The Dissertation Committee for Jacob Badger
certifies that this is the approved version of the following dissertation:

Scalable DPG Multigrid Solver with Applications in High-Frequency Wave Propagation

Committee:

Leszek Demkowicz, Supervisor

Björn Engquist

Omar Ghattas

George Biros

Jay Gopalakrishnan

Jacob Grosek

Socratis Petrides

Scalable DPG Multigrid Solver with Applications in High-Frequency Wave Propagation

by

Jacob Badger

DISSERTATION

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin
May 2024

Dedicated to my wonderful wife Jaclyn

Scalable DPG Multigrid Solver with Applications in High-Frequency Wave Propagation

by

Jacob Badger, Ph.D.
The University of Texas at Austin, 2024
Supervisor: Leszek Demkowicz

Abstract

Wave propagation is fundamental to applications including natural resource exploration, nuclear fusion research, and military defense, among others. However, developing accurate and efficient numerical algorithms for solving time-harmonic wave propagation problems is notoriously difficult. One difficulty is that classical discretization techniques (e.g., Galerkin finite elements, finite difference, etc.) yield indefinite discrete systems that preclude the use of many scalable solution algorithms. Significant progress has been made to develop specialized preconditioners for high-frequency wave propagation problems but robust and scalable solvers for general problems, including non-homogenous media and complex geometries, remain elusive. An alternative approach is to use minimum residual discretization methods—that yield Hermitian positive-definite discrete systems—and may be amenable to more standard preconditioners. Indeed, popularization of the first-order system least-squares methodology (FOSLS) was driven by the applicability of geometric and algebraic multigrid to otherwise indefinite problems. However, for wave propagation problems, FOSLS is known to be highly dissipative and is thus less competitive in the high-frequency regime.

The discontinuous Petrov–Galerkin (DPG) method of Demkowicz and Gopalakrishnan [28] is a minimum residual finite element method with several additional attractive properties: mesh-independent stability, a built-in error indicator, and applicability to a number of variational formulations. In the context of high-frequency wave propagation, the ultraweak DPG formulation has been observed to produce pollution error roughly commensurate to Galerkin discretizations. DPG discretizations may thus deliver accuracy typical of classical discretization techniques, but result in Hermitian positive-definite discrete systems that are often more amenable to preconditioning. A multigrid preconditioner for DPG systems, developed in the dissertation work of S. Petrides [100, 102], was shown to scale efficiently in a shared-memory implementation.

The primary objective of this dissertation is development of an efficient, distributed implementation of the DPG multigrid solver (DPG-MG). The distributed DPG-MG solver developed in this work will be demonstrated to be massively scalable, enabling solution of three-dimensional problems with $\mathcal{O}(10^{12})$ degrees of freedom on up to 460 000 CPU cores, an

unprecedented scale for high-frequency wave propagation. The scalability of the DPG-MG solver will be further combined with *hp*-adaptivity to enable efficient solution of challenging real-world high-frequency wave propagation problems including optical fiber modeling, simulation of RF heating in tokamak devices, and seismic simulation. These applications include complex three-dimensional geometries, heterogeneous and anisotropic media, and localized features; demonstrating the robustness and versatility of the solver and tools developed in this dissertation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Wave[1] simulation is important in applications including seismic and medical imaging, optical fiber design, and radio-frequency (RF) heating in tokamak devices for nuclear fusion research. For example, in optical fiber design, wave simulation can be used to predict the performance of specialized optical fiber amplifier designs, significantly reducing the time and capital required to develop improved high-power fiber lasers. In imaging applications, repeated forward wave simulation can be used in full waveform inversion (FWI) algorithms to create higher resolution images. However, efficient simulation of wave propagation problems remains a challenge in scientific computing, often necessitating use of less expensive and less accurate models that can limit the ability of simulations to predict real-world behavior, hindering their utility. Improving the efficiency of accurate wave simulation algorithms can thus enhance the predictive capabilities of simulations, enabling improved decision making in both scientific and commercial applications.

Wave simulation can be performed in the time- or frequency-domain. Time-domain approaches start from a known initial state and sequentially update the wavefield at discrete points in time. Time-domain simulation is popular in seismic imaging applications for its extreme scalability, but often require small time steps in high-contrast media. Frequency-domain approaches simulate discrete frequencies independently, and are thus useful for simulation of frequency-dependent phenomena including attenuation. Frequency-domain simulation can also be used to efficiently simulate long time windows and time-harmonic or steady state systems, and can significantly reduce memory and storage requirements. For example, the recent dissertations of Sriram Nagaraj [93] and Stefan Henneking [58], in collaboration with Jacob Grosek and the Air Force Research Laboratory [61, 62], developed a high-fidelity optical fiber amplifier model coupling frequency-domain optical fields to a time-domain thermal field to simulate transverse modal instability in high-power fiber lasers. In the optical fiber model, the significant timescale separation between optical and thermal fields would have required a prohibitive number of optical time steps to observe relevant perturbations in the thermal field; instead, the optical field was assumed to reach steady state between each thermal time step, effectively removing the optical timescale from the problem.

[1]This chapter contains passages adapted from "Scalable DPG multigrid solver for Helmholtz problems: A study on convergence," by J. Badger, S. Henneking, S. Petrides, and L.Demkowicz, *Comput. Math. Appl.*, 148:81–92, 2023. The author of this dissertation contributed to theory, software development, numerical simulations, and writing of that work.

The primary limitation of frequency-domain methods has been scale: frequency-domain simulation requires solving large linear systems of equations, and developing scalable and efficient solution algorithms for these systems is a notoriously challenging problem in mathematics and scientific computing [39]. The lack of scalable solvers has so-far limited frequency-domain simulation to problems with $\mathcal{O}(10^9)$ unknowns, a relatively small scale when compared to time-domain methods which have been demonstrated to solve problems with $\mathcal{O}(10^{12})$ spatial unknowns [41, 23]. A multigrid solver based on the discontinuous Petrov–Galerkin (DPG) finite element methodology was proposed in the dissertation work of Socratis Petrides [100]. A shared-memory implementation of the DPG multigrid solver (DPG-MG), when coupled with *hp*-adaptive mesh refinement, was shown to enable solution of high-frequency wave propagation problems with singular and localized features but the shared-memory implementation limited the solver to problems with $\mathcal{O}(10^7)$ unknowns.

1.2 Objective

This dissertation details development of a scalable and performant implementation of the DPG-MG multigrid solver. The scalable DPG-MG solver will be demonstrated to enable solution of high-frequency wave propagation problems with $\mathcal{O}(10^{12})$ unknowns—an unprecedented scale for high-frequency wave propagation. An alternative construction of coarse-grid operators, along with various algorithmic and implementational optimizations, improve the efficiency of the scalable solver and make the DPG-MG solver competitive for general wave propagation problems. Extension of the DPG-MG solver to support general unstructured meshes of all element types, as well as anisotropic *h*- and *p*- adaptivity, enable increased flexibility and further improve competitiveness for problems with singular and localized features. The distributed DPG-MG solver developed here is notable for its versatility, robustness, and extreme scalability; these features will be demonstrated via a number of challenging problems in acoustics, electromagnetics, and elasticity.

1.3 Background

1.3.1 Discretization methods for wave propagation

Developing accurate and efficient algorithms to simulate time-harmonic wave propagation is a perennial challenge in mathematics and scientific computing. The first challenge in solving high-frequency wave problems is defining an appropriate discretization. It is well known

that discretizations must satisfy the Nyquist criterion, establishing a minimum number of points per wavelength needed to capture waves. However, as the frequency grows, all known (volumetric) discretizations for three-dimensional wave problems suffer from so-called *pollution* error, arising either as phase (dispersive) error or as amplitude (dissipative) error. In the context of the h finite element method for the one-dimensional Helmholtz problem with wavenumber k, Ihlenburg and Babuška [70] proved the wavenumber explicit error bound:

$$|u - u_h|_{H^1} \leq (1 + Ck^2h) \inf_{w \in U^h} |u - w|_{H^1}, \tag{1.1}$$

indicating that to maintain a particular accuracy, k^2h should be held constant. The required number of elements per wavelength thus increases *linearly* with frequency.

Melenk and Sauter later demonstrated that pollution could be countered, for a fixed number of elements per wavelength, by increasing the polynomial order p logarithmically with the wavenumber [89]. Use of high-order methods is often the most efficient strategy to counter pollution (subject to regularity) and has motivated the development of numerous high-order finite difference, spectral element, and discontinuous Galerkin (DG) methods, among others, for both time-domain and frequency-domain wave propagation. Recently, finite difference operators with stencils optimized for pollution error have been proposed [1]; the so-called λ-adaptive schemes have been demonstrated to simulate high-frequency problems in heterogeneous media with relatively low-order discretizations and a surprisingly small number of points per wavelength.

Other approaches to mitigate pollution error largely fall into two categories: those that seek to improve *approximability* of a discretization (targeting the best approximation term $\inf_{w \in U^h} |u-w|_{H^1}$ in (1.1)), and those that seek to improve *stability* (targeting the wave number explicit factor k^2h in (1.1)). Methods in the former category attempt to enrich the approximation space, often supplementing piecewise polynomial bases with analytic wave solutions, plane waves, or Bessel functions. Methods with enriched trial spaces include plane wave DG methods [45, 66] and partition of unity method [87, 88], among others. Methods targeting stability include the Galerkin least-squares method of Harari and Hughes[56], and a variation by Monk and Wang [90] further enriching the approximation space. The discontinuous Petrov–Galerkin (DPG) method employed in this work can be seen as a method of optimal test functions. In the case of DPG, use of discontinuous ("broken") test spaces[2] enables effi-

[2]The 'D' in the DPG name

cient, on the fly computation of optimal test functions. A closely related method, leveraging continuous test spaces, was recently proven by Monsuur and Stevenson [91] to counter pollution through a modest increase in the order of the *test* space; however, test functions enter the system as additional unknowns in this discretization and cannot be efficiently removed, this approach thus similarly mitigates pollution at the expense of increasing the size of the global system. We note that discretizations based on integral equations [75] are practically pollution-free but assume piece-wise constant media; we thus limit the current discussion to volumetric discretizations.

There is (so far) no free lunch. Effectively mitigating pollution produces larger and, in the case of high-order methods, more densely coupled systems. Employing high-order methods with a fixed number of elements per wavelength (i.e. $h^{-1} \propto k$ and $p \propto \log k$) implies $\mathcal{O}(k^3 \log^3 k)$ spatial unknowns for three-dimensional wave propagation problems. The resulting systems quickly become unfeasible for existing solvers, necessitating development of increasingly fast, scalable, and efficient linear solvers for wave propagation problems.

1.3.2 Sparse Linear Solvers

Direct sparse solvers [4, 105] are among the most widely employed solvers for time-harmonic wave propagation problems. These solvers employ nested-dissection and other factorization-based techniques designed to minimize fill-in to preserve sparsity of the factored system. Direct sparse solvers are robust and applicable to a wide range of problems, but the computational and memory cost of factorization can be relatively expensive. When applied to systems arising from discretization of three-dimensional PDEs, leading direct sparse solvers have an $\mathcal{O}(N^2)$ computational complexity and $\mathcal{O}(N^{4/3})$ memory complexity. However, once a factorization is computed it can be applied in $\mathcal{O}(N^{4/3})$ complexity. Direct solvers can thus be competitive for problems with a large number of right-hand-sides (e.g. imaging applications [98]), in which case the factorization cost can be amortized over loads. Recently, techniques including hierarchically semi-separable structure (HSS), rank-revealing factorizations, and hierarchical compression [118, 76, 84] have been employed to reduce the memory and computational complexity of direct solvers, at the expense of incurring some algebraic error. Still, three-dimensional time-harmonic wave propagation represents a challenge for rank-exploiting approaches, so-far limiting the scale of direct solvers to $\mathcal{O}(10^8)$ (approaching $\mathcal{O}(10^9)$).

For elliptic problems, preconditioned iterative methods are well-established and highly effective, and can be used to solve problems with $\mathcal{O}(10^{13})$ unknowns [46]. However, developing scalable and effective preconditioners for high-frequency wave propagation problems is notoriously difficult. The primary challenge is that high-frequency wave propagation operators—and often the discrete systems they lead to—are highly indefinite, causing standard iterative solution schemes to break down [39]. Current leading-edge preconditioning techniques for wave operators such as multigrid methods [65, 113, 52], domain decomposition methods with special transmission conditions [16, 49, 112, 74], stabilized methods based on artificial absorption [54, 12], shifted Laplacian [110] and sweeping preconditioners [36, 37, 21, 81, 114] are very promising but they lose their efficiency in heterogeneous media and can be difficult to apply on complex geometries [43, 38].

An alternative approach instead employs *minimum-residual* discretization methodologies which, by construction, produce positive-definite discrete systems and may therefore be amenable to more standard preconditioning techniques [51, 101, 102, 100]. Indeed, popularization of the first-order system least-squares methodology (FOSLS) [17, 78], and other least-squares methodologies [24], was driven by the applicability of geometric and algebraic multigrid methods to otherwise indefinite problems. However, for wave propagation problems, FOSLS is known to be highly dissipative [47] and is thus not competitive in the high-frequency regime.

1.3.3 DPG methodology

The DPG finite element methodology of Demkowicz and Gopalakrishnan [28, 29, 31] is a minimum residual method with several attractive properties: mesh-independent stability, a built-in error indicator, and applicability to a number of variational formulations with different functional settings. A special case of the DPG method is the well-established FOSLS method in which the residual is minimized in the L^2 test norm. As mentioned previously however, other formulations are preferable in the context of wave propagation. Among the various DPG formulations, the so-called *ultraweak* variational formulation has proved to be superior: it is less dissipative than other DPG formulations [100], with pollution error roughly commensurate to Galerkin discretizations [47]. These properties were leveraged by Petrides and Demkowicz in [102] to define an *hp*-adaptive multilevel preconditioner for DPG systems discretized with conforming elements of the exact-sequence energy spaces [33].

Similar to hybridizable methods, the DPG methodology introduces additional trace unknowns on the mesh skeleton resulting from testing with larger discontinuous test spaces [19]. In the case of high-order discretizations, statically condensing all interior unknowns onto the mesh skeleton results in a smaller global system. The DPG multigrid solver (DPG-MG) is defined on this condensed global system of trace degrees of freedom. Constructing suitable prolongation operators for the condensed system is complicated by the fact that fine-grid unknowns resulting from h-refinement have no natural coarse-grid representatives; this is a challenge shared by hybridizable methods [103, 104]. Construction of a stable prolongation operator between such non-nested condensed systems for general DPG problems[3] was one of the contributions in the original DPG-MG work by Petrides and Demkowicz [101, 100] and will be outlined later in Chapter 5.

1.4 Accomplishments of this dissertation

The first accomplishment of this dissertation is development of a performant and scalable multigrid solver for 3D high-frequency wave propagation problems. An alternative construction of coarse-grid operators, along with numerous optimizations, will be shown to massively improve the efficiency of the scalable DPG-MG solver, enabling it to be competitive with leading-edge wave propagation solvers for general wave propagation problems.

The distributed implementation developed here employs hybrid MPI/OpenMP parallelism to enable efficient simulation on modern distributed manycore platforms. The extreme scalability of our implementation is notable, enabling solution of 3D high-frequency wave propagation problems in heterogeneous media with 1 000 wavelengths and over 800 billion degrees-of-freedom (DOFs) on 460 000 CPU cores. To the best of our knowledge, this is orders of magnitude larger than any previous result for high-frequency wave propagation.

The DPG-MG solver was additionally extended to support unstructured hybrid meshes with all element shapes (hexahedra, tetrahedra, prisms, and pyramids), general orientations, and anisotropic h- and p-refinements. These capabilities appear to be unique among geometric multigrid solvers and enable solution of challenging problems in science and engineering.

Finally, the DPG-MG solver is demonstrated on a number of challenging problems with complex geometries, high-contrast heterogeneous media, and localized features. Problems include bend-loss simulation in optical fibers, RF heating in a tokamak device, and acoustic

[3] discretized with exact-sequence energy spaces

and elastic simulation in complex seismic models including the GO_3D_OBS model [53] and the SEAM Arid model [99].

1.5 Outline

The remainder of this dissertation is organized as follows. Chapter 2 derives the acoustic, electromagnetic, and elastic wave operators used throughout this work. Chapter 3 provides an overview of the DPG methodology, starting with an abstract variational formulation and later defining specific formulations for the wave operators considered here. Details on the cost, implementation, and other considerations when computing with the DPG methodology are then outlined in Chapter 4.

Chapter 5 defines the DPG-MG preconditioner and introduces the distributed implementation; various convergence studies are performed to highlight aspects of the construction. Strong and weak scaling studies are provided to demonstrate the scalability of the approach.

Chapter 6 details application of the DPG-MG solver on various challenging problems including optical fiber modeling, seismic modeling, and simulation of RF heating in a tokamak device.

Chapter 2

Wave Operators

This chapter derives the various wave propagation problems considered throughout this work, aiming to establish common notation. Time-domain operators are first derived in Section 2.1 followed by time-harmonic operators in Section 2.2. Perfectly matched layer boundary conditions are derived in Section 2.3.

2.1 Time-Domain Wave Propagation

Let $\Omega \subset \mathbb{R}^3$ denote a bounded Lipschitz domain with boundary $\Gamma = \partial\Omega$ partitioned into three disjoint parts $\{\Gamma_i, i = 1, 2, 3\}$, i.e. Γ_i are (relatively) open in Γ and

$$\Gamma = \bigcup_i \bar{\Gamma}_i, \qquad \text{and} \qquad \Gamma_i \cap \Gamma_j = \emptyset \text{ for } i \neq j.$$

In the following, Γ_1 will correspond to type 1 (Dirichlet) boundary conditions, Γ_2 to type 2 (Neumann) boundary conditions, and Γ_3 to type 3 (Robin or *impedance*) boundary conditions.

2.1.1 Linear Acoustics

Classical linear acoustics can be derived by linearizing the isentropic form of the compressible Euler equation about a hydrostatic equilibrium state, i.e. $\rho = \rho_0$, $\mathbf{u} = 0$. Alternative derivations, for example linearized about a variable density state [6], are useful in geosciences but are not considered in this work. Consider linear perturbations of $\rho, \mathbf{u}$ about the equilibrium state, i.e.

$$\rho = \rho_0 + \delta\rho$$

$$\mathbf{u} = \delta\mathbf{u}.$$

For isentropic flow, the pressure is related to the density via an algebraic relation $p = p(\rho)$; linearization of this relation around the equilibrium state gives:

$$p = \underbrace{p(\rho_0)}_{:=P_0} + \underbrace{\frac{dp}{d\rho}(\rho_0)}_{:=c^2}\,\delta\rho,$$

where the hydrostatic pressure P_0 and wavespeed c have been defined. Perturbations in pressure and density are thus related as $\delta p = c^2\delta\rho$. Substituting these expressions into the isentropic form of the compressible Euler equation and subsequently eliminating $\delta\rho$ yields

the linear acoustic equations:

$$\begin{cases} \dfrac{1}{c^2}\dfrac{\partial(\delta p)}{\partial t} + \rho_0 \operatorname{div}(\delta \mathbf{u}) = 0 \\ \rho_0 \dfrac{\partial(\delta \mathbf{u})}{\partial t} + \nabla(\delta p) = 0. \end{cases}$$

We neglect the δ's from now on and introduce appropriate boundary conditions, these are:

- Prescribed pressure (acoustically soft):

$$p = p_0 \qquad \text{on } \Gamma_1$$

- Prescribed normal velocity (acoustically hard):

$$u_n = u_0 \qquad \text{on } \Gamma_2$$

- Impedance boundary:

$$u_n = dp + u_0, \qquad \text{on } \Gamma_3$$

  with impedance constant $d > 0$

Introducing pressure and velocity loads (f_p and $\mathbf{f_u}$, respectively) yields the final linear acoustic equations in the time domain:

$$\begin{cases} \dfrac{1}{c^2}\dfrac{\partial p}{\partial t} + \rho_0 \operatorname{div}(\mathbf{u}) = \mathrm{f}_p & \text{in } \Omega \\ \rho_0 \dfrac{\partial \mathbf{u}}{\partial t} + \nabla p = \mathbf{f_u}, & \text{in } \Omega \\ p = p_0 & \text{on } \Gamma_1 \\ u_n = u_0 & \text{on } \Gamma_2 \\ u_n - dp = u_0, & \text{on } \Gamma_3 \end{cases} \tag{2.1}$$

2.1.2 Electromagnetics

Electromagnetic wave equations for non-conductive media can be derived from Maxwell's equations:

$$\begin{aligned} \operatorname{div} \mathbf{D} &= \rho^{\mathrm{imp}} \\ \operatorname{div} \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{H} &= \mathbf{J}^{\mathrm{imp}} + \frac{\partial \mathbf{D}}{\partial t} \end{aligned} \tag{2.2}$$

where $\mathbf{D}$ is the electric displacement, $\mathbf{E}$ denotes the electric field, $\mathbf{B}$ is the magnetization, and $\mathbf{H}$ denotes the magnetic field. The impressed charge ρ^{imp} and impressed current $\mathbf{J}^{\text{imp}}$ are assumed to satisfy the compatibility condition $\text{div}(\mathbf{J}^{\text{imp}}) = \dfrac{\partial \rho^{\text{imp}}}{\partial t}$ and we have implicitly assumed the free charge and current density (typically ρ and $\mathbf{J}$) are negligible. The system is then closed with constitutive laws, relating electric and magnetic displacements to strength of the magnetic field:

$$\begin{aligned} \mathbf{B} &= \mu_0 \mathbf{H} + \mathbf{I}, \\ \mathbf{D} &= \epsilon_0 \mathbf{E} + \mathbf{P}, \end{aligned}$$

where μ_0 and ϵ_0 are, respectively, the magnetic and electric permeabilities of vacuum. $\mathbf{I}$ denotes the intensity of magnetization, and $\mathbf{P}$ denotes polarization; these may generally be nonlinear [2], but will assumed to be linear throughout this work, leading to the standard linear constitutive relation:

$$\begin{aligned} \mathbf{B} &:= \boldsymbol{\mu} \mathbf{H}, \\ \mathbf{D} &:= \boldsymbol{\varepsilon} \mathbf{E}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\mu} : \Omega \to \mathbb{M}$ and $\boldsymbol{\varepsilon} : \Omega \to \mathbb{M}$ denote, respectively, the permeability and permittivity of the material and $\mathbb{M}$ denotes the space of real-valued rank-2 tensors.

Substituting the constitutive laws (2.3) into Maxwell's equations (2.2) gives the time-domain electromagnetic wave equations:

$$\text{div}\left(\boldsymbol{\varepsilon} \mathbf{E}\right) = \rho \tag{2.4a}$$

$$\text{div}\left(\boldsymbol{\mu} \mathbf{H}\right) = 0 \tag{2.4b}$$

$$\nabla \times \mathbf{E} = -\frac{\partial (\boldsymbol{\mu} \mathbf{H})}{\partial t} \tag{2.4c}$$

$$\nabla \times \mathbf{H} = \mathbf{J}^{\text{imp}} + \frac{\partial (\boldsymbol{\varepsilon} \mathbf{E})}{\partial t} \tag{2.4d}$$

Let $\mathbf{n}$ denote the outward normal to the boundary; admissible boundary conditions include:

- Perfect electric conductor:

$$\mathbf{n} \times \mathbf{E} = \mathbf{n} \times \mathbf{E}_0 \qquad \text{on } \Gamma_1$$

- Prescribed electric surface current (Perfect magnetic conductor):

$$\mathbf{n} \times \mathbf{H} = \mathbf{J}_S^{\text{imp}} := \mathbf{n} \times \mathbf{H}_0 \qquad \text{on } \Gamma_2$$

- Impedance boundary:

$$\mathbf{n} \times \mathbf{H} + d\mathbf{E}_t = \mathbf{J}_S^{\text{imp}} \qquad \text{on } \Gamma_3$$

where $\mathbf{E}_t := -\mathbf{n} \times \mathbf{n} \times \mathbf{E}$ is the tangential component of the electric field $\mathbf{E}$, $d > 0$ is the prescribed impedance, and $\mathbf{J}_S^{\text{imp}}$ denotes the impressed electric surface current and is tangent to the boundary.

2.1.3 Linear Elasticity

The linear elastic model is given by

$$\operatorname{div} \boldsymbol{\sigma} + \mathbf{f} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2}, \tag{2.5a}$$

$$\mathsf{S} : \boldsymbol{\sigma} = \varepsilon(\mathbf{u}), \tag{2.5b}$$

where (2.5a) enforces conservation of linear momentum and (2.5b) is the linear elastic constitutive relation in compliance form. Vector field $\mathbf{u} : \Omega \to \mathbb{R}^3$ defines the displacement generated by body force $\mathbf{f} : \Omega \to \mathbb{R}^3$, and stress $\boldsymbol{\sigma} : \Omega \to \mathbb{S}$ is a real symmetric second-order tensor. Strain $\varepsilon(\mathbf{u})$ is defined to be the symmetric gradient of displacement $\mathbf{u}$, i.e.

$$\varepsilon(\mathbf{u}) := \frac{1}{2}\big(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathsf{T}}\big).$$

Compliance tensor $\mathsf{S} : \mathbb{S} \to \mathbb{S}$ is a fourth-order tensor with major and minor symmetries:

$$\mathsf{S}_{ijkl} = \mathsf{S}_{jikl} = \mathsf{S}_{ijlk} \quad \text{(minor symmetry)}$$

$$\mathsf{S}_{ijkl} = \mathsf{S}_{klij} \qquad\quad \text{(major symmetry)}$$

The compliance tensor is the inverse of the more common *stiffness* or *elastic* tensor C over symmetric rank-2 tensors, i.e. $\mathsf{S} = \mathsf{C}^{-1} : \mathbb{S} \to \mathbb{S}$. Indeed, constitutive relation (2.5b) can be expressed in equivalent stiffness form:

$$\mathsf{C} : \varepsilon(\mathbf{u}) = \boldsymbol{\sigma}.$$

In this form, the first-order elastic system reduces to the more compact second-order form

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} - \operatorname{div}(\mathsf{C} : \varepsilon(\mathbf{u})) = \mathbf{f};$$

however, the first-order system (2.5) will be a convenient starting point for deriving numerical methods in future sections.

Symmetries of the compliance tensor S and stiffness tensor C imply that only 21 of 81-components can be independently specified. In the context of wave propagation, isotropic stiffness tensors C are commonly specified with pressure (compressive) wavespeed V_p, shear wavespeed V_s, and density ρ. In Voigt notation, the stiffness tensor C takes the form:

$$\begin{bmatrix} C_{11} & C_{12} & C_{12} & & & \\ C_{12} & C_{11} & C_{12} & & & \\ C_{12} & C_{12} & C_{11} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{44} \end{bmatrix}, \tag{2.6}$$

with

$$C_{11} = \rho V_p^2$$
$$C_{44} = \rho V_s^2$$
$$C_{12} = C_{11} - 2C_{44}$$

Admissible boundary conditions for elasticity include:

- Prescribed displacement:

$$\mathbf{u} = \mathbf{u}_0 \qquad \text{on } \Gamma_1$$

- Prescribed traction:

$$\boldsymbol{\sigma}\mathbf{n} = \mathbf{t}_0 \qquad \text{on } \Gamma_2$$

Note that in the case of elasticity, boundary values are vector-valued, boundary conditions can thus be mixed. For example, a symmetric boundary condition can be enforced on a face normal to the x-direction by prescribing normal displacement $\mathbf{u}_x = 0$, and tangential tractions, $\boldsymbol{\sigma}_{xy} = \boldsymbol{\sigma}_{xz} = 0$. To simplify notation we neglect to enumerate each case. We further neglect impedance boundary conditions for elasticity and will instead leverage perfectly matched layer boundary conditions (Section 2.3) when a radiation-type boundary condition is required.

2.2 Frequency-Domain Wave Propagation

Time-harmonic wave equations are derived by assuming fields and sources are oscillating, monochromatic fields. Let $\mathfrak{u}$ and $\mathfrak{f}$ denote group variables and loads, the time-harmonic

equations will be derived by introducing, then factoring out the ansatz:

$$\mathfrak{u}(x,t) = \Re\big\{e^{i\omega t}\,\widetilde{\mathfrak{u}}(x)\big\}, \tag{2.7}$$

$$\mathfrak{f}(x,t) = \Re\big\{e^{i\omega t}\,\widetilde{\mathfrak{f}}(x)\big\}, \tag{2.8}$$

where i denotes the complex unit and $\widetilde{\mathfrak{u}}$, $\widetilde{\mathfrak{f}}$ are complex-valued fields. The remainder of this work considers only time-harmonic wave equations, we thus simplify notation by dropping the tilde $(\widetilde{\cdot})$, overloading time-dependent variables with corresponding complex-valued time-harmonic analogs.

Linear Acoustics. The group variable for the linear acoustics problem is composed of pressure p and velocity $\mathbf{u}$:

$$\mathfrak{u}_{\text{acoustic}} := (\,p,\,\mathbf{u}\,)^{\mathsf{T}}$$

Introducing the time-harmonic ansatz in the linear acoustic system (2.1) and nondimensionalizing leads to the time-harmonic system

$$\begin{cases} \dfrac{i\omega}{c^2}p + \operatorname{div}(\mathbf{u}) = \mathrm{f}_p & \text{in } \Omega \\ i\omega\mathbf{u} + \nabla p = \mathbf{f}_{\mathbf{u}} & \text{in } \Omega \\ p = p_0 & \text{on } \Gamma_1 \\ u_n = u_0 & \text{on } \Gamma_2 \\ u_n - dp = u_0, & \text{on } \Gamma_3 \end{cases} \tag{2.9}$$

where, $p : \Omega \to \mathbb{C}$ and $\mathbf{u} : \Omega \to \mathbb{C}^3$ are complex-valued fields.

System (2.9) can instead be expressed in more compact operator notation by defining

$$A_{\text{acoustic}} := \begin{pmatrix} i\dfrac{\omega}{c^2} & \operatorname{div} \\ \nabla & i\omega\mathbf{I} \end{pmatrix}, \tag{2.10}$$

where we have introduced $\mathbf{I}$, a rank-2 "diffusion" tensor that is currently defined to be the rank-2 identity tensor, this tensor will simplify the definition of PML boundary conditions later in Section 2.3. Homogeneous boundary conditions can then built-in to the definition of the domain of A_{acoustic}:

$$D(A) = \Big\{\mathfrak{u} \in \big(L^2(\Omega)\big)^4 : A\mathfrak{u} \in \big(L^2(\Omega)\big)^4,\, p = 0 \text{ on } \Gamma_1,\, u_n = 0 \text{ on } \Gamma_2,\, u_n = dp \text{ on } \Gamma_3\Big\}.$$

Note that the time-harmonic acoustic operator A_{acoustic} is formally skew adjoint, i.e.

$$A^*_{\text{acoustic}} = -A_{\text{acoustic}}.$$

Time-harmonic acoustic system (2.9) (and the remaining wave propagation operators), can then be written:

$$\begin{cases} \mathfrak{u} = \widetilde{\mathfrak{u}}_0 + D(A), \\ (A\mathfrak{u}, \mathfrak{v}) = 0 \qquad \mathfrak{v} \in L^2(\Omega). \end{cases} \tag{2.11}$$

where $\widetilde{\mathfrak{u}}_0$ is a lift of the boundary data and $\mathfrak{v} \in L^2(\Omega)$ is understood component-wise.

Electromagnetics. The electromagnetic group variable includes electric and magnetic fields,

$$\mathfrak{u}_{\text{EM}} := (\,\mathbf{E},\, \mathbf{H}\,)^{\mathsf{T}}$$

Substituting the time-harmonic ansatz in the electromagnetic wave equations (2.4) gives the time-harmonic electromagnetic wave equations:

$$\begin{cases} \nabla \times \mathbf{E} + i\omega\boldsymbol{\mu}\mathbf{H} = \mathbf{0} & \text{in } \Omega \\ \nabla \times \mathbf{H} - i\omega\boldsymbol{\varepsilon}\mathbf{E} = \mathbf{J}^{\text{imp}} & \text{in } \Omega \\ \mathbf{n} \times \mathbf{E} = \mathbf{n} \times \mathbf{E}_0 & \text{on } \Gamma_1 \\ \mathbf{n} \times \mathbf{H} = \mathbf{n} \times \mathbf{H}_0 & \text{on } \Gamma_2 \\ \mathbf{n} \times \mathbf{H} + d\mathbf{E}_t = \mathbf{J}^{\text{imp}}_S & \text{on } \Gamma_3 \end{cases} \tag{2.12}$$

where the impressed current is again assumed to satisfy the compatibility condition $\text{div}(\mathbf{J}^{\text{imp}}) = i\omega\rho^{\text{imp}}$. Note that we have neglected Gauss' laws (2.4a & 2.4b) in the time-harmonic electromagnetic system (2.12). Indeed, once the first two equations (Faraday's law and Ampere's law, respectively) are satisfied, Gauss' laws will be automatically satisfied, as can be seen by taking the divergence of Faraday's law and Ampere's law.

This problem can again be expressed in operator form by defining:

$$A_{\text{EM}} := \begin{pmatrix} -i\omega\boldsymbol{\varepsilon} & \nabla\times \\ \nabla\times & i\omega\boldsymbol{\mu} \end{pmatrix} \tag{2.13}$$

with domain

$$D(A) = \Big\{\mathfrak{u} \in \big(L^2(\Omega)\big)^6 : A\mathfrak{u} \in \big(L^2(\Omega)\big)^6, \\ \mathbf{n} \times \mathbf{E} = 0 \text{ on } \Gamma_1,\ \mathbf{n} \times \mathbf{H} = 0 \text{ on } \Gamma_2,\ \mathbf{n} \times \mathbf{H} + d\mathbf{E}_t = 0 \text{ on } \Gamma_3\Big\}.$$

Due to asymmetry of the cross-product, A_{EM} is not formally skew-adjoint (in this form, rotating one variable 90° in the complex plane yields a skew-adjoint operator).

Linear Elasticity. Substitution and subsequent elimination of the time-harmonic ansatz in the linear elastic system (2.5) yields the time-harmonic operator:

$$\begin{aligned} \operatorname{div} \boldsymbol{\sigma} + \rho\omega^2 \mathbf{u} &= -\mathbf{f}, \\ -\mathsf{S} : \boldsymbol{\sigma} + \varepsilon(\mathbf{u}) &= \mathbf{0}. \end{aligned}$$

However, to better mirror the acoustic and electromagnetic operators, it is convenient to formulate the system in terms of velocity $\mathbf{v}$ instead of displacement, i.e. $\mathbf{v} = i\omega\mathbf{u}$:

$$\begin{cases} \operatorname{div} \boldsymbol{\sigma} - i\omega\rho\mathbf{v} = \mathbf{f} & \text{in } \Omega, \\ -i\omega\mathsf{S} : \boldsymbol{\sigma} + \varepsilon(\mathbf{v}) = \mathbf{0} & \text{in } \Omega, \\ \mathbf{v} = \mathbf{v}_0 & \text{on } \Gamma_1, \\ \boldsymbol{\sigma}\mathbf{n} = \mathbf{t}_0 & \text{on } \Gamma_2. \end{cases} \tag{2.14}$$

One advantage of the velocity formulation is that variables are similarly scaled with respect to ω, which simplifies weighting of the adjoint graph test norm for the DPG discretization, introduced in Chapter 3. Still, this form of the elastic system presents a number of challenges for discretization and an alternative definition of the elastic operator will be presented in Chapter 3, introducing an infinitesimal rotation tensor $\mathbf{r}$.

Velocity and stress form the group variable for linear elasticity:

$$\mathfrak{u}_{\text{elastic}} := (\, \mathbf{v}, \boldsymbol{\sigma} \,)^{\mathsf{T}},$$

and the time-harmonic elastic wave operator is defined

$$A_{\text{elastic}} := \begin{pmatrix} -i\omega\rho & \operatorname{div} \\ \frac{1}{2}\big(\nabla(\cdot) + \nabla(\cdot)^{\mathsf{T}}\big) & -i\omega\mathsf{S} : \end{pmatrix} \tag{2.15}$$

with domain

$$D(A) = \Big\{ \mathfrak{u} \in \big(L^2(\Omega)\big)^9 : A\mathfrak{u} \in \big(L^2(\Omega)\big)^9,\ \mathbf{v} = \mathbf{0} \text{ on } \Gamma_1,\ \boldsymbol{\sigma}\mathbf{n} = \mathbf{0} \text{ on } \Gamma_2 \Big\}.$$

Note that symmetry of $\boldsymbol{\sigma}$ was strongly enforced and operator A_{elastic} is thus formally skew-adjoint.

2.3 Perfectly Matched Layers

Impedance boundary conditions, introduced previously for acoustic and electromagnetic operators, are a first-order approximation of radiation boundary conditions that permit outgoing wavefields to leave the domain. Radiation boundary conditions for the acoustic Helmholtz problem are given by the Sommerfeld radiation condition which, in three-dimensions, takes the form:

$$\lim_{r\to\infty} r\left(\frac{\partial p}{\partial r} - i\omega p\right) = 0,$$

where $r := |\mathbf{x}|$. More accurate approximations of the Sommerfeld-type radiation conditions can be obtained with perfectly matched layer (PML) boundary conditions, a class of non-reflecting boundary conditions that append finite-width absorbing layers to domain Ω. Absorbing layers dissipate outgoing (from the perspective of Ω) waves and are defined by specifying a complex-valued "stretching" function φ. Pullback maps will be used to derive the form of the absorbing material for each wave operator under a general complex stretching φ. On hexahedral domains, with $\Omega = (-l, l)^3$ and $\Omega_E = (-L, L)^3$ with $l < L$, the Cartesian stretching function is frequently adopted:

$$\varphi_j(\mathbf{x}) := \begin{cases} \mathbf{x}_j & |\mathbf{x}_j| \leq l \\ \mathbf{x}_j + \dfrac{iC}{\omega}\left(\dfrac{\mathbf{x}_j - l}{L - l}\right)^{\beta} & |\mathbf{x}_j| > l \end{cases} \tag{2.16}$$

where C, β are parameters used to modify the rate of decay in the PML absorbing layer.

Exact sequence energy spaces. Before introducing pullback maps we briefly recall the three-dimensional exact sequence energy spaces. We denote the L^2-inner product over Ω by $(\cdot, \cdot)$ and the L^2-norm by $\|\cdot\|$; the exact sequence energy spaces are defined:

$$\begin{aligned} L^2(\Omega) &= \{q : \Omega \to \mathbb{R}(\mathbb{C}) : \|q\| < \infty\}, \\ H^1(\Omega) &= \{w : \Omega \to \mathbb{R}(\mathbb{C}) : w \in L^2(\Omega), \nabla w \in (L^2(\Omega))^3\}, \\ H(\mathrm{curl}, \Omega) &= \{\mathbf{E} : \Omega \to \mathbb{R}^3(\mathbb{C}^3) : \mathbf{E} \in (L^2(\Omega))^3, \nabla \times \mathbf{E} \in (L^2(\Omega))^3\}, \\ H(\mathrm{div}, \Omega) &= \{\mathbf{v} : \Omega \to \mathbb{R}^3(\mathbb{C}^3) : \mathbf{v} \in (L^2(\Omega))^3, \mathrm{div}\, \mathbf{v} \in L^2(\Omega)\}. \end{aligned}$$

Energy spaces communicate regularity: $L^2(\Omega)$ corresponds to (an equivalence class of) functions that are L^2 integrable over Ω, $H^1(\Omega)$ to functions with value and gradient in $L^2(\Omega)$, $H(\mathrm{curl}, \Omega)$ to functions with value and curl in $L^2(\Omega)$, etc. In the present derivation of PML boundary conditions, identifying the energy setting of different variables reveals how they

are transformed under the complex stretching. For example, the p and ∇p terms in the acoustics operator indicate that for $A\mathbf{u}$ to be in $L^2(\Omega)$, p should be set in $H^1(\Omega)$. Similarly $\mathbf{u}$ and $\mathrm{div}(\mathbf{u})$ terms indicate $\mathbf{u}$ should be set in $H(\mathrm{div},\Omega)$.

Pullback maps. Let Ω_E denote an extended domain, (i.e. $\Omega \subset \Omega_E$) and $\varphi(\mathbf{x}) : \Omega_E \subset \mathbb{R}^3 \to \widetilde{\Omega}_E \subset \mathbb{C}^3$ denote a complex stretching of the extended domain such that $\varphi(\mathbf{x})\big|_\Omega = \mathbf{x}$. Pullback maps establish a mapping between exact sequences defined on the extended and stretched domains. Pullback maps can be derived from commutativity of the de Rham diagram:

$$\begin{array}{ccccccc} H^1(\Omega_E) & \xrightarrow{\nabla} & H(\mathrm{curl},\Omega_E) & \xrightarrow{\nabla\times} & H(\mathrm{div},\Omega_E) & \xrightarrow{\nabla\cdot} & L^2(\Omega_E) \\ \downarrow T^{\mathrm{grad}} & & \downarrow T^{\mathrm{curl}} & & \downarrow T^{\mathrm{div}} & & \downarrow T^{L^2} \\ H^1(\widetilde{\Omega}_E) & \xrightarrow{\widetilde{\nabla}} & H(\mathrm{curl},\widetilde{\Omega}_E) & \xrightarrow{\widetilde{\nabla}\times} & H(\mathrm{div},\widetilde{\Omega}_E) & \xrightarrow{\widetilde{\nabla}\cdot} & L^2(\widetilde{\Omega}_E). \end{array} \tag{2.17}$$

Let $\mathcal{J}$ denote the Jacobian of the complex stretching, i.e. $\mathcal{J} := \dfrac{\partial\varphi}{\partial\mathbf{x}}$, and $j := |\mathcal{J}|$, the pullbacks maps are defined:

$$H^1(\Omega) \ni u \longmapsto T^{\mathrm{grad}}u := \quad u\circ\varphi^{-1} = \widetilde{u} \quad \in H^1(\widetilde{\Omega}_E) \tag{2.18}$$

$$H(\mathrm{curl},\Omega_E) \ni \mathbf{E} \longmapsto T^{\mathrm{curl}}\mathbf{E} := \quad (\mathcal{J}^{-T}\mathbf{E})\circ\varphi^{-1} = \widetilde{\mathbf{E}} \quad \in H(\mathrm{curl},\widetilde{\Omega}_E) \tag{2.19}$$

$$H(\mathrm{div},\Omega_E) \ni \mathbf{v} \longmapsto T^{\mathrm{div}}\mathbf{v} := \quad (j^{-1}\mathcal{J}\mathbf{V})\circ\varphi^{-1} = \widetilde{\mathbf{v}} \quad \in H(\mathrm{div},\widetilde{\Omega}_E) \tag{2.20}$$

$$L^2(\Omega_E) \ni q \longmapsto T^{L^2}q := \quad (j^{-1}q)\circ\varphi^{-1} = \widetilde{q} \quad \in L^2(\widetilde{\Omega}_E). \tag{2.21}$$

Acoustics. Considering the acoustic system defined on the stretched domain,

$$\begin{cases} \dfrac{i\omega}{c^2}\widetilde{p} + \widetilde{\mathrm{div}}(\widetilde{\mathbf{u}}) = 0, \\ \quad i\omega\,\mathbf{I}\,\widetilde{\mathbf{u}} + \widetilde{\nabla}\widetilde{p} = 0. \end{cases} \tag{2.22}$$

The PML can be derived by pulling system (2.22) back to the extended domain. For example, $\widetilde{p} \in H^1(\widetilde{\Omega}_E)$ implies $\widetilde{p} = p\circ\varphi^{-1}$ while, according to diagram (2.17), $\widetilde{\nabla}\widetilde{p} \in H(\mathrm{curl},\widetilde{\Omega}_E)$ implies $\widetilde{\nabla}\widetilde{p} = (\mathcal{J}^{-T}\nabla p)\circ\varphi^{-1}$. Pulling back the acoustics operator leads to the following stretched system:

$$\begin{cases} \dfrac{i\omega}{c^2}p + j^{-1}\,\mathrm{div}(\mathbf{u}) = 0, \\ i\omega j^{-1}\mathcal{J}\,\mathbf{I}\,\mathbf{u} + \mathcal{J}^{-\mathsf{T}}\nabla p = 0. \end{cases}$$

Rearranging, stretching factors can be grouped with the wavespeed c and diffusion (identity) tensor $\mathbf{I}$ (introduced in (2.10) for precisely this purpose):

$$\begin{cases} \dfrac{i\omega j}{c^2} p + \operatorname{div}(\mathbf{u}) = 0, \\ i\omega j^{-1} \mathcal{J}^{\mathsf{T}} \mathcal{J}\, \mathbf{I}\, \mathbf{u} + \nabla p = 0. \end{cases}$$

This motivates definition of modified materials

$$c' = c j^{-1/2} \qquad \text{and} \qquad \mathbf{I}' = j^{-1} \mathcal{J}^{\mathsf{T}} \mathcal{J}. \tag{2.23}$$

Definition of modified materials simplifies the implementation of the PML: the extended domain and stretching function define the complex-valued modified materials which implement the complex stretching.

Electromagnetics. PML boundary conditions can be derived similarly for the electromagnetic operator (2.13). Pulling back the operator defined on the stretched domain $\widetilde{\Omega}_E$ to the extended domain Ω_E gives:

$$\begin{cases} j^{-1} \mathcal{J} \nabla \times \mathbf{E} + i\omega \boldsymbol{\mu} \mathcal{J}^{-\mathsf{T}} \mathbf{H} = 0, \\ j^{-1} \mathcal{J} \nabla \times \mathbf{H} - i\omega \boldsymbol{\varepsilon} \mathcal{J}^{-\mathsf{T}} \mathbf{E} = 0. \end{cases}$$

Rearranging to move the stretching factors onto $\boldsymbol{\varepsilon}$ and $\boldsymbol{\mu}$ terms yields the modified permittivity and permeability:

$$\boldsymbol{\varepsilon}' := j \mathcal{J}^{-1} \boldsymbol{\varepsilon} \mathcal{J}^{-\mathsf{T}} \qquad \text{and} \qquad \boldsymbol{\mu}' := j \mathcal{J}^{-1} \boldsymbol{\mu} \mathcal{J}^{-\mathsf{T}}. \tag{2.24}$$

Elasticity. To derive PML expressions for elasticity we further assume that $\mathcal{J} = \mathcal{J}^{\mathsf{T}}$, simplifying treatment of the symmetric gradient term. Pulling back the elastic system from the stretched domain gives:

$$\begin{cases} j^{-1} \operatorname{div} \boldsymbol{\sigma} - i\omega \rho \mathbf{v} = \mathbf{0}, \\ -i\omega \mathsf{S} : j^{-1} \mathcal{J} \boldsymbol{\sigma} + \mathcal{J}^{-1} \varepsilon(\mathbf{v}) = \mathbf{0}. \end{cases}$$

Straightforward manipulations again permit grouping of the stretching terms to define modified material properties:

$$\rho' = j\rho \qquad \text{and} \qquad \mathsf{S}'_{ijkl} = j^{-1} \mathcal{J}_{ni} \mathcal{J}_{mk} \mathsf{S}_{njml} \tag{2.25}$$

However, note that minor symmetries of S are violated in the modified compliance tensor S'.

Chapter 3

Discontinuous Petrov-Galerkin Method

3.1 Variational formulations

Variational formulations are at the heart of finite element methods; finite element practitioners are thus well aware of the classical weak formulation, commonly derived starting from the second order system. For example, the standard weak formulation for the Helmholtz (linear acoustics) problem starts from the Helmholtz operator:

$$-\Delta p - \frac{\omega^2}{c^2} p = f$$

Introducing test function v and integrating over the domain, we have:

$$-\int_\Omega \Delta p \bar{v} - \int_\Omega \frac{\omega^2}{c^2} p \bar{v} = \int_\Omega f \bar{v}.$$

where $\bar{\cdot}$ denotes complex conjugation. Applying integration by parts to the first term, yields the standard weak form:

$$\int_\Omega \nabla p \cdot \nabla \bar{v} - \int_{\Gamma_2} \partial_n p \bar{v} - \int_\Omega \frac{\omega^2}{c^2} p \bar{v} = \int_\Omega f \bar{v}.$$

In fact, this is just one of six variational formulations for linear acoustics. The other variational formulations can be derived starting from the first-order operators given in Chapter 2, then applying integration by parts to weaken (relax) none, either, or both of the equations (this gives four variational formulations, the remaining two are derived by eliminating one of the variables in the so-called mixed formulations). The various variational formulations for a variety of second-order operators are given in [26], and can be shown to simultaneously well- or ill-posed.

We note only two abstract variational problems for the wave operators defined previously. The first is the so-called *trivial* variational formulation that introduces group test variable $\mathfrak{v}$, and neglects to weaken either equation.

$$\begin{cases} \mathfrak{u} \in D(A) \\ (A\mathfrak{u}, \mathfrak{v}) = (\mathfrak{f}, \mathfrak{v}), \qquad \mathfrak{v} \in L^2(\Omega) \end{cases} \tag{3.1}$$

where $\mathfrak{v} \in L^2(\Omega)$ is used to denote that each component is individually an L^2 field.

At the other extreme we have the so-called *ultraweak* variational formulation, defined by weakening both equations:

$$\begin{cases} \mathfrak{u} \in L^2(\Omega) \\ (\mathfrak{u}, A^*\mathfrak{v}) = (\mathfrak{f}, \mathfrak{v}), \qquad \mathfrak{v} \in D(A^*), \end{cases} \tag{3.2}$$

Note that both trivial and ultraweak variational formulations have an asymmetric functional setting, requiring a Petrov–Galerkin discretization with different trial and test spaces.

3.2 The Ideal Petrov–Galerkin Method

The ideal Petrov–Galerkin methodology is a natural predecessor to the DPG methodology. This section briefly introduces the ideal Petrov–Galerkin method in the context of three characterizations or "hats" [30] that motivate important properties—namely Hermitian positive-definiteness, mesh-independent stability, and a built-in error indicator—of the DPG methodology.

Hat 1: A minimum residual method. To this end, consider a (well-posed) variational problem posed on complex-valued Hilbert spaces U, V

$$\begin{cases} u \in U \\ b(u,v) = l(v), \qquad v \in V, \end{cases} \tag{3.3}$$

where $b(\cdot,\cdot)$ is a continuous sesquilinear funcional on $U \times V$, and $l(\cdot) \in V'$. Sesquiliinear functional $b(\cdot,\cdot)$ defines linear operator $B : U \mapsto V'$ through duality pairing,

$$\langle Bu, v\rangle_{V'\times V} = b(u,v).$$

The abstract variational problem in (3.3) is then equivalent to minimization of the residual in dual space V', i.e.

$$u = \arg\min_{w\in U} \frac{1}{2}||l - Bw||^2_{V'}. \tag{3.4}$$

Instead of minimizing in the dual norm, we can introduce the Riesz operator $R_V : V \mapsto V'$, and recast problem (3.4) as a minimization in V:

$$u = \arg\min_{w\in U} \frac{1}{2}||R_V^{-1}(l - Bw)||^2_{V}. \tag{3.5}$$

Requiring the Gâteaux derivative of the cost functional in (3.5) to vanish, and considering now a finite dimensional trial space $U_h \subset U$, yields the variational problem:

$$\begin{cases} u_h \in U_h \\ (R_V^{-1}Bu_h, R_V^{-1}B\delta u_h)_V = (R_V^{-1}l, R_V^{-1}B\delta u_h), \qquad \delta u_h \in U_h. \end{cases} \tag{3.6}$$

where the left-hand side $(R_V^{-1}B\cdot, R_V^{-1}B\cdot)_V$ defines an inner product, and thus trivially defines a Hermitian positive-definite variational problem. The inner product defined by (3.6) further motivates definition of the energy norm for the ideal Petrov–Galerkin method as:

$$||u||_E = ||R_V^{-1}Bu||_V = ||Bu||_{V'} = \sup_{v\in V}\frac{|b(u,v)|}{||v||_V}. \tag{3.7}$$

Hat 2: A method of optimal test functions. An additional characterization of the ideal Petrov–Galerkin method can be observed by defining the *trial-to-test* operator $T : U_h \mapsto V$ by

$$(T\delta u_h, \delta v)_V = b(\delta u_h, \delta v), \qquad \delta u_h \in U_h, \delta v \in V; \tag{3.8}$$

i.e. $T = R_V^{-1}B$. Defining now the discrete test space $V_h^{opt} := T(U_h)$ leads to the equivalent Petrov–Galerkin problem

$$\begin{cases} u_h \in U_h \subset U \\ b(u_h, v_h) = l(v_h), \qquad v_h \in V_h^{opt}. \end{cases} \tag{3.9}$$

The test space V_h^{opt} trivially satisfies the discrete inf-sup condition since, for $u_h \in U_h$,

$$\frac{b(u_h, v_h)}{||v_h||_V} = \frac{b(u_h, Tu_h)}{||Tu_h||_V} = ||u_h||_E = \sup_{v \in V} \frac{|b(u_h, v)|}{||v||_V}.$$

As a method of optimal test functions, the ideal Petrov–Galerkin method is thus uniformly and unconditionally stable, inheriting stability from the continuous problem.

Hat 3: A mixed method. Finally, by introducing $\psi := R_V^{-1}(l - Bu_h)$, the Riesz representation of the residual, the ideal Petrov–Galerkin method can be cast as a mixed problem:

$$\begin{cases} u_h \in U_h, \psi \in V \\ (\psi, v)_V + b(u_h, v) & = l(v), \quad v \in V \\ b(w_h, \psi) & = 0, \quad w_h \in U_h. \end{cases} \tag{3.10}$$

Computing the error $e = u_h - u$ in the energy norm reveals,

$$||u_h - u||_E = ||B(u_h - u)||_{V'} = ||Bu_h - l||_{V'} = ||R_V^{-1}(l - Bu_h)||_V = ||\psi||_V. \tag{3.11}$$

The ideal Petrov–Galerkin methodology could therefore be seen to provide a built-in error indicator. Note however that ψ, arising from the continuous Riesz operator R_V, is set in continuous space V and problem (3.10) is not yet amenable to computation.

3.2.1 The Practical Petrov–Galerkin Method

The practical Petrov–Galerkin methodology with optimal test functions replaces infinite dimensional space V in (3.10) with an enriched finite dimensional space $V_h^r \subset V$:

$$\begin{cases} u_h \in U_h, \psi_h \in V_h^r \\ (\psi_h, v_h)_V + b(u_h, v_h) & = l(v_h), \quad v_h \in V_h^r \\ b(w_h, \psi_h) & = 0, \quad w_h \in U_h. \end{cases} \tag{3.12}$$

Substituting the finite-dimensional test space V_h^r diminishes stability of the problem compared to the ideal method, but the decrease in stability is often minor and can be quantified by constructing Fortin operators [50, 95].

Use of finite-dimensional test space V_h^r in the practical Petrov–Galerkin yields a fully discrete system that can be solved for u_h, ψ_h. Field ψ_h can further be used to define a built-in error indicator via element-wise contributions to $||\psi_h||_V$. While the practical Petrov–Galerkin methodology is computable, it yields a mixed system of the form

$$\begin{bmatrix} \mathsf{G} & \mathsf{B} \\ \mathsf{B}^* & 0 \end{bmatrix} \begin{bmatrix} \mathsf{s} \\ \mathsf{u} \end{bmatrix} = \begin{bmatrix} \mathsf{f} \\ 0 \end{bmatrix} \tag{3.13}$$

where G is discretized by the enriched test space V_h^r and is thus often significantly larger than B. Comparing system (3.13) to an analogous Galerkin discretization reveals that the practical Petrov–Galerkin has not only significantly *increased* the size of the problem, it has further imparted a saddle point structure. Indeed, while the practical Petrov–Galerkin method is computable, it is (at the moment) not computationally competitive. However, Monsuur and Stevenson recently showed that a practical Petrov–Galerkin method for Helmholtz is practically pollution-free, given a modest increase in order of the enriched test variable s with wavenumber [91]. Development of preconditioners and matrix-free solvers for mixed system (3.13) could improve the competitiveness of the practical Petrov–Galerkin methodology in the future.

3.3 The Discontinuous Petrov–Galerkin Method

The discontinuous Petrov–Galerkin methodology was developed to improve the computability of the practical Petrov–Galerkin method. The core idea is that testing with a larger discontinuous ("broken") enriched test space block-diagonalizes the Gram matrix G, allowing it to be eliminated at the element level. Breaking test spaces introduces additional trace variables [19] defined on the mesh skeleton. Denoting these new trace variables $\hat{u} \in \hat{U}_h$, the DPG system takes the form

$$\begin{cases} u_h \in U_h, \hat{u}_h \in \hat{U}_h, \psi_h \in V_h^r & & \\ (\psi_h, v_h)_V + b(u_h, v_h) + \hat{b}(\hat{u}_h, v_h) & = l(v_h), & v_h \in V_h^r \\ b(w_h, \psi_h) & = 0, & w_h \in U_h \\ \hat{b}(\hat{w}_h, \psi_h) & = 0, & \hat{w}_h \in \hat{U}_h. \end{cases} \tag{3.14}$$

The remainder of this section first introduces the energy spaces needed to rigorously define DPG systems for the wave propagation operators. Ultraweak variational formulations

are then given for the wave propagation operators, followed by the corresponding *broken* ultraweak forms. Corresponding DPG methods are finally defined.

3.3.1 Energy Spaces

Consider a bounded domain $\Omega \subset \mathbb{R}^3$ with Lipschitz boundary $\Gamma \equiv \partial\Omega$. L^2 inner products over Ω and Γ are denoted:

$$(u,v) := \int_\Omega u\bar{v} \qquad \text{and} \qquad \langle u,v\rangle := \int_\Gamma u\bar{v}.$$

The standard energy spaces were defined previously in Section 2.3 and are repeated here:

$$\begin{aligned}
L^2(\Omega) &= \{q : \Omega \to \mathbb{R}(\mathbb{C}) : \|q\| < \infty\},\\
H^1(\Omega) &= \{w : \Omega \to \mathbb{R}(\mathbb{C}) : w \in L^2(\Omega), \nabla w \in \boldsymbol{L}^2(\Omega)\},\\
H(\mathrm{curl},\Omega) &= \{\mathbf{E} : \Omega \to \mathbb{R}^3(\mathbb{C}^3) : \mathbf{E} \in \boldsymbol{L}^2(\Omega), \nabla\times\mathbf{E} \in (L^2(\Omega))^3\},\\
H(\mathrm{div},\Omega) &= \{\mathbf{v} : \Omega \to \mathbb{R}^3(\mathbb{C}^3) : \mathbf{v} \in \boldsymbol{L}^2(\Omega), \nabla\cdot\mathbf{v} \in L^2(\Omega)\},
\end{aligned}$$

where $\boldsymbol{L}^2(\Omega) := \big(L^2(\Omega)\big)^3$ is the space of vector-valued L^2 fields over Ω. We will similarly use, bold faced energy spaces to denote three-dimensional copy of other energy spaces, specifically:

$$\begin{aligned}
\boldsymbol{H}^1(\Omega) &:= \big(H^1(\Omega)\big)^3,\\
\boldsymbol{H}(\mathrm{div},\Omega) &:= \big(H(\mathrm{div},\Omega)\big)^3.
\end{aligned}$$

It is also convenient to define subspaces of the energy spaces vanishing on some part of the boundary, these will be denoted:

$$\begin{aligned}
H^1_{\Gamma_i}(\Omega) &= \{u \in H^1 : u|_{\Gamma_i} = 0\},\\
H_{\Gamma_i}(\mathrm{curl},\Omega) &= \{\mathbf{E} \in H(\mathrm{curl},\Omega) : \mathbf{E}|_{\Gamma_i} \times \mathbf{n}_{\Gamma_i} = 0\},\\
H_{\Gamma_i}(\mathrm{div},\Omega) &= \{\mathbf{v} \in H(\mathrm{div},\Omega) : \mathbf{v}|_{\Gamma_i} \cdot \mathbf{n}_{\Gamma_i} = 0\}.
\end{aligned}$$

Let Ω_h denote a finite-element mesh with elements $\{K\}_{K\in\Omega_h}$. Use of discontinuous test spaces in DPG will necessitate definition of broken energy spaces, these are defined as product spaces over elements:

$$\begin{aligned}
L^2(\Omega_h) &= \{q \in L^2(\Omega) : q|_K \in L^2(K)\ \forall K \in \Omega_h\} = L^2(\Omega),\\
H^1(\Omega_h) &:= \{w \in L^2(\Omega) : w|_K \in H^1(K)\ \forall K \in \Omega_h\} \supset H^1(\Omega),\\
H(\mathrm{curl},\Omega_h) &= \{E \in (L^2(\Omega))^3 : E|_K \in H(\mathrm{curl},K)\ \forall K \in \Omega_h\} \supset H(\mathrm{curl},\Omega),\\
H(\mathrm{div},\Omega_h) &:= \{v \in (L^2(\Omega))^3 : v|_K \in H(\mathrm{div},K)\ \forall K \in \Omega_h\} \supset H(\mathrm{div},\Omega).
\end{aligned}$$

Breaking test spaces [19] further introduces trace unknowns on the mesh skeleton $\Gamma_h := \{\partial K\}_{K\in\Omega_h}$. The trace spaces are understood as element-wise traces of globally conforming functions:

$$
\begin{aligned}
H^{1/2}(\Gamma_h) &:= \Big\{ \prod_{K\in\Omega_h} \gamma^K(w|_K) : w \in H^1(K) \Big\},\\
H^{-1/2}(\mathrm{div},\Gamma_h) &:= \Big\{ \prod_{K\in\Omega_h} \gamma_\top^K(\mathbf{E}|_K) : \mathbf{E} \in H(\mathrm{curl},K) \Big\},\\
H^{-1/2}(\mathrm{curl},\Gamma_h) &:= \Big\{ \prod_{K\in\Omega_h} \gamma_\perp^K(\mathbf{E}|_K) : \mathbf{E} \in H(\mathrm{curl},K) \Big\},\\
H^{-1/2}(\Gamma_h) &:= \Big\{ \prod_{K\in\Omega_h} \gamma_n^K(\mathbf{v}|_K) : \mathbf{v} \in H(\mathrm{div},K) \Big\},
\end{aligned}
$$

where γ^K, $\gamma_\top^K$, $\gamma_\perp^K$, and γ_n^K are element-wise continuous, tangential, rotated tangential, and normal trace operators [30, 27].

3.3.2 Ultraweak Time-Harmonic Wave Propagation

Before proceeding with the derivation of broken ultraweak formulations, we formalize the ultraweak variational problems for the time-harmonic wave propagation operators introduced in Section 2.2. To simplify the presentation we neglect impedance boundary conditions. Information on impedance boundary conditions and corresponding DPG formulations for the acoustic and electromagnetic problems can be found in [33, 60].

Acoustics. The ultraweak variational formulation, defined for the abstract setting in (3.2), takes the following form for the acoustics problem:

$$
\begin{cases}
p \in L^2(\Omega),\ \mathbf{u} \in \boldsymbol{L}^2(\Omega) & \\
i\omega(c^{-2}p,q) - (\mathbf{u},\nabla q) = -\langle u_0,q\rangle_{\Gamma_2} + (\mathrm{f}_p,q) \quad & q \in H^1_{\Gamma_1}(\Omega)\\
i\omega(\mathbf{u},\mathbf{v}) - (p,\mathrm{div}\,\mathbf{v}) = -\langle p_0,v_n\rangle_{\Gamma_1} + (\mathbf{f}_\mathbf{u},\mathbf{v}) \quad & \mathbf{v} \in H_{\Gamma_2}(\mathrm{div},\Omega).
\end{cases}
\tag{3.15}
$$

Note that p and $\mathbf{u}$ are set in $L^2(\Omega)$, the standard technique of using "finite energy lifts" [33] to enforce non-homogeneous boundary conditions is thus not well-defined. Instead, prescribed boundary conditions can be enforced either by including data as boundary loads (as above), or by introducing additional trace variables (see Section 3.3.3).

Electromagnetics. The ultraweak formulation for electromagnetics follows similarly:

$$\begin{cases} \mathbf{E}, \mathbf{H} \in \boldsymbol{L}^2(\Omega) & \\ -i\omega(\boldsymbol{\varepsilon}\mathbf{E}, \mathbf{F}) + (\mathbf{H}, \nabla \times \mathbf{F}) = \langle \mathbf{n} \times \mathbf{H}_0, \mathbf{F} \rangle_{\Gamma_2} & \mathbf{F} \in H_{\Gamma_1}(\mathrm{curl}, \Omega) \\ i\omega(\boldsymbol{\mu}\mathbf{H}, \mathbf{G}) + (\mathbf{E}, \nabla \times \mathbf{G}) = \langle \mathbf{n} \times \mathbf{E}_0, \mathbf{G} \rangle_{\Gamma_1} + (\mathbf{J}^{\mathrm{imp}}, \mathbf{G}) & \mathbf{G} \in H_{\Gamma_2}(\mathrm{curl}, \Omega) \end{cases} \tag{3.16}$$

Elasticity. Finally, the ultraweak formulation for elasticity is given by:

$$\begin{cases} \mathbf{v} \in \boldsymbol{L}^2(\Omega), \boldsymbol{\sigma} \in L^2(\Omega, \mathbb{S}) & \\ i\omega(\rho\mathbf{v}, \mathbf{w}) + (\boldsymbol{\sigma}, \nabla\mathbf{w}) = \langle \mathbf{t}_0, \mathbf{w} \rangle_{\Gamma_2} + (\mathbf{f}, \mathbf{v}) & \mathbf{w} \in \boldsymbol{H}^1_{\Gamma_1}(\Omega) \\ i\omega(\mathsf{S} : \boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{v}, \mathrm{div}\, \boldsymbol{\tau}) = \langle \mathbf{v}_0, \boldsymbol{\tau}\mathbf{n} \rangle_{\Gamma_1} & \boldsymbol{\tau} \in H_{\Gamma_2}(\mathrm{div}, \Omega; \mathbb{S}) \end{cases} \tag{3.17}$$

where $H_{\Gamma_2}(\mathrm{div}, \Omega; \mathbb{S}) \subset \boldsymbol{H}_{\Gamma_2}(\mathrm{div}, \Omega)$ denotes the space of symmetric rank-2 tensor-valued fields. Space $H_{\Gamma_2}(\mathrm{div}, \Omega; \mathbb{S})$ has proven challenging to discretize; we will thus introduce a second ultraweak formulation for elasticity, weakly enforcing symmetry of $\boldsymbol{\tau}$. To this end we introduce the infinitesimal rotation tensor $\mathbf{r} \in L^2(\Omega, \mathbb{A})$, where $\mathbb{A}$ denotes the space of complex-valued *anti-symmetric* tensors. The infinitesimal rotation tensor $\mathbf{r}$ is defined via three independent $L^2(\Omega)$ components and is thus simply discretized. The resulting ultraweak problem is then given by:

$$\begin{cases} \mathbf{v} \in \boldsymbol{L}^2(\Omega), \boldsymbol{\sigma} \in L^2(\Omega, \mathbb{S}), \mathbf{r} \in L^2(\Omega, \mathbb{A}) & \\ i\omega(\rho\mathbf{v}, \mathbf{w}) + (\boldsymbol{\sigma}, \nabla\mathbf{w}) = \langle \mathbf{t}_0, \mathbf{w} \rangle_{\Gamma_2} + (\mathbf{f}, \mathbf{v}) & \mathbf{w} \in \boldsymbol{H}^1_{\Gamma_1}(\Omega) \\ i\omega(\mathsf{S} : \boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{v}, \mathrm{div}\, \boldsymbol{\tau}) + i\omega(\mathbf{r}, \boldsymbol{\tau}) = \langle \mathbf{v}_0, \boldsymbol{\tau}\mathbf{n} \rangle_{\Gamma_1} & \boldsymbol{\tau} \in \boldsymbol{H}_{\Gamma_2}(\mathrm{div}, \Omega). \end{cases} \tag{3.18}$$

A comparison of elastic formulations (3.18) and (3.17) follows in Section 3.5. For now we will continue developing both, referring to them as elastic formulation I (with strongly symmetric test stresses) and elastic formulation II (with weakly symmetric test stresses), respectively.

3.3.3 Broken Ultraweak Time-Harmonic Wave Propagation

Use of broken test spaces necessitates introduction of additional trace variables [19]; these new trace variables can be understood as Lagrange multipliers, weakly enforcing continuity of the test space. Boundary conditions can then be enforced by building boundary data into the definition of the trace spaces. For example, trace spaces conforming to specified boundary data for the acoustics problem can be defined:

$$\begin{aligned} H^{1/2}_{p_0,\Gamma_1}(\Gamma_h) &:= \Big\{ \hat{p} \in H^{1/2}(\Gamma_h) : \hat{p} = p_0 \text{ on } \Gamma_1 \Big\}, \\ H^{-1/2}_{u_0,\Gamma_2}(\Gamma_h) &:= \Big\{ \hat{u}_n \in H^{-1/2}(\Gamma_h) : \hat{u}_n = u_0 \text{ on } \Gamma_2 \Big\}. \end{aligned} \tag{3.19}$$

Trace spaces for the electromagnetic problem will similarly be defined:

$$\begin{aligned} H^{-1/2}_{\mathbf{E}_0,\Gamma_1}(\mathrm{curl},\Gamma_h) &:= \Big\{\hat{\mathbf{E}} \in H^{-1/2}(\mathrm{curl},\Gamma_h) : \mathbf{n}\times\hat{\mathbf{E}} = \mathbf{n}\times\mathbf{E}_0 \text{ on } \Gamma_1\Big\}, \\ H^{-1/2}_{\mathbf{H}_0,\Gamma_1}(\mathrm{curl},\Gamma_h) &:= \Big\{\hat{\mathbf{H}} \in H^{-1/2}(\mathrm{curl},\Gamma_h) : \mathbf{n}\times\hat{\mathbf{H}} = \mathbf{n}\times\mathbf{H}_0 \text{ on } \Gamma_2\Big\}, \end{aligned} \tag{3.20}$$

and finally for the elastic problem:

$$\begin{aligned} \boldsymbol{H}^{1/2}_{\mathbf{v}_0,\Gamma_1}(\Gamma_h) &:= \Big\{\hat{\mathbf{v}} \in \boldsymbol{H}^{1/2}(\Gamma_h) : \hat{\mathbf{v}} = \mathbf{v}_0 \text{ on } \Gamma_1\Big\}, \\ \boldsymbol{H}^{-1/2}_{\mathbf{t}_0,\Gamma_2}(\Gamma_h) &:= \Big\{\hat{\boldsymbol{\sigma}}_n \in \boldsymbol{H}^{-1/2}(\Gamma_h) : \hat{\boldsymbol{\sigma}}_n = \mathbf{t}_0 \text{ on } \Gamma_2\Big\}. \end{aligned} \tag{3.21}$$

Broken ultraweak variational formulations all follow from unbroken forms by introducing the appropriate trace terms. Details can be found in various published works [33, 72] and are thus provided without additional commentary.

Acoustics.

$$\begin{cases} p \in L^2(\Omega_h),\ \mathbf{u} \in \boldsymbol{L}^2(\Omega_h),\ \hat{p} \in H^{1/2}_{p_0,\Gamma_1}(\Gamma_h),\ \hat{u}_n \in H^{-1/2}_{u_0,\Gamma_2}(\Gamma_h) & \\ i\omega(c^{-2}p, q) - (\mathbf{u}, \nabla q) + \langle \hat{u}_n, q\rangle_{\Gamma_h} = (\mathrm{f}_p, q) & q \in H^1(\Omega_h) \\ i\omega(\mathbf{u}, \mathbf{v}) - (p, \operatorname{div}\mathbf{v}) + \langle \hat{p}, v_n\rangle_{\Gamma_h} = (\mathbf{f}_\mathbf{u}, \mathbf{v}) & \mathbf{v} \in \boldsymbol{H}(\operatorname{div},\Omega_h) \end{cases} \tag{3.22}$$

Electromagnetics.

$$\begin{cases} \mathbf{E}, \mathbf{H} \in \boldsymbol{L}^2(\Omega_h),\ \hat{\mathbf{E}} \in H^{-1/2}_{\mathbf{E}_0,\Gamma_1}(\mathrm{curl},\Gamma_h),\ \hat{\mathbf{H}} \in H^{-1/2}_{\mathbf{H}_0,\Gamma_2}(\mathrm{curl},\Gamma_h) & \\ -i\omega(\boldsymbol{\varepsilon}\mathbf{E}, \mathbf{F}) + (\mathbf{H}, \nabla\times\mathbf{F}) - \langle \hat{\mathbf{H}}, \mathbf{n}\times\mathbf{F}\rangle_{\Gamma_h} = \mathbf{0} & \mathbf{F} \in H(\mathrm{curl},\Omega_h) \\ i\omega(\boldsymbol{\mu}\mathbf{H}, \mathbf{G}) + (\mathbf{E}, \nabla\times\mathbf{G}) - \langle \hat{\mathbf{E}}, \mathbf{n}\times\mathbf{G}\rangle_{\Gamma_h} = (\mathbf{J}^{\mathrm{imp}}, \mathbf{G}) & \mathbf{G} \in H(\mathrm{curl},\Omega_h) \end{cases} \tag{3.23}$$

Elasticity I.

$$\begin{cases} \mathbf{v} \in \boldsymbol{L}^2(\Omega_h),\ \boldsymbol{\sigma} \in L^2(\Omega_h,\mathbb{S}),\ \hat{\mathbf{v}} \in \boldsymbol{H}^{1/2}_{\mathbf{v}_0,\Gamma_1}(\Gamma_h),\ \hat{\boldsymbol{\sigma}}_n \in \boldsymbol{H}^{-1/2}_{\mathbf{t}_0,\Gamma_2}(\Gamma_h) & \\ i\omega(\rho\mathbf{v}, \mathbf{w}) + (\boldsymbol{\sigma}, \nabla\mathbf{w}) - \langle \hat{\boldsymbol{\sigma}}_n, \mathbf{w}\rangle_{\Gamma_h} = -(\mathbf{f}, \mathbf{v}) & \mathbf{w} \in \boldsymbol{H}^1(\Omega_h) \\ i\omega(\mathsf{S}:\boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{v}, \operatorname{div}\boldsymbol{\tau}) - \langle \hat{\mathbf{v}}, \boldsymbol{\tau}\mathbf{n}\rangle_{\Gamma_h} = \mathbf{0} & \boldsymbol{\tau} \in H(\operatorname{div},\Omega_h;\mathbb{S}) \end{cases} \tag{3.24}$$

Elasticity II.

$$\begin{cases} \mathbf{v} \in \boldsymbol{L}^2(\Omega_h),\ \boldsymbol{\sigma} \in L^2(\Omega_h,\mathbb{S}),\ \mathbf{r} \in L^2(\Omega,\mathbb{A}),\ \hat{\mathbf{v}} \in \boldsymbol{H}^{1/2}_{\mathbf{v}_0,\Gamma_1}(\Gamma_h),\ \hat{\boldsymbol{\sigma}}_n \in \boldsymbol{H}^{-1/2}_{\mathbf{t}_0,\Gamma_2}(\Gamma_h) & \\ i\omega(\rho\mathbf{v}, \mathbf{w}) + (\boldsymbol{\sigma}, \nabla\mathbf{w}) - \langle \hat{\boldsymbol{\sigma}}_n, \mathbf{w}\rangle_{\Gamma_h} = -(\mathbf{f}, \mathbf{v}) & \mathbf{w} \in \boldsymbol{H}^1(\Omega_h) \\ i\omega(\mathsf{S}:\boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{v}, \operatorname{div}\boldsymbol{\tau}) + i\omega(\mathbf{r}, \boldsymbol{\tau}) - \langle \hat{\mathbf{v}}, \boldsymbol{\tau}\mathbf{n}\rangle_{\Gamma_h} = \mathbf{0} & \boldsymbol{\tau} \in \boldsymbol{H}(\operatorname{div},\Omega_h) \end{cases} \tag{3.25}$$

3.3.4 Ultraweak DPG Method for Time-Harmonic Wave Propagation

The broken ultraweak variational problems given in (3.22)–(3.25) define sesquilinear functionals $b(\cdot,\cdot)$ and $\hat{b}(\cdot,\cdot)$ as well as function spaces U_h, $\hat{U}_h$, and V_h. All that remains to define the DPG systems of the form (3.14) is to specify an appropriate test inner product $(\cdot,\cdot)_V$. Definition of the test norm is not unique; specifying a different test norm will lead to a different method and convergence in a different norm. The choice of test norm can significantly impact the accuracy of DPG discretizations. Perhaps the most intuitive choice is the L^2 inner product; this leads to the FOSLS method [78] which is known to be highly dissipative for wave operators.

In the case of the (unbroken) ultraweak formulation, the adjoint norm $||\cdot||_{A^*} := (A^*\cdot, A^*\cdot)$ can be shown to be optimal [120]. However, the adjoint norm is not *localizable*; indeed, $||\cdot||_{A^*}$ is not definite over the discontinuous test space. The premise of DPG was to impart the Gram matrix G with a block diagonal structure that could be condensed locally. To this end, the adjoint norm is supplemented with a weighted L^2 contribution; the resulting *weighted adjoint graph norm*

$$\|\cdot\|_V^2 := \alpha\|\cdot\|^2 + \|\cdot\|_{A^*}^2 \tag{3.26}$$

is quasi-optimal for ultraweak DPG discretizations. The effect of the weighting parameter α will be demonstrated in Section 4.3. The broken ultraweak formulations and explicit form of the adjoint weighted graph norm are given in the notation of the abstract DPG system (3.14) in the following.

Acoustics.

$$\begin{aligned}
\mathfrak{u} &= (p,\mathbf{u}) \in L^2(\Omega_h)\times \boldsymbol{L}^2(\Omega_h)\\
\hat{\mathfrak{u}} &= (\hat{p},\hat{u}_n) \in H^{1/2}_{p_0,\Gamma_1}(\Gamma_h)\times H^{-1/2}_{u_0,\Gamma_2}(\Gamma_h)\\
\mathfrak{v} &= (q,\mathbf{v}) \in H^1(\Omega_h)\times H(\mathrm{div},\Omega_h)\\
\|\mathfrak{v}\|_V^2 &= \alpha\|q\|^2 + \alpha\|\mathbf{v}\|^2 + \|i\omega\bar{c}^{-2}q + \mathrm{div}_h\,\mathbf{v}\|^2 + \|i\omega\,\bar{\mathbf{I}}\,\mathbf{v} + \nabla_h q\|^2\\
b(\mathfrak{u},\mathfrak{v}) &= i\omega(c^{-2}p,q) - (\mathbf{u},\nabla_h q) + i\omega(\mathbf{I}\,\mathbf{u},\mathbf{v}) - (p,\mathrm{div}_h\,\mathbf{v})\\
\hat{b}(\hat{\mathfrak{u}},\mathfrak{v}) &= \langle \hat{u}_n, q\rangle_{\Gamma_h} + \langle \hat{p}, v_n\rangle_{\Gamma_h}\\
l(\mathfrak{v}) &= (\mathrm{f}_p, q) + (\mathbf{f}_\mathbf{u},\mathbf{v})
\end{aligned} \tag{3.27}$$

Recall that material properties ρ and $\mathbf{I}$ will be complex-valued in the PML (Section 2.3), and note the presence of complex conjugates on both in the definition of the test norm.

Electromagnetics.

$$
\begin{aligned}
\mathfrak{u} &= (\mathbf{E},\mathbf{H}) \in \big(\boldsymbol{L}^2(\Omega_h)\big)^2 \\
\hat{\mathfrak{u}} &= (\hat{\mathbf{E}},\hat{\mathbf{H}}) \in H^{-1/2}_{\mathbf{E}_0,\Gamma_1}(\mathrm{curl},\Gamma_h)\times H^{-1/2}_{\mathbf{H}_0,\Gamma_2}(\mathrm{curl},\Gamma_h) \\
\mathfrak{v} &= (\mathbf{F},\mathbf{G}) \in \big(H(\mathrm{curl},\Omega_h)\big)^2 \\
\|\mathfrak{v}\|_V^2 &= \alpha\|\mathbf{F}\|^2+\alpha\|\mathbf{G}\|^2+\|i\omega\boldsymbol{\varepsilon}^*\mathbf{F}+\nabla_h\times\mathbf{G}\|^2+\|i\omega\boldsymbol{\mu}^*\mathbf{G}-\nabla_h\times\mathbf{F}\|^2 \qquad (3.28)\\
b(\mathfrak{u},\mathfrak{v}) &= -i\omega(\boldsymbol{\varepsilon}\mathbf{E},\mathbf{F})+(\mathbf{H},\nabla_h\times\mathbf{F})+i\omega(\boldsymbol{\mu}\mathbf{H},\mathbf{G})+(\mathbf{E},\nabla_h\times\mathbf{G}) \\
\hat{b}(\hat{\mathfrak{u}},\mathfrak{v}) &= -\langle\hat{\mathbf{H}},\mathbf{n}\times\mathbf{F}\rangle_{\Gamma_h}-\langle\hat{\mathbf{E}},\mathbf{n}\times\mathbf{G}\rangle_{\Gamma_h} \\
l(\mathfrak{v}) &= (\mathbf{J}^{\mathrm{imp}},\mathbf{G})
\end{aligned}
$$

Elasticity I.

$$
\begin{aligned}
\mathfrak{u} &= (\mathbf{v},\boldsymbol{\sigma}) \in \boldsymbol{L}^2(\Omega_h)\times L^2(\Omega_h,\mathbb{S}) \\
\hat{\mathfrak{u}} &= (\hat{\mathbf{v}},\hat{\boldsymbol{\sigma}}_n) \in \boldsymbol{H}^{1/2}_{\mathbf{v}_0,\Gamma_1}(\Gamma_h)\times\boldsymbol{H}^{-1/2}_{\mathbf{t}_0,\Gamma_2}(\Gamma_h) \\
\mathfrak{v} &= (\mathbf{w},\boldsymbol{\tau}) \in \boldsymbol{H}^1(\Omega_h)\times H(\mathrm{div},\Omega_h;\mathbb{S}) \\
\|\mathfrak{v}\|_V^2 &= \alpha\|\mathbf{w}\|^2+\alpha\|\boldsymbol{\tau}\|^2+\|i\omega\bar{\rho}\mathbf{w}-\mathrm{div}_h\,\boldsymbol{\tau}\|^2+\|i\omega\bar{\mathsf{S}}:\boldsymbol{\tau}-\varepsilon_h(\mathbf{w})\|^2 \qquad (3.29)\\
b(\mathfrak{u},\mathfrak{v}) &= i\omega(\rho\mathbf{v},\mathbf{w})+(\boldsymbol{\sigma},\nabla_h\mathbf{w})+i\omega(\mathsf{S}:\boldsymbol{\sigma},\boldsymbol{\tau})+(\mathbf{v},\mathrm{div}_h\,\boldsymbol{\tau}) \\
\hat{b}(\hat{\mathfrak{u}},\mathfrak{v}) &= -\langle\hat{\boldsymbol{\sigma}}_n,\mathbf{w}\rangle_{\Gamma_h}-\langle\hat{\mathbf{v}},\boldsymbol{\tau}\mathbf{n}\rangle_{\Gamma_h} \\
l(\mathfrak{v}) &= (-\mathbf{f},\mathbf{v})
\end{aligned}
$$

Elasticity II.

$$
\begin{aligned}
\mathfrak{u} &= (\mathbf{v},\boldsymbol{\sigma},\mathbf{r}) \in \boldsymbol{L}^2(\Omega_h)\times L^2(\Omega_h,\mathbb{S})\times L^2(\Omega_h,\mathbb{A}) \\
\hat{\mathfrak{u}} &= (\hat{\mathbf{v}},\hat{\boldsymbol{\sigma}}_n) \in \boldsymbol{H}^{1/2}_{\mathbf{v}_0,\Gamma_1}(\Gamma_h)\times\boldsymbol{H}^{-1/2}_{\mathbf{t}_0,\Gamma_2}(\Gamma_h) \\
\mathfrak{v} &= (\mathbf{w},\boldsymbol{\tau}) \in \boldsymbol{H}^1(\Omega_h)\times\boldsymbol{H}(\mathrm{div},\Omega_h) \\
\|\mathfrak{v}\|_V^2 &= \alpha\|\mathbf{w}\|^2+\alpha\|\boldsymbol{\tau}\|^2+\|i\omega\bar{\rho}\mathbf{w}-\mathrm{div}_h\,\boldsymbol{\tau}\|^2+\|i\omega\bar{\mathsf{S}}:\boldsymbol{\tau}-\varepsilon_h(\mathbf{w})\|^2+\|\frac{\omega}{2}(\boldsymbol{\tau}^{\mathsf{T}}-\boldsymbol{\tau})\|^2 \qquad (3.30)\\
b(\mathfrak{u},\mathfrak{v}) &= i\omega(\rho\mathbf{v},\mathbf{w})+(\boldsymbol{\sigma},\nabla\mathbf{w})+i\omega(\mathsf{S}:\boldsymbol{\sigma},\boldsymbol{\tau})+(\mathbf{v},\mathrm{div}\,\boldsymbol{\tau})+i\omega(\mathbf{r},\boldsymbol{\tau}) \\
\hat{b}(\hat{\mathfrak{u}},\mathfrak{v}) &= -\langle\hat{\boldsymbol{\sigma}}_n,\mathbf{w}\rangle_{\Gamma_h}-\langle\hat{\mathbf{v}},\boldsymbol{\tau}\mathbf{n}\rangle_{\Gamma_h} \\
l(\mathfrak{v}) &= (-\mathbf{f},\mathbf{v})
\end{aligned}
$$

3.4 Discretization

3.4.1 Discrete Exact Sequence Energy Subspaces

Let $X_0,\ X_1, ...,\ X_{N_s}$ (with $N_s < \infty$) denote a family of vector spaces, and for each $i = 1, ..., N_s$, let $A_i : X_{i-1} \longrightarrow X_i$ be a linear operator. The sequence or *complex* formed by these operators

$$X_0 \xrightarrow{A_1} X_1 \xrightarrow{A_2} \cdots \xrightarrow{A_{N_s}} X_{N_s}$$

is said to be *exact* if, for $i = 1, 2, ..., N_s - 1$, it holds that $\mathsf{R}(A_i) = \mathsf{N}(A_{i+1})$ where R and N denote the operator range and nullspace. The three-dimensional energy spaces first introduced in Section 2.3 form an exact sequence. Similar exact sequences can be formed in one- and two- dimensions:

$$\begin{aligned}
&\textbf{1D:} && H^1(\Omega) \xrightarrow{\partial} L^2(\Omega) \\
&\textbf{2D:} && H^1(\Omega) \xrightarrow{\nabla} H(\mathrm{curl}, \Omega) \xrightarrow{\nabla_{vts}} L^2(\Omega) \\
& && H^1(\Omega) \xrightarrow{\nabla_{stv}} H(\mathrm{div}, \Omega) \xrightarrow{\mathrm{div}} L^2(\Omega) \\
&\textbf{3D:} && H^1(\Omega) \xrightarrow{\nabla} H(\mathrm{curl}, \Omega) \xrightarrow{\nabla\times} H(\mathrm{div}, \Omega) \xrightarrow{\mathrm{div}} L^2(\Omega).
\end{aligned} \tag{3.31}$$

where the ∇_{vts} and ∇_{stv} denote the 2D vector-to-scalar and scalar-to-vector curl operators respectively, defined by $\nabla_{vts} = \partial_1(\cdot)_2 - \partial_2(\cdot)_1$ and $\nabla_{stv} = (\partial_2, -\partial_1)$, and Ω denotes a bounded, simply connected, Lipschitz domain in $\mathbb{R}^N$ with $N = 1, 2, 3$, according to context.

It is important that discretizations of exact sequence energy spaces similarly form an exact sequence. For example, in discretization of the time-harmonic electromagnetic wave propagation problem (2.12) it was noted that, at the continuous level, Gauss' laws were implied by satisfaction of Faraday's law and Ampere's law. In particular, this requires that $\mathsf{R}(\nabla\times) \subset \mathsf{N}(\mathrm{div})$. Satisfaction of Gauss' laws on the discrete level thus requires that discrete finite element spaces inherit the exact sequence structure.

Perhaps the simplest polynomial exact-sequence subspaces are defined on hexahedral elements. Consider the space of tensor product polynomials

$$\mathcal{Q}^{p,q,r}(x, y, z) := \mathcal{P}^p(x) \otimes \mathcal{P}^q(y) \otimes \mathcal{P}^r(z),$$

where $\mathcal{P}^p(x) := \mathrm{span}\{x^j : j = 0, \ldots, p\}$. Polynomial exact-sequence subspaces on a hexahe-

dral element are then given by:

$$
\begin{aligned}
W^p &:= \mathcal{Q}^{p,q,r} \\
Q^p &:= \mathcal{Q}^{p-1,q,r} \times \mathcal{Q}^{p,q-1,r} \times \mathcal{Q}^{p,q,r-1} \\
V^p &:= \mathcal{Q}^{p,q-1,r-1} \times \mathcal{Q}^{p-1,q,r-1} \times \mathcal{Q}^{p-1,q-1,r} \\
Y^p &:= \mathcal{Q}^{p-1,q-1,r-1}
\end{aligned}
$$

Note that throughout this work we will use discretization order to indicate the order of the corresponding exact sequence. This implies that, for an order p discretization, variables set in H^1 will be discretized with order p polynomials, while L^2 variables are discretized with order $p-1$ polynomials, and $H(\text{curl})$ and $H(\text{div})$ variables have mixed order. This dissertation leverages the open-source orientation embedded high-order shape functions (defined for all element shapes) developed by Keith, Fuentes, Demkowicz, and Nagaraj [42]. Exact sequence conforming shape functions are also implemented in a number of modern finite element packages including FEniCS (Basix) [108, 109], NGSolve [107], and MFEM [5].

3.4.2 Discrete Linear System

The Discontinuous Petrov Galerkin methodology leads to a global linear problem with saddle-point structure:

$$
\begin{bmatrix} \mathsf{G} & \mathsf{B} & \hat{\mathsf{B}} \\ \mathsf{B}^* & 0 & 0 \\ \hat{\mathsf{B}}^* & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathsf{s} \\ \mathsf{u} \\ \hat{\mathsf{u}} \end{bmatrix} = \begin{bmatrix} \mathsf{f} \\ 0 \\ 0 \end{bmatrix} \tag{3.32}
$$

where G is element-wise block-diagonal. The DPG system in *mixed* form (3.32) has condition number $\mathcal{O}(h^{-1})$ [73], and can be useful for solving poorly conditioned problems. Solving the system in mixed form may also permit use of matrix-free solvers. However, a more economical approach is often to leverage the element-wise block-diagonal structure of G to condense s at the element level, forming the *normal* system:

$$
\begin{bmatrix} \mathsf{B}^*\mathsf{G}^{-1}\mathsf{B} & \mathsf{B}^*\mathsf{G}^{-1}\hat{\mathsf{B}} \\ \hat{\mathsf{B}}^*\mathsf{G}^{-1}\mathsf{B} & \hat{\mathsf{B}}^*\mathsf{G}^{-1}\hat{\mathsf{B}} \end{bmatrix} \begin{bmatrix} \mathsf{u} \\ \hat{\mathsf{u}} \end{bmatrix} = \begin{bmatrix} \mathsf{B}^*\mathsf{G}^{-1}\mathsf{f} \\ 0 \end{bmatrix}. \tag{3.33}
$$

In the context of the ultraweak variational formulation, field unknowns u are L^2 fields with no coupling between elements, $\mathsf{B}^*\mathsf{G}^{-1}\mathsf{B}$ is thus again element-wise block diagonal and can be condensed on the element level. Condensing both s and u leaves only trace variables $\hat{\mathsf{u}}$ defined on the mesh skeleton, drastically reducing the number of unknowns in the DPG system without increasing matrix bandwidth. Still, forming and condensing element Gram

matrices G_K and forming element stiffness matrices B_K is expensive; these costs will be outlined in Chapter 4.

3.5 Convergence Studies

With the exception of $H(\text{div}, \Omega_h; \mathbb{S})$, the ultraweak DPG problems in Section 3.3.4 leverage the standard energy spaces and are discretized accordingly. At the heart of energy spaces is continuity: H^1 is composed of continuous scalar-valued functions, $H(\text{curl})$ of vector-valued functions with tangential continuity, $H(\text{div})$ of vector-valued functions with normal continuity, and L^2 of discontinuous scalar-valued functions. However, since DPG leverages *broken* test spaces, it is perhaps natural to ask whether it is strictly necessary to use an $H(\text{div}, \Omega_h; \mathbb{S})$-conforming discretization for the test stress $\boldsymbol{\tau}$ in (3.29). We follow intuition, partially discarding mathematical rigor in the process, and take $\boldsymbol{\tau} \in H^1(K, \mathbb{S})$, i.e. we use six scalar valued $H^1(K)$ variables to represent $\boldsymbol{\tau}$ in the first elastic formulation.

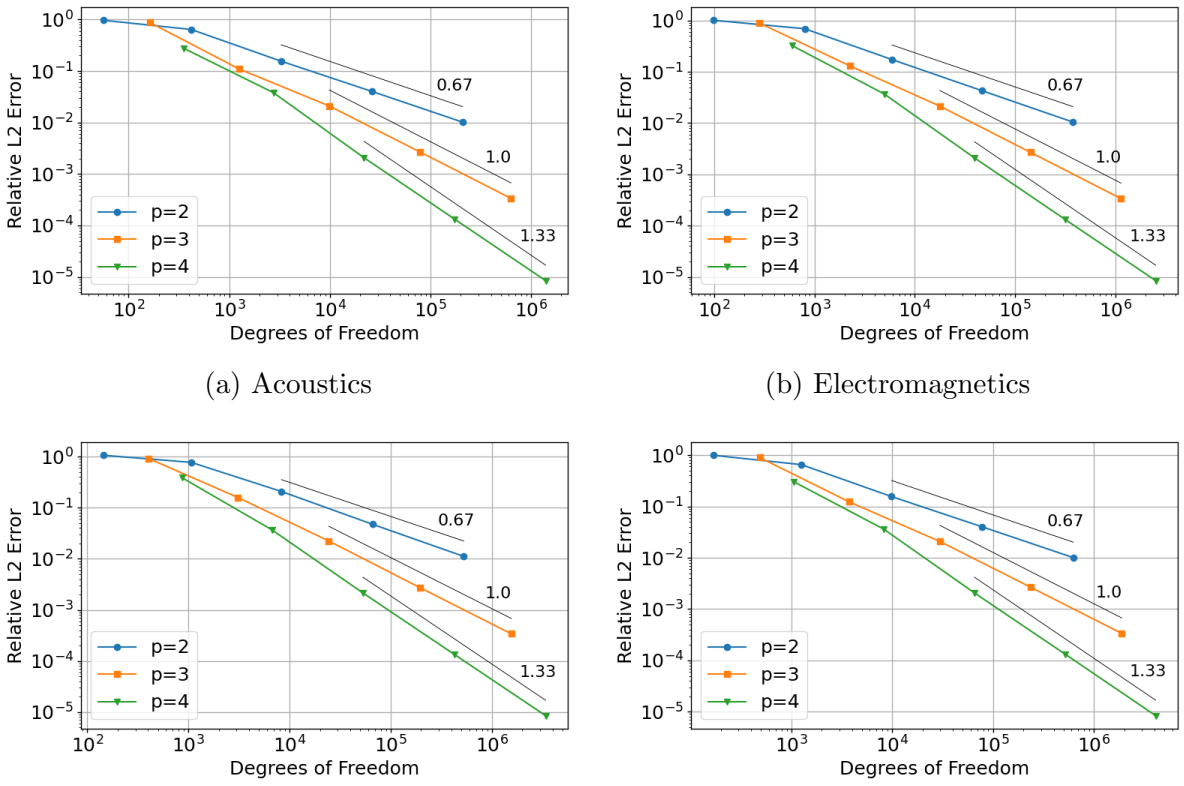
Convergence studies were performed to verify the correctness of the implementations. For each wave problem we considered a unit cube domain, $\Omega = (0, 1)^3$, with manufactured solution

$$\phi(\mathbf{x}) = \sin(\omega x) \sin(\omega y) \sin(\omega z)$$

with $\omega = 2\pi$ for components p, $\mathbf{E}_x$, and $\mathbf{v}_z$ in the acoustic, elactromagnetic, and elastic problem, respectively. Type 1 boundary conditions with manufactured (homogeneous) boundary data were employed in each case. Each study began from a single hexahedral element, four uniform h-refinements were then performed. Test norm weight $\alpha = 1$ was used throughout. All material parameters were taken to be unit valued (with $\boldsymbol{\epsilon} = \boldsymbol{\mu} = I$), with the exception of elasticity where $V_p = 2$ was used (with $V_s = 1$, $\rho = 1$).

Results of the convergence studies are illustrated in Figure 3.1. Errors were computed in the L^2 norm, with respect to all L^2 field variables, except in the case of elastic formulation II where the infinitesimal rotation tensor $\mathbf{r}$ was neglected in error computations. Convergence rates are depicted for each problem and for orders $p \in \{2, 3, 4\}$. The first mesh (corresponding to a single element per wavelength) in each case is too coarse to resolve the manufactured solution, but optimal convergence rates were observed in subsequent refinements. Note that elastic formulation II (Fig. 3.1d) demonstrated marginally higher accuracy than elastic formulation I (Fig. 3.1d) in every case, although the difference was was often insignificant.

The most substantial differences were observed on the second mesh for $p = 2$ and $p = 3$, although even these differences are difficult to observe visually.



(a) Acoustics

(b) Electromagnetics

(c) Elastic Formulation I

(d) Elastic Formulation II

Figure 3.1: Convergence studies for ultraweak DPG discretization of wave operators. Optimal rates are observed.

3.5.1 Comparison of Elastic Formulations with PML

Elastic formulation II was observed to be marginally more accurate than formulation I in the previous convergence study. Further testing revealed challenges of formulation II in the presence of PML boundary conditions. We thus briefly consider a slightly more complex problem.

Consider a simple 0.5 km cube domain ($\Omega = (-0.25, 0.25) \times (-0.25, 0.25) \times (0, 0.5)$), with a free-surface boundary condition at $z = 0$, and surrounded by a 0.25 km-wide PML layer on the remaining sides. The PML is defined via the Cartesian stretching function (2.16) with parameters $\beta = 3, C = 40$. The domain is again taken to be homogeneous with $\rho = 1$ g/cm^3, $V_p = 1$ km/s, and $V_s = 2$ km/s. A similar configuration is illustrated in Fig. 4.2, although piecewise-constant material data was assumed in that case. The methods are compared at

frequency $f = 12$ Hz ($\omega = 24\pi$) and loaded by a point source located at $\mathbf{x}_{\text{src}} = [0, 0, 0]$. The point source was implemented as a simple radial step function:

$$f = \begin{cases} 1 & |\mathbf{x}| \leq 0.0125, \\ 0 & \text{otherwise.} \end{cases}$$

A $32 \times 32 \times 24$ element hexahedral mesh (3 elements per shear wavelength) was used for both formulations. To ensure accurate integration of the load, elements incident to the source were h-refined once.

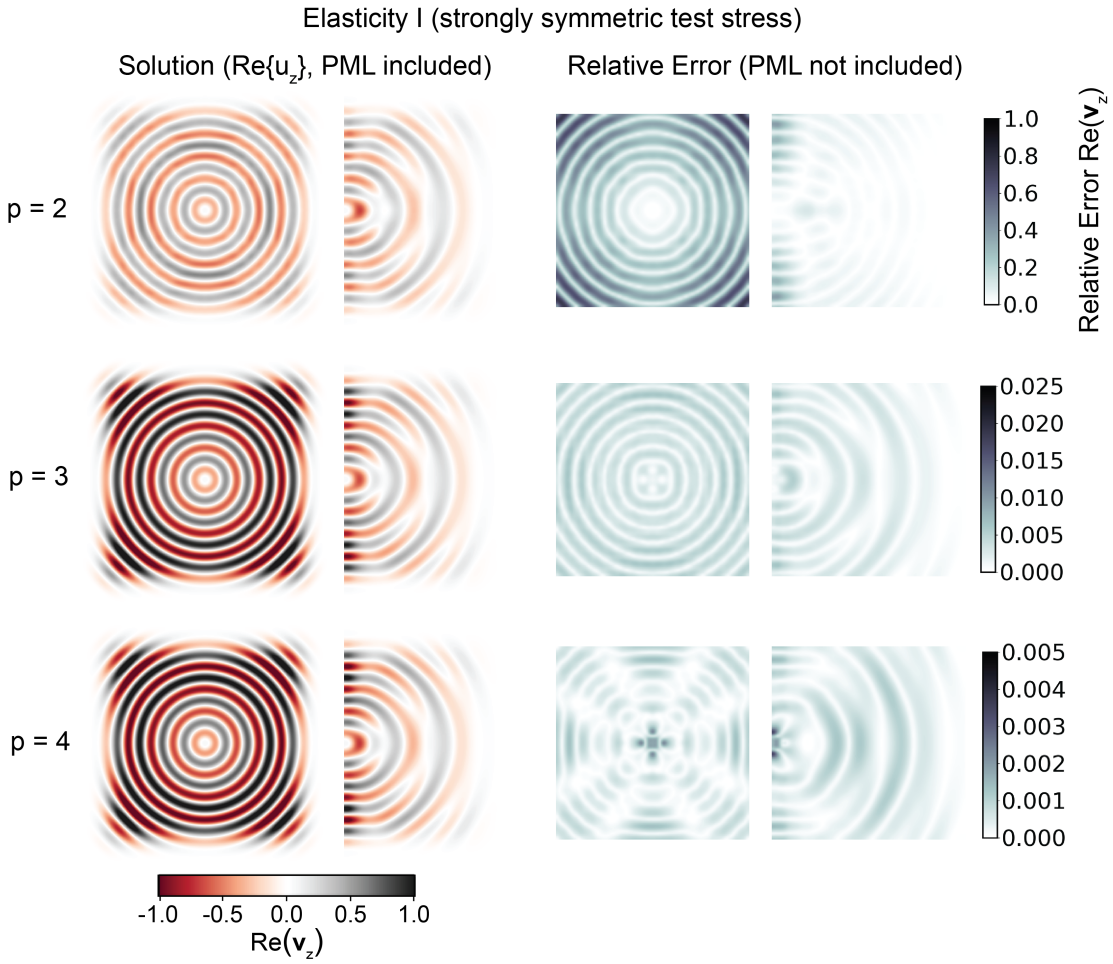


Figure 3.2: Solution and relative errors for simulation of a point source with elastic formulation I (strongly symmetric test stress). Rows correspond to increasing polynomial order (labeled); the first and second column show slices of $\Re\{\mathbf{v}_z\}$ at $z = 0$, $x = 0$, respectively. The third and fourth columns correspond to the relative error in the same slices. All fields are amplitude compensated.

Slices of the solution ($\Re\{\mathbf{v}_z\}$) are shown in Fig. 3.2 and Fig. 3.3. Relative error was computed against a reference solution produced on a fourth-order ($p = 4$) $128 \times 128 \times 96$

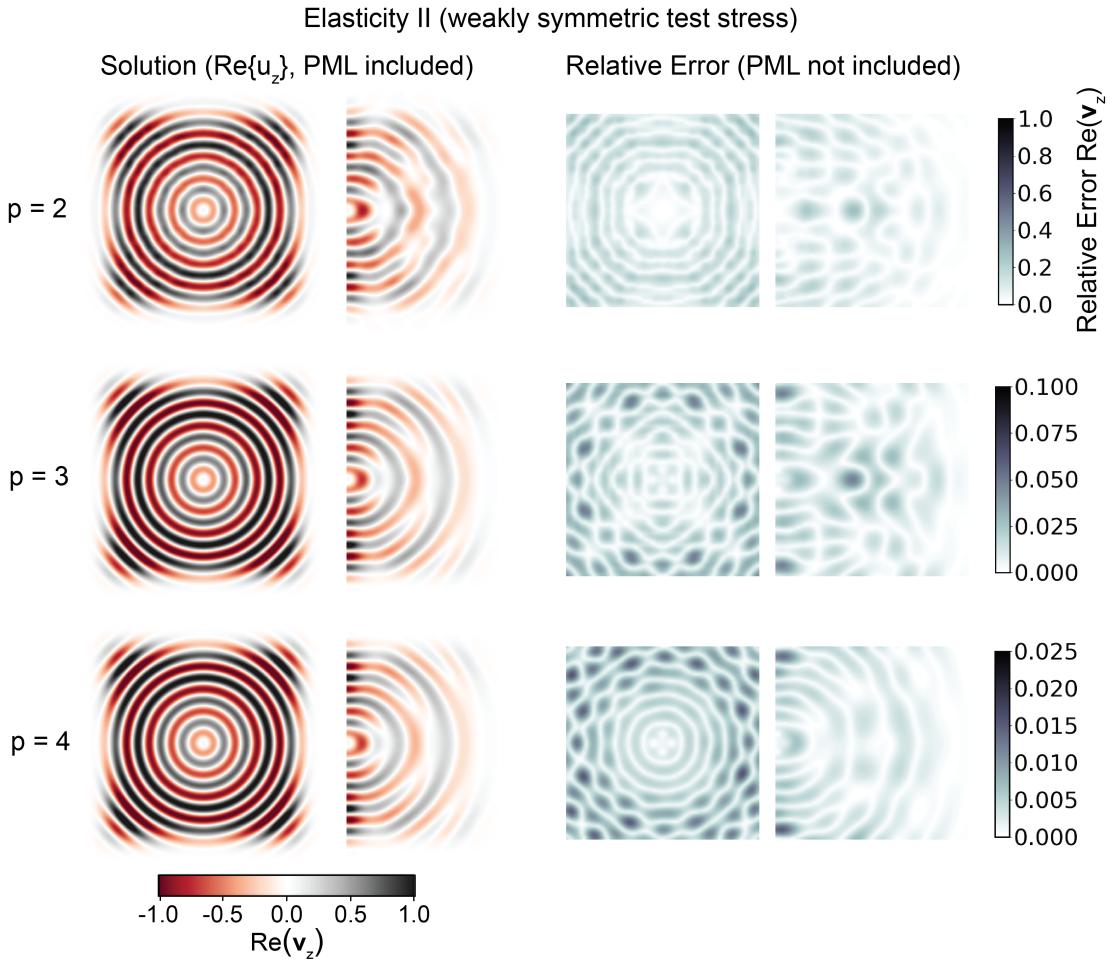


Figure 3.3: Solution and relative errors for simulation of a point source with elastic formulation II (weakly symmetric test stress). Rows correspond to increasing polynomial order (labeled); the first and second column show slices of $\Re\{\mathbf{v}_z\}$ at $z = 0$, $x = 0$, respectively. The third and fourth columns correspond to the relative error in the same slices. All fields are amplitude compensated. Note difference in colorbar scales compared to Fig 3.2.

element hexahedral mesh, corresponding to 12 fourth-order elements (48 points) per shear wavelength, and is also provided in the figures. Both solution and error fields are *amplitude compensated*, i.e. scaled by the distance from the source ($\widetilde{u}(\mathbf{x}) := u(\mathbf{x})|\mathbf{x} - \mathbf{x}_{\text{src}}|$). This compensates for the natural decay of point source solutions in three-dimensions and helps visualize wavefields uniformly in space. The scale in the relative error plots differ between Fig. 3.2 and Fig. 3.3; indeed, elastic formulation I is observed to produce ca. $5\times$ smaller error than elastic formulation II, with $p = 4$ achieving ca. 0.2% error in the L^2_r norm (where L^2_r is the weighted norm implied by amplitude compensation). Finally, it can be observed that although the lowest order fields for both formulations demonstrated significant inaccuracies,

the error is largely *dissipative*. Indeed, pollution in DPG discretizations manifests primarily as amplitude error, often producing very little phase error.

The underperformance of formulation II was somewhat unexpected in this case. To understand the difficulty of the method in the context of PML boundary conditions, recall the second equation in (3.25):

$$i\omega(\mathsf{S} : \boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{v}, \operatorname{div} \boldsymbol{\tau}) + i\omega(\mathbf{r}, \boldsymbol{\tau}) - \langle \hat{\mathbf{v}}, \boldsymbol{\tau}\mathbf{n} \rangle_{\Gamma_h} = \mathbf{0}.$$

We consider only the antisymmetric part of this expression. Assuming S conforms to major and minor symmetries leaves only the infinitesimal rotation term:

$$i\omega(\mathbf{r}, \boldsymbol{\tau}) = \mathbf{0}.$$

The infinitesimal rotation tensor $\mathbf{r}$ can thus be understood as a Lagrange multiplier, enforcing (weakly) symmetry of $\boldsymbol{\tau}$. However, as noted in Section 2.3, PML stretching violates minor symmetries of S. The asymmetry thus couples the reaction from weak enforcement of test stress symmetry to the symmetric terms enforcing the constitutive relation. Indeed, as can be seen in Fig. 3.3 (especially $p = 4$), the error appears to originate from the PML boundary. Further refinements and computing the solution on a larger domain (not depicted) were observed to further localize the error near the PML.

Further investigations into the treatment of the compliance asymmetry induced by the PML are required for formulation II to be competitive. We leave such investigations to future work, noting only that formulation I avoids these challenges and was observed to achieve optimal convergence rates and comparable accuracy to formulation II. We will thus consider formulation I exclusively throughout the remainder of this dissertation.

Chapter 4

Computing with DPG

The DPG method was first proposed by Demkowicz and Gopalakrishnan nearly 15 years ago [28] but, despite the various compelling qualities outlined in the previous chapter, DPG has struggled to gain traction among practitioners. This delayed adoption is perhaps partially due to the significant mathematical and implementational complexity of DPG relative to other numerical methods. A second likely factor is cost: DPG has earned a reputation of being exorbitantly expensive relative to more standard Galerkin finite element approaches, which are in turn often considered expensive relative to finite difference and other methods. The additional cost of DPG is two-fold: 1) forming element normal matrices is typically orders of magnitude more expensive than forming Galerkin element matrices, and 2) DPG introduces additional variables that result in larger global systems with increased coupling and are often significantly more expensive to solve. This chapter summarizes the costs and complexities of computing with DPG and outlines a number of practical considerations.

4.1 Element Matrix Formation

4.1.1 Numerical Integration

Forming DPG element normal matrices requires forming Gram matrix G, stiffness matrix B, and interface stiffness matrix $\hat{\mathsf{B}}$, all using a larger *enriched* test space. To illustrate the size of these matrices, Table 4.1 outlines the number of bubble (N_b), interface (N_i), and enriched test unknowns (N_t) for a hexahedral element with modest order ($p \in \{2, 3, 4\}$) for each wave operator studied here. Standard numerical integration of the matrices (with the exception of $\hat{\mathsf{B}}$) in three-dimensions has $\mathcal{O}(p^9)$ complexity for evaluating and accumulating $\mathcal{O}(p^3)$ trial functions against $\mathcal{O}(p^3)$ test functions at $\mathcal{O}(p^3)$ quadrature points.

Numerical integration was implemented with matrix-matrix multiplication routines that implicitly perform the accumulation over quadrature points [11]. That is, let S denote a matrix where each column corresponds to the values of a particular shape function at each quadrature point, then the numerical integration can be expressed as $\mathsf{S}_1^*\mathsf{D}\mathsf{S}_2$, where D is a diagonal (for scalar-valued unknowns) or block-diagonal (for vector- and tensor-valued unknowns) incorporating the quadrature weight, material data, etc. Despite having the same $\mathcal{O}(p^9)$ complexity, this approach enables use of highly-optimized general matrix-matrix multiplication (GEMM) routines to compute matrix-matrix products and significantly outperforms classical loop-based implementations. Numerical integration via matrix-matrix multiplication is employed by numerous finite element codes, but is notable here as it signifi-

Table 4.1: Number of bubble (N_b), interface (N_i), and enriched test (N_t) degrees of freedom for wave propagation problems with different orders.

Problem	p	N_b	N_i	N_t
Acoustics	2	32	50	127
	3	108	110	365
	4	256	194	666
EM	2	48	96	288
	3	162	216	600
	4	384	384	1080
Elasticity	2	72	150	576
	3	243	330	1125
	4	576	582	1944

cantly reduces numerical integration times previously reported in *hp*3D [92, 8, 58]. Numerical integration times for the DPG system are reported in Table 4.2.

4.1.2 Static condensation

Static condensation of the Gram matrix can be performed in three main stages:

1. Cholesky factorization of the Gram matrix: $\mathsf{G} = \mathsf{U}^*\mathsf{U}$
2. Back substitution for columns of the stiffness, interface stiffness, and load: $\mathsf{W} = \mathsf{U}^{-1}[\mathsf{B}\ \hat{\mathsf{B}}\ \mathsf{f}]$
3. Matrix-matrix multiplication for the weighted normal matrix $\widetilde{\mathsf{B}} = \mathsf{W}^*\mathsf{W}$; the resulting matrix $\widetilde{\mathsf{B}}$ contains the blocks of the element normal system and the modified load in 3.33

Each of these operations has $\mathcal{O}(p^9)$ complexity. Timings are reported in Table 4.2, from which it can be seen that each of the three stages have a similar cost. The total cost of numerical integration and static condensation are also relatively similar, with the exception of elasticity, where the number of trial and test unknowns significantly exceeds the number of quadrature points. Static condensation is significantly more expensive in the elastic case.

Table 4.2: Timings for forming DPG element normal matrices for wave propagation problems with various discretization orders p.

		Numerical Integration (s)				Static Condensation (s)			
	p	B	$\hat{\mathsf{B}}$	G	Total	Chol.	Backsub.	$\mathsf{W}^*\mathsf{W}$	Total
Acoustic	2	3.2e-4	1.2e-4	6.2e-4	1.1e-3	2.3e-4	1.9e-4	9.3e-5	5.2e-4
	3	2.1e-3	6.5e-4	3.7e-3	6.5e-3	2.2e-3	2.5e-3	1.3e-3	6.0e-3
	4	1.2e-2	1.2e-4	2.0e-2	3.2e-2	1.0e-2	1.5e-2	8.8e-3	3.4e-2
EM	2	1.5e-3	1.0e-3	3.2e-3	5.8e-3	1.2e-3	8.8e-4	5.8e-4	2.7e-3
	3	8.1e-3	2.9e-3	1.6e-2	2.7e-2	8.3e-3	9.2e-3	6.2e-3	2.4e-2
	4	3.4e-2	8.2e-3	7.8e-2	1.2e-1	5.2e-2	5.5e-2	4.6e-2	1.5e-1
Elastic	2	1.5e-3	1.3e-3	5.2e-3	8.1e-3	5.9e-3	5.2e-3	2.3e-3	1.3e-2
	3	9.2e-3	3.8e-3	4.2e-2	5.5e-2	3.7e-2	4.4e-2	2.5e-2	1.1e-1
	4	3.3e-2	1.0e-2	1.4e-1	1.8e-1	1.6e-1	2.5e-1	1.7e-1	5.7e-1

4.1.3 Sum-factorization

Sum factorization can be used to reduce the complexity of numerical integration from $\mathcal{O}(p^9)$ to $\mathcal{O}(p^7)$ on elements with tensor product [92] and semi-tensor product [8] structure. Figure 4.1 compares the standard GEMM-based numerical integration to an optimized sum-factorization routine. Results were computed on *Frontera* [111] at the Texas Advanced Computing Center (TACC). A single dual socket (Intel 8280) node with 56 total cores and 192GB of RAM was employed in a 2 MPI rank by 28 thread configuration. Each thread computed a 100-run average cost of forming G, B, and $\hat{\mathsf{B}}$; timings were then averaged over threads. The sum-factorized routine is ca. 2–3× faster even for low-order elements; however the results are somewhat misleading since evaluating 3D shape functions is the dominant cost for low-order elements ($p < 4$). The *observed* complexity of the GEMM-based routine is also less than the expected $\mathcal{O}(p^9)$ complexity; this was likely due to bandwidth limitations in the all-core configuration and the improving intensity of the computation for larger matrices.

Timings for full and partial sum-factorized integration of electromagnetic element matrices were provided in [8] but the fastest sum-factorized routines outlined there are again only ca. 3× faster than the GEMM-based routines. However, implementing, optimizing, and maintaining sum-factorization routines is cumbersome and DPG element matrix formation is still often limited by the cost of static condensation. GEMM-based routines are also often much better suited to GPU computing and other accelerators. Still, sum-factorization can provide substantial speedups for numerical integration and, especially if coupled with fast

static condensation routines, can be an effective tool for reducing the cost of DPG element matrix formation.

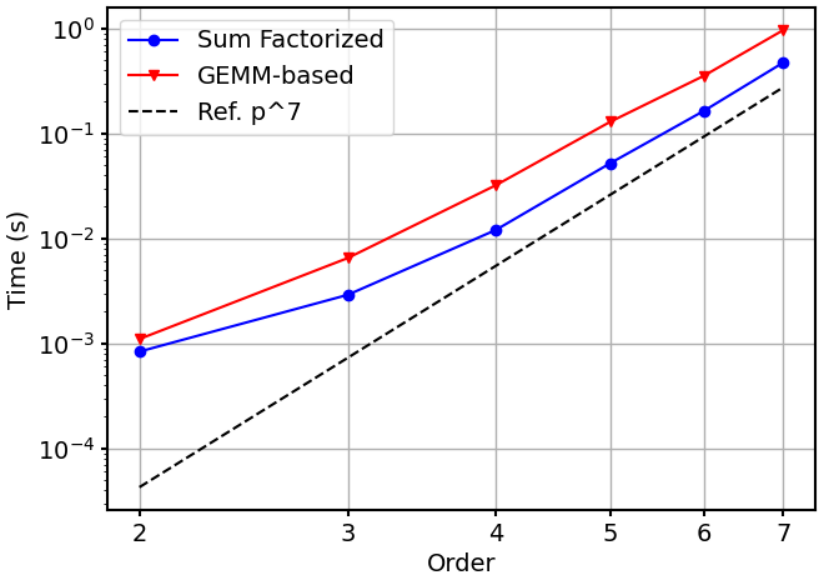


Figure 4.1: Timings for numerically integrating G, B, and $\hat{\mathsf{B}}$ for the acoustics problem with GEMM-based and sum-factorization routines.

4.1.4 Future directions

Numerically integrating and statically condensing relatively large Gram matrices is expensive and, in applications where this cost is limiting and cannot be amortized, one approach may be to remove it entirely. Hybridizable discontinuous Galerkin (HDG) methods with impedance-based fluxes have been proposed [69] and proved [83] to be pre-asymptotically stable. These HDG methods have a similar (not identical) functional setting to the ultraweak DPG methods employed here and seem to be compatible with the DPG-MG preconditioner (the solve may require an alternative Krylov iteration), although we have not yet verified this assertion. Critically, HDG methods avoid formation and static condensation of a Gram matrix. The cited HDG methods do advocate for static condensation of the field variables, but condensation of the field variables in both HDG and DPG is relatively inexpensive due to the reduced size of these matrices compared to the Gram matrix. Indeed, as can be noted in Table 4.1, the number of bubble unknowns (N_b) is typically 3–4$\times$ smaller than the number of enriched test unknowns (N_t) for the polynomial orders shown; this implies a 27–64$\times$ reduced cost for static condensation of the field variables compared to the cost

for statically condensing the Gram matrix. A comparison of the cited HDG methods with DPG methods for wave propagation problems would make a compelling future study. Still, DPG provides substantial benefits including a built-in error indicator and Hermitian positive definite structure; methods for fast condensation of Gram matrices would thus be a welcome direction for further inquiry.

4.2 Additional Unknowns and Linear Solvers

DPG methods are typically derived starting from first-order operators that have additional variables relative to the second-order operators used for standard Galerkin discretizations. For example, linear acoustics supplements scalar-valued pressure p with vector-valued velocity $\mathbf{u}$, electromagnetics uses both electric and magnetic fields ($\mathbf{E}$ and $\mathbf{H}$, respectively), and linear elasticity supplements vector-valued displacement (velocity) $\mathbf{v}$ with the tensor-valued stress $\boldsymbol{\sigma}$. The mixed DPG system (3.10) further introduces additional enriched test variables and trace variables on the mesh skeleton and, for the problems and orders considered here, has ca. $10\times$ as many unknowns than a standard Galerkin system. However, as noted previously, standard DPG implementations condense the enriched test variables and any bubble trial degrees of freedom (DOFs) on the element level, leaving only DOFs supported on the mesh skeleton. Galerkin discretizations may similarly condense bubble DOFs on the element level, leaving only DOFs supported on the mesh skeleton; in this case the DPG system will typically have $2\times$ as many unknowns as the Galerkin system.

This simple enumeration of unknowns confirms that DPG systems do, at best, have roughly twice as many unknowns as a Galerkin system on a given mesh. DPG's additional degrees of freedom are coupled, resulting in $2\times$ larger matrix bandwidth and requiring $4\times$ more memory. Applying a direct sparse solver with $\mathcal{O}(N^2)$ complexity would, in an optimistic estimate, increase the cost of solving the system by 4–$8\times$ (when accounting for the increased coupling); however, in practice the performance of direct sparse solvers significantly degrades with increased coupling, often incurring a cost at least $10\times$ higher than for applying a direct solver to an analogous Galerkin system.

The preceding estimate neglects many of the benefits of the DPG methodology and is intended to demonstrate how, under common workflows, DPG can prove exorbitantly expensive and discourage potential adopters. Benefits of DPG—including mesh-independent stability, a built-in error indicator, and compelling dispersion characteristics—will be illus-

trated throughout this dissertation. For now we simply note that the Hermitian positive-definite structure of DPG systems enables use of advanced preconditioners (e.g. multigrid) and use of the preconditioned conjugate gradient iteration with improved memory and computational complexity compared to the preconditioned GMRES iteration typically employed for Galerkin methods. For example, the largest problems considered in Chapter 6 required ca. 500 iterations; storing the Krylov history in these cases would require substantially more memory than the DPG system while use of restarting, BiCGSTAB, or some other method to limit the Krylov history would be expected to increase the number of iterations and computational expense. Thus, at scale, DPG may become more economical than comparable Galerkin systems. The Hermitian structure can further be leveraged to store only the upper or lower triangular part of the operator, reducing memory consumption to only $2\times$ that of a Galerkin system (while still requiring $4\times$ more operations to apply, however bandwidth limitations of modern computing hardware typically imply a less substantial increase).

Finally, we note that the number of DOFs reported in future studies count all trial variables, including bubble and interface DOFs (neglecting enriched test DOFs). Counting in this way is somewhat biased in favor of DPG discretizations. As can be verified in Table 4.1, ultraweak DPG discretization of the fourth-order ($p = 4$) acoustic problem will have 450 DOFs per element, while a Galerkin discretization will have 125 DOFs per element. In the examples considered in this dissertation, the ultraweak DPG systems have ca. 3–4$\times$ as many DOFs as a comparable Galerkin discretization on the same mesh. However, even when accounting for this advantageous counting, the scale of problems shown in Chapter 6 represents an unprecedented scale for time-harmonic wave propagation problems.

4.3 Test Norm Scaling

As observed in Section 3.5, definition of the test norm can significantly impact the quality of the solution, particularly in the preasympotic regime. This section demonstrates the effect of test norm weight α on solution accuracy for a linear elastic problem with piecewise constant material properties. A similar test-norm scaling study, in the context of an ultraweak acoustic problem, was conducted in [47].

The study is conducted on a 0.5 unit cube domain, partially surrounded by a 0.25 unit PML layer (see Fig. 4.2), with a free-surface boundary condition on the $z = 0$ face. The Cartesian stretching function (2.16) with parameters $\beta = 3$, $C = 40$ was used to define the

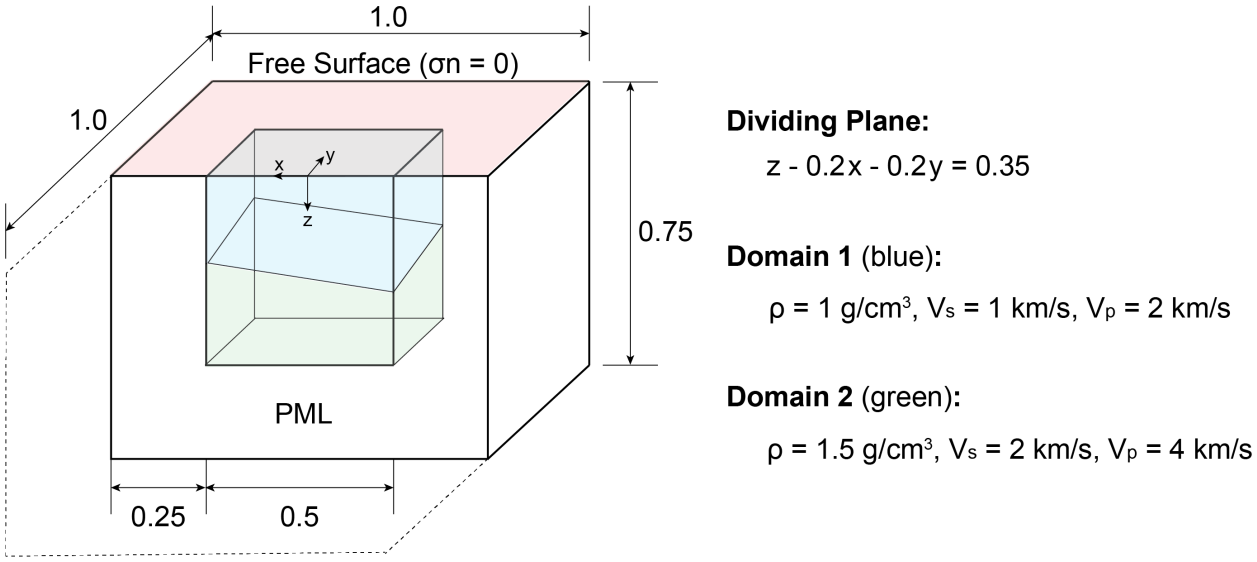


Figure 4.2: Configuration for test norm (α) scaling study. All length measurements shown are in km. The domain has piecewise constant material properties, with separating plane and material data specified. The location of the origin (at the center of the free surface) is noted.

PML. Piecewise constant material data and separating plane are additionally specified in Fig. 4.2. In the PML absorbing layer, material data was simply extruded in the direction of stretching. The problem is driven by a point source located at $\mathbf{x}_{\text{src}} = [0, 0, 0.15]$ with frequency $\omega = 32\pi$ ($f = 16$ Hz).

Solutions were computed on a uniform hexahedral mesh ($h = 1/32$), corresponding to two elements per minimum shear wavelength, for orders $p \in \{2, 3, 4\}$ and $\alpha \in \{10^{-2}, 10^{0}, 10^{2}\}$; the resulting solutions are shown in Fig. 4.3–4.5. The depicted relative error was computed against a reference solution computed on a twice h-refined fourth-order mesh ($h = 1/128$, $p = 4$). Meshes were *not* fitted to the material discontinuity. Although fitting meshes would have been relatively simple for this problem, it will be nearly impossible for the more complex applications considered in Chapter 6. This study thus fulfills a dual purpose of validating accuracy of DPG discretizations when meshes are not aligned to material discontinuities.

Second-order solutions are depicted in Fig. 4.3, where it can be observed that the solution becomes more dissipative as α is increased. It is well-documented [47, 59] that increasing the test norm weight α tends to increase pollution error which, in the case of DPG discretizations, tends to manifest as dissipation. Increased dissipation is thus expected for larger α. It can also be seen from Fig. 4.3 that for small α the solution appears to become less stable,

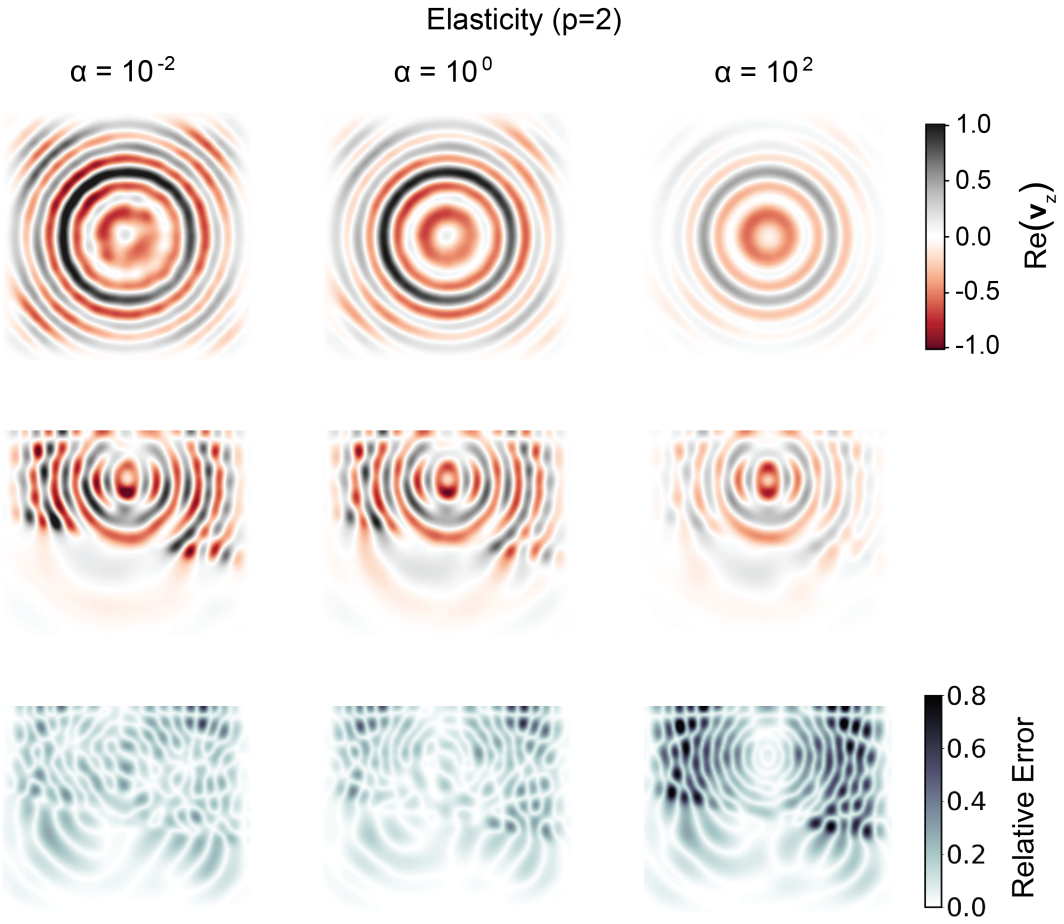


Figure 4.3: Test norm (α) scaling study for $p = 2$. Solution ($\Re\{\mathbf{v}_z\}$) slices at $z = 0$ (top row) and $x = 0$ (middle row) and relative error slices at $x = 0$ (bottom row) are depicted. Fields are amplitude compensated.

exhibiting additional oscillations.

Third- and fourth-order solutions are depicted in Fig. 4.4 and Fig. 4.5. In both cases, additional error is incurred for $\alpha = 10^2$, but the induced changes are relatively less pronounced. Only minor differences are visible in relative errors for $\alpha = 10^{-2}$ and $\alpha = 10^0$ and the overall accuracy ($\mathcal{O}(3\%)$ and $\mathcal{O}(1\%)$ for $p = 3$ and $p = 4$, respectively) is acceptable given the two element per minimum shear wavelength mesh.

Finally, we note that the DPG-MG solver with a stopping criterion of 10^{-8} relative ℓ^2 residual was used to solve this problem. Additional information on the solver and run configurations will be provided in future studies; for now we note only that the solve required 128, 46, and 37 iterations for $\alpha = 10^{-2}, 10^0, 10^2$, respectively, reflecting a general trend that

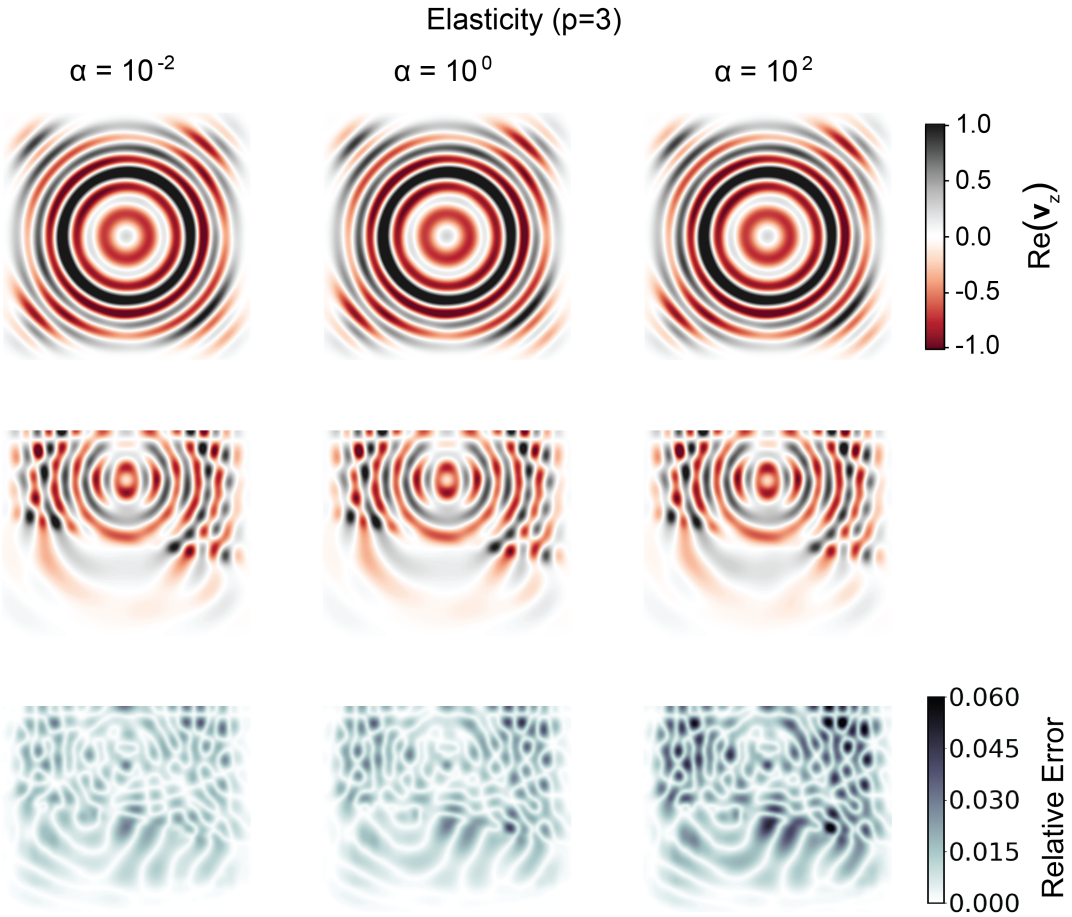


Figure 4.4: Test norm (α) scaling study for order $p = 3$.

decreasing α tends to increase the number of iterations required for convergence. Selecting an appropriate test weight α thus requires balancing competing objectives of accuracy and solution time. As defined here, the choice of α is not robust with respect to frequency ω, element size h, domain size L [59], or the wave operator. Fortunately, the sensitivity of both accuracy and iteration count to α is often relatively mild. Unless otherwise noted, test norm weight $\alpha = 1$ was used throughout this work.

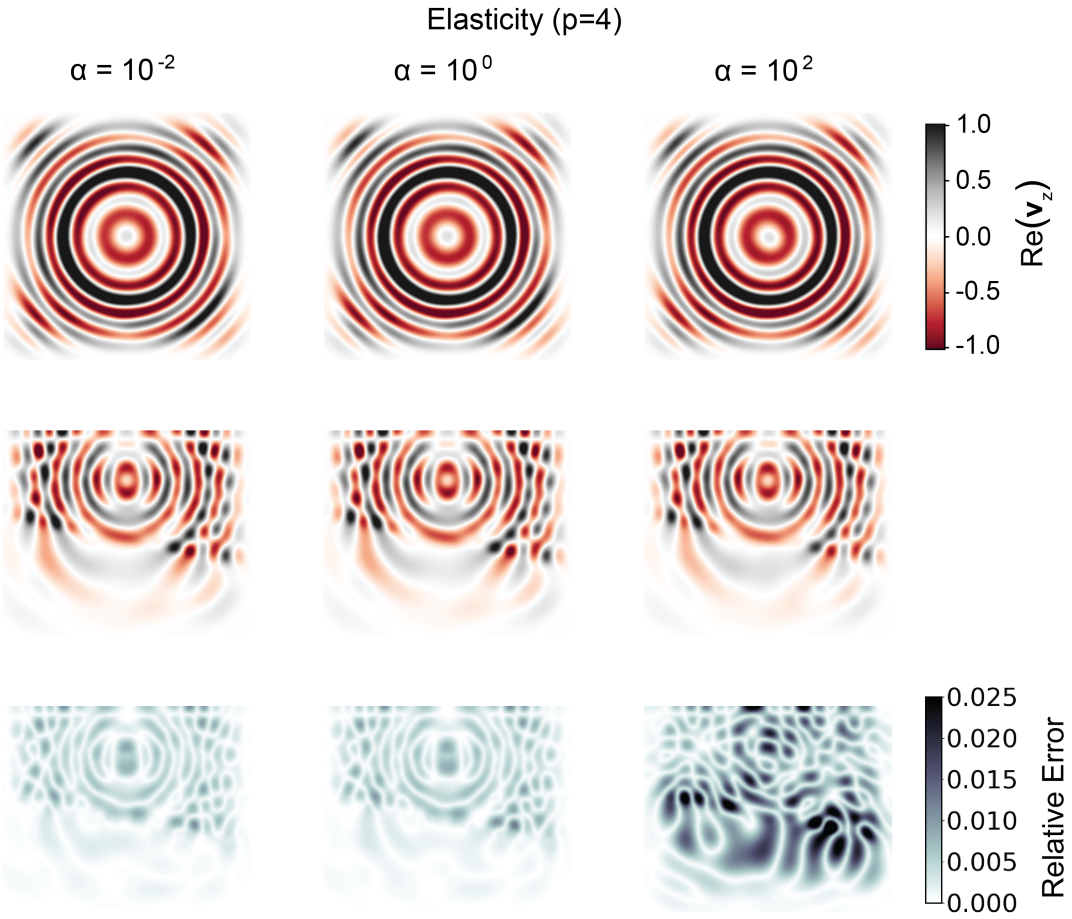


Figure 4.5: Test norm (α) scaling study for order $p = 4$.

Chapter 5

DPG Multigrid Solver

5.1 Overview

Multigrid[1] algorithms are a cornerstone of modern scalable solver technology but their application to problems with indefinite operators, such as high-frequency wave propagation, is challenging and remains an active area of research. The DPG methodology—as a minimum residual method—produces Hermitian positive definite discrete systems and can thus be used to circumvent many of the challenges associated with applying multigrid to indefinite problems. That is not to say DPG is without its own unique challenges in applying multigrid. For example, as outlined in Chapter 3, the DPG methodology introduces additional trace degrees of freedom (DOFs) on the mesh skeleton, resulting from testing with discontinuous ("broken") test spaces [19]. The DPG system is typically condensed onto a system composed entirely of trace unknowns defined on the mesh skeleton. Constructing suitable prolongation operators for these trace unknowns is complicated by the fact that h-refinement alters the mesh skeleton, introducing additional fine-grid unknowns with no natural coarse-grid representatives; this is a challenge shared by hybridizable methods [103, 104]. The construction of a stable prolongation operator between such non-nested condensed systems for general DPG problems[2] was one of the contributions in the original work by Petrides and Demkowicz [100].

The DPG multigrid preconditioner is a non-standard multilevel preconditioner that operates on DPG trace systems and is applicable—without modification—to any DPG formulation, for any well posed problem involving the standard energy spaces. The DPG multigrid solver (DPG-MG) leverages this multilevel preconditioner to precondition a conjugate gradient iteration, defining a fast iterative solver for general DPG systems. The DPG-MG solver can be leveraged in conjunction with mesh-independent stability and built-in error indicator of the DPG method to drive hp-adaptivity. In this context, the solution on intermediate meshes is only needed to sufficient accuracy to produce a subsequent refinement; achieving the needed accuracy typically requires only a relatively small number of iterations, this approach was demonstrated in Petrides' original works [100, 102].

This dissertation extends Petrides' implementation in a number of key ways:

[1] This chapter contains a number of sections adapted from "Scalable DPG multigrid solver for Helmholtz problems: A study on convergence." by J. Badger, S. Henneking, S. Petrides, and L.Demkowicz, *Comput. Math. Appl.*, 148:81–92, 2023. The author of this dissertation contributed to theory, software development, numerical simulations, and writing of that work.

[2] discretized with exact-sequence energy spaces

1. An alternative construction of coarse-grid operators is proposed that significantly improves convergence of the solver.

2. A fully distributed implementation is developed, enabling efficient and scalable solution on modern distributed manycore architectures.

3. The implementation is extended to support anisotropic refinements and hybrid meshes of all element shapes.

4. A performant CPU implementation achieves significant acceleration compared to the original implementation, enabling the solver to be competitive for a much larger class of problems.

5.1.1 Software

The DPG-MG solver was built on top of *hp*3D and grown out of Petrides' original shared-memory solver implementation [100]. An open-source version of the distributed DPG-MG solver was intended to be included in the *hp*3D repository; however, *hp*3D was not intended for massively parallel computation at the scale demonstrated in this work. Indeed, among other limitations, *hp*3D employs a replicated-mesh data structure to enable distributed parallelism, this approach requires each process to store and maintain a copy of the entire mesh and limits scalability of the solver to $\mathcal{O}(10^9)$ DOFs. The scalable DPG-MG solver leveraged throughout this work is based on an optimized version of *hp*3D that can be run in replicated or fully distributed modes and will be demonstrated in Chapter 6 to solve problems with $\mathcal{O}(10^{12})$ DOFs.

As part of the development process, the performant version of *hp*3D has diverged significantly from the open source code and we are exploring publishing a performant version of *hp*3D along with an open source version of the scalable solver in the FrequenSolver Github repository[3]. However, at the time of writing no open source implementation of the scalable DPG-MG solver is available. In the meantime, a simplified version of the solver based on a previous version of *hp*3D will be made available in that repository, or can be obtained by contacting the author[4]. A version of the shared-memory implementation by S. Petrides [100] is available in *hp*3D (although it is not compatible with the current tagged release); the

[3] https://github.com/FrequenSol/FrequenSolver

[4] `jcbadger@utexas.edu`

following presentation will at times point to that implementation but will seek to emphasize commonality in operations, better paralleling the approach of the scalable implementation.

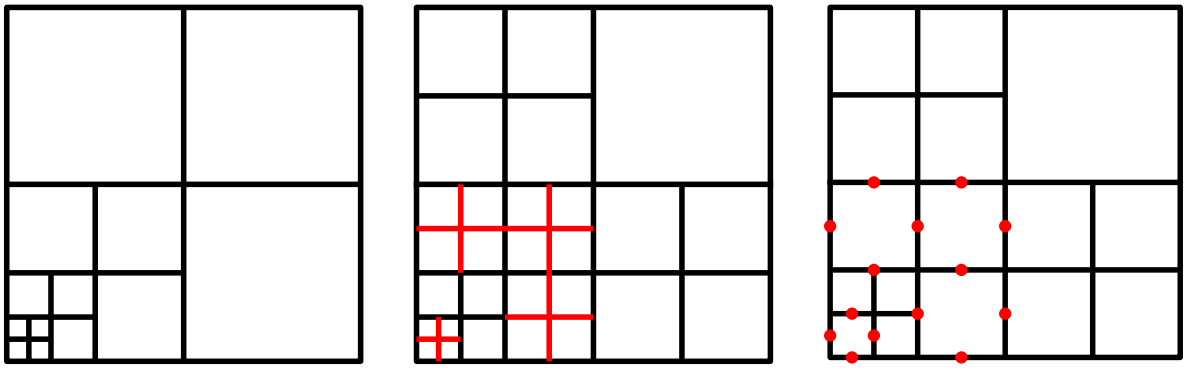
5.2 DPG Multigrid Preconditioner

The DPG multigrid preconditioner was defined in [100]. This section provides a brief overview of the preconditioner for completeness and to establish necessary terminology for the distributed implementation, outlined Section 5.3. We begin by introducing the concept of the macro grid and necessary components for a two-grid V-cycle: smoothing, prolongation, and coarse-grid correction operators. Finally, we review the multigrid V-cycle and preconditioned conjugate gradient iteration. This section largely summarizes Petrides' construction, the most notable exception being the alternative coarse grid construction in Section 5.2.5; the alternative construction will be observed in Section 5.4 to drastically improve convergence of the DPG-MG solver.

5.2.1 Macro Grid

The DPG-MG solver operates on trace unknowns on the mesh skeleton; h-refinements produce fine-grid edges and faces that do not coincide with the previous-grid skeleton and thus have no natural representatives on the previous mesh. To ameliorate this, Petrides and Demkowicz [100] introduced a *macro* grid, where fine-grid unknowns not supported on the coarse mesh skeleton are statically condensed. In the case of p-refinements, macro and fine grids coincide. Figure 5.1 illustrates coarse-, fine-, and macro-grids in the context of h-adapted meshes.

As shown by Petrides and Demkowicz [100], macro- and fine-grid systems are spectrally equivalent. This allowed analysis to be performed on the uncondensed DPG normal system and implies that similar convergence should be expected when applying the preconditioned conjugate gradient iteration on the condensed fine-grid system or macro-grid system. Throughout this work, we chose to apply operators on the macro grid. In the case when the fine grid is defined via a single h-refinement of the coarse grid, the macro grid will have ca. 50% fewer degrees of freedom than the fine grid system, and thus reduces the cost of vector operations such as communicating and computing dot products. Note however that this choice can increase the expense of storing and applying the global system (by ca. $2\times$) due to the increased density of the macro-grid system induced by static condensation. Coarse-grid



(a) Coarse grid (b) Fine grid (c) Macro grid

Figure 5.1: Macro grid construction. The fine grid unknowns that do not lie on the skeleton of the coarse grid are statically condensed onto the remaining unknowns. Dots on the macro grid indicate the location of vertex degrees of freedom. Illustrated in 2D for clarity, 3D construction follows analogously. Reproduced from *Adaptive multilevel solvers for the discontinuous Petrov–Galerkin method with an emphasis on high-frequency wave propagation problems* by S. Petrides., 2019, PhD thesis. Copyright 2019 by S. Petrides. Reproduced with permission.

smoothing patches are ca. $2\times$ smaller, albeit more dense, on the macro grid.

We emphasize that this construction is intended for a *two*-grid V-cycle with fine and coarse grid separated by a single h- or p-refinement. This differs from the original two-grid definition in [101, 100], where the fine and coarse grids were separated by multiple refinements. Performing multiple levels of refinement between grid levels necessitates more extensive static condensation that degrades sparsity and increases the memory consumption and the computational cost of applying the preconditioner and further requires more complicated restriction operators defined via recursion. Multiple levels of refinements can be more efficiently exploited in a multigrid V-cycle, defined in Section 5.2.7.

5.2.2 Preconditioned Conjugate Gradient Iteration

The preconditioned conjugate gradient (PCG) iteration requires the preconditioner to be symmetric (Hermitian) and positive definite. Positive definiteness is guaranteed by the DPG discretization but symmetry requires symmetric smoother and prolongation operators, an equal number of pre- and post-smoothing steps, and a linear coarse grid correction operator. The PCG iteration is standard and will not be detailed here. However, as will be noted in Section 5.5, we will at times define the coarse-grid correction via an inner PCG iteration. This will imply that the preconditioner for the outer PCG iteration will become nonlinear

and require use of the flexible PCG iteration. This will be revisited in Section 5.5.

5.2.3 Additive Schwarz (Overlapping Block Jacobi) Smoother

The DPG-MG solver employs an additive Schwarz (overlapping block-Jacobi) smoother with blocks defined by coarse-grid vertex patches. As noted previously, smoothing can be performed on the fine-grid or macro-grid system, both patch definitions are illustrated in 5.2. In either case, let R_j denote a restriction operator, extracting the unknowns supported on patch j. The additive Schwarz smoother is then defined

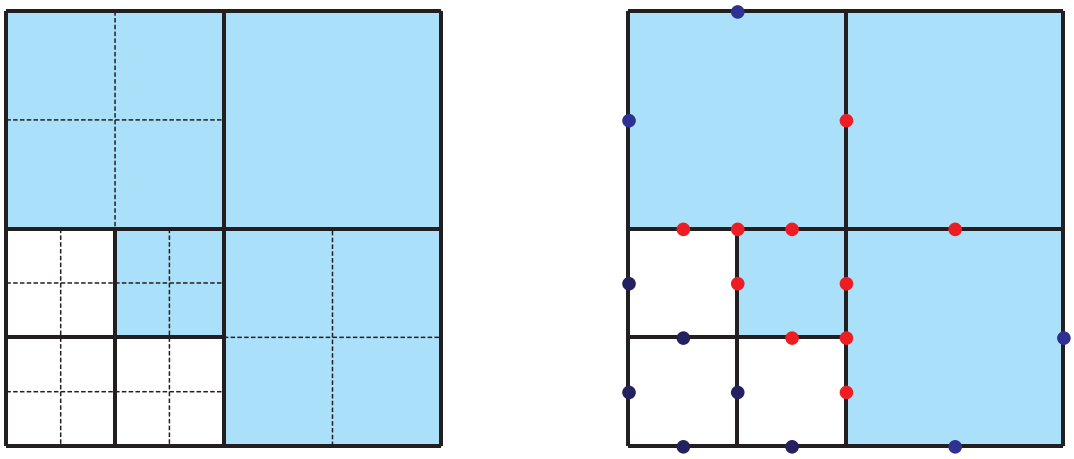
$$\mathsf{S} = \sum_j \theta \mathsf{R}_j^{\mathsf{T}} \mathsf{B}_j^{-1} \mathsf{R}_j, \tag{5.1}$$

where $\mathsf{B}_j := \mathsf{R}_j \mathsf{B} \mathsf{R}_j^{\mathsf{T}}$ is the patch stiffness matrix, and $\theta \in (0, 1)$ is a relaxation parameter ($\theta = 0.2$ was adopted throughout this work). A diagonal weighting matrix is sometimes included in (5.1), such a diagonal weighting often corresponds to a partition of unity that ensures smoother corrections are equally weighted throughout the domain. Inclusion of the diagonal weighting term makes the preconditioner asymmetric and is thus neglected in our case.

Note that the definition of smoothing patches based on the support of coarse-grid vertex patches can lead to large smoothing blocks, especially in the case of high-order discretizations, but avoids additional complexities for vector-valued variables set in $H(\text{curl})$ and $H(\text{div})$ [7]. In particular, this definition of patches implies that the DPG-MG solver is applicable—without modification—to any well-posed DPG problem discretized with exact-sequence energy spaces. Alternative definitions of smoothing patches lead to smaller patch sizes for vector-valued variables [64, 65].

5.2.4 Prolongation

The macro grid was introduced in Section 5.2.1 to facilitate restriction of h-refinements with no coarse-grid representatives. In the two-grid case defined thus far, the macro-grid system is simply defined via static condensation and a single-stage prolongation between the macro-grid and coarse-grid system is required. In the multigrid case, and when the PCG iteration is applied on the fine-grid, we will instead define a two-stage prolongation operator, with the first stage restricting from the fine-grid to the macro-grid system using Schur complements, and the second stage restricting from the macro-grid to coarse-grid



(a) Vertex patch - coarse grid

(b) Vertex patch - macro grid

Figure 5.2: Construction of a smoother patch. A smoother patch is defined by the support of a coarse grid vertex basis function. Adapted from "An adaptive multigrid solver for DPG methods with applications in linear acoustics and electromagnetics." by S. Petrides and L. Demkowicz, *Comput. Math. Appl.*, 87:12–26, 2021.

system. The corresponding restriction operators will be denoted:

- I_m^f – Restriction from fine-grid to macro-grid
- I_c^m – Restriction from macro-grid to coarse-grid

Fine-to-macro prolongation operators. The fine-to-macro restriction is defined via Schur complements. Let B_e denote a macro stiffness matrix corresponding to macro-element e; assembled from fine-grid stiffness matrices B_K with $K \subset e$. Partition the system into unknowns supported on the macro-element interface (denoted with subscript i) and interior (denoted with subscript b); i.e.

$$\begin{bmatrix} \mathsf{B}_{e,ii} & \mathsf{B}_{e,ib} \\ \mathsf{B}_{e,bi} & \mathsf{B}_{e,bb} \end{bmatrix}$$

The local restriction on macro element e is then defined via the Schur complement as:

$$\mathsf{I}_m^f(e) = \begin{bmatrix} I & -\mathsf{B}_{e,ib}\mathsf{B}_{e,bb}^{-1} \end{bmatrix}.$$

where I denotes the $N_i \times N_i$ identity.

The global fine-to-macro restriction is then defined by assembling local element contributions:

$$\mathsf{I}_m^f = \prod_e \mathsf{I}_m^f(e) \prod_{K \in e}, \tag{5.2}$$

where $\prod$ denotes the finite element assembly operator. Note that the definition of the macro-to-fine restriction operator in (5.2) is applied to element-wise unassembled fine-grid arrays. This definition reflects the shared memory implementation and the approach outlined in Section 5.3. The definition could have alternatively been given in global assembled form:

$$\mathsf{I}_m^f = \left[\begin{array}{cc} I & -\mathsf{B}_{ib}\mathsf{B}_{bb}^{-1} \end{array} \right]$$

where indices i denote the global unknowns supported on the macro-grid skeleton.

The macro-to-fine extension operator, denoted I_f^m is simply defined as the Hermitian transpose of the restriction operator; i.e. $\mathsf{I}_f^m := \left(\mathsf{I}_m^f\right)^*$. The definition of I_f^m via the transpose is required for the classical preconditioned conjugate gradient iteration since the preconditioner is required to be symmetric.

Macro-to-coarse prolongation operators. The coarse-to-macro extension operator (I_m^c) is defined simply via the natural inclusion operator, expressing coarse-grid basis functions as a linear combination of macro-grid basis functions. The natural inclusion operator (or rather its transpose) can thus be constructed by solving a series of projection problems, projecting basis functions defined on macro-grid edges and faces onto basis functions defined on the coarse-grid mesh skeleton. Efficient implementation of these projection problems can be achieved via *constrained approximation* [32]. In the case of p-refinements with hierarchical shape functions (as employed here), the new unknowns resulting from p-refinement are simply set to zero.

The macro-to-coarse restriction operator I_m^c is real-valued and defined via the transpose i.e. $\mathsf{I}_m^c := \left(\mathsf{I}_c^m\right)^{\mathsf{T}}$ to preserve symmetry of the preconditioner.

5.2.5 Coarse-Grid Correction

Smoothers, such as the overlapping block Jacobi smoother introduced in Section 5.2.3, are known to effectively reduce high-frequency error. The motivation behind multilevel methods is that low-frequency error can be more effectively corrected on coarse meshes; smoothing iterations are thus combined with a coarse-grid correction operator to improve convergence.

Coarse-grid operators can be constructed either by direct assembly (i.e. by forming the discrete operators on coarser meshes directly), or by restricting the already formed fine-grid operators. In the case of classical Galerkin methods with nested approximation spaces (i.e.

$U_C \subset U_F$, where U_C and U_F denote coarse- and fine-grid approximation spaces, respectively) these approaches coincide. Indeed, let $\mathsf{I} : U_C \to U_F$ denote the natural inclusion operator and $b(\cdot,\cdot)$ denote a bilinear form defining a Galerkin discretizaiton. Abusing notation we have,

$$\mathsf{B}_C = b(U_C, U_C) = b(\mathsf{I}^\mathsf{T} U_F, \mathsf{I}^\mathsf{T} U_F) = \mathsf{I}\,\mathsf{B}_F\,\mathsf{I}^\mathsf{T},$$

where B_C and B_F denote the resulting coarse- and fine-grid stiffness matrices, respectively.

In the context of the DPG methodology, where the mixed or saddle-point structure of the system resembles a Stokes problem, these approaches do not coincide. The direct assembly approach was used in Petrides' original works, storing and re-using operators assembled on coarse grids to precondition fine grids. However, we observed that for high-frequency problems, convergence rapidly deteriorated with increasing frequency. As will be demonstrated in Section 5.4, computing coarse-grid stiffness matrices as restrictions of fine-grid stiffness matrices restores the expected convergence rates. Computing coarse-grid operators as restrictions of fine-grid operators is relatively inexpensive, typically requiring 1%–10% of the cost of forming the fine-grid DPG system. It may be possible to define operator-dependent prolongation operators that enable direct assembly of coarse-grid operators to be effective, this would be an interesting direction for future inquiry.

Coarse grid correction with restricted coarse operators. Let I_c^m again denote the macro-to-coarse restriction operator defined in Section 5.2.4, the restricted coarse-grid system is given by

$$\mathsf{B}_c := \mathsf{I}_c^m\,\mathsf{B}\,\left(\mathsf{I}_c^m\right)^*.$$

The coarse-grid correction operator is then defined:

$$\mathsf{Q} := \left(\mathsf{I}_c^m\right)^*\,\mathsf{B}_c^{-1}\,\mathsf{I}_c^m.$$

5.2.6 Two-Level Preconditioner

A two-level preconditioner may be defined via additive (parallel) or multiplicative (serial) coupling between smoothing operators and coarse-grid corrections. We employ multiplicative coupling, the two-level preconditioner is then given as:

$$\mathsf{z}_n = \mathsf{M}\mathsf{r}_n = \mathsf{z}_n^s + \mathsf{z}_n^{sq} + \mathsf{z}_n^{sqs}$$

where

$$\mathsf{z}_n^s = \mathsf{S}\mathsf{r}_n$$
$$\mathsf{z}_n^{sq} = \mathsf{Q}(I - \mathsf{B}\mathsf{z}_n^s)$$
$$\mathsf{z}_n^{sqs} = \mathsf{S}(I - \mathsf{B}\mathsf{z}_n^{sq})$$

The preconditioner M can be expressed compactly [100] as:

$$\mathsf{M} = \big(\mathsf{Q} + \mathsf{S}(I - \mathsf{A}\mathsf{Q}) + (I - \mathsf{Q}\mathsf{A})\mathsf{S} - \mathsf{S}(I - \mathsf{A}\mathsf{Q})\mathsf{A}\mathsf{S}\big)$$

5.2.7 Multigrid V-cycle

Until this point we have only considered the two-level preconditioner, with a single fine and coarse mesh. The multilevel preconditioner operates on a hierarchy of grids, we will thus need to distinguish between various grids. Multigrid can be expressed as recursive application of the two-level preconditioner, where the coarse grid correction operator B_c^{-1}, is approximated by a further two-level preconditioner. The recursive algorithm is given in Algorithm 1, and motivates our enumeration of meshes. The initial mesh is labeled `coarse(1)`, the i-th subsequent refinement is labeled `coarse(i)`, coinciding with `fine(i-1)`. The macro-grid between `coarse(i)` and `fine(i)` is denoted `macro(i)` and is uniquely ennumerated. This convention is illustrated in Fig. 5.3.

Algorithm 1 Multigrid V(1,1)-cycle

```
procedure V_CYCLE(r,i)                          ▷ r: residual, i: Grid level
    if i == 1 then
        z = coarse_correction(r)                ▷ Coarsest-grid correction
    else
        z = smooth(r,i)
        r = r - B_i z                           ▷ Update residual
        r_c = restrict(r,i)                     ▷ Restrict residual
        z_c = V_CYCLE(r_c,i-1)                  ▷ Recurse
        z = z + extend(z_c,i)                   ▷ Prolong update
        r = r - B_i z                           ▷ Update residual
        z = z + smooth(r,i)
    return z
```

This grid numbering convention may appear unintuitive; indeed, grids could simply be enumerated based on the level of mesh refinement; however, in the distributed implementation outlined next, `coarse(i)` and `fine(i-1)` correspond to two different partitions of the

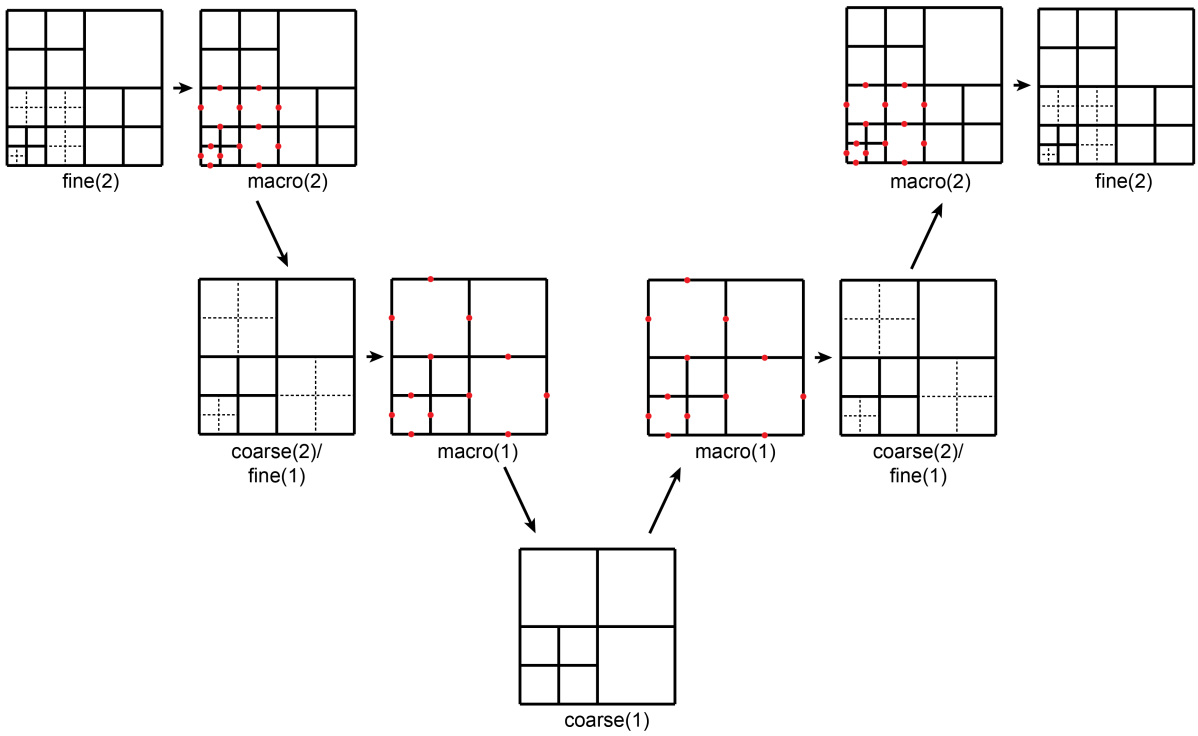


Figure 5.3: Multigrid V-cycle schematic. Adapted from "An adaptive multigrid solver for DPG methods with applications in linear acoustics and electromagnetics." by S. Petrides and L. Demkowicz, *Comput. Math. Appl.*, 87:12–26, 2021.

same mesh. Note that the definition of smoothing patches, etc. is also based on this two-grid structure, with smoothing patches on `macro(i)` (or `fine(i)`) defined via the support of `coarse(i)` vertex patches.

5.3 Distributed Implementation

The distributed DPG-MG solver leverages a domain-partitioning approach to distribute work among parallel processors. Domain partitioning is necessary since the DPG-MG solver stores the macro-element stiffness matrices and (optionally) patches, these operators consume a large amount of memory and cannot be efficiently communicated during the solve. However, domain partitioning presents a challenge in the context of mesh adaptivity since local mesh refinement quickly imbalances a partition; repartitioning is thus necessary for load balancing. Repartitioning is somewhat more complex in the context of multigrid, which operates on hierarchy of meshes. Our implementation partitions grids as they are constructed; the top level grid may be repartitioned while it is active, but once an additional grid is added the

coarser grids becomes static. During prolongation, the solution or residual array is migrated to coincide with the partition on that mesh. The multigrid V-cycle with repartitioning is illustrated in Fig. 5.4 and outlined in Algorithm 2.

We intend to publish a version of the solver in the FrequenSolver Github repository[5] but it is not available at the time of writing. A simplified version of the solver, based on a previous version of *hp*3D will be made available in the repository, or can be obtained by contacting the author[6]. A shared memory version of the solver is currently available in *hp*3D under the directory `trunk/src/solver/MGCG/`, but is not compatible with the current *hp*3D tagged release. We will point to that code when possible to provide a concrete instantiation of the ideas presented here.

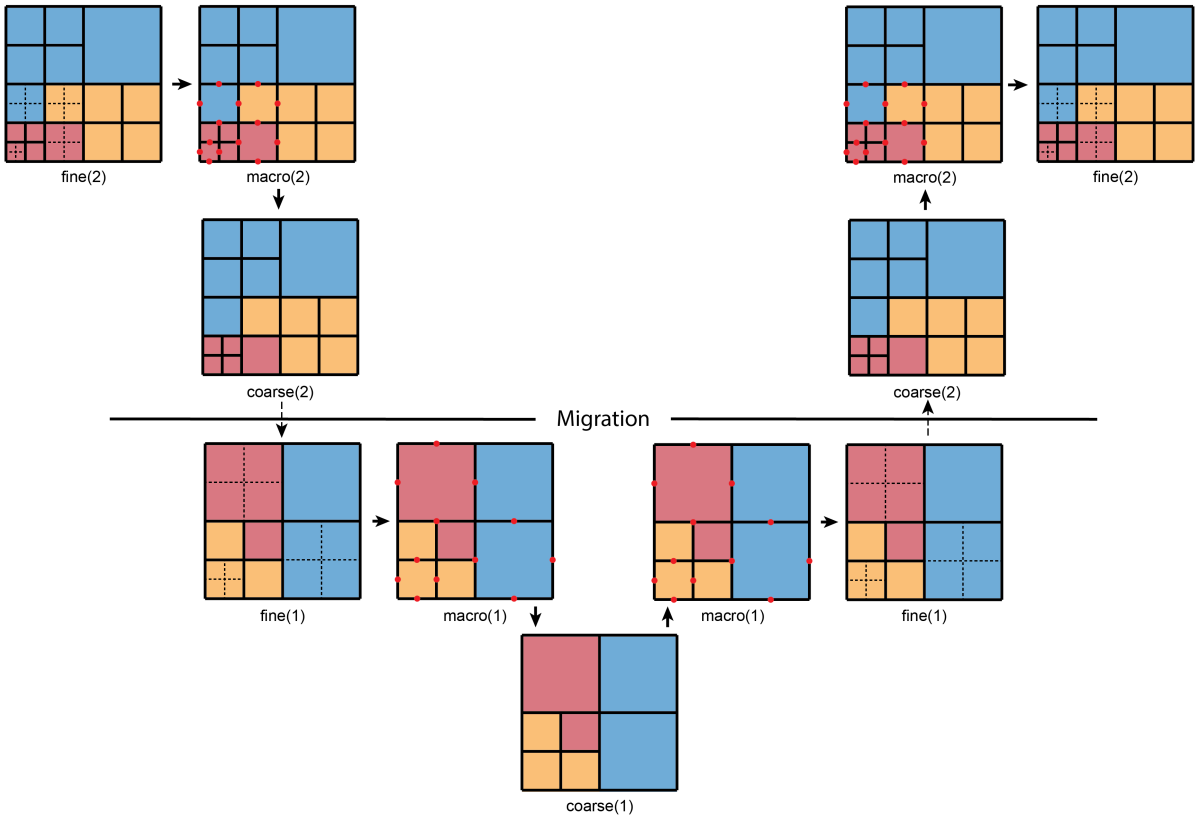


Figure 5.4: Multigrid V-cycle schematic with repartitioning.

[5] https://github.com/FrequenSol/FrequenSolver

[6] `jcbadger@utexas.edu`

Algorithm 2 Distributed multigrid V(1,1)-cycle with migration

```
procedure V_CYCLE(r,i)                                  ▷ r: residual, i: Grid level
    if i = 1 then
        z = coarse_correction(r)                        ▷ Coarsest-grid correction
    else
        z = smooth(r,i)
        r = r - B_i z                                   ▷ Update residual

        r_c = I_c^m(i) r                                ▷ Restrict from macro(i) to coarse(i)
        r_f = migrate_c2f(r_c,i)                        ▷ Migrate from coarse(i) to fine(i-1)
        r_m = I_m^f(i-1) r_f                            ▷ Restrict from fine(i-1) to macro(i-1)

        z_m = V_CYCLE(r_m,i-1)                          ▷ Recurse

        z_f = I_f^m(i-1) z_m                            ▷ Prolong from macro(i-1) to fine(i-1)
        z_c = migrate_f2c(z_f,i)                        ▷ Migrate from fine(i-1) to coarse(i)
        z = z + I_m^c(i) z_c                            ▷ Prolong from coarse(i) to macro(i)

        r = r - B_i z                                   ▷ Update residual
        z = z + smooth(r,i)
    return z
```

5.3.1 Mesh Partitioning and Ghosting

The distributed solver uses standard, element-wise partitioning, but partitions are required to conform to the coarse grid on each two-level cycle. As can be noted in Fig. 5.4, this implies that the mesh partition lags the mesh by one refinement level and can lead to some imbalance. This imbalance can be mitigated using a weighted partition, with weights reflecting the increased cost of coarse grid elements that have been refined.

After partitioning macro elements, each process further claims all coarse-grid vertex patches incident to its subdomain. Each process then appends ghost elements to its subdomain to complete the claimed patches. The resulting *padded* subdomain thus includes a single layer of macro-grid elements. The partitioning and ghosting strategies are illustrated in Fig. 5.5. This partitioning strategy is somewhat inefficient, patches on the interface between multiple subdomains will be claimed by multiple processes. More efficient partitioning strategies based on unique ownership of patches are possible but necessitate additional communication stages; the ghosting strategy illustrated in Fig. 5.5 was chosen for simplicity of presentation and implementation. An alternative partitioning strategy is outlined in Sec-

tion 5.6.2.

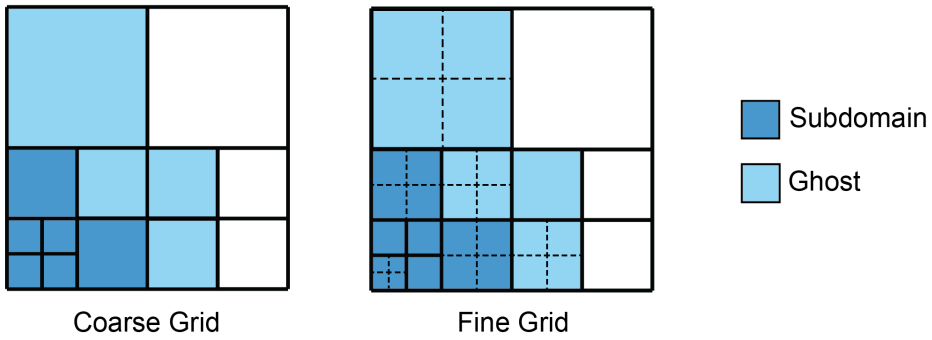


Figure 5.5: Partitioning and ghosting strategy. Coarse-grid elements are partitioned; ghost elements are then appended to complete vertex patches incident to subdomain elements.

5.3.2 Data Structures

***hp*3D data structures.** We briefly review the main data structures in *hp*3D—particularly `ELEMS` and `NODES`. A more extensive review of these structures can be found in Leszek Demkowicz's book series on *hp*3D [25, 32, 60]. The `ELEMS` data structure stores element and connectivity information for *initial* (unrefined) mesh elements. The `ELEMS` structure is thus defined once, when a mesh is read in, and remains static for the remainder of the computation. Information about mesh refinement, approximation order, etc. is thus encoded in the `NODES` data structure, a dynamic array storing 'vertex', 'edge', 'face', and 'middle' node types. The `NODES` data structure is initialized for the initial mesh; the connectivity of these initial nodes is stored in the `ELEMS` array. Refining elements "breaks" nodes, generating a number of children nodes of different types that are appended to the data structure. For example, breaking an edge node (`NODES(nod)%ntype = MEDG`) results in two additional edge nodes and a single vertex node. `NODES` is thus a tree data structure, with the connectivity of nodes implied by their genealogy and the initial mesh connectivity. In addition to implicitly storing mesh connectivity (via genealogy) for refined elements, `NODES` stores the subdomain that middle nodes belong to, as well as geometry and solution degrees of freedom for nodes that are active in a given mesh.

The distributed implementation of *hp*3D utilizes a replicated-mesh approach, maintaining a full copy of the `ELEMS` and `NODES` data structure on each distributed process, but deallocating geometry and solution degrees of freedom not on a process' partition. The rather spartan design of `NODES` is central to *hp*3D's distribution strategy (indeed, each node occupies only

56 bytes) and enables solution of problems with $\mathcal{O}(10^9)$ degrees of freedom on large scale distributed compute resources. Indeed, during development of the distributed version of *hp*3D [58], considerable effort was devoted to 'slimming' the data structures to enable some additional scalability at the expense of disabling features including mesh unrefinement, etc..

The largest problem considered in this work has 2.6 billion elements and 25 billion nodes, the replicated mesh approach would have required nearly 1.5 TB of memory per MPI rank. A fully distributed approach was thus required. Before introducing DPG-MG specific data structures, we take a brief detour to outline the design of our fully distributed data structures. More extensive details may be provided in the future, when an open-source version is available for reference.

Mostly-distributed data structures. The `ELEMS` and `NODES` data structures are fundamental to *hp*3D; the initial idea behind our fully distributed data structure was thus to leave these data structures unperturbed and 'trick' the code into thinking it was working on a full mesh. A 'translation layer' was then defined to reconcile the differing enumeration of elements and nodes on different processes. The first incarnation of the translation layer (Global Local mesh Unification or GLU) was defined starting from a replicated initial mesh. The replicated mesh was partitioned and a replicated data structure was constructed containing all nodes on elements incident to any subdomain interface, these will be referred to as global nodes.

Local `ELEMS` and `NODES` data structures, defined only on a process' ghosted subdomain (with a single layer of ghost elements) were then created and the replicated `ELEMS` and `NODES` data structures were deleted. Nodes on the interface were then bidirectionally linked to corresponding global nodes, this is illustrated in Fig. 5.6. This correspondence defined the translation layer: when a node was communicated it would first be routed to the translation layer, translated to its global node, communicated, then translated back on the receiving process. Reduction of an array indexed by node number is a common operation in *hp*3D. For example, unique node ownership can be decided by initializing an integer array; processes then mark all entries corresponding to nodes on their subdomain elements with their MPI rank. The array is then reduced with a min or max operation to decide ownership. In the mostly distributed approach, local nodes on the global interface are copied from the local array to the global array; the global array is then reduced, and the local entries are extracted

from the global array; this is outlined in Algorithm 3.

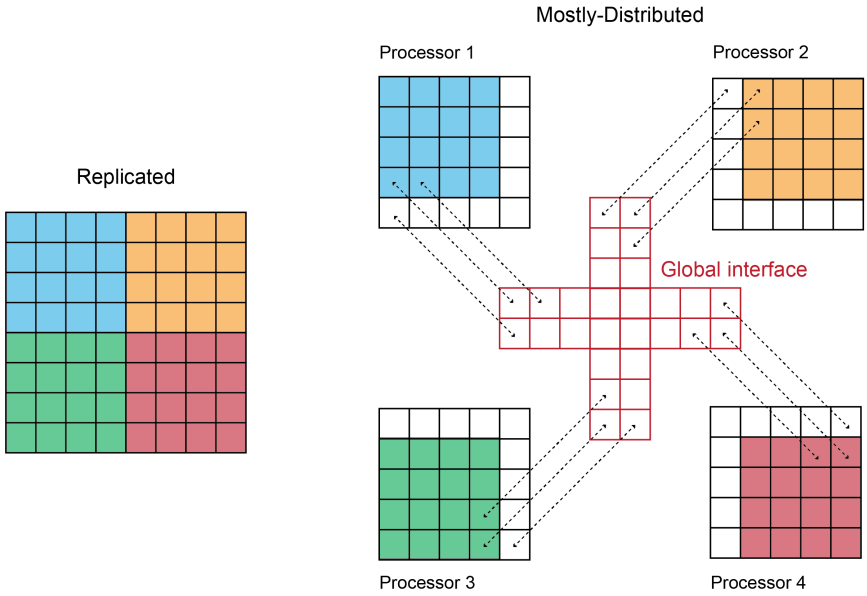


Figure 5.6: Replicated mesh (left), with all processes store a full copy of the mesh. In the mostly distributed data structure (right), each process stores its ghosted subdomain, and all processes store a consistent global interface.

Algorithm 3 Global reduction for local distributed arrays.

procedure GLU_AllReduce(`larray`)
- `garray(:)` $= 0$ ▷ Initialize global array (value depends on operation)
- **call** copy_to_glob(`larray`, `garray`) ▷ Extract global entries from local array
- **call** MPI_AllReduce(`garray`) ▷ Global reduction
- **call** copy_from_glob(`garray`, `larray`) ▷ Return global entries to local array
- **return** `larray`

The idea is thus relatively simple so far, next we outline how to produce globally consistent mesh refinements and update the global data structure accordingly. For now we will consider only isotropic element refinements so that an element (and consequently, a node) is either broken or not. We will allow processes to execute a single batch of refinements (in a refinement batch any number of elements on a process' subdomain may be refined *once*). After the refinements have been executed, an array indexed by local node number is initialized and any local nodes that were broken are marked. The array is then globally reduced as in Algorithm 3. Processes then loop through the resulting local array and reproduce the indicated refinements. At this point all local data structures are consistent; the global data

structure is then updated by performing the reduction in Algorithm 3 once more, but retaining the global array. Processes then loop through the global array *in order* and break any indicated global nodes. Because the global interface was constructed with the same ordering across processes (via the replicated mesh), breaking nodes in the same order will result in a consistent updated global data structure.

A few notes on this approach. First, all refinements to the local data structure are produced in parallel which, once we have removed the global data structure, will allow refinements to be produced quickly. Second is that the approach was not intrusive, it largely did not require changes to existing data structures or routines, with the exception of wrapping node-related MPI calls to redirect them to corresponding 'translated' (GLU) routines. Finally, while the global data structure was simple and convenient for illustration, the only requirement for any pair of nodes to communicate is that there exist a consistent translation layer between them. This motivates a fully-distributed implementation.

Fully-distributed data structures. The fully distributed analog (GLUd) of the mostly-distributed data structure follows the same logical outline but instead of creating a single translation layer on the global interface, each process creates a local translation layer with each of it's neighbors. There are multiple ways to implement this; our initial approach was for each process defined a data structure containing all the nodes that are shared between the process and any other process (either in their subdomain or ghost layer). Local nodes were bidirectionally linked to shared nodes. Additional data structures were then created containing the nodes shared with each neighbor, and the shared nodes were bidirectionally linked to neighbor data structures, this is illustrated in Fig. 5.7. The call to the global MPI collective in Algorithm 3 was then replaced by point-to-point communications gathering shared entries from each neighbor, performing the reduction in the order of shared nodes, and then communicating the result back. The translation layers were then updated similar to the mostly-distributed context. We have of course neglected to outline every routine needed to support the fully distributed code but what has been outlined is largely sufficient to define and update translation layers. The logic is rather intuitive once a consistent node ordering is established.

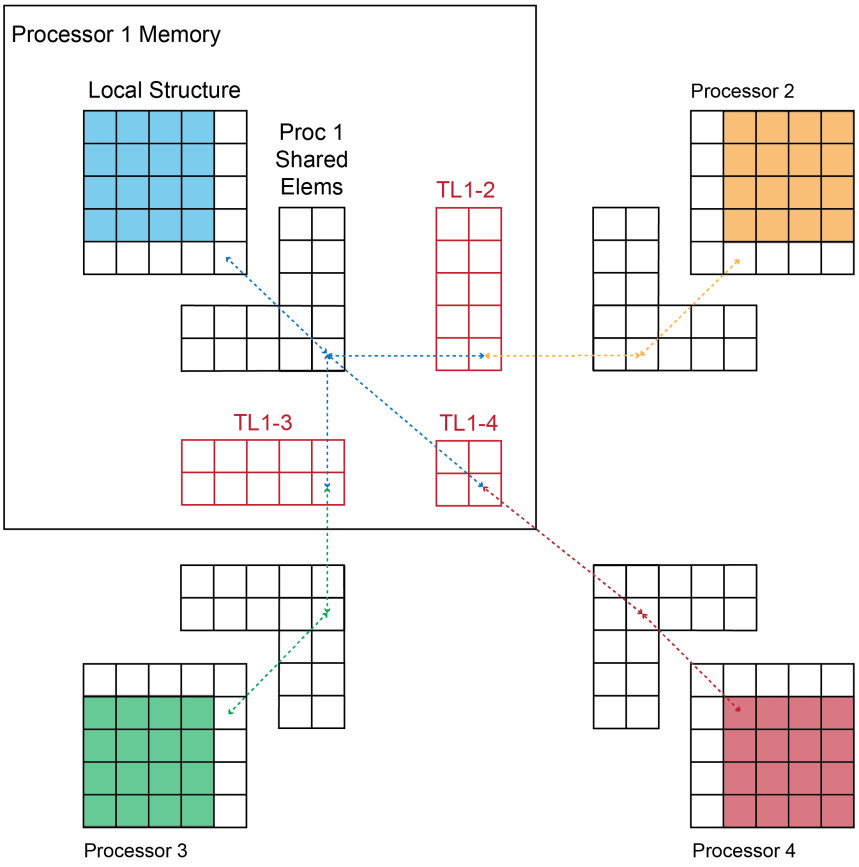


Figure 5.7: Fully distributed data structure where each process stores its ghosted subdomain, a structure storing elements shared with any neighbor, and a *consistent* interface for each neighboring process. Only structures relating to process 1 are shown.

Limitations. We have made a number of assumptions in defining the fully-distributed data structure. The first implicit assumption in *hp*3D was that translation layers (and the distributed mesh) were defined starting on the initial mesh, before any refinements have been produced. This restriction is largely because connectivities are explicitly stored only on the initial mesh, and traversing the mesh and reconstructing connectivities thus requires descending the data structure down to the initial mesh. A second assumption was that only isotropic refinements were permitted (at least on shared elements), this avoids more complex reconciliation when processes produce different refinements of the same element. Finally, we have implicitly assumed that once a mesh has been fully distributed it cannot be redistributed. This is a severe restriction, especially in the context of adaptive mesh refinement.

The implementation used in this work did not address the first limitation. The second limitation can be addressed rather simply by redefining the reduction operator to compute the union of anisotropic refinements. For example, if one process refined an element in the x-direction and another process performed a refinement in the y-direction, then the reduction operator would return an xy-refinement; local data structures then remove inconsistent refinements and reproduce the reduced refinement. The final limitation was addressed by introducing the concept of mesh 'parts,' a fully-distributed subdomain that is shared by multiple processes. In particular, the local data structures (`NODES` and `ELEMS`) defining the mesh part were replicated by a number of processes, allowing the part to be repartitioned as usual among those processes, averaging any imbalance over a number of processes. This is thus a two-level partitioning approach where the lower level is fully distributed and the top level is replicated. On the top level, standard MPI collectives were used to communicate among processes sharing the part. On the lower (fully distributed) level, one process per part is nominated as the manager and handles communication with other parts. A hierarchical version of Algorithm 3 would thus first reduce the local array from workers in a part to the manager; the manager would then proceed with the global reduction, and the local result would then be broadcast back to the workers.

We emphasize that one of the benefits of this translation layer model is that it is largely non-invasive. The translation layer code itself can become rather complex but is only ever encountered via redirection of fairly intuitive MPI collectives over nodes and elements. That is not to say our implementation was fully non-invasive; as expected, additional bottlenecks were encountered in scaling from $\mathcal{O}(10^9)$ DOFs to $\mathcal{O}(10^{12})$ DOFs and required various changes to algorithms and data structures.

DPG-MG data structures. The DPG-MG solver defines two additional data structures, `NODES_MG` and `GRID`. The `NODES_MG` structure is simply an extension of `NODES`, storing the following additional information for each node:

- `orderC` – Integer storing the order of approximation of the node on the previous mesh
- `master(:)` – Array indicating whether the node is an active node in a particular grid
- `subd_old` – The subdomain containing the node in the previous mesh partition

The `GRID` data structure is the workhorse of the DPG-MG solver, storing operators and other data required for prolongation, smoothing, and residual updates. Our implementations of the DPG-MG solver primarily operate using element-wise unassembled operators. For example, no global sparse stiffness matrix is stored, the stiffness matrix is instead stored and applied element-wise (in unassembled form). This framework of block-wise operations is defined next. As operations are defined their location in the `GRID` structure will be noted.

5.3.3 Block-Wise Operations

The distributed solver relies primarily on block-wise operations. This basic operation is given in Algorithm 4.

Algorithm 4 Gather-Operate-Reduce

```
procedure BLOCKWISE_OPERATION(v_g)
    for i_blk=1 to N_blk do                                   ▷ Loop over blocks
        for i_l = 1 to block_size(i_blk) do ▷ Gather entries from global array into local array
            i_g = lcon(i_l)
            v_l(i_l) = v_g(i_g)

        call block_operation(v_l)                             ▷ Operate on local array

        for i_l = 1 to block_size(i_blk) do                   ▷ Reduce entries back into global array
            i_g = lcon(i_l)
            u_g(i_g) = u_g(i_g) + v_l(i_l)
    return v_g
```

The shared-memory implementation loosely follows this paradigm; each block (e.g. an element or patch) stores a connectivity array (typically named `lcon`) containing the global indices where data can be found. Threads loop through the blocks, extract entries for a particular block, apply the operator, and then reduce the result. The reduction is often particularly expensive; use of standard reduction clauses cause threads to create copies of the entire array and use of locks tends to serialize operations and limit efficiency. A simultaneously more maintainable and higher-performance approach defines these operations as abstract algebraic blocks. Operations including smoothing, applying stiffness matrices, and applying prolongation operators, as well as communication routines can be defined in this abstract framework, simplifying routines and interfaces and further enabling analysis and optimization. For example, block dependencies can be analyzed and independent blocks and

be grouped, removing the need for threaded reduction entirely. Abstraction could further facilitate offloading of blocks to accelerators in heterogenous compute contexts, or enable exchanging blocks with neighboring processes to enable improved dynamic load-balancing.

We briefly note that an alternative approach to this block-wise approach is to assemble blocks into sparse matrices. We briefly explored this approach, implementing the stiffness matrix product in PETSc[10], but found it to be less performant than the block-wise approach. The large systems considered here require use of 64-bit integer indices in PETSc; indices thus consume 33–50% of memory bandwidth, depending on whether they system is stored in (complex) single or double precision. PETSc was further unable to take advantage of the Hermitian structure of DPG systems, this required reading $2\times$ the number of entries. More optimal sparse-matrix approaches could be perhaps be devised and may simplify implementation. We finally note that storing the global system in unassembled form does require storing additional entries; however, when accounting for the cost of storing 64-bit sparse matrix indices, memory consumption was comparable (when storing operators in single precision, the block-wise approach required substantially less memory).

Residual update and smoothing. The most basic examples of block-wise operations include routines used to compute global matrix-vector products. These are defined in `modules/pcg_info.F90` in the shared-memory implementation. In this case, blocks correspond to macro-elements in the *padded*-subdomain and the operator applies the macro-element stiffness matrix.

The overlapping block-Jacobi smoother, defined in `smoother/smoothers.F90` is similar: blocks correspond to patches and the operator is the patch stiffness matrix inverse. Storing the patch in Cholesky-factorized form reduces the cost of applying the patch stiffness matrix from $\mathcal{O}(N_p^3)$ to $\mathcal{O}(N_p^2)$, where N_p denotes the number of DOFs supported on the patch. Patches can alternatively be re-assembled from macro-element stiffness matrices and refactored at each step. While often significantly more expensive, recomputing patches can reduce memory requirements and enable solution of larger problems; this will be revisited in Section 5.6.

- Stiffness matrix application (`modules/pcg_info.F90`)
  - *Connectivity*: Macro-element to global (`GRID%dloc(iel)%lcon`)

  - *Operator*: Element matrix-vector multiply (`GRID%dloc(iel)%zstiff`)

- Overlapping Block-Jacobi Smoother (`smoother/smoothers.F90`)
  - *Connectivity*: Patch to global connectivities (`GRID%patch(iptch)%lcon`)
  - *Operator*: Patch matrix inverse (`GRID%patch(iptch)%zAp`)

Sharing. Smoothing produces an update vector; in the distributed context each process only applies a subset of patches, communication is thus needed to assemble process-local updates into a globally consistent update. The ghosting strategy was previously defined such that a process claims all patches incident to its subdomain elements; this implies that patches on interfaces will be applied by multiple processes but ensures that all unknowns on *subdomain* elements will be fully assembled and globally consistent. Communication is required to update unknowns on ghost elements; this is accomplished by communicating macro-element DOFs from processes on which the element is owned, to processes on which the element is ghosted.

Sharing is also required after applying the stiffness matrix. We again perform redundant operations to simplify the required communication. In particular, each process applies and assembles all of the elements in its *padded* subdomain; again producing a result in which subdomain entries are already assembled and globally consistent. The same communication routine used after smoothing can thus be used to share the solution from owned elements to ghosted elements. Communication operators are not implemented in the shared-memory version of the code available in *hp*3D.

Prolongation and Migration. The distributed multigrid V-cycle in Algorithm 2 separates prolongation operations into three stages: macro-to-coarse prolongation, migration, and fine-to-macro prolongation. Each stage has a similar block-wise structure, but are relatively more involved because the input and output are defined on different meshes.

Restriction begins by restricting from the macro to coarse grid, by applying constrained approximation operators to subdomain elements only. The element-wise operators are defined in (`prolongation/macro2coarse.F90`). The element-wise output of the macro-to-coarse restriction is stored in element-local arrays (defined in `GRID%loc(iel)`) and are not assembled into a global array.

Migration communicates local arrays from the (unique) process owning an element on the previous partition to the (unique) process owning it on the next partition.

The final stage of the restriction is defined via the Schur complement. As outlined in Section 5.2.4, a number of fine grid elements are first assembled into a partitioned array separating interface DOFs, defined on the macro-grid skeleton, from the remaining DOFs. The assembly and partiton are defined by `GRID%sch(iel_m)%gloc(iel_f)%lcon`; `iel_m` corresponds to the macro-grid element, `iel_f` denotes the index of the fine-grid element *within the macro element*, and the connectivity array `lcon` stores a positive index for interface DOFs and a negative index for bubble DOFs (simultaneously encoding the ordering and partitioning of the assembly). The Schur complement operator (`GRID%sch(iel_m)%A21`) is then applied and the result is again stored in local arrays.

At this point each process has local restricted residuals on its subdomain elements only, the distributed implementation thus communicates once more, sharing the local arrays from processes on which the element is owned, to processes on which the element is ghosted. The local arrays are then finally assembled into global arrays using the standard macro-element connectivities (`GRID%dloc(iel)%lcon`).

5.4 Convergence Studies

This section outlines a number of convergence studies to investigate the robustness of the DPG-MG solver with respect to element size h, polynomial order p, and angular frequency ω. Previous expositions of the DPG-MG solver considered a variety of physical problems, smoothing steps, and tolerances, illustrating the versatility of the DPG-MG solver but somewhat confounding the scaling behavior. To elucidate the convergence characteristics of the DPG-MG solver, we fix the following parameters:

- Conjugate gradient iterations are terminated when the relative ℓ^2-norm of the discrete residual is less than 10^{-7}.
- After each refinement, the (initial) solution is reset to zero; in other words, solutions from previous grids are *not* used to generate initial guesses for following grids.
- A single pre- and post-smoothing step is performed on each grid level (V(1,1)-cycle), except in one case in Section 5.4.2 in which both *one* and *five* smoothing steps are employed to aid in comparison; this case will be noted.

- The initial mesh is a single second-order element ($p = 2$); however, iterations are not reported for the initial mesh, which is solved using the MUMPS direct sparse solver [4].
- No initial-grid solver is employed during the iteration. We have observed no effect on convergence when the initial grid is far from resolving the wave. This will be explored further in Section 5.5.
- Test norm weight $\alpha = 9$ was used[7].

All experiments in this section were performed on *Frontera*'s Cascade Lake (CLX) nodes at the Texas Advanced Computing Center [111]. Timing statistics are neglected in this section, but will be provided in Section 5.6.

5.4.1 Problem Setup

Throughout this section we consider propagation of an acoustic Gaussian beam with waist-radius 0.1 and direction given by spherical angles $(\theta, \phi) = (45^\circ, 55^\circ)$ in a homogeneous unit cube domain $[0, 1]^3$. Homogeneous impedance boundary conditions are imposed on all surfaces except near the origin, where the Gaussian beam is injected through a manufactured impedance load. More precisely, let $g(\mathbf{x})$ denote the prescribed Gaussian beam solution; the impedance load was defined on boundary $\Gamma_3 = \partial\Omega$ as:

$$u_0(\mathbf{x}) = e^{-1000\,|\mathbf{x}|^6(\mathbf{x})}\big(Z^{-1}g + (i\omega)^{-1}\partial_n g\big)$$

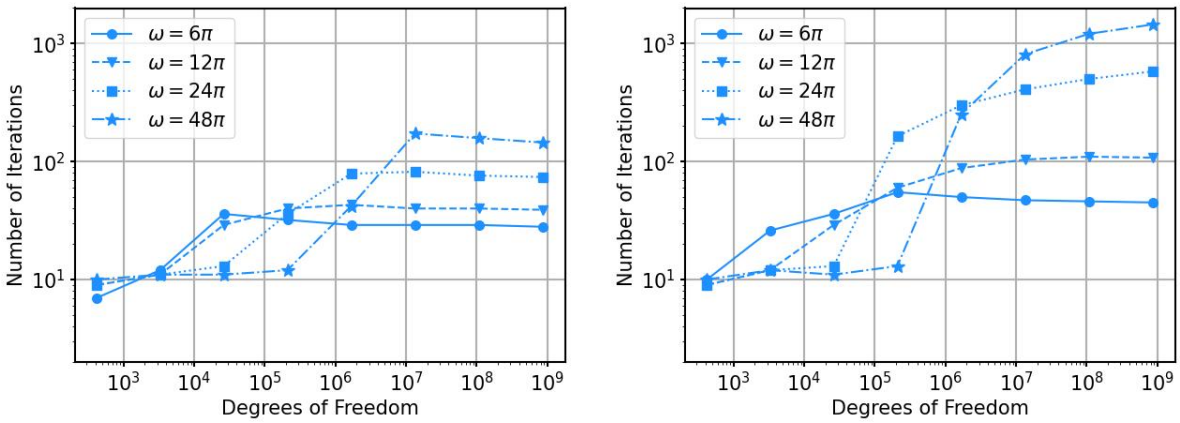
where the exponential term corresponds to a fast-decaying window function that localizes the load near the origin, and ∂_n denotes the outward normal derivative. Ultraweak DPG system (3.27) was solved with acoustic impedance $d = 1$. The solution is depicted in Fig. 5.11.

5.4.2 Direct Assembly vs. Fine-Grid Restriction of Coarse-Grid Operators

As indicated in Section 5.2.5, coarse-grid systems can be either formed directly (or stored from previous meshes) or computed from fine-grid systems by applying the restriction operator. The two approaches are referred to as *store* and *restrict*, respectively. As will be demonstrated, the construction of coarse-grid systems has significant implications for convergence of the DPG-MG solver.

[7] In the original study in [9], $\alpha = 1$ was claimed but a length-scale of 1/3 km was chosen that effectively increased the size of the domain and decreased the frequency (both by a factor of 3×); we neglected to consider the effect of the non-dimensionalization on α.

Uniform h-refinements. We begin by studying convergence of the DPG-MG solver under uniform h-refinements; i.e. each subsequent grid is produced by a uniform h-refinement of the previous grid. The number of iterations required for convergence under each of the approaches, for a variety of frequencies, is reported in Fig. 5.8. Examining the results in Fig. 5.8a (*restrict*), it can be seen that the number of iterations increases roughly linearly with frequency but demonstrates clear h-robustness in the asymptotic regime. The increase in number of iterations with frequency is expected: meshes that cannot resolve the wave do not contribute to preconditioning. Note that unlike multigrid preconditioners for the standard Galerkin method, which would diverge in this setting due to lack of discrete stability on the coarse grid, the DPG solver remains stable. Next, comparing Fig. 5.8a (*restrict*) and Fig. 5.8b (*store*), it can be seen that storing the coarse-grid system consistently resulted in a larger number of iterations than restricting. Storing the coarse-grid system additionally does not appear to demonstrate h-robustness.



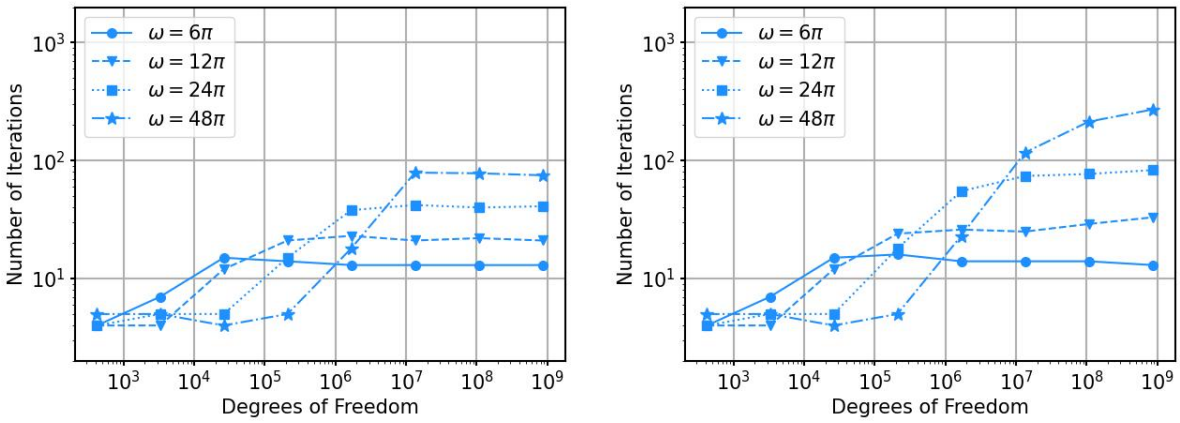
(a) Coarse-grid system restricted from fine-grid system

(b) Coarse-grid system stored from previous meshes

Figure 5.8: Convergence of the DPG-MG solver with *one smoothing step* applied to uniformly h-refined meshes. The solver convergence is h-robust and the iteration numbers are lower when using fine-grid restriction. The iterations until convergence depend linearly on the frequency ω.

Uniform h-refinements; five smoothing steps. In the original implementation of the DPG-MG solver, a relatively large number of smoothing iterations (typically between 5 and 10) were used in numerical experiments. For comparison, we repeat the previous study using five smoothing steps per iteration (V(5,5)-cycle); the results are depicted in Fig. 5.9.

Using a large number of smoothing steps tends to restore the h-robust convergence when the coarse-grid system is stored (Fig. 5.9b); however, the number of smoothing steps needed to attain h-robustness tends to increase with frequency. In particular, note that for the higher frequencies, the number of iterations until convergence is lower when using one smoothing step with restriction from fine-grid systems (Fig. 5.8a) than when using five smoothing steps with coarse-grid operators stored from previous meshes (Fig. 5.9b). Comparing Fig. 5.8a and Fig. 5.9a, it can be seen that when the coarse-grid systems are defined via restriction, increasing the number of smoothing steps by a factor of five results in a decrease of the number of smoothing iterations by only a factor of two. We neglect an explicit study of convergence in terms of the number of smoothing steps but qualitatively report that a single smoothing step per grid level has been optimal in all of our numerical experiments to date.

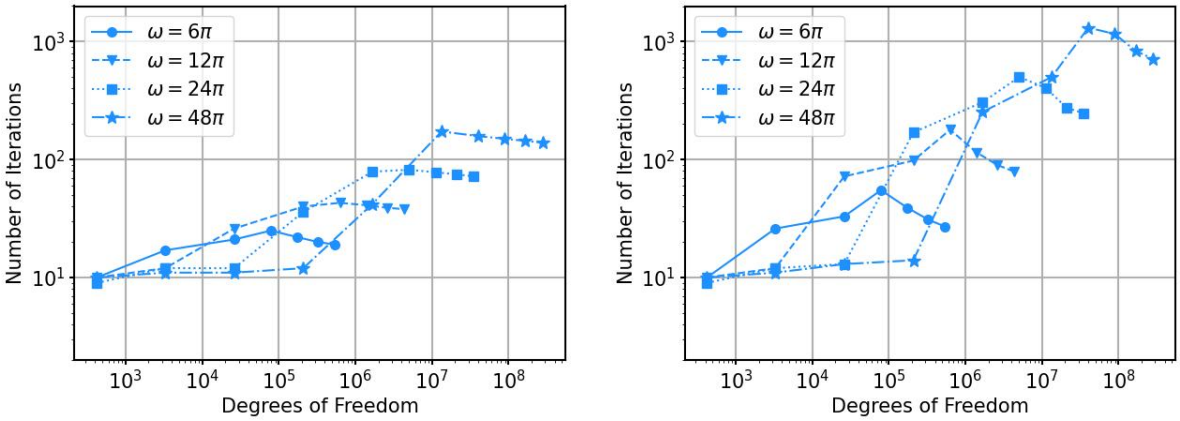


(a) Coarse-grid system restricted from fine-grid system

(b) Coarse-grid system stored from previous meshes

Figure 5.9: Convergence of the DPG-MG solver with *five smoothing steps* applied to uniformly h-refined meshes. Doing additional smoothing tends to restore h-robustness in (b) to some extent; in (a), h-robustness is still observed, however using five smoothing steps per level only reduces the number of iterations until convergence by a factor of approximately two when compared to one smoothing step. Iterations depend linearly on the frequency ω.

Uniform p-refinements. To investigate convergence of the DPG-MG solver under p-refinements, we first perform h-refinements until there are at least two elements per wavelength, p is then incremented on each subsequent grid. The results of this study are shown in Fig. 5.10, where it can be seen that both storing and restricting lead to p-robust convergence; however, restricting again requires far fewer iterations.



(a) Coarse-grid system restricted from fine-grid system

(b) Coarse-grid system stored from previous meshes

Figure 5.10: Convergence of the DPG-MG solver applied to uniformly *hp*-refined meshes: grids are uniformly *h*-refined until two elements per wavelength, then uniformly *p*-refined. Convergence is *p*-robust and the iteration numbers are lower when using fine-grid restriction. Iterations depend linearly on the frequency ω

5.4.3 *hp*-Adaptive Refinements

We now consider *hp*-adaptive refinements, employing the Dörfler marking strategy [34] to determine elements to be refined. Marked elements are *h*-refined until the maximum edge-length is less than one-half the wavelength, otherwise they are *p*-refined. We end refinements one mesh after no additional *h*-refinements are requested. As shown in Fig. 5.11, *hp*-adaptive refinements produce a series of meshes with a "sweeping" structure, i.e., they follow the direction of propagation of the beam.

In the case of uniform *h*-refinements, the observed linear increase in iterations with frequency is expected and is related to the inadequacy of coarse-space corrections when meshes are not sufficiently fine to resolve the wave. With *hp*-adaptive refinements, the behavior of the number of iterations until convergence with respect to frequency is less obvious since intermediate meshes are able to *partially* resolve the wave. Indeed, we initially hypothesized the "sweeping" structure of meshes would reduce the frequency dependence of convergence. The convergence study in Fig. 5.12 seems to indicate this is not the case; the number of iterations increases linearly with frequency. Further, note the maximum number of iterations required for convergence was consistently higher than for uniform refinements; this was unexpected since the adaptive case smoothes on each grid level, thus a much larger number of smoothing steps are performed overall.

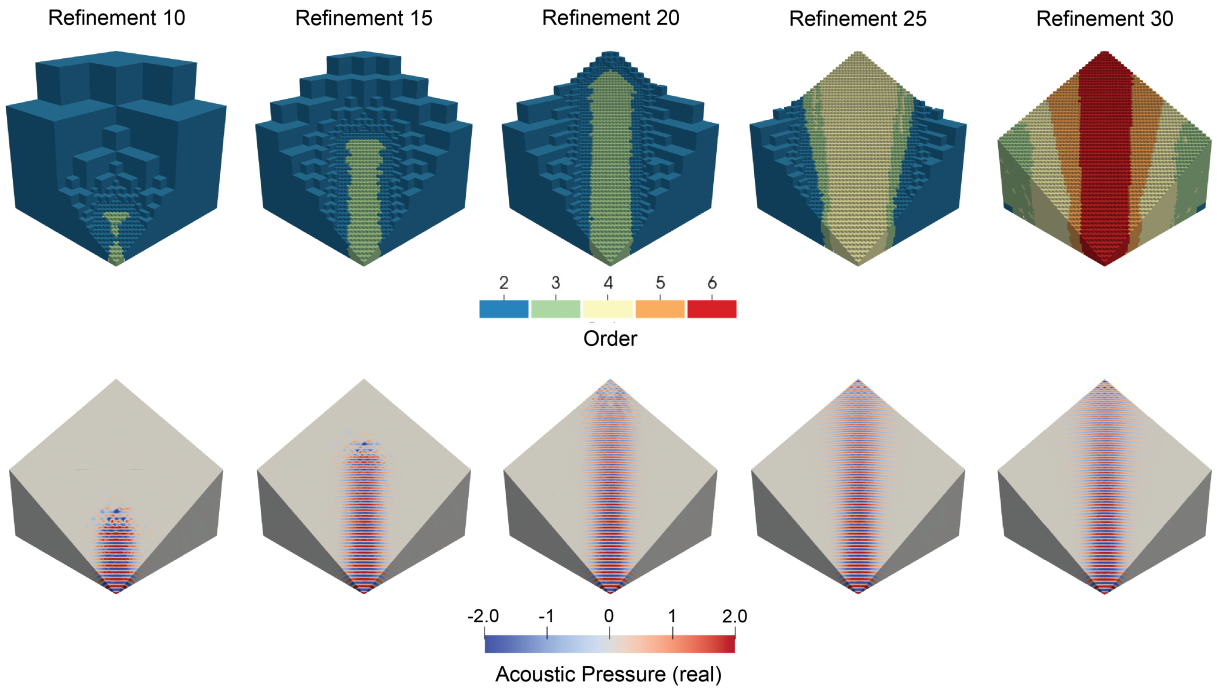
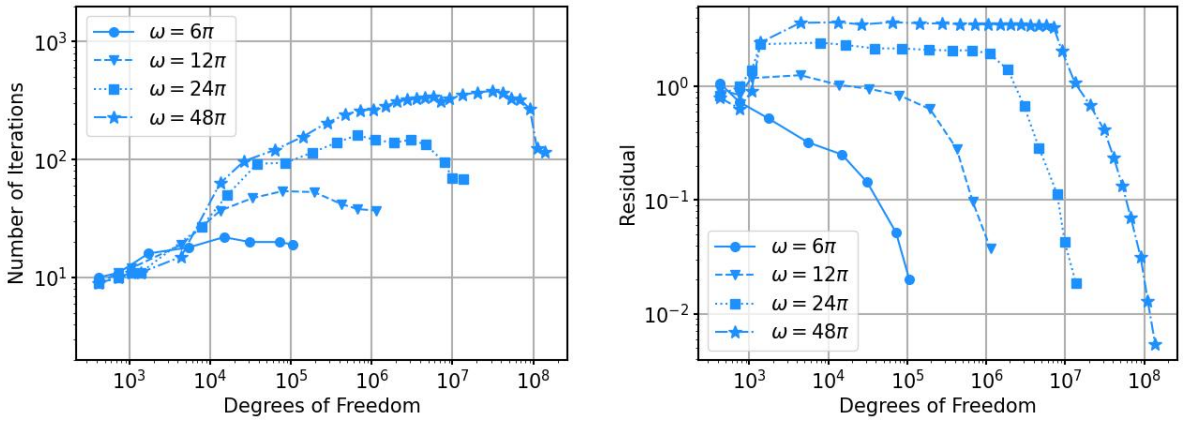


Figure 5.11: *hp*-adaptive propagation of a Gaussian beam in a cube domain; crinkle cut of *hp*-adaptive meshes (top) and surface cut of the real part of the acoustic pressure field (bottom). The *hp*-adaptive meshes have a "sweeping" structure as refinements first accumulate near the corner and then propagate into the domain; solutions on intermediate meshes are stable and partially resolve the wave.



(a) Number of iterations to 10^{-7} tolerance

(b) Convergence of DPG residual

Figure 5.12: Convergence DPG-MG solver applied to *hp*-adaptive meshes. The number of iterations required for convergence again increases linearly with frequency (ω) and the maximum number of iterations is higher than for uniform refinements (compare Fig. 5.8a).

An initial hypothesis on the cause of the deteriorated convergence of the DPG-MG solver on *hp*-adaptive meshes implicates a similar phenomenon underlying the deteriorated con-

vergence when coarse-grid systems are stored from previous meshes. In that case, roughly speaking, different scales between fine and coarse systems, imparted by element-wise Gram matrices, were ameliorated by constructing coarse-grid systems as restrictions of fine-grid systems, so that all systems inherit the fine-grid scale. Under *hp*-adaptive refinements, multiple element sizes—with various scales imparted by element-wise Gram matrices—are simultaneously present. A more rigorous investigation is required; however, note that in Fig. 5.12a, the number of iterations for convergence on the final meshes decreases considerably. Returning to Fig. 5.11, we can see that these final meshes are characterized exclusively by *p*-refinements with a consistent, uniform element size; from which coarse-grid systems are restricted.

5.5 Coarse-Grid Correction

Initial implementations of the DPG-MG solver leveraged a parallel direct sparse solver [4] on the initial mesh; this direct solver was also used for the coarse-grid correction on the initial mesh during the multigrid V-cycle. However, when the initial mesh did not at least partially resolve the wavefield, the initial-grid solver was not observed to improve convergence. This motivated terminating the V-cycle at some intermediate grid depth, when the mesh was still fine enough to provide a reasonable correction. The size of the system on the coarsest grid that improves convergence grows with frequency and quickly becomes too large for direct sparse solvers, we thus investigated a number of other approaches for producing the coarse-grid correction.

We have employed a number of different strategies throughout this work. The first was to simply neglect the coarse grid solver and use a number of smoother iterations on the coarsest grid. A slight modification of this strategy is to use a PCG iteration on the coarsest grid, preconditioned by one or more smoothing steps. Note however that the later strategy implies that the outer-loop preconditioner will be nonlinear since step sizes, as well as potentially the number of steps (when using a tolerance to terminate the inner iteration), on the coarse grid varies based on the residual. Use of a PCG iteration on the coarse grid thus requires that flexible PCG [96] be employed in the outer iteration.

The classical PCG iteration defines β via the Fletcher–Reeves formula:

$$\beta_k = \frac{\mathbf{r}_{k+1}^* \mathbf{z}_{k+1}}{\mathbf{r}_k^* \mathbf{z}_k},$$

where $\mathbf{z}_k$ is the preconditioned residual, i.e. $\mathbf{z}_k = \mathsf{M}\mathbf{r}_k$. When M is Hermitian positive-definite, this can be rewritten as:

$$\beta_k = \frac{\|\mathbf{r}_{k+1}\|_{\mathsf{M}}^2}{\|\mathbf{r}_k\|_{\mathsf{M}}^2},$$

and it is clear that $\beta \in \mathbb{R}$.

In contrast, the flexible PCG iteration leverages the Polak–Ribière formula:

$$\beta_k = \frac{\mathbf{r}_{k+1}^*(\mathbf{z}_{k+1} - \mathbf{z}_k)}{\mathbf{r}_k^*\mathbf{z}_k}.$$

The Polak–Ribière formula is often employed in the nonlinear conjugate gradient iteration but requires storing an additional vector. For linear problems, when M is linear $\mathbf{r}_{k+1}^*\mathbf{z}_k = 0$ and the methods coincide. Convergence of the Polak–Ribière formula was observed to stall occasionally when nesting PCG iterations. Note that in this case the additional term implies that $\beta \in \mathbb{C}$. Stalling seemed to coincide with β developing an appreciable imaginary component, we thus adopted the slightly modified formula:

$$\beta_k = \frac{\Re\{\mathbf{r}_{k+1}^*(\mathbf{z}_{k+1} - \mathbf{z}_k)\}}{\mathbf{r}_k^*\mathbf{z}_k},$$

taking only the real part of the Polak–Ribière formula. This was observed to eliminate stalling and generally improved convergence. We have not yet investigated this observation further and simply note it in interest of reproducibility.

Table 5.1 outlines the effect of coarsest-grid on convergence of the tilted interface problem illustrated in Fig. 4.2. In this case, the domain was taken to be 1 km $\times$ 1 km $\times$ 0.75 km (the same size as the extended (PML) domain in that problem). Impedance boundary conditions were assumed on all sides except at $z = 0$, where a free-surface boundary condition ($u_n = 0$) was adopted. The frequency was taken to be 32 Hz with test norm weight $\alpha = 10^2$. Iterations were terminated at 10^{-5} relative ℓ^2 residual. The initial mesh consisted of $4 \times 4 \times 3$ second order ($p = 2$) hexahedral elements. The mesh was h-refined three times, then p-refined twice, so that the final mesh was composed of $32 \times 32 \times 24$ fourth-order ($p = 4$) elements and 8.4 million DOFs. The same material properties from Fig. 4.2 were adopted, implying two elements per wavelength on the finest grid. In Table 5.1, grid depth 0 corresponds to preconditioning using only the smoother on the finest level. Grid depth 1 terminated the multigrid V-cycle at the third-order mesh ($p = 3$) with two elements per wavelength. Grid depth 2 terminated the V-cycle at the second-order mesh ($p = 2$) with two elements per wavelength. Finally, grid depth 3 terminated the V-cycle at the second-order mesh ($p = 2$)

with one element per wavelength; coarser grids had no effect on convergence. The coarse grid correction on the given grid depth was produced either with a direct solver, or PCG with N_{PCG} iterations. The study was performed on a single *Frontera* CLX node.

We first explain a few seeming anomalies in Table 5.1. First, note that the V$(1,1)$-cycle uses two smoothing steps on each of the intermediate grids; when $N_{\text{PCG}} = 1$, only a single smoothing step is performed on the coarse grid, resulting in an increase in the number of iterations with decreasing depth. Also note that the number of iterations increases from grid depth 2 to grid depth 1; this is because the coarse-grid patches on grid depth 1 were defined by vertex patches on grid depth 2 (corresponding to a p-refinement), the smoothing patches thus have smaller support on grid depth 1 than grid depth 2.

At grid depth 3, with one second-order element per wavelength (two points per wavelength), the coarse-grid correction is limited by discretization error and increasing the algebraic accuracy of the coarse-grid correction (by increasing N_{PCG}) only marginally improves convergence. On grid depth 2, the coarse-grid correction meaningfully improves convergence, and the coarse grid correction with 8 PCG iterations achieves nearly the same benefit as the direct solver; this configuration was also fastest among PCG coarse-grid corrections. As noted previously, grid depth 1 used a smaller patch size and thus required more iterations and was generally more expensive, except when the direct solver was used. Timings with the direct solver are somewhat irrelevant in the context of the large problems we are targeting with the distributed implementation, but the small number of iterations is compelling and indicates a more accurate correction on that grid could improve performance. We have investigated use of W-cycles to this end, results have been mixed and will be presented in the future. With a few exceptions that will be noted, in the remainder of this work we used grid depth 2 with $N_{\text{PCG}} = 8$.

Table 5.1: Number of iterations and, in parentheses, total solve time for the acoustic test problem using either N_{PCG} PCG iterations or a direct sparse solver for the coarse-grid correction.

N_{PCG} \ Grid Depth	3	2	1	0
1	118 (31.8)	158 (41.3)	328 (57.7)	636 (49.16)
2	111 (29.8)	103 (28.1)	198 (44.9)	
4	109 (30.1)	76 (22.2)	131 (36.2)	
8	110 (31.3)	65 (21.2)	79 (30.0)	
Direct	109 (28.8)	64 (24.8)	15 (18.2)	

5.6 Scaling Studies

The scalable DPG-MG solver was developed during the past five years and performance has changed significantly throughout its development. Application problems considered in Chapter 6 were also produced over a number of years, timing and scaling were thus produced with different versions of the code and may not be comparable or reflect the current state of the solver. This section provides recent timing and scaling data from a number of the application problems. In particular, this section focuses on solver iteration costs to illustrate scaling. Convergence and overall timing data will be given in Chapter 6.

The timing and scaling data reported in this section are for an optimized implementation of the fully distributed DPG-MG solver. Optimizations include use of a modified ghosting strategy based on unique patch-ownership that eliminates redundant operations, overlapping computation and communication, use of mixed precision, and various other improvements. With respect to precision, operators (including element matrices and patch assembly and factorization) were formed in double precision. Smoothing operators and stiffness matrices were then stored and applied in single precision. Storing stiffness matrices and performing residual updates in double precision enables double precision results at a 20–30% higher cost per iteration. However, the applications demonstrated here operate primarily in the early-asymptotic regime with $\mathcal{O}(10^{-3})$–$\mathcal{O}(10^{-1})$ relative discretization error, single precision algebraic error error is thus acceptable. Results were obtained on *Frontera*. Reported timings were averaged over 100 iterations.

5.6.1 Strong Scaling

Many of the applications in Chapter 6 will be based on adapted meshes, with mesh refinements defined either via the DPG error indicator or adapted to material properties. We

will demonstrate strong scaling in the context of an *hp*-adapted mesh and a uniform mesh. This comparison is intended to demonstrate that *hp*-adapted and uniform meshes often demonstrate comparable performance per DOF, and will motivate use of simple uniform meshes in later weak scaling studies (weak scaling studies are often not convenient in the context of *hp*-adapted meshes as it can be difficult to generate meshes with the desired multiples of DOFs).

We consider elastic simulation on a small section of the SEAM Arid model [99]. The study was performed on a 0.5 km cube centered at $(5, 5, 0.25)$ km (see Fig. 6.17). A free-surface boundary condition was assumed at the surface and a 0.25 km wide PML was appended on each of the remaining boundaries. Fig. 5.13 compares the time per iteration on an *hp*-adapted mesh ($2 \leq p \leq 4$) with 51 200 elements (18.4 million DOFs) and on a uniform mesh ($p = 3$) with 49 152 elements (18.7 million DOFs). In the case of uniform refinements, multigrid levels were constructed via a single h-refinement followed by a single p-refinement; the coarsest mesh is thus one level coarser with order $p = 2$. The adapted mesh demonstrates slightly deteriorated strong scaling relative to the uniform mesh, mainly due to the load imbalance, but achieves acceptable performance given that adapted meshes often require far fewer DOFs to achieve similar accuracy. Timings in Fig. 5.13 demonstrate that the cost of applying the solver on adaptive meshes is comparable on a per-DOF basis with uniform meshes.

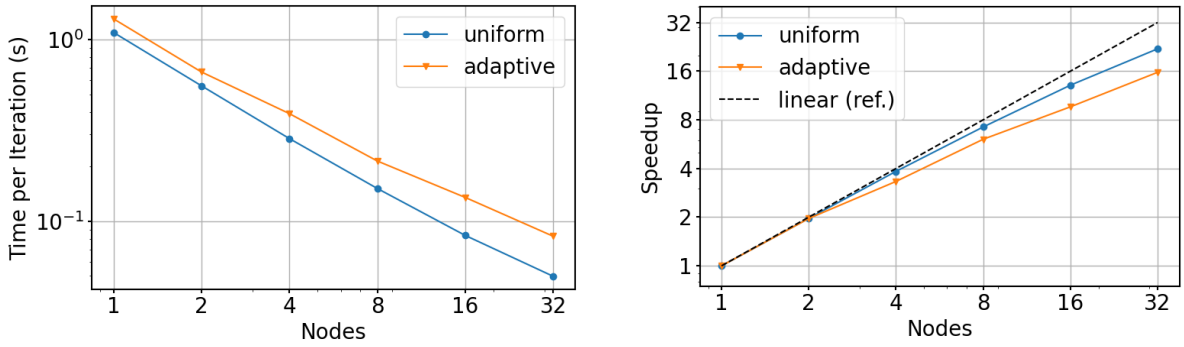


Figure 5.13: Strong scaling of elastic model with uniform ($p = 3$) and *hp*-adaptive mesh.

5.6.2 Weak Scaling – Storing Patches.

Weak scaling was investigated with elastic simulation on the uniform mesh used in the previous study; results are shown in Fig. 5.14. Each two-fold increase in problem size corre-

sponds to doubling the mesh size in one of the dimensions. Acceptable efficiency (ca. 80%) on up to 512 nodes was achieved. The decline in efficiency is largely due to imperfect partitioning. The employed partitioning strategy first performs a weighted partition of elements with weights defined by the number of DOFs supported on an element. After partitioning elements, we partition the patches supported on the interfaces between subdomains, this avoids redundant computation of patches and reduces the number of ghost elements. We required that patches only be claimed by a subdomain they were incident to, resulting in a constrained discrete optimization problem. A relatively simple damped iterative strategy was adopted to perform the partitioning. An initial partition was assigned, processes computed the cost of all currently claimed patches, and neighboring processes sequentially compared costs and exchanged patches corresponding to a (damped) fraction of the difference.

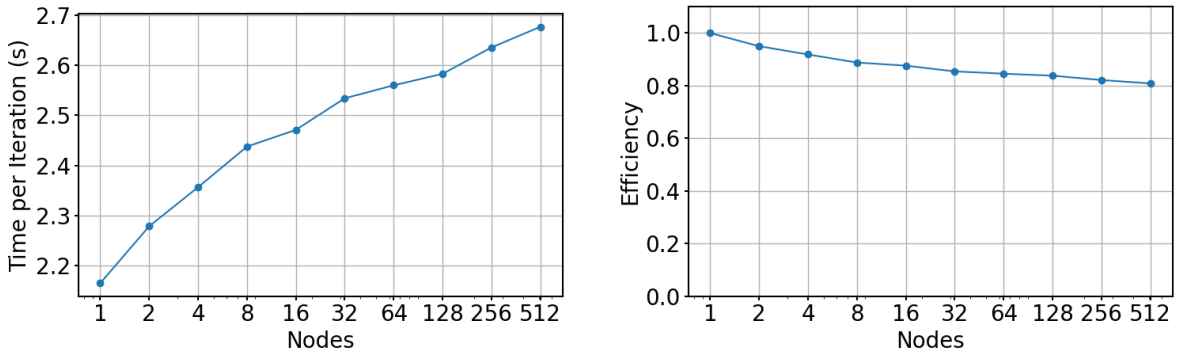


Figure 5.14: Weak scaling of the elastic model on a uniform $p = 3$ mesh.

5.6.3 Weak Scaling – Recomputing Patches

Smoothing patches were defined via coarse-grid vertex patches, resulting in relatively local smoothing blocks. Smoothing patches are stored in Cholesky-factorized form, this reduces the complexity of applying patch inverse from $\mathcal{O}(N_p^3)$ to $\mathcal{O}(N_p^2)$. Storing patches typically requires 2–3× more memory than storing macro-element stiffness matrices. One approach to reduce memory usage is thus to reconstruct patch stiffness matrices (from macro-element stiffness matrices) and recompute the factorization at each smoothing step. However, because Cholesky-factorization has a higher compute-intensity than back-substitution applied to a single right-hand-side, the refactorization is often less expensive than complexity alone would suggest on modern bandwidth-limited compute hardware.

This approach is demonstrated via a weak-scaling study on acoustic simulation of a

100 km ×100 km ×30 km section of the GO_3D_OBS model (see Fig. 6.9). In the case of the largest simulation on 8 192 *Frontera* nodes (458 752 cores), the coarsest mesh consisted of ca. 2.7 billion second-order ($p = 2$) hexahedral elements. Two uniform p-refinements were performed, resulting in a fine grid with 829 billion fourth-order ($p = 4$) DOFs. Scaling and timing data are depicted in Fig. 5.15. Over 80% efficiency was observed on 8 192 nodes, but this is largely due to the increased expense of re-assembly and factorization of patches, which hides communication and other costs. Indeed, recomputing patches increased the cost of smoothing by ca. 15×, and increased the cost of the full iteration by ca. 7×; these factors were determined by comparing costs on a smaller problem for which the patches could be stored. In problems with many loads that can be solved simultaneously (e.g. seismic imaging), the cost of repeated factorization can be amortized over loads.

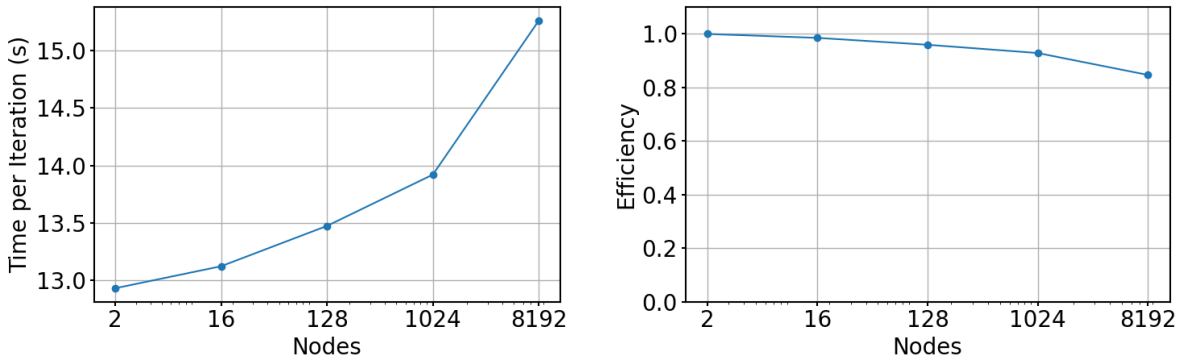


Figure 5.15: Weak scaling of acoustic simulation on the GO_3D_OBS model with a uniform hexahedral mesh. Patches were recomputed during the iteration.

Memory scaling for this problem is shown in Fig. 5.15, where it can be seen that the solver achieves near perfect linear memory scaling. Linear memory scaling is achieved by leveraging fully distributed data structures and use of the PCG iteration, which does not require storing a Krylov history. Note that in this case we only use ca. 50% of the total available memory per node. We could likely solve a larger problem (ca. 1.6 trillion DOFs) in the future but had limited attempts to run at this scale. Storing patches would have required ca. 270 GB per node and was thus not feasible for this problem.

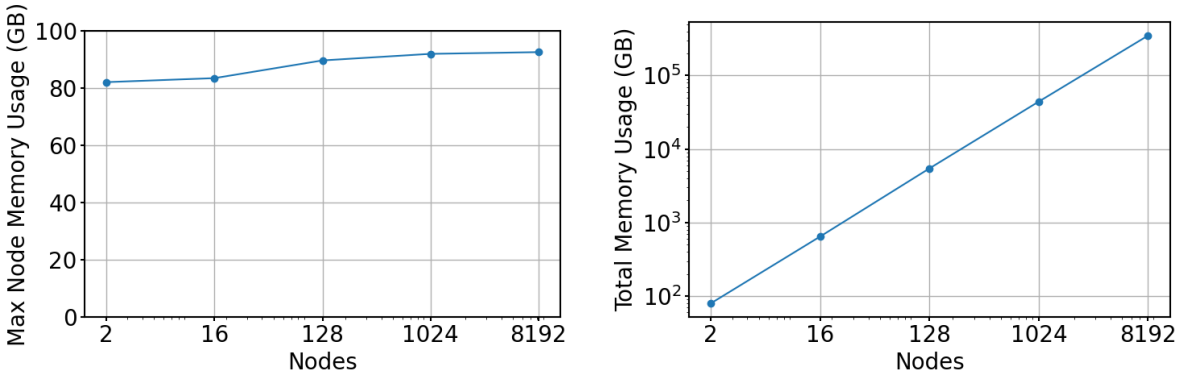


Figure 5.16: Memory scaling for acoustic simulation on the GO_3D_OBS model with a uniform hexahedral mesh. Maximum memory usage over nodes (left) and total memory usage (right) demonstrate near linear memory scaling of the solver.

Chapter 6

Applications

The DPG-MG solver is unique in a number of aspects. First, it is applicable—without modification—to general DPG systems[1] and can thus be used to simulate a number of physical problems. Second, it is adaptive and flexible, supporting isotropic and anisotropic h- and p-refinements on unstructured hybrid meshes with elements of all shapes. Third, it is massively scalable. This chapter focuses on a number of applications designed to illustrate these properties. Section 6.1 details simulation of bent optical fibers, leveraging anisotropic h-adaptivity to adaptively resolve the portion of the signal that is lost from the fiber core. Section 6.2 details simulation of RF heating in a Tokamak device to illustrate simulation on general unstructured meshes. Section 6.3 is dedicated to simulation of a number of challenging seismic wave propagation models; both acoustic and elastic models are simulated, illustrating the benefit of hp-adaptivity in these contexts and demonstrating the extreme scalability of the solver for practical problems of interest.

The applications in this chapter are presented in chronological order, with the most-recent studies presented at the end, in Section 6.3. The chronology is important since results were produced over a number of years, with different versions of the code. The performant fully distributed code is still under development and some of the older results have not yet been ported to the new code. Timings are presented for the version of the code used at the time and are thus not comparable across applications since the structure and performance of the code has changed significantly in recent years. Recent scaling data was given in Section 5.6, and more recent studies will provide more relevant indicators of the performance of the DPG-MG solver.

6.1 Optical fiber bending

Fiber lasers operate using the principle of active gain amplification, where power is transferred from a pump field to a signal field via rare-earth doping of the fiber core. Power scaling of single-mode fiber lasers is limited by the onset of nonlinear effects such as stimulated Raman scattering (SRS) and stimulated Brillouin scattering (SBS) at high optical intensities [2]. The optical intensity can be decreased by increasing the mode diameter, thus higher power fibers can be realized by increasing the area of the fiber core. Such large mode area (LMA) fibers have higher gain-saturation characteristics but lower efficiency and permit propagation of additional high-order modes that can degrade signal coherence. Transverse mode

[1] discretized with the exact sequence energy spaces

instability (TMI), a nonlinear phenomenon in which energy is chaotically transferred between high-order modes [71], causes a severe degradation in beam quality and is a dominant barrier to power scaling of LMA fiber lasers.

Bending optical fibers facilitates energy leakage from core-guided modes. Losses due to bending are highly mode-dependent, thus bending can be used to preferentially filter high-order modes at the expense of lowering overall amplification efficiency. Bending has been observed to delay the onset of TMI [71, 22] thus an improved understanding of bend losses could help to improve power scaling of fiber lasers. The observed effect of bending on nonlinear effects and the ubiquity of bending in applications has lead to renewed interest in accurate modeling of bend losses, however this remains a challenging problem.

6.1.1 Bent Fiber Model

A high-fidelity optical fiber amplifier model was developed in the dissertation work of S. Nagaraj [93] and S. Henneking [58], in collaboration with J. Grosek and the Air Force Research Lab [94, 61, 62]. In this dissertation, we consider only a linear time-harmonic Maxwell fiber model (2.12) of a weakly guiding, continuous wave, single-clad, large mode area, step-index fiber, as illustrated in Fig. 6.1. We will simulate only the signal field, with frequency ω in a bent configuration. Magnetization of silica glass is assumed to be negligible, and the polarization will be assumed to be isotropic, implying that the background permeability and permittivity take the form:

$$\boldsymbol{\mu} = I, \qquad \boldsymbol{\varepsilon} = n^2(x)I$$

where $n : \Omega \to \mathbb{R}$ is the index of refraction and I is the 3×3 identity tensor.

A host of computational and analytic methods have been proposed to model bend losses in optical fibers [57, 85, 106]. Bending perturbs optical properties and induces birefringence, a separation in the propagation speed of differently polarized modes. Birefringence is a result of *geometric* effects due to the spatial deformation of the fiber and *photoelastic* effects, describing stress-induced perturbations to optical properties. These effects can be modeled by modifying the permittivity and permeability of the fiber, as will be described next.

Photoelasticity. The photoelastic effect [97] relates strains (e) to perturbations in the inverse relative permittivity ($\boldsymbol{\varepsilon}^{-1}$) through the photoelastic tensor P as

$$\Delta(\boldsymbol{\varepsilon}^{-1})_{ij} = P_{ijkl}e_{kl}. \tag{6.1}$$

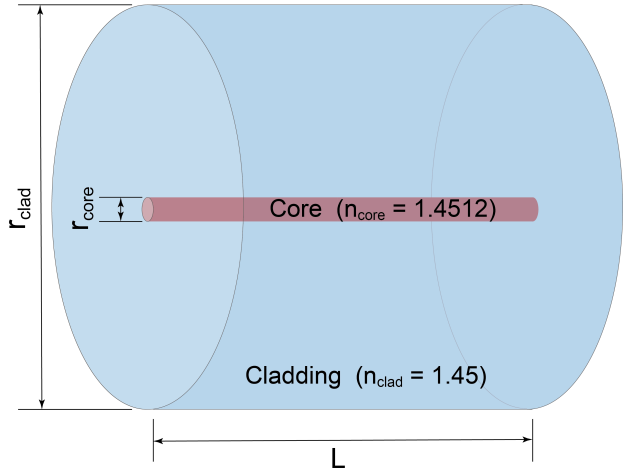


Figure 6.1: Optical fiber schematic.

Silica glass has an amorphous structure and is thus characterized as an isotropic crystal; the photoelastic tensor takes the form

$$P_{ijkl} = \begin{cases} p_{11} & \text{if } i = j = k = l \\ p_{12} & \text{if } i = j \neq k = l \\ \dfrac{1}{2}(p_{11} + p_{12}) & \text{otherwise} \end{cases} \tag{6.2}$$

for parameters p_{11} and p_{12}. These values were reported for silica glass in [55] as $p_{11} = 0.121$, $p_{12} = 0.270$.

Geometric birefringence. Spatial deformation of a waveguide changes the propagation guided fields even in the absence mechanical stresses. Pullback maps, introduced in Section 2.3 in the context of PML boundary conditions, can be used to quantify these purely geometric effects by defining equivalent straight fiber models. The use of pullback maps to transform fields defined on deformed spaces has only recently (ca. 2010) been adopted by the optics community under the name transform optics [20]. Pullback maps are a generalization of the commonly used conformal mapping techniques used to analyze 2D scalar optics models [57, 82]. Various transformed equations have been proposed to model bent optical fibers, but these transformed equations typically do not agree with those produced by the mathematically rigorous pullback maps.

Consider the deformation of an optical fiber from a straight domain $\hat{\Omega}$ to a bent domain Ω as illustrated in Fig. 6.2, given by deformation $\varphi : \hat{\Omega} \to \Omega$. Pullback maps can be employed to transfer fields between bent and straight domains. The same transformations introduced

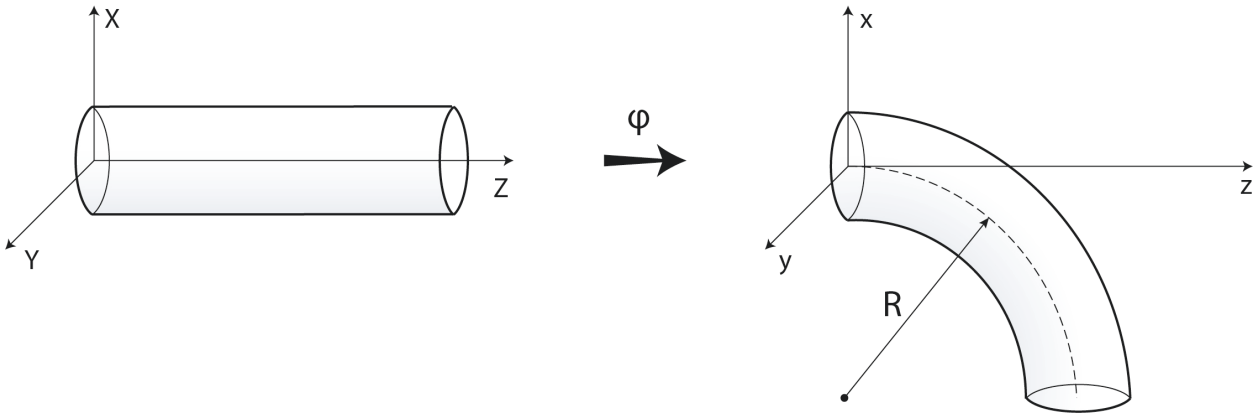


Figure 6.2: Deformation of an optical fiber to a bent configuration

in Section 2.3 can be used here, however in this case the real-valued deformation will result in real-valued modified permittivity and permeability:

$$\boldsymbol{\varepsilon}' := |\mathcal{J}|\mathcal{J}^{-1}\boldsymbol{\varepsilon}\mathcal{J}^{-\mathsf{T}}, \qquad \boldsymbol{\mu}' := |\mathcal{J}|\mathcal{J}^{-1}\mathcal{J}^{-\mathsf{T}}. \tag{6.3}$$

Note that although the magnetization of silica glass was assumed to be negligible, the equivalent straight fiber has non negligable magnetization due to the nontrivial relative permeability $\boldsymbol{\mu}'$. This magnetization is an artifact of computing on the curvilinear manifold defined by mapping physical space under φ^{-1} and is not physically realized in bent fibers.

Assuming the isotropic and divergence-free permittivity, i.e.

$$\boldsymbol{\varepsilon} = n^2 I, \qquad \nabla \cdot \boldsymbol{\varepsilon} = \mathbf{0},$$

the time-harmonic Maxwell (vectorial) model (2.12) can be reduced to an acoustic (scalar) model (2.9) with wavespeed $c = n^{-1}$. In this case, the modified material properties derived in Section 2.3 take the form:

$$n' = nj^{1/2}, \qquad \mathbf{I}' = j^{-1}\mathcal{J}^{\mathsf{T}}\mathcal{J},$$

In two-dimensions, straight configurations can be defined such that φ is a conformal map; in this case $\mathbf{I}$ remains identity by the Cauchy-Riemann equations. This conformally mapped scalar model was proposed in [57] and has formed the basis of much of the bending literature including the work of Schermer and Cole [106]; one of the most widely used bend loss models. However, in three-dimensions there is no generalization of the conformal mapping and the diffusion tensor $\mathbf{I}$ is generally nontrivial. Perhaps more challenging however, is that straight-fiber models introduce a radially dependent index of refraction such that $n \to \infty$ as $r \to \infty$.

This complicates analysis and even the definition of PML and other radiative-type boundary conditions. Some of these challenges will be noted in Section 6.1.2.

Bent configurations. Bent configurations can be derived from the theory of elasticity. Consider again the deformation $\varphi : \hat{\Omega} \to \Omega$ mapping between reference and current configurations. The deformation gradient is simply the Jacobian used to define pullback maps; we will use the symbol $\mathcal{F}$ in the context of elasticity to conform to classical notation:

$$\mathcal{F}(\mathbf{X}) := \mathcal{J} = \frac{\partial \phi}{\partial \mathbf{X}_j}, \tag{6.4}$$

where $\mathbf{X} \in \hat{\Omega}$ is used to denote coordinates in reference the reference configuration. The Green-St. Venant strain tensor is then:

$$\mathcal{E}(\mathbf{X}) := \frac{1}{2}(\mathcal{F}^T\mathcal{F} - I) = \frac{1}{2}\big[\nabla\mathbf{u} + (\nabla\mathbf{u})^\mathsf{T} + (\nabla\mathbf{u})^\mathsf{T}\nabla\mathbf{u}\big] \tag{6.5}$$

where $\mathbf{u}(\mathbf{X}) := \phi(\mathbf{X}) - \mathbf{X}$ is the displacement field. In the case of thin beam bending, strains are small and the linear constitutive relation (Hooke's law) is given in terms of Young's modulus E and Poisson ratio ν:

$$\mathcal{E}_{ij} = \frac{1}{E}\big[(1+\nu)\sigma_{ij} - \nu\sigma_{kk}\delta_{ij}\big]. \tag{6.6}$$

Deformations due to pure bending are classically derived by assuming the form of the stress tensor as

$$\boldsymbol{\sigma} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \dfrac{E}{R}X \end{bmatrix}.$$

In the case of small deformations, the nonlinear term in (6.5) is frequently neglected, reducing to (2.5b). Use of the linearized strain leads to a parabolic profile and is unsuitable for the large deformations encountered in fiber bending.

An alternate approach is to assume the form of the displacements directly then compute the corresponding stresses in the beam. Assuming the simple longitudinal deformation of a fiber into a circular profile as

$$\mathbf{x} = \varphi(\mathbf{X}) = \begin{bmatrix} (X+R)\cos\left(\dfrac{Z}{R}\right) - R \\ Y \\ (X+R)\sin\left(\dfrac{Z}{R}\right) \end{bmatrix}. \tag{6.7}$$

Stress can be recovered from Hooke's law as

$$\boldsymbol{\sigma} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} \nu\dfrac{X}{R} & 0 & 0 \\ 0 & \nu\dfrac{X}{R} & 0 \\ 0 & 0 & (1-\nu)\dfrac{X}{R} \end{bmatrix}. \tag{6.8}$$

Note however that when $\nu \neq 0$ (indeed, $\nu = 0.17$ in silica glass) the normal stress on the outer surface of the fiber does not vanish. The deformation given by (6.7) thus only represents a pure bending state when $\nu = 0$; otherwise, for a fiber with radius a, the radial traction boundary conditions are violated with error $\mathcal{O}(\nu a/R)$. More accurate deformations can be derived based on the theory of inflexed beams [40].

6.1.2 Spectral Analysis of Open Waveguides

This section takes a brief detour to review spectral theory for waveguide structures and is intended to motivate some of the challenges in modeling, and indeed even rigorously analyzing bent optical fibers. Waveguide structures are commonly assumed to be homogeneous in the longitudinal direction and naturally give rise to eigenvalue problems defined in the transverse plane. The modes obtained by solving the transverse eigenvalue problem are fundamental to waveguide design and form the basis for coupled mode theory (CMT) [119] commonly used to simulate complex and nonlinear waveguide structures.

Consider an open waveguide structure defined on the half-space $\Omega = \{\mathbb{R}^3 : z \geq 0\}$. We assume the waveguide geometry and material are longitudinally homogeneous and make the right-propagating ansatz:

$$\mathfrak{u}(x, y, z) = \mathfrak{u}(x, y)e^{-i\beta z} \tag{6.9}$$

Vectorial transverse eigenproblem. Let $\mathsf{E} = (\mathbf{E}, E_3)$, where $\mathbf{E}$ denotes transverse components and E_3 denotes longitudinal components. Similarly assume the permittivity $\boldsymbol{\varepsilon}$ can be separated into transverse ($\boldsymbol{\epsilon} : \underline{\Omega} \to \mathbb{R}^{2x2}$) and longitudinal contributions ($\epsilon_3 : \underline{\Omega} \to \mathbb{R}$). Substituting (6.9) into the time-harmonic Maxwell system (2.12) and assuming longitudinal homogeneity gives the following generalized transverse polynomial eigenproblem [116]:

$$\begin{cases} \nabla \times \big(\text{curl } \mathbf{E}\big) - \omega^2 \boldsymbol{\epsilon}\mathbf{E} + \beta^2 \mathbf{E} - j\beta \nabla E_3 = 0, \\ \nabla \cdot \nabla E_3 + \omega^2 \epsilon_3 E_3 + j\beta \nabla \cdot \mathbf{E} = 0 \end{cases} \tag{6.10}$$

where $\nabla\times := (\partial_2, -\partial_1)$ denotes the scalar-to-vector curl operator, and $\text{curl} := \partial_1(\cdot)_2 - \partial_2(\cdot)_1$ denotes the vector-to-scalar curl. As shown in [79], this system can be rescaled to yield a classical eigenproblem where the linear operator $A : \left\{\mathsf{E} \in \boldsymbol{L}^2(\underline{\Omega}) : A(\mathsf{E}) \in \boldsymbol{L}^2(\underline{\Omega})\right\} \to \boldsymbol{L}^2(\underline{\Omega})$ is self-adjoint.

Scalar transverse eigenproblem. The acoustic model leads to the scalar transverse eigenproblem:

$$\Delta u + k^2 n^2 u = \beta^2 u, \tag{6.11}$$

where $k^2 := \omega^2 - \beta^2$. Let $n \in \mathbb{R}$, then linear operator $A := \Delta + k^2 n^2$ defined on domain

$$D(A) = \left\{u \in L^2(\mathbb{R}^2) : A(u) \in L^2(\mathbb{R}^2)\right\}$$

is self-adjoint.

Theory of self-adjoint operators on unbounded domains. Spectral theory requires that the spectrum of *essentially* self-adjoint operators defined on an unbounded domain $\underline{\Omega}$ consist of only discrete and continuous spectrum [35]. The discrete and continuous spectrum are complete in $L^2(\underline{\Omega})$ with representation

$$\mathfrak{u}(\mathbf{x}) = \sum_{i=0}^{N} e_i(\mathbf{x}) \int_{\underline{\Omega}} \overline{e_i(\mathbf{x})}\mathfrak{u}(\mathbf{x})d\mathbf{x} + \int_{\kappa} e(\kappa;\mathbf{x}) \int_{\underline{\Omega}} \overline{e(\kappa;\mathbf{x})}\mathfrak{u}(\mathbf{x})d\mathbf{x}\, d\kappa \tag{6.12}$$

where e_i denote guided modes corresponding to the discrete spectrum $\{\beta_i\}_{i=1}^N$, and $e(\kappa;\cdot)$ represent the continuum of *radiation* or scattering modes corresponding to the continuous spectrum $\{\beta(\kappa) : \kappa \in \mathbb{R}\}$. Orthogonality of the guided modes corresponding to the discrete spectrum can be understood in the L^2 sense,

$$\int_{\underline{\Omega}} e_i(\mathbf{x})\overline{e_j(\mathbf{x})}d\mathbf{x} = \delta_{ij} \tag{6.13}$$

Radiation modes are not in $L^2(\underline{\Omega})$ and their orthogonality must be understood in the distributional sense

$$\int_{\underline{\Omega}} e(\kappa,\mathbf{x})\overline{e(\kappa',\mathbf{x})}d\mathbf{x} = \delta(\kappa - \kappa'); \tag{6.14}$$

i.e. for any test function $\varphi \in \mathcal{D}(\underline{\Omega})$,

$$\int_{\underline{\Omega}}\int_{\underline{\Omega}} e(\kappa,\mathbf{x})\overline{e(\kappa',\mathbf{x})}\varphi(\kappa')d\kappa' d\mathbf{x} = \varphi(\kappa).$$

Leaky modes. Lossy waveguides (e.g. W-waveguides [68], microstructure waveguides [48]) have no discrete spectrum; thus any signal propagated in the waveguide will experience radiative losses. Radiation modes define a *spectral* transform $\mathcal{S} : L^2(\underline{\Omega}) \mapsto L^2(\underline{\Omega})$ given by

$$\hat{\mathfrak{u}}(\kappa) := \mathcal{S}(\mathfrak{u})(\kappa) = \int_{\underline{\Omega}} \mathfrak{u}(\mathbf{x})\overline{e(\kappa, \mathbf{x})}d\mathbf{x}. \tag{6.15}$$

This spectral transform can be used to represent a solution to the waveguide problem on half-space Ω with input $\mathfrak{u}_0(\mathbf{x}) = \mathfrak{u}(\mathbf{x}, 0)$ as

$$\mathfrak{u}(\mathbf{x}, z) = C_N \int_{\mathbb{R}} \hat{\mathfrak{u}}_0(\kappa) e^{i\beta\kappa z} e(\kappa, \mathbf{x}) d\kappa. \tag{6.16}$$

where C_N denotes a constant scaling factor. An analogous representation holds for the vectorial problem (6.10).

The complex extension of the spectral representation is a meromorphic function with poles in the upper half plane $\mathbb{R} \otimes i\mathbb{R}^+$ [86, 80]. The modes corresponding to these poles are called *leaky* modes. By the Residue Theorem, we can express the solution u as a discrete sum of residues corresponding to the poles,

$$\begin{aligned} \mathfrak{u}(\mathbf{x}, z) &= C_N \int_{\mathbb{R}} \hat{\mathfrak{u}}_0(\kappa) e^{i\beta(\kappa)z} e(\kappa, \mathbf{x}) d\kappa \\ &= \sum_{i=0}^{\infty} \operatorname{Res}\left[\bar{\mathfrak{u}}_0(\cdot) E(\cdot, \mathbf{x}), s_i\right] e^{i\beta(s_i)z} \end{aligned} \tag{6.17}$$

where $\bar{\cdot}$ and E represent the complex extension of the spectral transform $\hat{\cdot}$ and radiation mode e respectively. Leaky modes are not individually L^2 integrable, are not orthogonal, and grow exponentially in the transverse direction; however their discrete representation has lead to widespread adoption. Appendix A shows that leaky modes in a simple W-waveguide can be equivalently derived by enforcing a radiation condition on modes, i.e. requiring the modes to be outgoing only. The equivalence of leaky modes derived by meromorphic extension or by radiation condition for more general problems can be established by representing the radiation condition via complex stretching. This implies that mode solvers enforcing radiation boundary conditions will compute leaky modes.

Bent waveguides and non-self-adjoint operators. We briefly consider the simplest possible bent waveguide structure: a 2D acoustic slab waveguide. Formulation of the problem

in polar coordinates and subsequent separation of variables with separation constant β^2 leads to a Bessel problem in the radial coordinate:

$$r\frac{\partial}{\partial r}\left(r\frac{\partial \phi}{\partial r}\right) + \left(k_0^2 n(r)^2 r^2 - \beta^2\right)\phi = 0, \qquad r \in (0,\infty). \tag{6.18}$$

Note the similarity of this problem to the problem defining the Hankel transform; in that case n is constant and $n^2k_0^2$ takes the role of the separation constant for fixed β. The resulting operator in that case is self-adjoint in $L^2((0,\infty), r)$ and thus generates a spectral transform, i.e. the Hankel transform. We would thus expect a similar result in the bent waveguide problem. Indeed, the operator is symmetric in the weighted space $L^2((0,\infty), 1/r)$; it is not however *essentially* self-adjoint. Indeed, for constant n, this problem permits solutions of the form

$$\phi_\beta(r) = cJ_\beta(k_0 nr) + dY_\beta(k_0 nr) \tag{6.19}$$

where J_ν, Y_ν represent the ν-th order Bessel functions of first and second kind respectively; it can be verified that modes are *not* orthogonal in $L^2((0,\infty), 1/r)$. The Bessel function of the first kind further has finite energy, i.e. $\|J_\beta\|_{1/r} < \infty$. This is incompatible with the notion of generalized eigenfunctions (i.e. modes corresponding to the continuous spectrum) under the spectral theory for self-adjoint operators.

Spectral theory for non-self-adjoint operators is a mathematically formidable subject, we do not attempt to produce any new insights in this regard but note a few surprising results. First, note that the well-posedness of even the simplest bent slab waveguide problem has not been established. Second, Hiremath et al. [67] observed that 'leaky' modes derived for the bent slab waveguide problem by enforcing a radiation condition satisfy an orthogonality relationship in the indefinite inner product

$$[u, v]_{1/r} = \int_0^\infty \frac{uv}{r} dr, \tag{6.20}$$

which differs from the $L^2((0,\infty), 1/r)$ inner product only by the lack of the complex conjugate on the second argument. This structure seems to suggest an analysis via Krein spaces [77], although we are not aware of any existing work on the application of Krein spaces to bent waveguides.

6.1.3 Adaptive Simulation

The preceding mathematical challenges illustrate the lack of rigorous understanding of bent waveguides and appropriate boundary conditions. We had intended to use large scale three-dimensional simulations to investigate the effect of various boundary conditions on bend losses. This section outlines initial simulations in this direction, however we quickly discovered that the replicated version of *hp*3D and corresponding version of the solver were not sufficiently scalable to simulate meaningful lengths of fiber. Some work is still required to port the fiber models to the fully distributed solver, we have thus not yet had occasion to run these problems at a larger scale. These initial simulations are intended to demonstrate the applicability of adaptive mesh refinement in this context and provide some insight into the scale needed for high-fidelity simulation of bent optical fibers.

We will consider a simple step-index fiber with core radius $r_{\text{core}} = \sqrt{162}$ μm and cladding radius $r_{\text{clad}} = \sqrt{16200}$ μm. The index of refraction in core and cladding was taken to be $n_{\text{core}} = 1.4512$ and $n_{\text{clad}} = 1.45$, respectively. We consider propagation of a signal field at wavelength $\lambda = 1024$ nm, this configuration corresponds to the Yitterbium-doped optical fiber amplifier simulated in [58].

A fiber length of $L = 384$ μm, corresponding to 512 fundamental mode wavelengths, is adopted. The simple circular deformation (6.7) with a bend radius of 3 mm was used to define the bent configuration, with 20 μm straight sections appended at the beginning and end of the fiber to inject the straight-fiber fundamental mode at the entrance and implement a PML at the exit. Perfectly electric conducting (PEC) boundary conditions were assumed on all boundaries except the exit where with a simple longitudinal PML $\beta = 3$, $C = 25$ was implemented in the final 20 μm straight section. Note that the bend radius is exaggerated by 1–2 orders of magnitude relative to practice and we have neglected to define an appropriate radiation condition on the transverse surface in these initial simulations. A transverse cross-section of the initial mesh is depicted in Fig. 6.3; this mesh was extruded by 512 elements in the longitudinal direction.

Simulations were performed on 256 *Frontera* CLX nodes (14336 cores). The DPG-MG solver was used with a stopping criterion of 10^{-8} relative ℓ^2 residual. A simple V$(1,1)$ multigrid cycle was used with a coarse-grid correction produced via a sparse direct solver on the initial mesh. The initial fourth-order mesh ($p = 4$) was longitudinally refined twice to define meshes 2 and 3; subsequent meshes were defined via anisotropic (in the transverse

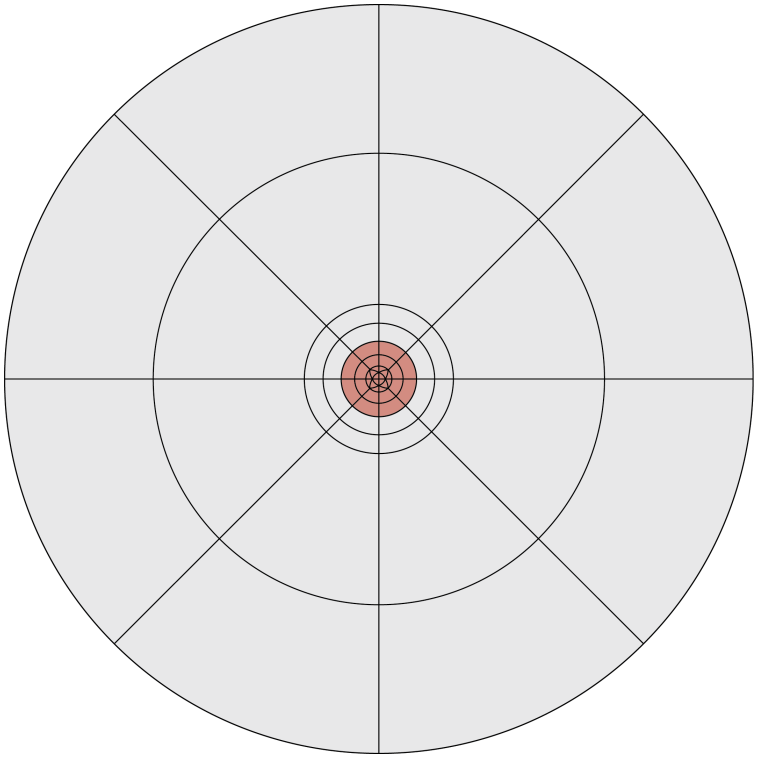


Figure 6.3: Transverse cross-section of initial mesh. Mesh is composed of hexahedral and prismatic elements (near center). The fiber core is marked in red and has radius $r_{\text{core}} = \sqrt{162}\ \mu$m; the radius of the cladding $r_{\text{clad}} = \sqrt{16200}\ \mu$m.

plane) h-adaptive mesh refinement using a Dörfler marking strategy [34] and the DPG error indicator funciton. Cross-sections of the solution ($|\mathbf{E}_x|$) are depicted in Fig. 6.4, where values are plotted only at element second-order Lagrange nodes to enable visualization of the refined mesh. Iteration and timing data are provided in Table 6.1.

It can be observed in Fig. 6.4 that even in the presence of the extreme bend radius, the majority of the signal leaves the fiber core but does not reach the boundary. Adaptive mesh refinement was able to resolve the signal leaving the core. However, in the case of less extreme bends, losses often have several orders of magnitude smaller amplitude than core-guided modes. Indicator-based adaptivity may struggle to mark and resolve the losses unless the core-guided modes are accurately resolved and the solution is computed to sufficient accuracy. Indeed, the relatively small magnitude of bend losses further implies that a highly accurate solution would likely be needed to accurately predict loss coefficients, especially since the DPG method primarily exhibits dissipative pollution error. The relatively small tolerance (10^{-8}) used here was chosen to reflect the need for high-accuracy solutions to

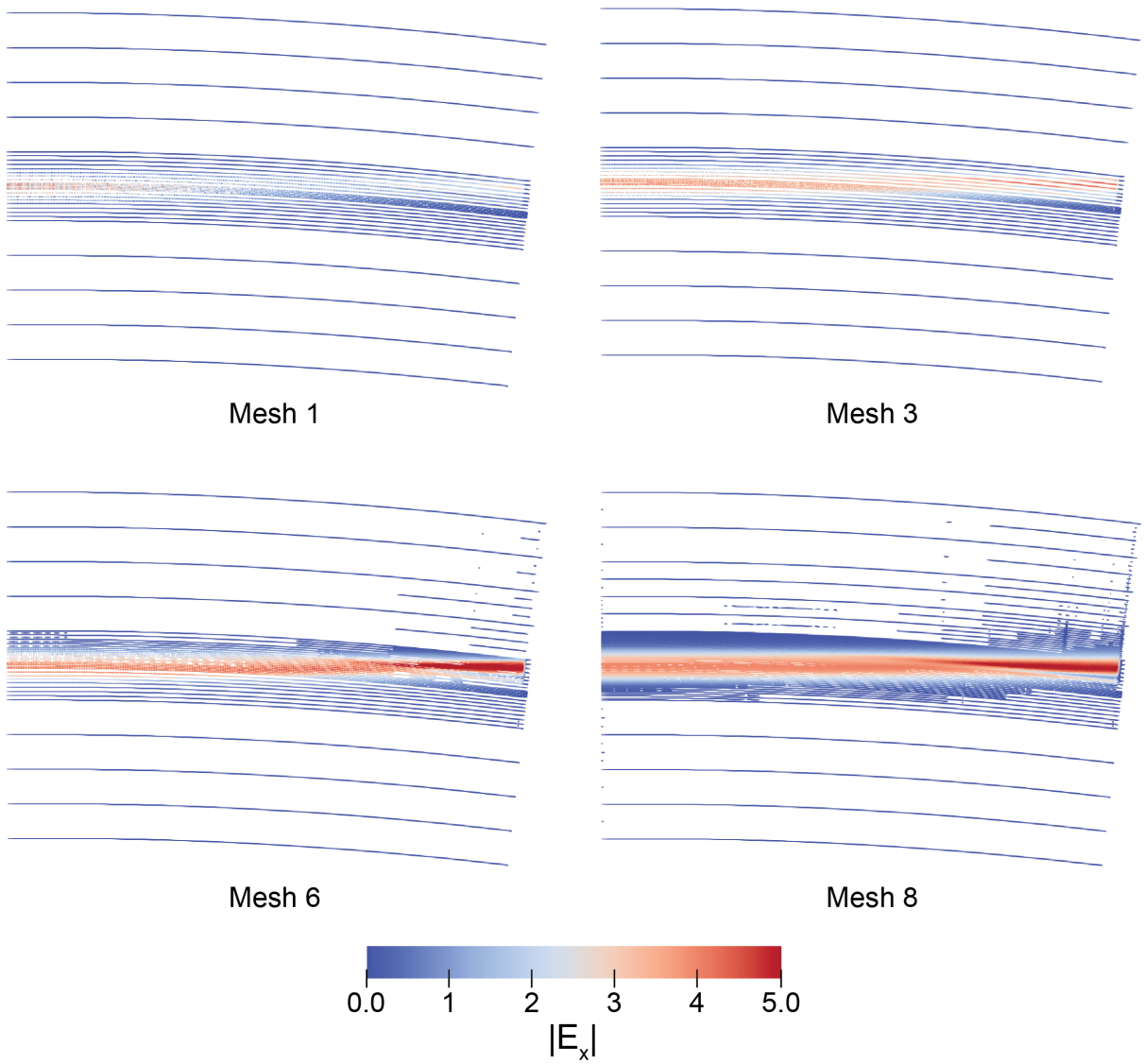


Figure 6.4: Adaptive simulation of bend loss in an optical fiber with band radius $R = 3$ mm. The magnitude of E_x is plotted at second-order Lagrange nodes (vertices and middle of edges, faces, elements) along a longitudinal slice ($y = 0$).

accurately model bend losses. As can be seen in Table 6.1, this requires a large number of iterations and is thus relatively expensive.

Future directions. Although preliminary simulations revealed a number of challenges with this approach, high-fidelity three-dimensional bend loss simulations may be feasible with a number of further developments. First, a longitudinal wave ansatz could likely be used to ease longitudinal discretization requirements. Such an approach leverages the com-

Table 6.1: Timing and convergence history for adaptive simulation of bent fiber on 256 *Frontera* CLX nodes (14336 cores).

Grid	DOFs	Iterations	Solve Time
1	29 401 120	1042	21.7
2	54 323 240	1123	56.5
3	108 646 440	1678	172.1
4	122 188 170	1701	256.6
5	161 432 655	1671	599.8
6	247 612 480	1789	762.1
7	400 902 405	1756	1201.0
8	563 642 835	1871	1851.1

mon direction of propagation of fields along the optical fibers, computing a less-oscillatory envelope around the common wave-vector component. The original wavefield can then be computed by reintroducing the ansatz into the envelope solution. This approach has been successfully deployed in the high-fidelity straight fiber models developed in [58] to simulate TMI on meter-long fibers [63]. While the envelope approach is likely to be substantially less effective in the case of bending, when combined with the significantly improved scaling and performance of newer versions of the DPG-MG solver, it may be possible to simulate 10–100× longer fibers, enabling more realistic and relevant studies, including investigation of appropriate boundary conditions and comparison with existing bend loss models.

6.2 Radio-Frequency Heating in a Tokamak

Nuclear fusion is a process in which two atomic nuclei (commonly hydrogen isotopes) combine to form a heavier nucleus (commonly hydrogen isotopes) releasing a significant amount of energy in the process. Nuclear fusion has the potential to generate vast amounts of clean and sustainable energy, but achieving and controlling nuclear fusion on Earth is challenging due to the extremely high temperatures and pressure required to initiate and maintain the reaction. Various concepts including tokamaks devices (e.g. Joint European Torus (JET) and International Thermonuclear Experimental Reactor (ITER)) and laser fusion facilities (e.g. National Ignition Facility (NIF)), are being explored for this purpose.

Tokamaks are large, toroidal shaped devices used to contain and heat plasmas for nuclear

fusion research. During fusion experiments, plasmas are heated to millions of degrees Celsius. Strong magnets arranged along the exterior of the torus are thus used to magnetically confine the plasma to the interior of the device, away from walls and other equipment. Heating is achieved via radio-frequency (RF) heating, directing RF electromagnetic waves into the plasma where they are absorbed by particles. This section details preliminary simulations of RF heating in a tokamak device; we thank S. Shiraiwa and J. Hillairet for the geometrical information of the simulation domain[2].

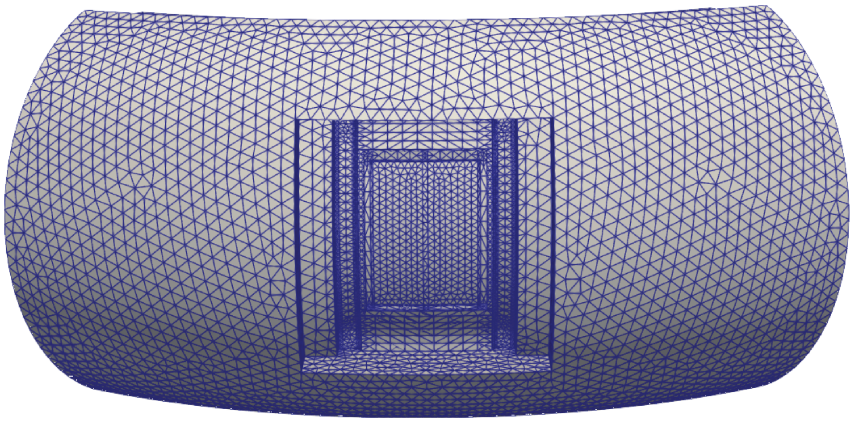


Figure 6.5: Initial tokamak mesh, composed of ca. 300 000 tetrahedral elements

Problem Setup. The initial mesh is shown in Fig. 6.5 and was composed of ca. 300 000 linear ($p = 1$) tetrahedral mesh elements. Perfect electric conductor (PEC) boundary conditions were assumed on all boundaries, with the z-component of the boundary data specified (the remaining components are homogeneous) as shown in Fig. 6.6. The material was assumed to be isotropic and homogeneous with index of refraction $n = 1$ and frequency $\omega = 100$ was adopted. These preliminary simulations were only intended to demonstrate the applicability of the DPG-MG solver to this problem; more realistic simulations including the plasma would employ highly heterogeneous and anisotropic material data.

The first mesh was produced via a single uniform p-refinement of the initial mesh. Mesh 2 and mesh 3 were defined by uniform h-refinement, and the remaining three meshes were defined via uniform p-refinement. The final mesh was composed of ca. 19 million fifth-order ($p = 5$) tetrahedral elements, supporting 6.7 billion unknowns. The MUMPS direct solver was employed for the initial solver and coarse-grid correction. Two pre- and post-smoothing

[2]Fig. 9 of S. Shiraiwa et al 2023 Nucl. Fusion, 63, 026024

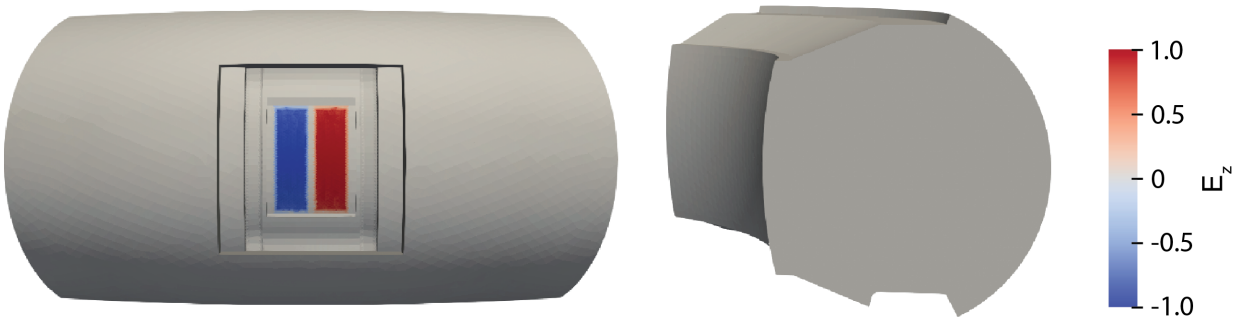


Figure 6.6: z-component of the PEC boundary data ($\mathbf{E}_0$).

steps were employed and the iteration was terminated at 10^{-4} relative residual. Test norm weight $\alpha = 10^4$ was used.

The magnitude of the z-component of the simulated electric field (E_z) is illustrated in Fig. 6.7; timing and convergence data are given in Table 6.2. The number of iterations required for convergence was relatively small; this is largely due to the large test norm weight α which, as noted in Section 4.3, tends to improve convergence but increases pollution error.

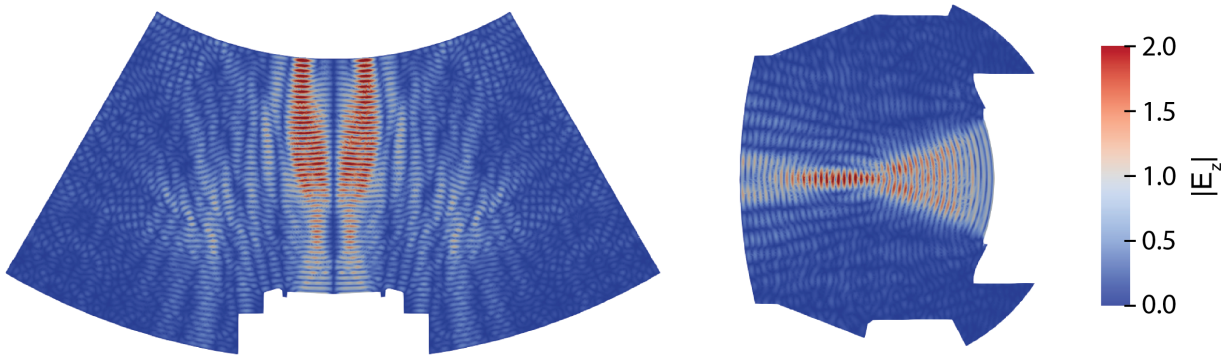


Figure 6.7: Simulated electric field in the tokamak device. Slices of the z-component of the electric field on the final mesh at $z = 0$ (left) and $x = 0$ (right)

Adaptivity studies. Adaptive mesh refinement is likely required to accurately resolve singularities induced by the large number of reentrant corners and irregular features present in the tokamak geometry. Adaptivity is also expected to be beneficial in realistic simulations with material heterogeneity and anisotropy.

Table 6.2: Timing and convergence data for Tokamak simulation on 512 *Frontera* nodes CLX nodes (24 576 cores).

Grid	DOFs	Iterations	Solve Time (s)
1	19 993 630	28	0.4
2	159 976 860	17	1.4
3	703 920 440	23	7.0
4	1 747 020 756	37	33.5
5	3 404 467 824	33	91.5
6	6 704 649 952	24	196.2

Preliminary adaptivity studies were attempted on this problem but were largely unsuccessful. The difficulty in enabling adaptivity was that the mesh contained nearly singular elements. Two of these elements are shown in Fig. 6.8 (marked in yellow) and have edge length $\mathcal{O}(10^{-1}$ m) and volume $\mathcal{O}(10^{-9}$ m$^3)$. These near-singular elements necessitated use of a large test norm weight to enable inversion of the element Gram matrix. The singular elements were produced at interfaces where the size of mesh elements changes abruptly, these elements were thus often adjacent to large elements. Near-singular elements induced errors in the neighboring larger elements, causing those elements to be marked first for refinement (the small volume of the near-singular elements typically prevented them from being marked directly). Repeated refinement of the large elements subsequently necessitated refinements in the near-singular elements to maintain a one-irregular mesh. The resulting repeated refinement of near-singular elements caused them to become numerically singular even when employing large test norm weights.

Future directions. These preliminary results demonstrate the applicability of the DPG-MG solver to problems with unstructured meshes and complex geometries. Substantial further work is required to produce useful insights into tokamak control and design. Use of a more suitable mesh would enable refinement studies and permit use of a smaller test norm weight, mitigating pollution error and significantly reducing the number of unknowns required for a given accuracy. A curvilinear mesh would also likely be beneficial for adaptivity as it would avoid localized sharp features induced by the triangulation of the coarse-mesh boundary. Heterogeneous and highly anisotropic material data was made available to us but

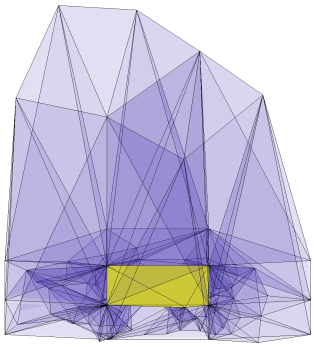


Figure 6.8: Tokamak mesh elements in the neighborhood of two nearly singular tetrahedral elements (yellow) with edge length $\mathcal{O}(10^{-1})$ and volume $\mathcal{O}(10^{-9})$.

tended to exacerbate challenges related to the numerical singularity of the element Gram matrix on small aspect-ratio elements. The abrupt changes in element size further tended to produce vertices shared by up to nearly 100 mesh elements, producing a number of very large smoothing patches that were expensive to factor and apply and contributed significantly to load imbalance. A more regular mesh would thus also reduce the cost of the DPG-MG solver significantly. When the normal-system was able to be formed, the DPG-MG solver demonstrated satisfactory convergence even in the case of highly anisotropic material data, but use of very large test norm weights in these cases likely obscured true convergence trends.

6.3 Seismic Simulation

6.3.1 Visco-Acoustic Simulation on the `GO_3D_OBS` model

This section outlines acoustic simulation on the `GO_3D_OBS` model [53]. We thank Andrzej Górszczyk of the Institute of Geophysics at the Polish Academy of Sciences for kindly sharing the model with us, it proved to be a valuable tool in helping to benchmark and develop the DPG-MG solver and in investigating adaptivity strategies.

Model. The `GO_3D_OBS` model is an open-source multiparameter seismic model with high-contrast heterogeneous structures representing a subduction zone, inspired by the geology of the Nankai Trough. The model provides isotropic visco-elastic properties (V_p, V_s, ρ, Q_p, Q_s). In the following we used only the compressive wavespeed V_p (c in the acoustic model) and quality factor Q_p. These are shown on a 100 km $\times$ 100 km $\times$ 30 km section

of the GO_3D_OBS model in Fig. 6.9. Constant density was assumed. Indeed, recall that the derivation of the linear acoustic equations in Section 2.1.1 performed linearization about a constant density state. Linearization around a variable density state [6] leads to models with so-called fractional density ($\nabla\rho/\rho$), although this term is often neglected leading to the classical variable density Helmholtz problem.

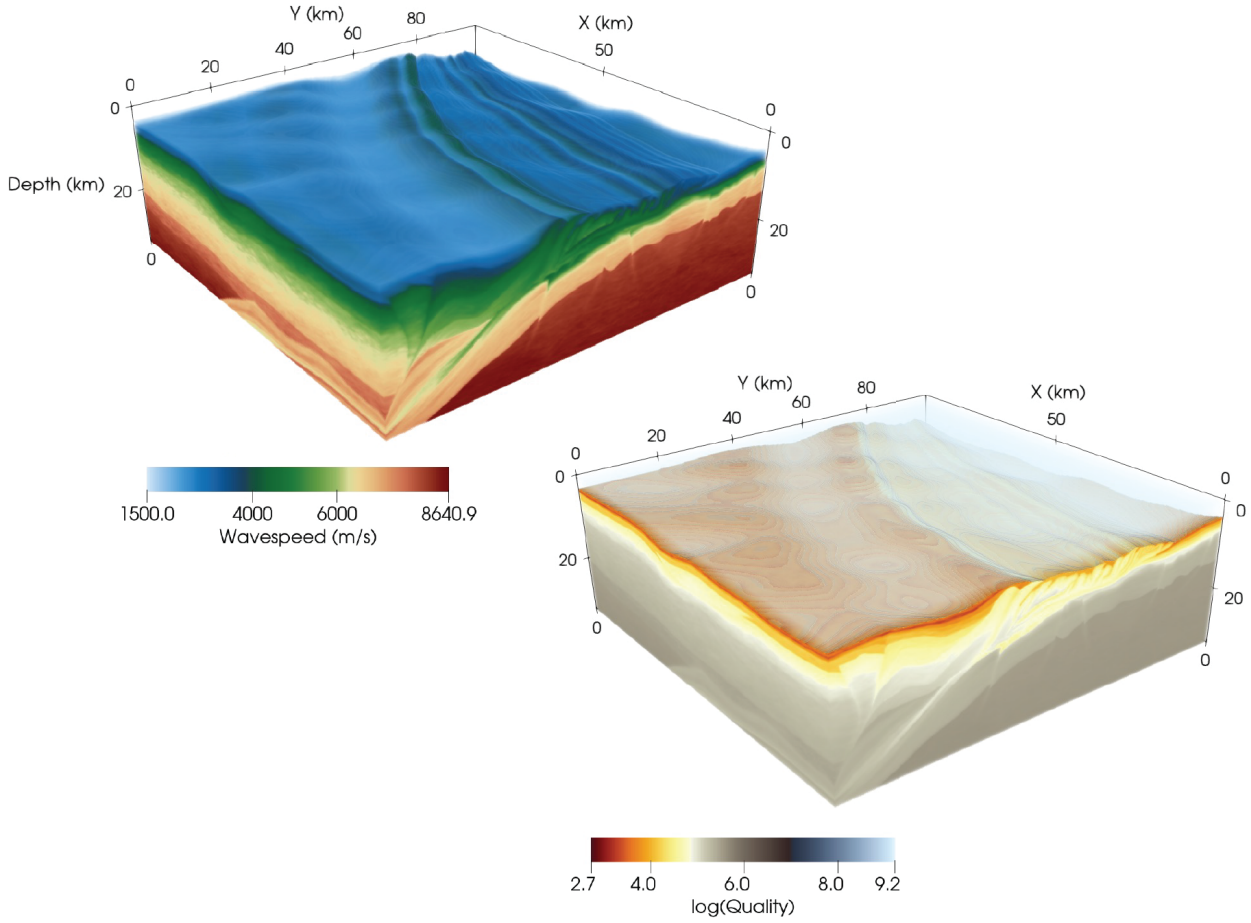


Figure 6.9: Wavespeed c (or V_p) (left) and quality factor Q_p (right) for a section of the GO_3D_OBS model. The model is hexahedral, water was set to be transparent to reveal the bathymetry.

Material properties for the GO_3D_OBS model are defined on a 25 m uniform grid. Throughout the following examples the full-resolution seismic model is read and stored separately from the finite element mesh. This implies that material properties are evaluated using the full-resolution model (not interpolated on the finite-element mesh) and will introduce some error into the numerical integration of element matrices. To counter this error, the quadrature order was increased by two over the enriched test space order ($p_r = p + 1$ throughout

this work). Thus, for a fourth order ($p = 4$) discretization, a total of 8^3 quadrature points are used for numerical integration.

Attenuation is parameterized by the quality factor (Q_p) which, in the frequency domain, defines a complex-valued wavespeed [3]. For simplicity, we assume frequency-independent attenuation leading to the complex-valued wavespeed:

$$\widetilde{c}(\mathbf{x}) = c(\mathbf{x}) \left(1 + i\mathrm{sgn}(\omega)\frac{1}{2Q_p(\mathbf{x})}\right),$$

although frequency dependence would be rather trivial to implement in the frequency domain. Impedance parameter $d(\mathbf{x}) = c^{-1}(\mathbf{x})$ was adopted at impedance boundaries for the remainder of this work.

Indicator-based adaptive simulation. Building on our experience in previous applications, we attempted to apply indicator-based adaptivity to seismic simulation. Following [115], these initial studies considered a 20 km × 102 km × 28.3 km section of the model. That particular section largely overlaps with the section depicted in Fig. 6.9, but is not depicted here. Material data was downsampled to 100 m spacing for these initial indicator-based simulations. The following computations were performed on 512 *Frontera* compute nodes (28 672 cores).

The frequency was taken to be 3.75 Hz and the problem was loaded with a pressure point source (f_p) at $(10.0, 12.5, 0.0)$ km, implemented as a tight Gaussian with standard deviation of 50 m. Impedance boundary conditions were adopted on all boundaries. The initial mesh consisted of $8 \times 42 \times 12$ hexahedral quadratic ($p = 2$) elements with a total of 205 757 DOFs. After solving the problem on a given mesh and evaluating the DPG error indicator, elements were marked for refinement using the Dörfler marking strategy [34]. The solver iteration was terminated at 10^{-7} relative ℓ^2 residual; we note however that the large number of h-refinements near the source tended to produce a large initial residual. The 10^{-7} tolerance used here is thus somewhat artificially small but is adopted in absence of a more appropriate termination criterion. A simple V$(1, 1)$ cycle was used without a coarse-grid correction on the initial mesh.

Seven initial h-adaptive refinements were used to resolve the region around the point source, followed by hp-adaptive refinements. The hp-adaptive strategy selected h-refinements until the maximum edge length of an element was less than one-half the minimum wavelength

over the element. In regions of high contrast, where the variation in wavespeed over the element was greater than 10% of the minimum wavespeed over the element, an additional h-refinement was used. Otherwise, p-refinements were selected until a maximum order of $p = 5$. A subset of the resulting hp-adaptive meshes are depicted in Fig 6.10. In total, the mesh is refined 34 times, resulting in a final mesh with over 6.3 million elements and 1.9 billion DOFs. The corresponding wavefields ($\Re\{p\}$) are depicted in Fig. 6.11.

Convergence and timings for indicator-based adaptive simulation of the `GO_3D_OBS` model are compared to a uniformly refined mesh with a minimum of two fourth-order ($p = 4$) elements per wavelength on the finest mesh in Fig. 6.12. The number of solver iterations on each grid is depicted in Fig. 6.12a; however, in contrast to the convergence studies in Section 5.4, the solution on the previous mesh is used to initialize the solution. The effect of initializing with previous solutions becomes apparent in later iterations, when the solution is reasonably well resolved in much of the domain and further refinements result in fairly localized perturbations to the solution. Fig. 6.12b shows that the DPG residual decreases early in the adaptive process, when compared to the hp-adaptive Gaussian beam problem in Fig. 5.12b; this is likely because the wave decays relatively quickly away from the point source.

The solution on the finest mesh was completed in 210 seconds, whereas the total runtime for the job (including forming and solving the system on 34 meshes) was 3 029 seconds (51 minutes). The time *per iteration* is depicted in Fig. 6.12d to illustrate scaling of the solver with indicator-based adaptive refinement. After an early preasymptotic regime, it can be seen that, in the adaptive case, the time per iteration scales roughly linearly with the size of the system; however, super-linear scaling is observed in later iterations. This super-linear scaling is due to the sub-geometric growth of adaptive meshes late in the adaptive process.

Comparing uniform and adaptive refinements reveals that the uniform refinement case required nearly 3× as many DOFs and only reaches a DPG residual ten times larger than for hp-adaptive refinements. This is however not as compelling as it may otherwise indicate. First, the DPG test norm was defined via the L^2 norm (i.e. the L^2 norm of the graph and adjoint terms). The error indicator is thus not amplitude compensated and the far-field is effectively weighted less than the near-field in simulation of point sources. The resulting error indicator is thus perhaps not an incredibly relevant metric for seismic simulation and imaging contexts where an accurate far-field is desired. Second, it can be seen in Fig. 6.12d

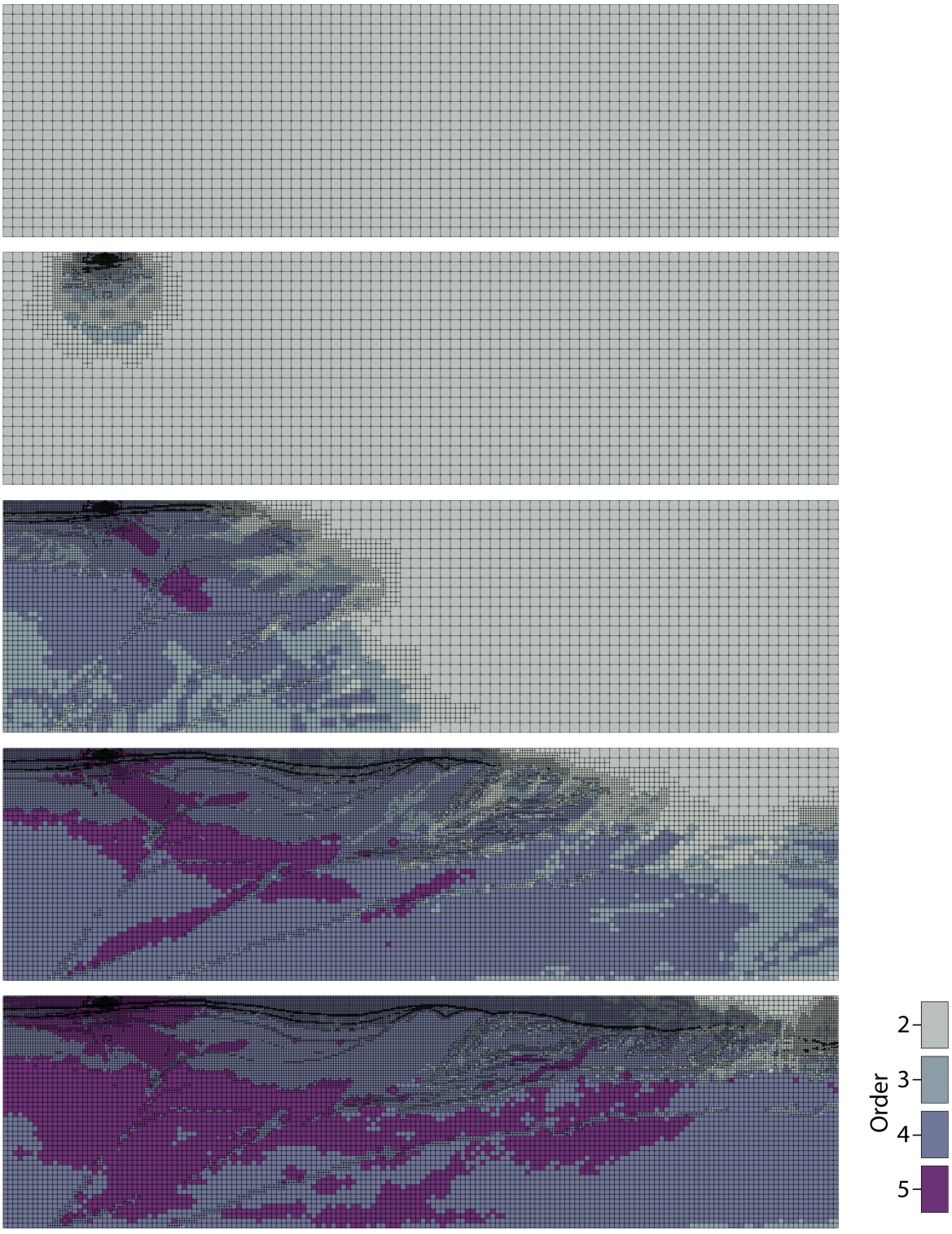


Figure 6.10: 2D cross-section of *hp*-adaptive meshes produced via indicator-based adaptive refinement of the `GO_3D_OBS` model. The depicted meshes are, from top to bottom, after 0, 9, 18, 26, and 33 adaptive mesh refinements.

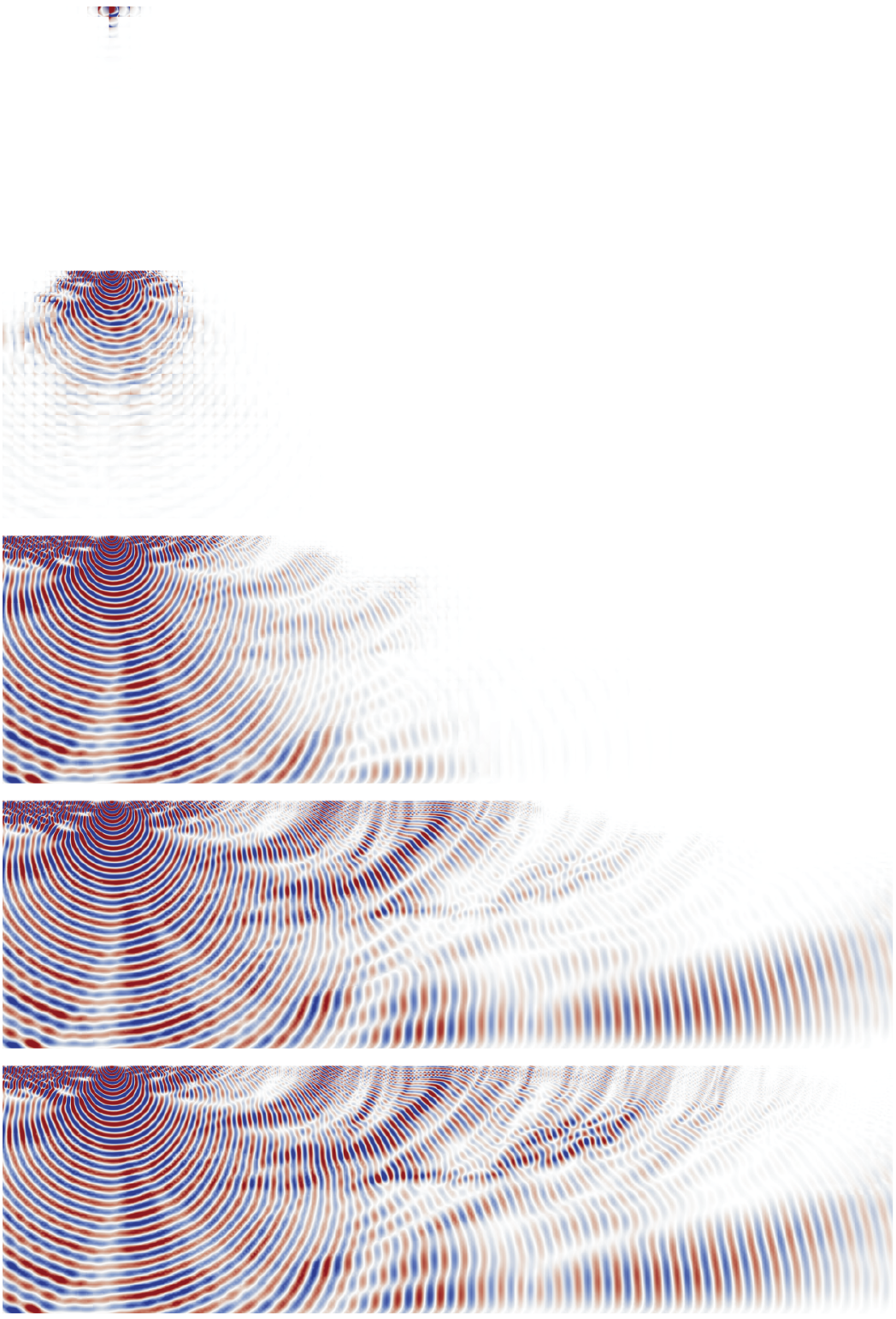
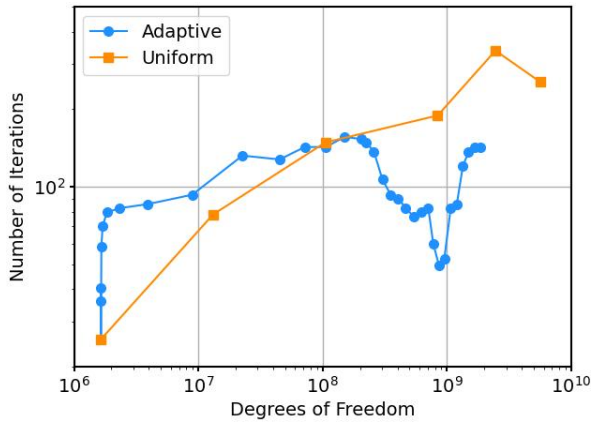
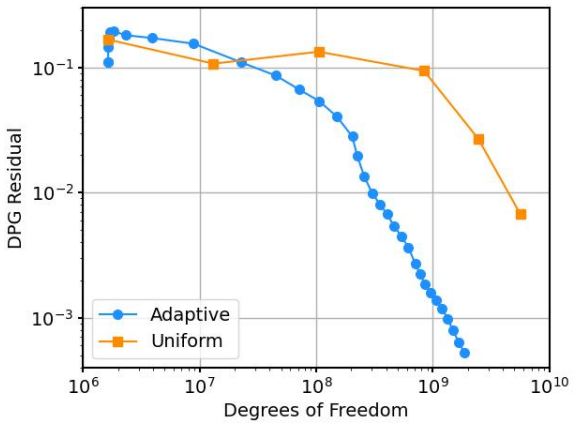


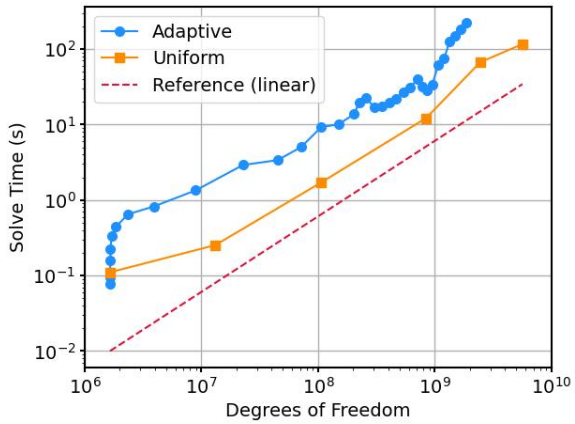
Figure 6.11: 2D cross-sections depicting the real part of the pressure after (from top to bottom) 0, 9, 18, 26, and 33 adaptive mesh refinements. Fields are amplitude-compensated.



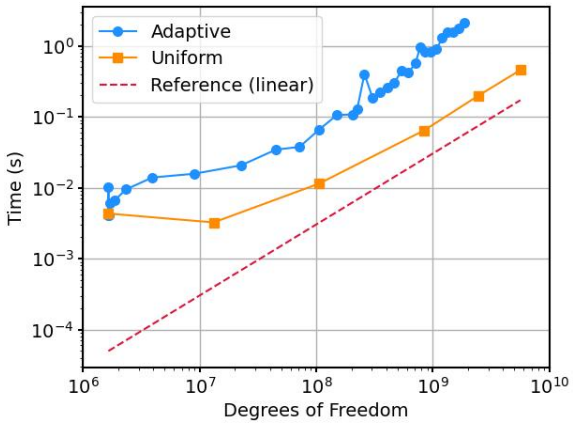
(a) Iterations to convergence (10^{-7} tolerance)



(b) Convergence of DPG residual



(c) Solution time



(d) Time *per iteration*

Figure 6.12: Convergence and timings for *hp*-adaptive solution of the `GO_3D_OBS` model with the DPG-MG solver (28 672 cores). The number of iterations is better controlled when solutions on previous grids are used to initialize subsequent grids. For a fixed frequency, the DPG-MG solver scales nearly linearly with respect to degrees of freedom; super-linear scaling is due to sub-geometric growth in DOFs.

that the time per iteration for the uniformly refined case is three times smaller than for adaptive meshes, or nine times smaller when normalizing for DOFs.

Indicator-based adaptivity is attractive for resolving singular and localized features. Indeed, it did manage to resolve the near-singular point source in Fig. 6.11, this is however typically not the feature of interest in seismic modeling contexts. The DPG test norm (and thus error indicator) could be defined via a weighted norm (e.g. L^2_r) to better account for far-field errors, this could make a compelling future study.

Wavespeed-adapted simulation. In the context of seismic wave simulation, the wavefield is expected to be supported globally and the wavespeed provides an initial indicator of the mesh size and order needed to resolve the wavefield. Adapting the mesh to wavespeed is often a much more efficient approach and does not require repeated solution of the problem on a large number of meshes. Tournier et al. [115] used this approach with advanced mesh generation tools to generate a regular tetrahedral mesh with element size adapted to wavespeed. While the DPG-MG solver does accommodate general unstructured meshes with all element types, mesh generation is often cumbersome and difficult to automate. We thus adopt a slightly different approach, starting from trivial uniform hexahedral meshes and leveraging *hp*-adaptivity to define wavespeed adapted meshes.

Wavespeed adaptive meshing was accomplished by defining a number of rules that determine when elements are adequately refined. Perhaps the simplest rule is that element edge length should not exceed a given ratio of the minimum wavelength over the element. The wavespeed-adapted mesh is then produced by looping over elements in the current mesh, checking whether the given rules are satisfied, and marking them for h-refinement if not. A sequence of meshes produced in this way is illustrated later in Fig. 6.17. This initial h-adaptation defines a coarse-grid for the DPG-MG solver with relatively uniform approximability, additional h- and p-refinements are then specified to define multigrid levels. The initial round of refinements is typically based on wavespeed only; subsequent refinements select between h- and p-refinements based on the number of elements per wavelength and the variation of wavespeed over the element, leveraging p-refinements in elements with low wavespeed variation and h-refinements elsewhere. This strategy thus deploys h-refinements to resolve sharp interfaces and irregular features while deploying p-refinements to mitigate pollution error. A cross-section of a wavespeed-adapted mesh for the `GO_3D_OBS` model is depticted in Fig. 6.13. Wavespeed adaptivity is not limited to simple hexahedral meshes, it could also be useful starting from general meshes conforming to topography or some other feature of interest.

Adapting meshes to wavespeed in this way is conceptually simple, does not require complex meshing tools, and can be done in parallel in the fully distributed version of *hp*3D. Generating the wavespeed adapted mesh for the problem illustrated in Fig. 6.14 and Fig. 6.15, with nearly 500 million elements took a total of 12 seconds, with a majority of that time spent updating geometry DOFs between refinements. The previously outlined fully-distributed

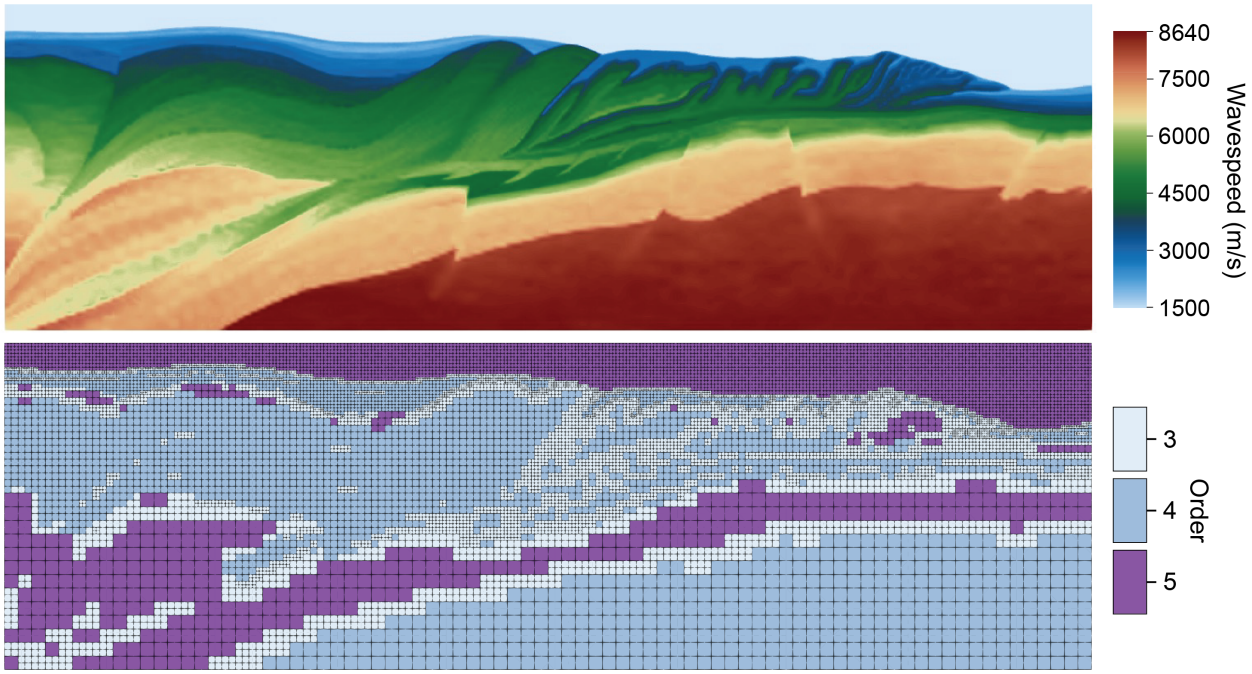


Figure 6.13: Cross-section of the wavespeed (top) and wavespeed-adapted mesh (bottom) for the `GO_3D_OBS` model; the mesh is intended to resolve a 2 Hz wavefield.

data structures solve many of the challenges of scaling this routine. We note however that the large seismic models used in this section occupy 50–300GB, depending on the model and number of properties stored. The models are thus distributed as well, with each process storing the section of model corresponding to its current subdomain.

Wavespeed adaptivity was used to simulate acoustic propagation in the 100 km × 100 km × 30 km section of the `GO_3D_OBS` model depicted in Fig. 6.9 at 15 Hz, this is the same problem simulated in the weak scaling study in Section 5.6.3. In contrast to the indicator-based adaptive simulation previously, a free-surface boundary condition was implemented at the surface. The free-surface boundary condition generates reflections, effectively forming a waveguide with the seabed; this tends to roughly double the number of iterations required for convergence. The coarsest mesh had uniform order $p = 2$ and was adapted to have a minimum of 2 elements per wavelength. Two p-refinements were then performed, resulting in a final adapted mesh with over 500 million hexahedral elements and 157 billion DOFs, a 5.3× reduction compared to the uniform mesh considered in Section 5.6.3 with 829 billion DOFs.

The simulation was performed on 2048 *Frontera* nodes (114 688 cores). The top-level

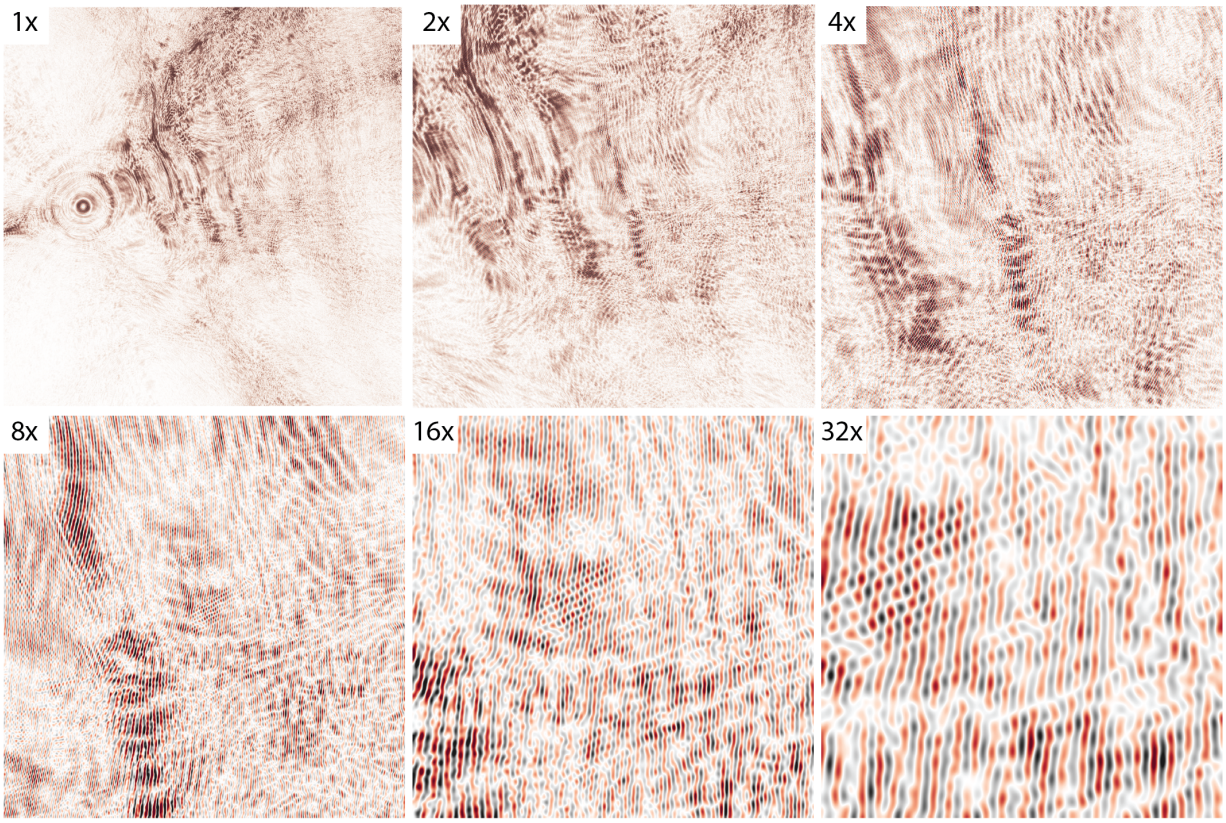


Figure 6.14: 15 Hz time-harmonic acoustic wavefield simulated on a 100 km × 100 km × 30 km section of the `GO_3D_OBS` using a wavespeed adapted mesh on 2 048 *Frontera* nodes (114 688 cores). The real part of the pressure ($\Re\{p\}$) is plotted at the surface ($z = 0$) in all panels, with the magnification factor indicated. Fields are amplitude compensated.

patches were refactored during the simulation, while patches on the remaining grid levels were stored. Smoothing iterations on the coarsest grid were thus relatively inexpensive and 16 PCG iterations were used for the coarse-grid correction. The iteration was terminated at 10^{-4} relative ℓ^2 residual, requiring 492 iterations for convergence. Construction of the DPG normal system took 532 seconds and the solve took 3 891 seconds; however, the maximum per-node memory usage was 76 GB, indicating that top-level patches perhaps could have been stored, which would have significantly reduced the solve time.

Cross-sections of the (amplitude-compensated) solution are depicted in Fig. 6.14 and Fig. 6.15. To our knowledge this is the largest time-harmonic visco-acoustic simulation to date on a complex heterogeneous model, corresponding to ca. 1 000 acoustic wavelengths in the water. We did not verify accuracy but computing the ratio of the DPG error-indicator at the final solution and under a zero wavefield indicates an L^2 error of $\mathcal{O}(5\%)$.

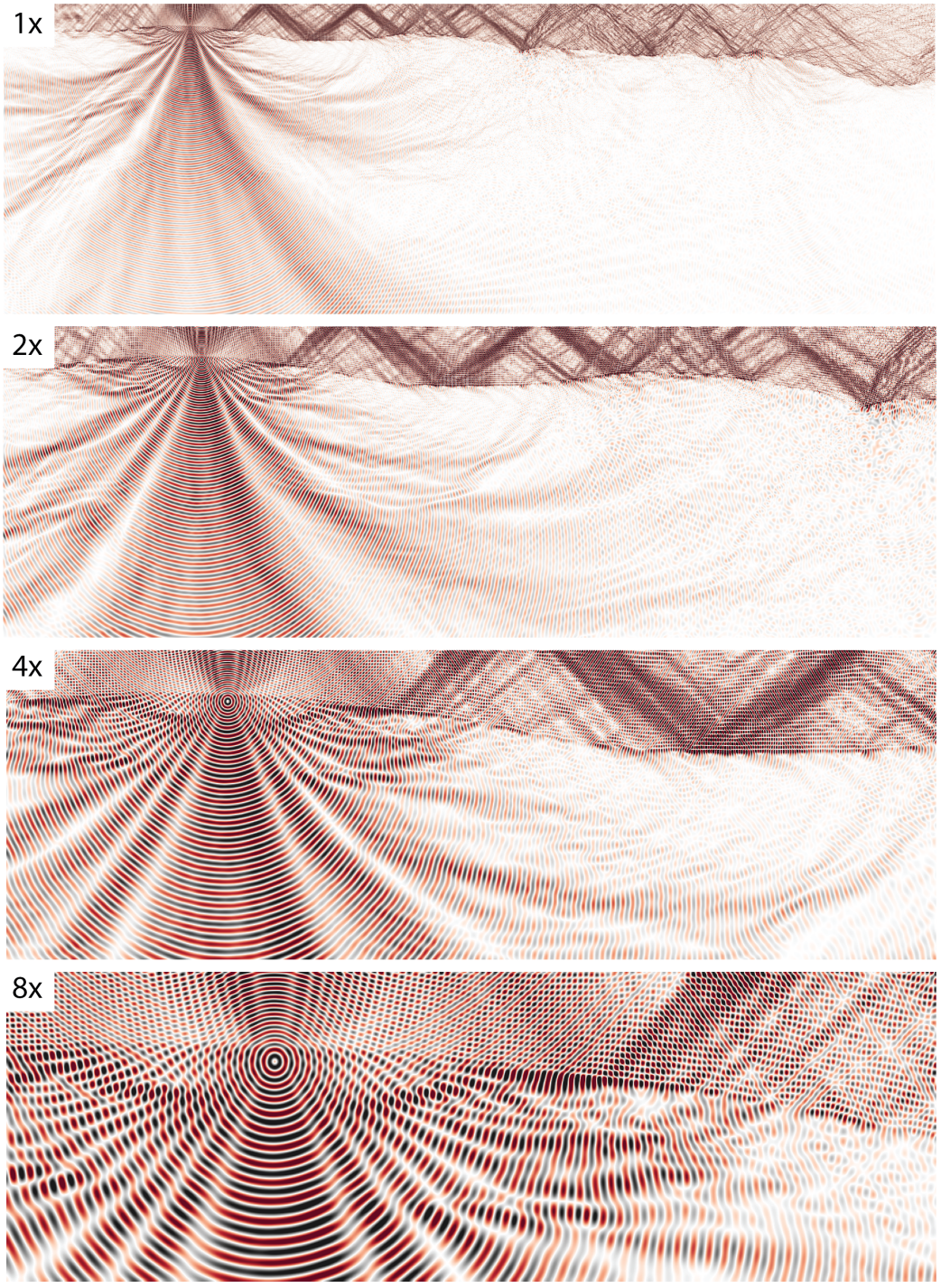


Figure 6.15: 15 Hz time-harmonic acoustic wavefield simulated on a 100 km × 100 km × 30 km section of the `GO_3D_OBS` using a wavespeed adapted mesh on 2 048 *Frontera* nodes (114 688 cores). The real part of the pressure ($\Re\{p\}$) is plotted at ($y = 50$ km) in all panels, with the magnification factor indicated. Fields are amplitude compensated.

6.3.2 Visco-Elastic Simulation on the SEAM Arid Model

We briefly conclude with visco-elastic simulation on the SEAM Arid model, produced in collaboration with Sergey Fomel and Andrey Bakulin of the Bureau of Economic Geology at the University of Texas at Austin. We thank Saudi Aramco for providing the model, it has helped us probe the limits of the DPG-MG solver and wavespeed adaptive meshing.

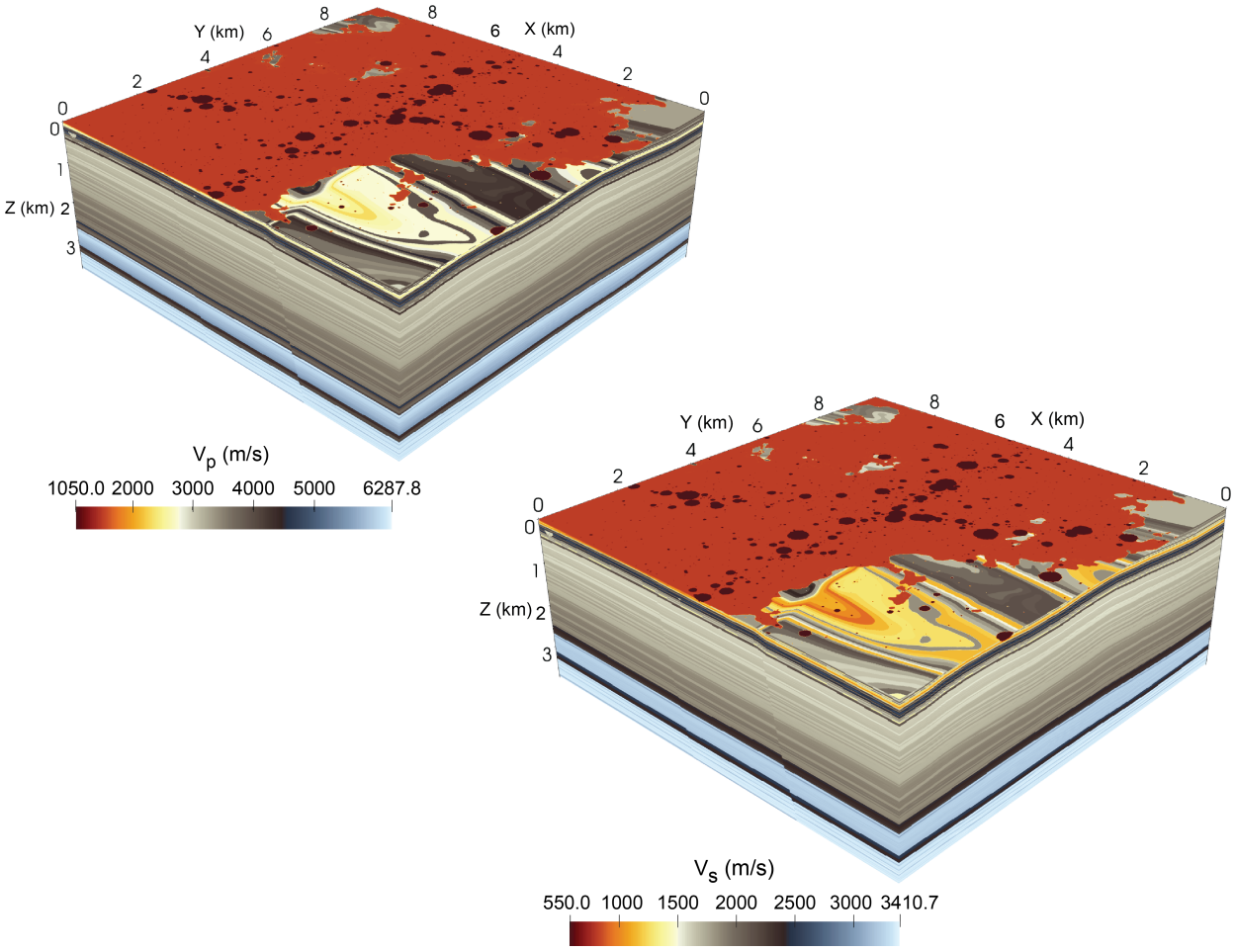


Figure 6.16: Compressive wavespeed (V_p, left) and shear wavespeed (V_s, right) for the SEAM Arid model.

Model. The Arid model has complex near-surface structure including karsts and significant high-contrast structures. Within the first 150 m, the minimum (shear) wavespeed is 550 m/s and the maximum (compressive) wavespeed is ca. 5 000 m/s, a contrast of nearly 10×. The model is characterized by horizontal transverse isotropy (HTI), defined via compressive wavespeed (V_p), shear wavespeed (V_s), density (ρ), Thomsen's parameters (γ, ϵ, δ), and a

rotation angle defining the orientation of the isotropic plane (ϕ). The full domain is 10 km $\times$ 10 km $\times$ 3.75 km, and is depicted in Fig. 6.16. All material properties are specified on a 6.25 m regular grid. We will consider only a 7 km $\times$ 7 km patch of the model, centered at $(5, 5)$ km. The top 25 m of the model is attenuating, with constant compressive and shear Q-factors, $Q_s = Q_p = 75$. A free-surface boundary condition was assumed at the surface, a 0.5 km wide PML was appended to the remaining boundaries. The PML was defined via the Cartesian stretching function (2.16) with parameters $\beta = 3$, $C = 50$. A frequency of was 25 Hz ($\omega = 50\pi$) was assumed.

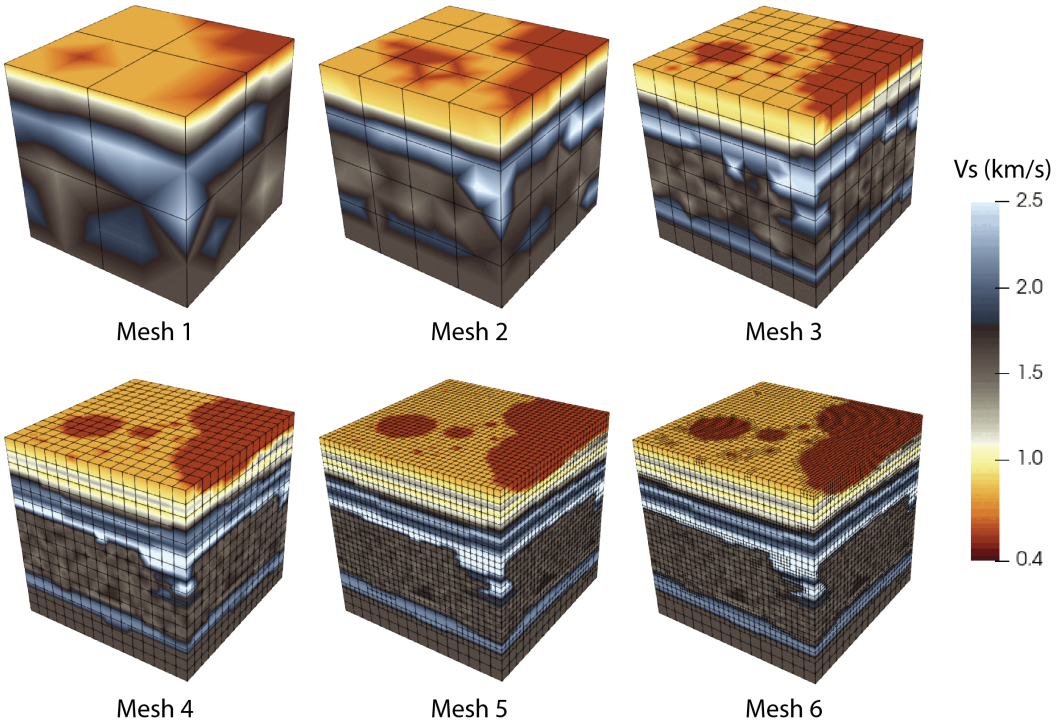


Figure 6.17: Wavespeed-adaptive meshing of a 0.5 km cube section of the SEAM Arid model.

Mesh. The mesh was defined starting from an $80 \times 80 \times 44$ second-order ($p = 2$) uniform hexahedral mesh. The initial-mesh was then adaptively h-refined until at least 2.2 elements per shear wavelength. In the final refinement, elements in which the shear wavespeed varied more than 30% of the minimum shear wavespeed were h-refined; the remaining elements p-refined. The final mesh had 20 million elements and 7.4 billion DOFs with a minimum of ca. 7 points per shear wavelength. The complex near-surface structure of the Arid model makes hp-adaptivity particularly advantageous. A uniform mesh with similar minimum resolution

would have ca. 20× more DOFs than the *hp*-adapted mesh. The mesh adaptivity process is visualized on a small section of the Arid model in Fig. 6.17. The material properties are visualized using second-order hexahedral elements, the full-resolution model is however stored separately from the mesh. The full-resolution model was used during adaptivity and throughout the simulation.

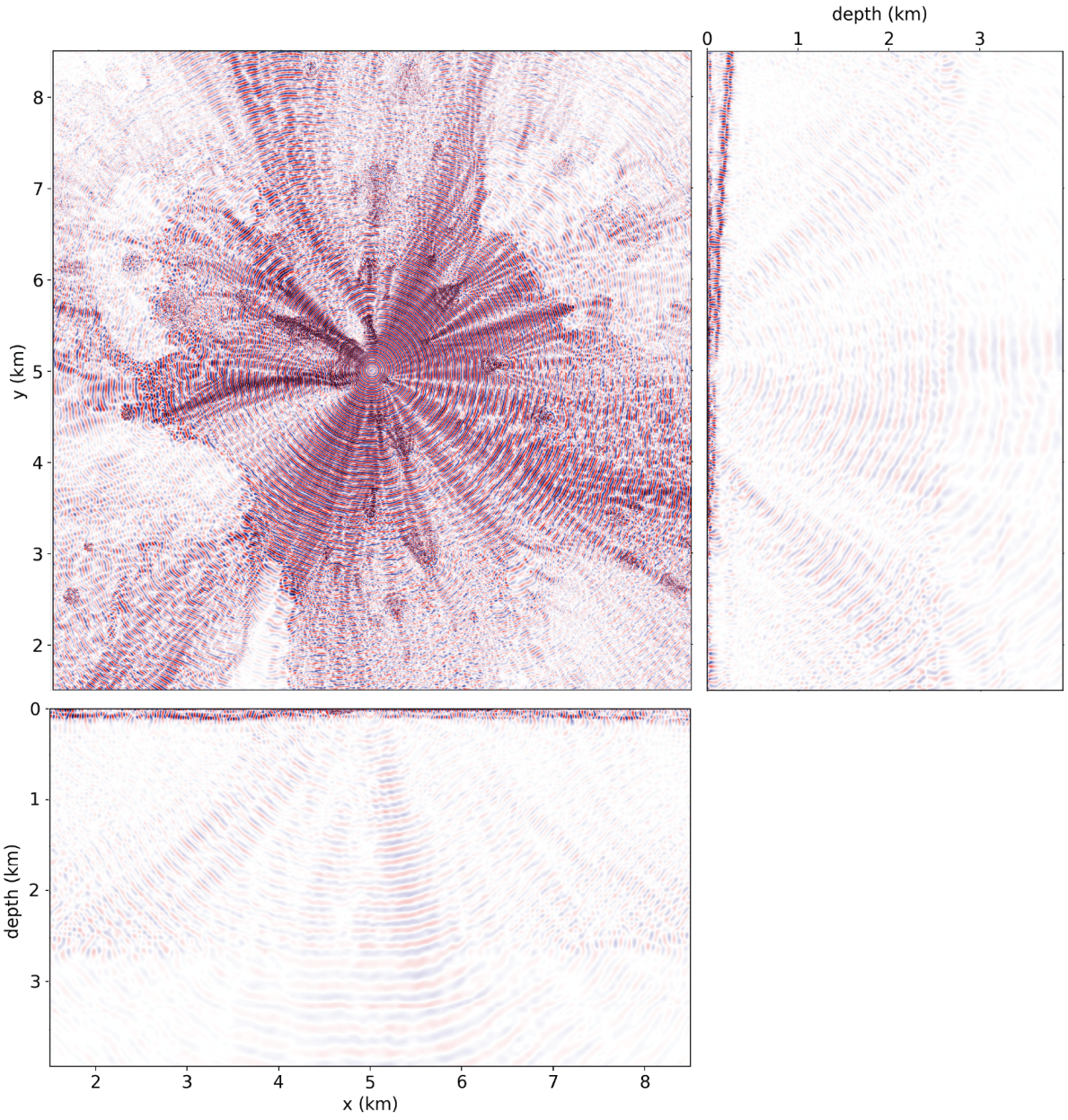


Figure 6.18: Time-harmonic visco-elastic (HTI) simulation of a 25 Hz source on the SEAM Arid model. Slices through the shot show the real part of z-displacement.

Simulation. The simulation was performed on 256 Frontera CLX nodes (14 336 cores). Test norm weight $\alpha = 10^{-2}$ was used and assembly of the DPG normal system took 348 seconds. The iteration was terminated at a 10^{-5} relative ℓ^2 residual, requiring 273 iterations. A single source was solved in 493 seconds. The real part of the z-displacement ($\Re\{\mathbf{v}_z\}$) is depicted in Fig. 6.18. While this problem was run on 256 Frontera nodes (only 3% of the total machine) it is to our knowledge significantly larger than any previous time-harmonic elastic simulation. Convergence studies were conducted on a 0.5 km cube section of the arid model at 25 Hz, these will be published at a later date and indicated $\mathcal{O}(2\%)$ L^2 error, we expect the error on the full model to be modestly higher.

Chapter 7

Concluding Remarks

Contributions. This dissertation outlined development of a scalable, fully distributed multigrid solver for high-frequency wave propagation problems discretized with the DPG finite element methodology. The distributed DPG-MG solver was shown to enable solution of time-harmonic wave propagation problems with over 800 billion fourth-order ($p = 4$) unknowns, a scale nearly 2–3 orders of magnitude larger than any previous work we are aware of. As noted in Section 4.2, a portion of the achieved scale is due to a rather favorable counting of unknowns in DPG systems. Indeed, counting interior and interface unknowns of the first-order system results in ca. 3–4× as many unknowns as a Galerkin system on a comparable mesh. Perhaps an even more significant factor was our use of significantly larger computational resources than any previous time-harmonic wave propagation work we are aware of. Still, the DPG-MG solver is notable for its scalability and efficiency and enables solution of previously intractable problems in high-frequency wave propagation. Indeed, the DPG-MG solver was shown to enable solution of challenging problems with over 1 000 wavelengths, high-contrast and anisotropic materials, and complex geometries.

While the DPG-MG solver was the primary focus of this work, supporting technologies including fully distributed finite element data structures were outlined. These distributed data structures enabled fast and parallel isotropic and anisotropic *hp*-adaptive mesh refinement on meshes with elements of all types and with limited dynamic load balancing. These capabilities seem to be fairly unique among scalable finite element codes.

Conclusions and future directions. The observed performance of the DPG-MG solver is perhaps surprising, especially for a method that is notoriously slow. We thus briefly compare the DPG-MG solver to existing methods, summarize what we view as the main points of differentiation, and provide an outlook for future work.

First, note that the observed $\mathcal{O}(N^{4/3})$ computational complexity is in line with a host of other solvers including shifted Laplacian [44, 110], multilevel [113, 18, 52], and domain decomposition [16, 74, 112] methodologies, but is worse than the $\mathcal{O}(N \log(N))$ computational complexity for sweeping-type preconditioners [37, 117] including source-transfer [81], and others. The later class of methods are however difficult to scale. A comparison of leading scalable wave propagation solvers thus reduces to a comparison of constants, which often depends significantly on hardware and implementation. We thus limit our comparison to structural differences.

Domain decomposition (DD) methods are a promising alternative to the DPG-MG solver, the optimized restricted additive Schwarz (ORAS) preconditioner [15, 115] seems to be particularly noteworthy. The ORAS preconditioner [115] was recently applied with an optimized finite difference discretization [1] to enable solution of very high-frequency time-harmonic acoustics problems. Investigation of optimized DPG discretizations to minimize pollution (e.g. in the spirit of [14]) may be a compelling direction for future research.

Both ORAS and DPG-MG solvers leverage a parallel subspace correction strategy; the notable difference being that the ORAS solver leverages a smaller number of subspaces defined over larger subdomains as opposed to the coarse-grid vertex patches employed by the DPG-MG solver. Use of more localized vertex patches in the DPG-MG solver may prove to be a significant advantage compared to DD approaches. Perhaps the first benefit is that the solver can be implemented with dense matrix-vector operations. When solving multiple loads simultaneously, this enables use of dense matrix-matrix operations which often have high-compute intensity and perform well on GPUs and other accelerators such as advanced matrix extension (AMX) coprocessors. The localized definition of operators is also expected to be an advantage in the case of problems such as elasticity, where additional unknowns and increased coupling diminishes the performance of sparse solvers and necessitates use of smaller subdomains, often leading to deteriorated convergence. A final advantage was that patch factorizations could be recomputed on the fly, reducing the overall memory footprint and enabling solution of very large problems. Still, storing and applying smoothing patches represents the most significant cost of the DPG-MG solver, investigation of optimized smoothers for particular problems thus may prove a fruitful area of inquiry. For example, edge- or face-based definitions for $H(\text{div})$ and $H(\text{curl})$ variables, respectively [64, 65], would significantly reduce patch sizes.

An additional advantage of the DPG-MG solver relative to other scalable Helmholtz solvers is the Hermitian positive-definite structure of the resulting system. This structure enables use of the conjugate gradient iteration, or of a GMRES iteration based on the Lanczos factorization with a three-term recurrence. Both algorithms limit the size of the Krylov history and enable truly linear memory scaling, as demonstrated in Section 5.6. The Hermitian positive-definite structure of DPG systems further guarantees the invertibility of sub-blocks, playing a key role in the definition of the patch-based smoother, and seems to enable considerable flexibility in defining preconditioners.

Flexibility and adaptivity are also notable advantages. Indeed, the DPG-MG solver is applicable—without modification—to general DPG systems including coupled multiphysics problems. In the context of seismic modeling, simple wavespeed-adaptive mesh refinement was shown to reduce the number degrees of freedom by 5.3× in the case of the acoustic simulation shown in Fig. 6.14 and ca. 20× in the case of the elastic simulation shown in Fig. 6.18, relative to a uniform mesh with similar minimum resolution. In the context of optical fiber modeling, indicator-based adaptivity was used to progressively refine the localized beam solution, drastically reducing the size of the final system. Adaptivity is certainly not limited to the DPG method but the built-in error indicator makes adaptivity relatively more accessible since developing robust error indicators is problem-dependent and often requires significant work.

The various examples in this dissertation demonstrated the compelling pollution characteristics of DPG. In particular, DPG demonstrates relatively little dispersive (phase) error, with pollution largely manifesting as dissipative (amplitude) error. This property could prove useful in imaging contexts including full waveform inversion (FWI). Jaime Bramwell's dissertation work [13] on applying DPG to FWI was motivated precisely by the improved dispersion characteristics of DPG; however, we are not aware of any further work in this direction in the decade since that pioneering work. The limited work in this direction was likely due to the relative expense of DPG and the lack of scalable solvers for both DPG and high-frequency wave propagation, but likely merits fresh consideration. Additional benefits in the context of seismic modeling include 1) trivial handling of attenuation and other frequency-dependent phenomena, 2) relatively inexpensive resolution of near-surface structures with *hp*-adaptivity, and 3) support for general unstructured meshes to enable simulation of non-trivial topography and other complex features. Finally, the DPG-MG solver is expected to support simulation of additional problems of interest including poroelasticity and coupled multiphysics problems. Simulation of coupled time-harmonic acoustics, elasticity, poroelasticity, and perhaps other phenomena at scale would represent a significant advancement in forward modeling capabilities and support novel research in geosciences and beyond.

Appendices

Appendix A

Leaky Modes in W-type Slab Waveguide

In this section we consider the the scalar transverse eigenproblem (6.11) associated with a W-type slab waveguide defined on the half-space $\{(x, z) \in \mathbb{R}^2 : z > 0\}$. The W-type waveguide has discontinuous index of refraction (depicted in Fig. A.1) defined on five 'slabs' as:

$$n(x) := \begin{cases} n_0 & |x| < a \\ n_1 & a < |x| < b \\ n_0 & |x| > b. \end{cases}$$

where $n_0 > n_1$. In the following we will show that the W-waveguide has only continu-

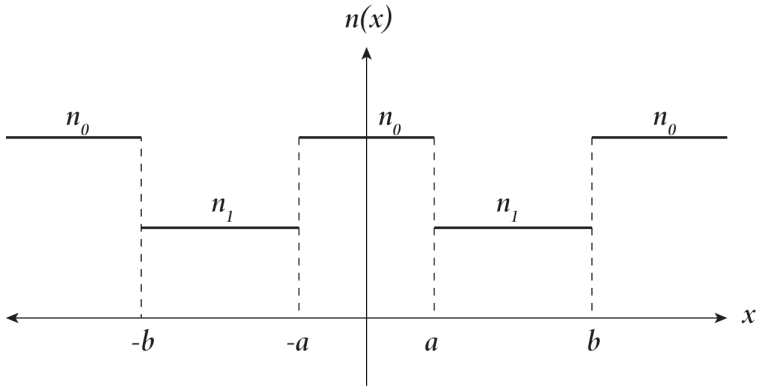


Figure A.1: Refractive index profile for W-type slab waveguide

ous spectrum and, in accordance with Section 6.1.2, defines a spectral representation with a meromorphic complex extension. Finally, we will show that the poles of the meromorphic continuation coincide with zeros of a dispersion relationship arising from enforcing a radiation-type condition on eigenmodes.

The scalar transverse eigenproblem is given by

$$\Delta u + \left(k_0^2 n(x)^2 - \beta^2\right) u = 0, \qquad x \in \mathbb{R}. \tag{A.1}$$

Define:

$$k := (k_0^2 n_0^2 - \beta^2)^{1/2} \qquad \text{and} \qquad \alpha := (\beta^2 - k_0^2 n_1^2)^{1/2},$$

and let $V := k_0^2(n_0^2 - n_1^2)$. Problem (A.1) permits solutions of the form

$$u(x) = \Re \circ \begin{cases} C_0 e^{-ikx} + \overline{C_0} e^{ikx} & |x| < a \\ C_1 e^{-\alpha x} + \overline{C_1} e^{\alpha x} & a < |x| < b \\ C_2 e^{-ikx} + \overline{C_2} e^{ikx} & b < |x| \end{cases} \tag{A.2}$$

where C_0, C_1, C_2 are complex constants that are partially constrained by requiring $u \in D(A) \equiv H^2(\mathbb{R}) \hookrightarrow C^1(R)$. Explicit expressions for modes can be derived more easily by considering separately *even* and *odd* modes characterized by $C_0 \in \mathbb{R}$ and $C_0 \in i\mathbb{R}$, respectively.

Dipsersion condition via radiation condition. For brevity we consider only even modes and the right half of the waveguide, $x \in \mathbb{R}^+$:

$$u(x) = \Re \circ \begin{cases} c_0 \cos(kx) & x < a \\ c_1 e^{-\alpha x} + d_1 e^{\alpha x} & a < x < b \\ c_2 e^{-ikx} + d_2 e^{ikx} & b < x \end{cases} \tag{A.3}$$

Enforcing C^1 continuity at $x = a$ and at $x = b$ will lead to four constraints on constants c_0, c_1, c_2, d_1, d_2; a fifth constraint will arise from the normalization implied by the orthogonality condition (6.14) but for now we simply set $c_0 = 1$.

Continuity (C^1) at $x = a$ gives rise to the linear system

$$\begin{bmatrix} e^{-\alpha a} & e^{\alpha a} \\ -\alpha e^{-\alpha a} & \alpha e^{\alpha a} \end{bmatrix} \begin{bmatrix} c_1 \\ d_1 \end{bmatrix} = \begin{bmatrix} \cos ka \\ -k \sin ka \end{bmatrix}$$

which permits solution

$$\begin{aligned} c_1 &= \frac{1}{2a}\big(\alpha e^{\alpha a} \cos ka + k e^{\alpha a} \sin ka\big), \\ d_1 &= \frac{1}{2a}\big(\alpha e^{-\alpha a} \cos ka - k e^{-\alpha a} \sin ka\big). \end{aligned}$$

Similarly, continuity at $x = b$ yields

$$\begin{bmatrix} e^{-\alpha b} & e^{\alpha b} \\ -\alpha e^{-\alpha b} & \alpha e^{\alpha b} \end{bmatrix} \begin{bmatrix} c_1 \\ d_1 \end{bmatrix} = \begin{bmatrix} e^{-ikb} & e^{ikb} \\ -ike^{-ikb} & ike^{ikb} \end{bmatrix} \begin{bmatrix} c_2 \\ d_2 \end{bmatrix}.$$

which leads to

$$\begin{aligned} c_2 &= \frac{1}{2ik}\Big(c_1(\alpha + ik)e^{-(\alpha - ik)b} - d_1(\alpha - ik)e^{(\alpha + ik)b}\Big) \\ d_2 &= \frac{1}{2ik}\Big(-c_1(\alpha - ik)e^{-(\alpha + ik)b} + d_1(\alpha + ik)e^{(\alpha - ik)b}\Big). \end{aligned}$$

Now, with the time-harmonic ansatz defined as in (2.8), d_2 is the coefficient on an *incoming* signal. Enforcing a radiation boundary condition (i.e. enforcing that the solution be outgoing only) results in the following dispersion relationship:

$$e^{-2\alpha(b-a)}(a\cos ka + k\sin ka)(\alpha - ik) - (\alpha\cos ka - k\sin ka)(\alpha + ka) = 0. \tag{A.4}$$

This relationship in general cannot be satisfied for $k \in \mathbb{R}$ so we consider roots of the complex extension, $k \in \mathbb{C}$. Figure A.2 shows the complex magnitude and phase of the left-hand side (LHS) of (A.4); the zeros can be identified visually.

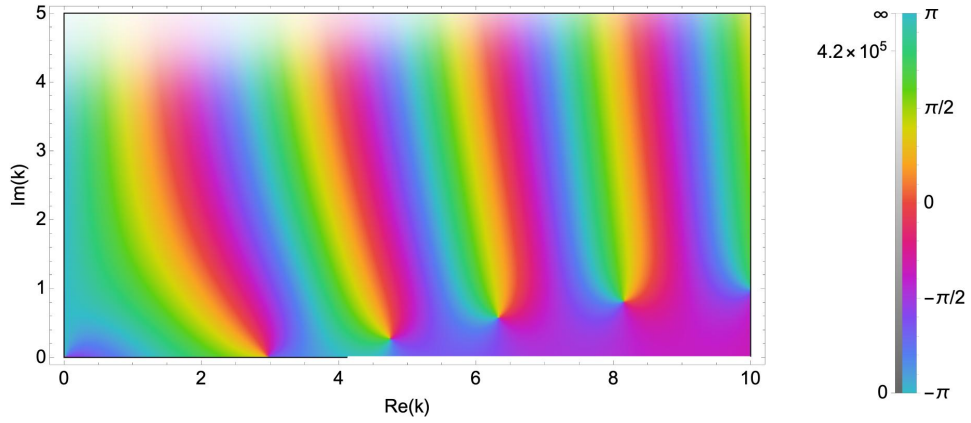


Figure A.2: Dispersion relation (A.4) for a W-type slab waveguide with parameters $V = 16, a = 0.25$, and $b = 1.5$. The complex phase is indicated by color and the magnitude is indicated by brightness. Zeros correspond to dark spots near the real axis.

Spectral representation. Finally, we can derive leaky modes by finding the pole of the complex extension of the spectral representation. In order to derive the complex extension we need a slightly different representation:

$$u(x) = \begin{cases} c_0\cos(kx) & x < a \\ c_1 e^{-\alpha x} + d_1 e^{\alpha x} & a < x < b \\ \hat{c}_2\cos(kx) + \hat{d}_2\sin(kx) & b < x. \end{cases} \tag{A.5}$$

The reason for the alternative representation is that the complex modulus is not an analytic function, thus cannot be used in the complex extension. By rewriting the solution in this form we ensure that $\hat{c}_2, \hat{d}_2 \in \mathbb{R}$. In this form, the normalization implied by the orthogonality condition (6.14) requires that $\hat{c_2}^2 + \hat{d}_2^2 = 1$. The proper normalization is thus given by

$$c_0 = \frac{1}{\sqrt{\hat{c_2}^2 + \hat{d}_2^2}}. \tag{A.6}$$

The poles of the spectral representation (6.16) behave as c_0^2, this is plotted in Fig. A.3. The poles can be identified as simple poles since contours around the poles traverse a single period in the complex phase. Finally, to see that the poles coincide we relate $\hat{c}_2, \hat{d}_2$ to c_2, d_2

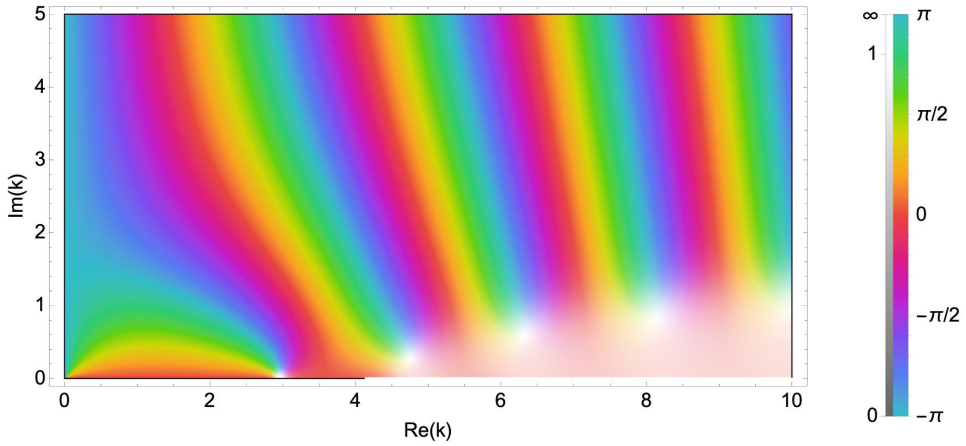


Figure A.3: Normalization factor (A.6) for a W-type slab waveguide with parameters $V = 16, a = 0.25$, and $b = 1.5$. The complex phase is indicated by color and the magnitude is indicated by brightness. Poles correspond to white spots near the real axis.

as

$$\hat{c}_2 = \frac{c_2 + d_2}{2} \qquad \text{and} \qquad \hat{d}_2 = i\frac{d_2 - c_2}{2},$$

and we can re-express the normalization:

$$\hat{c}_2^2 + \hat{d}_2^2 = \frac{(c_2 + d_2)^2}{4} - \frac{(c_2 - d_2)^2}{4} = 2c_2 d_2.$$

Zeros of the dispersion relation (A.4) thus correspond to poles of the normalization (A.6) and leaky modes derived via either method coincide.

Bibliography

[1] Hossein S Aghamiry, Ali Gholami, Laure Combe, and Stéphane Operto. Accurate 3D frequency-domain seismic wave modeling with the wavelength-adaptive 27-point finite-difference stencil: A tool for full-waveform inversion. *Geophysics*, 87(3):R305–R324, 2022.

[2] G.P. Agrawal. *Nonlinear fiber optics*. Academic Press, 5$^{\text{th}}$ edition, 2012.

[3] Keiiti Aki and Paul G Richards. *Quantitative seismology*. University Science Books, 2002.

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

[5] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.

[6] Phil D Anno. A Klein-Gordon acoustic theory. Technical report, Colorado School of Mines, Golden, CO (United States). Center for Wave Phenomena, 1992.

[7] Douglas N Arnold, Richard S Falk, and Ragnar Winther. Multigrid in H(div) and H(curl). *Numer. Math.*, 85(2):197–217, 2000.

[8] Jacob Badger, Stefan Henneking, and Leszek Demkowicz. Sum factorization for fast integration of DPG matrices on prismatic elements. *Finite Elem. Anal. Des.*, 172:103385, 2020.

[9] Jacob Badger, Stefan Henneking, Socratis Petrides, and Leszek Demkowicz. Scalable DPG multigrid solver for Helmholtz problems: A study on convergence. *Comput. Math. Appl.*, 148:81–92, 2023.

[10] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L. McInnes, R. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.13, Argonne National Laboratory, 2020.

[11] Natalie Beams, Ahmad Abdelfattah, Stan Tomov, Jack Dongarra, Tzanio Kolev, and Yohann Dudouit. High-order finite element method using standard and device-level batch GEMM on GPUs. In *2020 IEEE/ACM 11th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pages 53–60. IEEE, 2020.

[12] Marcella Bonazzoli, Victorita Dolean, Ivan G Graham, Euan A Spence, and Pierre-Henri Tournier. Domain decomposition preconditioning for the high-frequency time-harmonic Maxwell equations with absorption. *arXiv preprint arXiv:1711.03789*, 2017.

[13] Jamie Ann Bramwell. *A discontinuous Petrov-Galerkin method for seismic tomography problems*. PhD thesis, The University of Texas at Austin, 2013.

[14] Ignacio Brevis, Ignacio Muga, and Kristoffer G van der Zee. Neural control of discrete weak formulations: Galerkin, least squares & minimal-residual methods with quasi-optimal weights. *Comput. Methods Appl. Mech. Engrg.*, 402:115716, 2022.

[15] Xiao-Chuan Cai and Marcus Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21(2):792–797, 1999.

[16] Xiao-Chuan Cai and Olof B Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM J. Sci. Statist. Comput.*, 13(1):243–258, 1992.

[17] Zhiqiang Cai, Thomas A Manteuffel, and Stephen F McCormick. First-order system least squares for the Stokes equations, with application to linear elasticity. *SIAM J. Numer. Anal.*, 34(5):1727–1741, 1997.

[18] Henri Calandra, Serge Gratton, Xavier Pinel, and Xavier Vasseur. An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media. *Numer. Linear Algebra Appl.*, 20(4):663–688, 2013.

[19] Carsten Carstensen, Leszek Demkowicz, and Jay Gopalakrishnan. Breaking spaces and forms for the DPG method and applications including Maxwell equations. *Comput. Math. Appl.*, 72(3):494–522, 2016.

[20] Huanyang Chen, Che Ting Chan, and Ping Sheng. Transformation optics and metamaterials. *Nature Materials*, 9(5):387–396, 2010.

[21] Z. Chen and X. Xiang. A source transfer domain decomposition method for Helmholtz equations in unbounded domain. *SIAM J. Numer. Anal.*, 51(4):2331–2356, 2013.

[22] Dan Cheng et al. Bend distortion effect on transverse mode instability. *IEEE Photonics J.*, 14(1):1509606, 2022.

[23] Yifeng Cui, Kim B Olsen, Thomas H Jordan, Kwangyoon Lee, Jun Zhou, Patrick Small, Daniel Roten, Geoffrey Ely, Dhabaleswar K Panda, Amit Chourasia, et al. Scalable earthquake simulation on petascale supercomputers. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–20. IEEE, 2010.

[24] Hans De Sterck, Thomas A Manteuffel, Stephen F McCormick, and Luke Olson. Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs. *SIAM J. Sci. Comput.*, 26(1):31–54, 2004.

[25] Leszek Demkowicz. *Computing with hp Finite Elements. I. One and Two Dimensional Elliptic and Maxwell Problems.* Chapman & Hall/CRC Press, Taylor and Francis, 2006.

[26] Leszek Demkowicz. Various variational formulations and closed range theorem. *ICES Report*, 2015.

[27] Leszek Demkowicz. Energy Spaces, 2018. Lecture notes; The University of Texas at Austin.

[28] Leszek Demkowicz and Jay Gopalakrishnan. A class of discontinuous Petrov–Galerkin methods. Part I: the transport equation. *Comput. Methods Appl. Mech. Engrg.*, 199(23-24):1558–1572, 2010.

[29] Leszek Demkowicz and Jay Gopalakrishnan. A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions. *Numer. Methods Partial Differ. Equ.*, 27(1):70–105, 2011.

[30] Leszek Demkowicz and Jay Gopalakrishnan. Discontinuous Petrov–Galerkin (DPG) method. *Encyclopedia of Computational Mechanics Second Edition*, pages 1–15, 2017.

[31] Leszek Demkowicz, Jay Gopalakrishnan, and Antti H. Niemi. A class of discontinuous Petrov–Galerkin methods. Part III: adaptivity. *Appl. Numer. Math.*, 62(4):396–427, 2012.

[32] Leszek Demkowicz, Jason Kurtz, David Pardo, Maciej Paszyński, Waldemar Rachowicz, and Adam Zdunek. *Computing with hp Finite Elements. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications.* Chapman & Hall/CRC, 2007.

[33] Leszek F Demkowicz. *Mathematical theory of finite elements.* SIAM, 2023.

[34] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM J. Numer. Anal.*, 33(3):1106–1124, 1996.

[35] David Eric Edmunds and W Desmond Evans. *Spectral theory and differential operators.* Oxford University Press, 2018.

[36] Bjorn Engquist and Lexing Ying. Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation. *Comm. Pure Appl. Math.*, 64(5):697–735, 2011.

[37] Bjorn Engquist and Lexing Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Model. Simul.*, 9(2):686–710, 2011.

[38] Yogi A. Erlangga. Advances in iterative methods and preconditioners for the Helmholtz equation. *Arch. Comput. Methods Eng.*, 15(1):37–66, 2008.

[39] Oliver G Ernst and Martin J Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. In *Numerical Analysis of Multiscale Problems. Lect. Notes Comput. Sci. Eng.*, volume 83, pages 325–363. Springer, 2012.

[40] Federico Falope, Luca Lanzoni, and Angelo Marcello Tarantino. The bending of fully nonlinear beams. Theoretical, numerical and experimental analyses. *Int. J. Eng. Sci.*, 145:103167, 2019.

[41] Arnau Folch, Claudia Abril, Michael Afanasiev, Giorgio Amati, Michael Bader, Rosa M Badia, Hafize B Bayraktar, Sara Barsotti, Roberto Basili, Fabrizio Bernardi, et al. The EU Center of Excellence for Exascale in Solid Earth (ChEESE): Implementation, results, and roadmap for the second phase. *Future Generation Computer Systems*, 146:47–61, 2023.

[42] Federico Fuentes, Brendan Keith, Leszek Demkowicz, and Sriram Nagaraj. Orientation embedded high order shape functions for the exact sequence elements of all shapes. *Comput. Math. Appl.*, 70(4):353–458, 2015.

[43] M. Gander and H. Zhang. A class of iterative solvers for the Helmholtz equation: factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods. *SIAM Rev.*, 61(1):3–76, 2019.

[44] M. J. Gander, I. G. Graham, and E. A. Spence. Applying GMRES to the Helmholtz equation with shifted Laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed? *Numer. Math.*, 131(3):567–614, 2015.

[45] Claude J Gittelson, Ralf Hiptmair, and Ilaria Perugia. Plane wave discontinuous Galerkin methods: analysis of the h-version. *ESAIM Math. Model. Numer. Anal.*, 43(2):297–331, 2009.

[46] Björn Gmeiner, Markus Huber, Lorenz John, Ulrich Rüde, and Barbara Wohlmuth. A quantitative performance analysis for Stokes solvers at the extreme scale. *arXiv preprint arXiv:1511.02134*, 2015.

[47] Jay Gopalakrishnan, Ignacio Muga, and Nicole Olivares. Dispersive and dissipative errors in the DPG method with scaled norms for Helmholtz equation. *SIAM J. Sci. Comput.*, 36(1):A20–A39, 2014.

[48] Jay Gopalakrishnan, Benjamin Quanah Parker, and Pieter Vandenberge. Computing leaky modes of optical fibers using a FEAST algorithm for polynomial eigenproblems. *Wave Motion*, 108:102826, 2022.

[49] Jay Gopalakrishnan and Joseph Pasciak. Overlapping Schwarz preconditioners for indefinite time harmonic Maxwell equations. *Math. Comput.*, 72(241):1–15, 2003.

[50] Jay Gopalakrishnan and W. Qiu. An analysis of the practical DPG method. *Math. Comput.*, 83(286):537–552, 2014.

[51] Jay Gopalakrishnan and Joachim Schöberl. Degree and wavenumber [in] dependence of Schwarz preconditioner for the DPG method. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*, pages 257–265. Springer, 2015.

[52] Jayadeep Gopalakrishnan, Joseph E Pasciak, and Leszek F Demkowicz. Analysis of a multigrid algorithm for time harmonic Maxwell equations. *SIAM J. Numer. Anal.*, 42(1):90–108, 2004.

[53] Andrzej Górszczyk and Stéphane Operto. GO_3D_OBS: the multi-parameter benchmark geomodel for seismic imaging method assessment and next-generation 3D survey design (version 1.0). *Geosci. Model Dev.*, 14(3):1773–1799, 2021.

[54] Ivan Graham, Euan Spence, and Eero Vainikko. Domain decomposition preconditioning for high-frequency Helmholtz problems with absorption. *Math. Comput.*, 86, 07 2015.

[55] Dwight E Gray. American institute of physics handbook. *American Journal of Physics*, 32(5):389–389, 1964.

[56] Isaac Harari and Thomas JR Hughes. Finite element methods for the Helmholtz equation in an exterior domain: model problems. *Comput. Methods Appl. Mech. Engrg.*, 87(1):59–96, 1991.

[57] Mordehai Heiblum and Jay Harris. Analysis of curved optical waveguides by conformal transformation. *IEEE J. Quantum Electron.*, 11(2):75–83, 1975.

[58] Stefan Henneking. *A scalable hp-adaptive finite element software with applications in fiber optics*. PhD thesis, The University of Texas at Austin, 2021.

[59] Stefan Henneking and Leszek Demkowicz. A numerical study of the pollution error and DPG adaptivity for long waveguide simulations. *Comput. Math. Appl.*, 95:85–100, 2021.

[60] Stefan Henneking, Leszek Demkowicz, Socratis Petrides, Federico Fuentes, and Brendan Keith. *Computing with hp Finite Elements. III. Parallel hp3D Code.* In preparation, 2024.

[61] Stefan Henneking, Jacob Grosek, and Leszek Demkowicz. Model and computational advancements to full vectorial Maxwell model for studying fiber amplifiers. *Comput. Math. Appl.*, 85:30–41, 2021.

[62] Stefan Henneking, Jacob Grosek, and Leszek Demkowicz. Parallel simulations of high-power optical fiber amplifiers. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2020+1*, pages 349–360. Springer, 2022.

[63] Stefan Henneking, Jacob Grosek, and Leszek Demkowicz. Numerical simulation of transverse mode instability with a full envelope DPG Maxwell model, 2024. In preparation.

[64] Ralf Hiptmair. Multigrid method for H(div) in three dimensions. *Electron. Trans. Numer. Anal*, 6(1):133–152, 1997.

[65] Ralf Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1998.

[66] Ralf Hiptmair, Andrea Moiola, and Ilaria Perugia. Plane wave discontinuous Galerkin methods for the 2D Helmholtz equation: analysis of the p-version. *SIAM J. Numer. Anal.*, 49(1):264–284, 2011.

[67] K.R. Hiremath, M. Hammer, R. Stoffer, L. Prkna, and J. Čtyrokỳ. Analytic approach to dielectric optical bent slab waveguides. *Optical and quantum electronics*, 37:37–61, 2005.

[68] Jonathan Hu and Curtis R Menyuk. Understanding leaky modes: slab waveguide revisited. *Advances in Optics and Photonics*, 1(1):58–106, 2009.

[69] Martin Huber. *Hybrid discontinuous galerkin methods for the wave equation.* PhD thesis, Vienna University of Technology, 2013.

[70] F. Ihlenburg and I. Babuška. Finite element solution of the Helmholtz equation with high wave number part I: the h-version of the FEM. *Comput. Math. Appl.*, 30(9):9–37, 1995.

[71] Cesar Jauregui et al. Transverse mode instability. *Advances in Optics and Photonics*, 12(2):429–484, 2020.

[72] Brendan Keith, Federico Fuentes, and Leszek Demkowicz. The DPG methodology applied to different variational formulations of linear elasticity. *Comput. Methods Appl. Mech. Engrg.*, 309:579–609, 2016.

[73] Brendan Keith, Socratis Petrides, Federico Fuentes, and Leszek Demkowicz. Discrete least-squares finite element methods. *Comput. Methods Appl. Mech. Engrg.*, 327:226–255, 2017.

[74] Seungil Kim and Hui Zhang. Optimized Schwarz method with complete radiation transmission conditions for the Helmholtz equation in waveguides. *SIAM J. Numer. Anal.*, 53(3):1537–1558, 2015.

[75] R. Kleinman and G. Roach. Boundary integral equations for the three-dimensional Helmholtz equation. *SIAM review*, 16(2):214–236, 1974.

[76] Victor Kostin, Sergey Solovyev, Hongwei Liu, and Andrey Bakulin. HSS cluster-based direct solver for acoustic wave equation. In *SEG International Exposition and Annual Meeting*, pages SEG–2017. SEG, 2017.

[77] Heinz Langer. Spectral functions of definitizable operators in Krein spaces. In *Functional Analysis: Proceedings of a Conference Held at Dubrovnik, Yugoslavia, November 2–14, 1981*, pages 1–46. Springer, 2006.

[78] Barry Lee, Thomas A Manteuffel, Stephen F McCormick, and J Ruge. First-order system least-squares for the Helmholtz equation. *SIAM J. Sci. Comput.*, 21(5):1927–1949, 2000.

[79] Jin-Fa Lee. Finite element analysis of lossy dielectric waveguides. *IEEE Transactions on Microwave Theory and Techniques*, 42(6):1025–1031, 1994.

[80] San-Liang Lee et al. On leaky mode approximations for modal expansion in multilayer open waveguides. *IEEE J. Quantum Electron.*, 31(10):1790–1802, 1995.

[81] W. Leng and L. Ju. A diagonal sweeping domain decomposition method with source transfer for the Helmholtz equation. *arXiv preprint arXiv:2002.05327*, 2020.

[82] Ulf Leonhardt. Optical conformal mapping. *Science*, 312(5781):1777–1780, 2006.

[83] Michael Leumüller and Joachim Schöberl. Error analysis of an HDG method with impedance traces for the Helmholtz equation. *arXiv preprint arXiv:2307.03845*, 2023.

[84] Xiao Liu, Jianlin Xia, and Maarten V De Hoop. Parallel randomized and matrix-free direct solvers for large structured dense linear systems. *SIAM J. Sci. Comput.*, 38(5):S508–S538, 2016.

[85] Dietrich Marcuse. Curvature loss formula for optical fibers. *J. Opt. Soc. Am.*, 66(3):216–220, 1976.

[86] Dietrich Marcuse. *Theory of dielectric optical waveguides*. Elsevier, 2013.

[87] Jens Markus Melenk. *On generalized finite element methods*. PhD thesis, University of Maryland at College Park, 1995.

[88] Jens Markus Melenk and Ivo Babuška. The partition of unity finite element method: basic theory and applications. *Comput. Methods Appl. Mech. Engrg.*, 139(1-4):289–314, 1996.

[89] Jens Markus Melenk and Stefan A. Sauter. Wavenumber explicit convergence analysis for Galerkin discretizations of the Helmholtz equation. *SIAM J. Numer. Anal.*, 49(3):1210–1243, 2011.

[90] Peter Monk and Da-Qing Wang. A least-squares method for the Helmholtz equation. *Comput. Methods Appl. Mech. Engrg.*, 175(1-2):121–136, 1999.

[91] Harald Monsuur and Rob Stevenson. A pollution-free ultra-weak FOSLS discretization of the Helmholtz equation. *arXiv preprint arXiv:2303.16508*, 2023.

[92] Jaime David Mora and Leszek Demkowicz. Fast integration of DPG matrices based on sum factorization for all the energy spaces. *Comput. Methods Appl. Math.*, 19(3):523–555, 2019.

[93] Sriram Nagaraj. *DPG methods for nonlinear fiber optics*. PhD thesis, The University of Texas at Austin, 2018.

[94] Sriram Nagaraj, Jacob Grosek, Socratis Petrides, Leszek Demkowicz, and Jaime David Mora. A 3D DPG Maxwell approach to nonlinear Raman gain in fiber laser amplifiers. *J. Comput. Phys.*, 2:100002, 2019.

[95] Sriram Nagaraj, Socratis Petrides, and Leszek Demkowicz. Construction of DPG Fortin operators for second order problems. *Comput. Math. Appl.*, 74(8):1964–1980, 2017.

[96] Yvan Notay. Flexible conjugate gradients. *SIAM J. Sci. Comput.*, 22(4):1444–1460, 2000.

[97] John Frederick Nye et al. *Physical properties of crystals: their representation by tensors and matrices*. Oxford University Press, 1985.

[98] S. Operto, P. Amestoy, H. Aghamiry, S. Beller, A. Buttari, L. Combe, V. Dolean, M. Gerest, G. Guo, P. Jolivet, et al. Is 3D frequency-domain FWI of full-azimuth/long-offset OBN data feasible? The Gorgon data FWI case study. *The Leading Edge*, 42(3):173–183, 2023.

[99] Michael Oristaglio. SEAM Update: SEAM Phase II: The Arid Model—Seismic exploration in desert terrains. *The Leading Edge*, 32(6):606–608, 2013.

[100] Socratis Petrides. *Adaptive multilevel solvers for the discontinuous Petrov–Galerkin method with an emphasis on high-frequency wave propagation problems*. PhD thesis, The University of Texas at Austin, 2019.

[101] Socratis Petrides and Leszek Demkowicz. An adaptive DPG method for high frequency time-harmonic wave propagation problems. *Comput. Math. Appl.*, 74(8):1999–2017, 2017.

[102] Socratis Petrides and Leszek Demkowicz. An adaptive multigrid solver for DPG methods with applications in linear acoustics and electromagnetics. *Comput. Math. Appl.*, 87:12–26, 2021.

[103] Sander Rhebergen and Garth N Wells. Preconditioning of a hybridized discontinuous Galerkin finite element method for the Stokes equations. *J. Sci. Comput.*, 77(3):1936–1952, 2018.

[104] Sander Rhebergen and Garth N Wells. Preconditioning for a pressure-robust HDG discretization of the Stokes equations. *SIAM J. Sci. Comput.*, 44(1):A583–A604, 2022.

[105] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[106] Ross T Schermer and James H Cole. Improved bend loss formula verified for optical fiber by simulation and experiment. *IEEE J. Quantum Electron.*, 43(10):899–909, 2007.

[107] Joachim Schöberl. C++ 11 implementation of finite elements in NGSolve. *Institute for analysis and scientific computing, Vienna University of Technology*, 30, 2014.

[108] Matthew W Scroggs, Igor A Baratta, Chris N Richardson, and Garth N Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022.

[109] Matthew W Scroggs, Jørgen S Dokken, Chris N Richardson, and Garth N Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software (TOMS)*, 48(2):1–23, 2022.

[110] Abdul H Sheikh, Domenico Lahaye, and Cornelis Vuik. On the convergence of shifted Laplace preconditioner combined with multilevel deflation. *Numer. Linear Algebra Appl.*, 20(4):645–662, 2013.

[111] Dan Stanzione, John West, R Todd Evans, Tommy Minyard, Omar Ghattas, and Dhabaleswar K Panda. Frontera: The evolution of leadership computing at the National

Science Foundation. In *Practice and Experience in Advanced Research Computing*, pages 106–111. 2020.

[112] Christiaan C Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *J. Comput. Phys.*, 241:240–252, 2013.

[113] Christiaan C Stolk, Mostak Ahmed, and Samir Kumar Bhowmik. A multigrid method for the Helmholtz equation with optimized coarse grid corrections. *SIAM J. Sci. Comput.*, 36(6):A2819–A2841, 2014.

[114] Matthias Taus, Leonardo Zepeda-Núñez, Russell J Hewett, and Laurent Demanet. L-Sweeps: A scalable, parallel preconditioner for the high-frequency Helmholtz equation. *J. Comput. Phys.*, 420:109706, 2020.

[115] Pierre-Henri Tournier, Pierre Jolivet, Victorita Dolean, Hossein S Aghamiry, Stéphane Operto, and Sebastian Riffo. 3D finite-difference and finite-element frequency-domain wave simulation with multilevel optimized additive Schwarz domain-decomposition preconditioner: A tool for full-waveform inversion of sparse node data sets. *Geophys.*, 87(5):T381–T402, 2022.

[116] Leon Vardapetyan and Leszek Demkowicz. Full-wave analysis of dielectric waveguides at a given frequency. *Math. Comput.*, 72(241):105–129, 2003.

[117] Alexandre Vion and Christophe Geuzaine. Double sweep preconditioner for optimized Schwarz methods applied to the Helmholtz problem. *J. Comput. Phys.*, 266:171–190, 2014.

[118] Shen Wang, Maarten de Hoop, and Jianlin Xia. On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver. *Geophysical Prospecting*, 59:857–873, 2011.

[119] Amnon Yariv. Coupled-mode theory for guided-wave optics. *IEEE J. Quantum Electron.*, 9(9):919–933, 1973.

[120] Jeffery Zitelli, Ignacio Muga, Leszek Demkowicz, Jay Gopalakrishnan, David Pardo, and Victor M. Calo. A class of discontinuous Petrov–Galerkin methods. Part IV: the

optimal test norm and time-harmonic wave propagation in 1D. *J. Comput. Phys.*, 230(7):2406–2432, 2011.