{-2}.
* Coefficients specifically for axis-aligned positions $(\pm3,0,0),(0,\pm3,0),(0,0,\pm3)$.
ğ Not applicable for the $5\times5\times5$-stencil A_{2h}.

Table B.2: Computational stencils for the wavenumber terms $\mathcal{K}(k^2_{i,j,k})_{2^{l-1}h}$

r^2	$\mathcal{K}(k^2_{i,j,k})_{2h}$ *	$\mathcal{K}(k^2_{i,j,k})_{4h}$	$\mathcal{K}(k^2_{i,j,k})_{8h}$	$\mathcal{K}(k^2_{i,j,k})_{16h}$	$\mathcal{K}(k^2_{i,j,k})_{32h}$
0	1.3084411621E+0	7.7105929767E+0	5.7646951751E+1	4.5365194446E+2	3.6144249807E+3
1	5.2337646484E-1	3.6428073344E+0	2.8130913668E+1	2.2307352540E+2	1.7806657512E+3
2	2.0935058593E-1	1.7210148837E+0	1.3727496073E+1	1.0969157818E+2	8.7725448291E+2
3	8.3740234375E-2	8.1307957250E-1	6.6988278685E+0	5.3938459540E+1	4.3218410151E+2
4	1.8692016601E-2	3.0682290392E-1	2.7181045758E+0	2.2245542085E+1	1.7895246359E+2
5	7.4768066406E-3	1.4495600119E-1	1.3263973695E+0	1.0938763864E+1	8.8161885974E+1
6	2.9907226562E-3	6.8483291222E-2	6.4726353708E-1	5.3789003851E+0	4.3433423505E+1
8	2.6702880859E-4	1.2209215899E-2	1.2816102605E-1	1.0908454128E+0	8.8600494952E+0
9^*	-ğ	9.5286616124E-4	1.8785593324E-2	1.7731919155E-1	1.4749848620E+0
9	1.0681152343E-4	5.7681453763E-3	6.2540804848E-2	5.3639962283E-1	4.3649506558E+0
10	-	4.5017391676E-4	9.1671092391E-3	8.7192874764E-2	7.2665916194E-1
11	-	2.1268102864E-4	4.4734222844E-3	4.2875209068E-2	3.5799251315E-1
12	3.8146972656E-6	4.8583385068E-4	6.0429053190E-3	5.3491333688E-2	4.3866664633E-1
13	-	3.7916819565E-5	8.8575727984E-4	8.6951275889E-3	7.3027432091E-2
14	-	1.7913494957E-5	4.3223727904E-4	4.2756408050E-3	3.5977354051E-2
17	-	1.5088007785E-6	4.1764236309E-5	4.2637936213E-4	3.6156342854E-3
18	-	1.1775409802E-7	6.1217203306E-6	6.9308852471E-5	6.0191603226E-4
19	-	5.5631971918E-8	2.9873146956E-6	3.4081128166E-5	2.9653714476E-4
22	-	4.6857167035E-9	2.8864450829E-7	3.3986694277E-6	2.9801242915E-5
27	-	1.4551915228E-11	1.9949042417E-9	2.7090790279E-8	2.4563161237E-7

* All $\mathcal{K}(k^2_{i,j,k})$ values are scaled by the corresponding $k^2_{i,j,k}$ on the current grid.
* Coefficients specifically for axis-aligned positions $(\pm3,0,0),(0,\pm3,0),(0,0,\pm3)$.
ğ Not applicable for the $5\times5\times5$-stencil $\mathcal{K}(k^2_{i,j,k})_{2h}$.

ACKNOWLEDGEMENTS

First and foremost, I extend my deepest gratitude to my promotor, **Prof. Vuik**, whose unwavering support made this doctoral journey possible. Without his invaluable assistance, I would never have successfully relocated to TU Delft during the Corona time to begin my PhD studies. Throughout these four years, his guidance has been nothing short of exceptional – his advice consistently precise, his insights profoundly enlightening, and his expertise truly inspiring. He exemplifies the very essence of academic excellence and educational leadership, serving as a lifelong role model whose influence will forever shape my career pursuits.

I am equally indebted to my promotor, **Prof. van Gijzen**. His scholarly rigor and intellectual depth have been instrumental to my development. I really appreciate his generous support during times of need.

My heartfelt appreciation extends to my co-promotor, **Dr. Dwarka**, who has been an extraordinary mentor throughout this journey. From serving as my senior colleague and daily supervisor to becoming my co-promotor, her exceptional expertise has been beyond question. This thesis builds upon her outstanding foundational work, and I feel privileged to have been her first doctoral student under her guidance. I cherish the hope that we may one day fulfill our shared vision of watching an Ajax match together.

I wish to express my sincere gratitude to the members of my doctoral defense committee: **Prof. Bisseling**, **Prof. Mulder**, **Prof. Nabben**, **Prof. Schuttelaars**, and **Prof. Heemink**. Their dedication in reviewing this thesis and providing invaluable feedback has significantly enhanced the quality of this work. Special thanks are due to Prof. Heemink, who also served as a committee member during my GO/NOGO meetings.

I remain deeply grateful to my master's supervisors who provided the foundation for all that followed. **Dr. Yu** and **Prof. Ouyang** served as my academic mentors during those formative years, introducing me to the rigorous world of research. I am profoundly thankful for their cultivation of my academic potential and for believing in my capabilities when I was still finding my research voice.

This journey has been enriched immeasurably by the friendship and collegiality of many remarkable individuals. **Alice** deserves special recognition for her selfless assistance and wise counsel, both professionally and personally. I am grateful to **Roel** and **Mo** for welcoming me into the Krylov Tigers football team – the memories of our games at Delft X Sports Centre will hold a special place in my heart. **Jonas** provided invaluable expertise in high-performance computing. The conversations with **Alexander** have always been enlightening. I sincerely hope for future opportunities to collaborate. Sharing an office with **Xiujie**, **Elisa**, **Merel**, **Hugo**, **Buu-Van**, **Corne**, and **Kevin** has been an absolute privilege, and the pleasant memories of daily interactions and academic exchanges with **Chen Kewang**, **Li Jun**, and **Ji Ye** have added warmth to my doctoral experience.

My involvement with the Amsterdam Torch Football Team has been an integral part of these precious four years. I am particularly grateful to our captain **Mao** for his trust and the multifaceted support he has provided throughout my life here.

至亲之恩，山高海深。谨以最诚挚的谢意献予我的家人：父母一生劬劳，以三十年如一日的坚忍托举我一路前行，他们质朴的勤勉是我人生最深的启蒙。寸草之心难报春晖，游子远行，愧怍殊深。惟愿流光缓步，许我以反哺之时。外祖父期许如炬，长姐一家周末围炉的暖意，皆成异乡岁月里的甘霖。王叔王姨的关切，亦如春风化雨，润物无声。

Finally, my deepest appreciation goes to my beloved partner, **Aru**. You were among the driving forces that inspired me to embark on this journey. Together, we have weathered countless storms and celebrated numerous joys over these four years – experiencing moments of happiness and sorrow, triumph and challenge. As we look toward the future, regardless of what mountains we may face or what horizons await us, my greatest wish is to continue walking this path of life hand in hand with you.

This thesis stands not merely as a culmination of academic endeavor, but as a testament to the collective support, wisdom, and love of all those mentioned above. To each of you, I offer my profound and enduring gratitude.

Curriculum Vitæ

Jinqiang Chen

01-12-1994	Born in Maoming, Guangdong, China.

Education

2013–2017	Bachelor in Energy and Power Engineering Shanghai Jiao Tong University (SJTU), China	
	Thesis:	Flexible Performance Measuring Method of Fans in Pasture
	Supervisor:	Prof. Tong Wang
2017–2020	Master in Power Engineering and Engineering Thermophysics Shanghai Jiao Tong University (SJTU), China	
	Thesis:	Direct Numerical Simulation of Trailing Edge Noise Based on High-Fidelity Algorithm
	Supervisors:	Prof. Hua Ouyang, Dr. Peixiang Yu
2021-present	PhD. in Applied Mathematics Delft University of Technology (TU Delft), the Netherlands	
	Thesis:	High-Performance Iterative Methods for the Helmholtz Equation
	Promotors:	Prof.dr.ir. C. Vuik, Prof.dr.ir. M.B. van Gijzen
	Co-promotor:	Dr.ir.mr. V.N.S.R. Dwarka

Work Experience

2025–present	Postdoc researcher in Section of Applied Geophysics and Petrophysics Delft University of Technology (TU Delft), the Netherlands

Scientific Contributions

Journal Papers

4. **J. Chen**, V. Dwarka and C. Vuik. *A robust parallel multilevel deflation solver for large-scale 3D frequency-domain wave simulation: A tool for full-waveform inversion*, submitted to GEOPHYSICS (2025)

3. **J. Chen**, V. Dwarka and C. Vuik. *Matrix-Free Parallel Scalable Multilevel deflation Preconditioning for Heterogeneous Time-Harmonic Wave Problems*, Journal of Scientific Computing, **102** (2025), 47. DOI: 10.1007/s10915-024-02786-w

2. **J. Chen**, V. Dwarka and C. Vuik. *A Matrix-free parallel two-level deflation preconditioner for two-dimensional heterogeneous Helmholtz problems*, Journal of Computational Physics, **514** (2024), 113264. DOI: 10.1016/j.jcp.2024.113264

1. **J. Chen**, V. Dwarka and C. Vuik. *A matrix-free parallel solution method for the three-dimensional heterogeneous Helmholtz equation*, Electronic Transactions on Numerical Analysis, **59** (2023), 270-294. DOI: 10.1553/etna_vol59s270

Peer-reviewed Proceedings

1. **J. Chen**, V. Dwarka and C. Vuik. *Matrix-Free Parallel Preconditioned Iterative Solvers for the 2D Helmholtz Equation Discretized with Finite Differences*, In: Scientific Computing in Electrical Engineering, 2024, 61-68. DOI: 10.1007/978-3-031-54517-7_7

Oral Presentations

4. *Matrix-Free Parallel Scalable Multilevel Deflation Preconditioning for the Helmholtz Equation*. International Conference On Preconditioning Techniques For Scientific and Industrial Applications (Precond24). Atlanta, USA. June 2024.

3. *Matrix-free parallel scalable multilevel deflation for time-harmonic wave problems*. The European Conference on Numerical Mathematics and Advanced Applications (ENUMATH 2023). Lisbon, Portugal. September, 2023.

2. *A Matrix-Free Parallel Two-Level Deflation Preconditioner for 2D Helmholtz Problems*. SIAM Conference on Computational Science and Engineering (CSE 2023). Amsterdam, the Netherlands. February, 2023.

1. *Towards matrix-free parallelization of the scalable deflation method for the 3D heterogeneous high-frequency Helmholtz equation* (Online). Numerical Methods for Large Scale Problems (NMLSP 2022). Belgrade, Czech. June, 2022.

POSTER PRESENTATIONS

3. *Scalable Solvers for Time-harmonic Wave Problems.* 48th Woudschoten Conference. Zeist, the Netherlands. September, 2024.

2. *A matrix-free parallel solution method for 2D Helmholtz equation.* 14th International Conference on Scientific Computing in Electrical Engineering (SCEE 2022). Amsterdam, the Netherlands. July, 2022.

1. *Matrix-free parallel solution methods for Helmholtz equations.* 45th Woudschoten Conference. Zeist, the Netherlands. October, 2021.