UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Igor Almeida Baratta

HIGH-PERFORMANCE DOMAIN DECOMPOSITION PRECONDITIONERS FOR TIME-HARMONIC WAVE PROBLEMS

Belo Horizonte

2021

Igor Almeida Baratta

High-performance Domain Decomposition Preconditioners for Time-Harmonic Wave Equations

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Prof. Elson José da Silva

Belo Horizonte

2021

B226h

Baratta, Igor Almeida.
High-performance domain decomposition preconditioners for time-harmonic wave problems [recurso eletrônico] / Igor Almeida Baratta. – 2021.
1 recurso online (140 f. : il., color.) : pdf.

Orientador: Elson José da Silva.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 132-140.

Bibliografia: f. 121-131.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Método dos elementos finitos - Teses. 3. Eletromagnetismo - Simulação por computador - Teses. 4. Computação paralela. - Teses. 5. Método de decomposição - Teses. I. Silva, Elson José da. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

D E C L A R A Ç Ã O

Declaramos, para os devidos fins, que **Igor Almeida Baratta**, sob orientação do Prof. Elson José da Silva, concluiu os créditos e defendeu publicamente no dia 27 de Agosto de 2021 sua Tese de Doutorado intitulada **"High-performance Domain Decomposition Preconditioners For Time-harmonic Wave Problems"** perante banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, a qual foi aprovada por unanimidade, fazendo jus, portanto, ao grau de *Doutor* em Engenharia Elétrica.

Belo Horizonte, 15 de setembro de 2021.

Programa de Pós-Graduação em Engenharia Elétrica
EEUFMG- UFMG

Resumo

Esta tese investiga e propõe novas técnicas para a solução de problemas eletromagnéticos harmônicos discretizados pelo método dos elementos finitos. Reconhecemos que os métodos de solução estritamente algébricos não funcionam corretamente para tais problemas e, em vez disso, uma alternativa atraente é usar métodos iterativos pré-condicionados enriquecidos com informações física. Essa abordagem, no entanto, requer uma forte integração entre os métodos de discretização e solução. A implementação de pré-condicionadores de decomposição de domínio eficazes para problemas eletromagnéticos harmônicos é extremamente desafiadora. No entanto, mostramos que a implementação pode ser simplificada significativamente, empregando uma estrutura de elementos finitos que permite que cada algoritmo seja expresso em um nível de abstração adequado. Para gerenciar todas as abstrações e seus inter-relacionamentos, usamos e estendemos o ambiente de solução de problemas FEniCS. Além da disponibilidade de infraestrutura de software adequada, a eficácia dos métodos de decomposição de domínio depende de dois fatores extras que não podem ser obtidos algebricamente: as condições de transmissão aplicadas na interface entre subdomínios adjacentes e a correção de espaço grosseiro que permite uma transferência global de informações. Neste trabalho propomos um novo processo de otimização para o desenvolvimento de condições de transmissão que considerem automaticamente a natureza propagativa das ondas. Experimentos numéricos mostram que pré-condicionadores Schwarz de um nível utilizando as condições de transmissão otimizadas propostas conduzem sistematicamente a taxas de convergência rápidas. Também propomos um novo pré-condicionador de decomposição de domínio de dois níveis para problemas eletromagnéticos harmônicos que é eficiente e escalável: a construção do problema de espaço grosseiro é barata, e o GMRES pré-condicionado depende fracamente do número de processos MPI.

Palavras-chave: Método dos elementos finitos. Computação paralela, Métodos de decomposição de domínio.

Abstract

This thesis investigates and proposes new techniques for solving time-harmonic electromagnetic problems discretized by the finite element method. We reckon that strictly algebraic solving methods do not work correctly for such problems, and instead, an appealing alternative is to use physics-informed preconditioned iterative methods. This approach, however, requires a strong integration between the discretization and solution methods. As the current trend in high-performance computing trends toward increased parallelism, domain decomposition preconditioners come into play, and it is the class of method we consider in this thesis. The implementation of effective domain decomposition preconditioners for time-harmonic electromagnetic problems is remarkably challenging. However, we show that the implementation can be simplified significantly by employing a finite-element framework that allows each algorithm to be expressed at a suitable abstraction level. To manage all the abstractions and their interrelationships, we use and extend the FEniCS Problem Solving Environment. In addition to the availability of suitable software infrastructure, the effectiveness of domain decomposition methods depends on two extra factors that cannot be obtained purely algebraically: the transmission conditions applied at the interface between adjacent subdomains and the coarse space correction that allows a global transfer of information. We propose a new optimization process for devising transmission conditions that automatically considers the propagative nature of waves. Numerical experiments show that one-level Schwarz preconditioners with our optimized transmission conditions lead systematically to fast convergence rates. We also propose a new two-level domain decomposition preconditioner for time-harmonic electromagnetic problems that is efficient and scalable in parallel: the construction of the coarse space problem is cheap overall, and the preconditioned GMRES depends weakly on the number of processes. We present some numerical experiments using the FEniCSx library, highlighting our approach's scalability on up to 2240 processes and 2×10^9 unknowns.

Keywords: Computational electromagnetics. Finite element method. Parallel Computing.

List of Figures

List of Tables

Contents

1 Introduction

Contents

1.1 Summary

There is no known method for finding an analytical solution for many problems arising from scientific and industrial applications. In such circumstances, fast numerical methods capable of finding a solution to an acceptable accuracy level are fundamental. Nevertheless, finding the approximate solution in an adequate time and finite computational resources is still challenging.

The demand for simulations of complex and large-scale electromagnetic problems has grown continuously over the past decades and requires increasingly efficient numerical methods. Some areas even have their development linked to the availability of fast and accurate tools, such as microwave medical imaging (BONAZZOLI et al., 2017b), antenna design and integration (BARKA, 2013), electromagnetic compatibility (WANG et al., 2017) in addition to the design of photonic devices and structures (LIU; MILLER; FAN, 2013). Furthermore, computational solutions are also used to understand physical phenomena (LIU; MILLER; FAN, 2013), thus supporting scientific discovery. Computational electromagnetism research is an active field per se, as can be verified with an upward trend in the number of scientific publications in the field, as shown in Figure 1.

The solution of the time-harmonic electromagnetic equations is remarkably challenging; from a computational perspective, the simulation of these wave phenomena often leads to massive computations. Due to the oscillatory nature of the solutions and the intrinsic pollution effect (IHLENBURG; BABUŠKA, 1995), the resulting linear system of equations is rather large. Besides, it inherits the sign-indefiniteness of the discretized partial differential operator (ERNST; GANDER, 2012).

While the pollution effect can be alleviated to some extent using higher-order polynomial approximations (IHLENBURG; BABUSKA, 1997), the indefiniteness is an intrinsic property of time-harmonic wave problems (MOIOLA; SPENCE, 2014). This indefiniteness has severe implications on classical iterative solvers' convergence and performance. Direct solvers such as

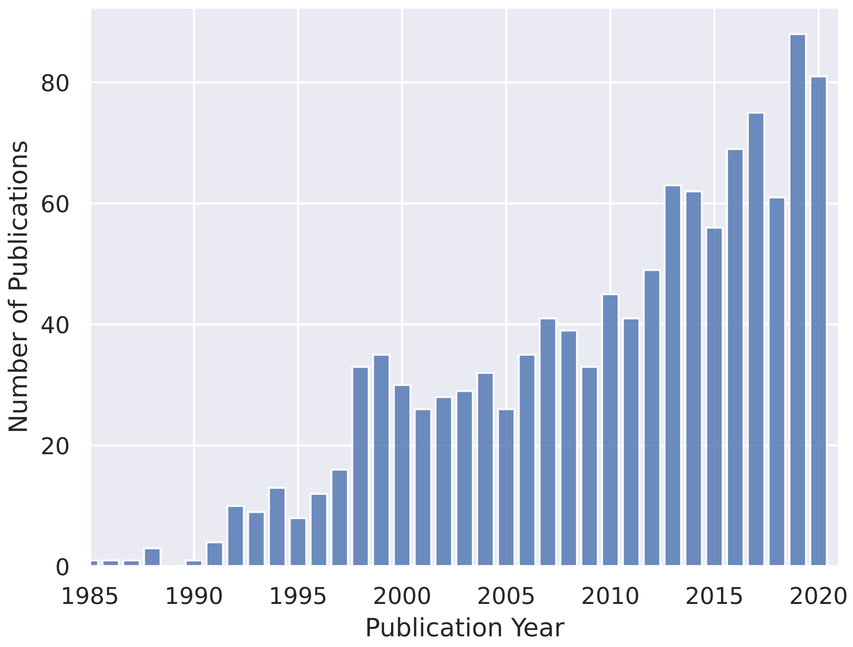


Figure 1 – Number of publications indexed in the Web of Science (WoS) with keywords *electromagnetics* and *finite element method* since 1985. The graph shows a smooth growth trend with peak in 2019.

LU-factorization, on the other hand, are not memory efficient and lack scalability on parallel architectures. An appealing alternative is the use of preconditioned iterative methods. As the current trend in high-performance computing trends toward increased parallelism, domain decomposition preconditioners come into play.

The thesis's central theme is developing robust parallel domain decomposition preconditioners for the iterative solution of time-harmonic wave boundary value problems, discretized by the finite element method. As will become apparent in later chapters, purely algebraic preconditioners are not suited for our problem of interest. Thus, the development of the preconditioner should be strongly linked to the physical problem and the corresponding discretization. Consequently, we will also focus on the discretization method, as it will be a fundamental part of constructing the preconditioner.

The effectiveness of modern domain decomposition methods stems from two extra factors (that cannot be obtained purely algebraically): the transmission conditions applied at the interface between adjacent subdomains and the coarse space correction that allows a global transfer of information. Unfortunately, the standard choice for any of these factors is not adequate for time-harmonic wave problems. Classical one-level and two-level domain decomposition preconditioners are not scalable in modern parallel architectures nor have satisfactory convergence rates. Even high-performance implementations of purely algebraic methods suffer from insufficient scalability as a result of poor convergence.

After a brief introduction to the equations that describe the problem of interest and a review of the finite element method, we start the exposition of domain decomposition methods by studying transmission conditions tailored for time-harmonic problems. In the case of scalar waves, governed by the Helmholtz equation, we describe in detail a new optimization process for devising transmission conditions. Numerical experiments show that the Schwarz preconditioner with our optimized transmission conditions leads systematically to faster convergence rates than using the conditions available in the literature. The extension of this method to vectorial Maxwell's equation follows naturally through a similarity analysis and is confirmed by numerical experiments.

Even with optimal transmission conditions, the convergence rates of one-level preconditioners depend on the number of subdomains, preventing such methods from achieving weak or strong scalability. Two-level methods introduce a coarse space correction, a mechanism by which global communication is achieved. Unfortunately, robust two-level preconditioners for time-harmonic Maxwell's equations are still elusive. Many of the advances discussed in the scientific literature for scalar waves have not yet been achieved for vectorial problems. After an in-depth bibliographic review, we introduce new coarse space corrections guided by physical principles that achieve good parallel scalability and robustness, reducing the gap in the scientific literature.

All in all, in this thesis, we propose contributions of varying degrees in many aspects of the construction of scalable and robust methods for the iterative solution of time-harmonic waves. In addition to the scientific contribution associated with the development of domain decomposition methods, the implications of using the proposed methods and implementations can range from reducing the time to market of electromagnetic devices to improving microwave imaging diagnosis.

Some components of this Ph.D. have been directly software-related; other components have been more technical but have been accompanied by an implementation in open-source software within the FEniCSx framework or add-on libraries. Besides the impact on academia, the inclusion of these components in open-source software also impacts both the industry and the engineering community.

1.2 Scope and Goals

When it comes to wave propagation, the scope of traditional Computational Electromagnetic approaches (CEM), such as the finite element method (FEM) (JIN, 2015), the Boundary Element Method (BEM) (HARRINGTON, 1993), or the Finite Difference Method (FDM) (TAFLOVE; HAGNESS, 2005), is limited to problems of moderate electrical size and simplified geometrical complexity. This limitation is, in general, due to the high computational cost associated with the solution of the discretized system.

Since semiconductor manufacturing physics no longer lets sequential processors get faster, not only are the fastest computers parallel (STROHMAIER et al., 2015), but almost all computers are parallel, including consumer portable devices. The current computational potential, measured by the number of floating-point operations per second, is only available through parallelism. However, traditional numerical methods were developed for sequential computers and did not fully advantage these new architectures' processing power.

Although different efforts have been made, robust and highly scalable methods for solving time-harmonic electromagnetic problems remain elusive, creating great research opportunities. This thesis aims to achieve further progress in this field and, more expressly, in iterative domain decomposition preconditioners supporting the solution of high-frequency problems discretized with the finite element method that scales up to thousands of computing units.

1.2.1 Specific Goals and Proposed Contributions

This project aims to contribute to the development of efficient techniques for solving time-harmonic electromagnetic wave propagation problems. Furthermore, we aim to reduce the computational cost of complex engineering problems by exploring the capabilities of massively parallel computational architectures. This work, therefore, aims to study, implement and improve domain decomposition methods taking into account the current trends of highly parallelizable computational architectures.

The specific contributions can be divided into five topics that roughly define the remaining chapters of the thesis.

- Improve the availability of state of the art of open-source finite element software for solving large-scale electromagnetic problems. Several contributions were made to the FEniCS project, including the support for complex numbers, improved parallelism, and automatic high-performance code generation for Nédélec elements. (Detailed in Chapter 2, and Appendices A and B)

- Improve the convergence rates of one-level domain decomposition methods for time-harmonic problems by developing optimized transmission conditions that consider the propagative nature of the fields and are more suited for multiple subdomains. This means to develop methods that are better suited for parallel architectures up to a couple of hundred computing units. (Detailed in Chapter 3)

- Increase the robustness of plane wave-based two-level domain decomposition methods for the scalar Helmholtz equation. This class of preconditioners is scalable, and its construction has a moderate computational cost. However, it is challenging to choose which directions will go into the construction of coarse space. Using signal processing techniques, we propose a technique to choose wave directions that is automatic, cheap, and effective. (Detailed in Chapter 4)

- Develop a two-level domain decomposition for time-harmonic Maxwell's equations. Based on the Shifted-Laplacian approach, we propose a new physics-based coarse space with a sound base in the finite element theory, and we show its effectiveness and scalability on up to 2240 cores and 2 billion unknowns. (Detailed in Chapter 5)

Moreover, along with the thesis, we provide a friendly (research) open-source software on top of FEniCS to solve time-harmonic wave problems and test different domain decomposition techniques, facilitating the development of new methods and the solution to larger problems.

1.3 Dissemination of Research Results

This section details the publications, presentations, and awards emerging from the work carried out during the course of the Ph.D.

1.3.0.1 Journal papers

The following manuscripts have been accepted or are currently under review in peer-reviewed journals:

- **Igor A. Baratta**, and Elson J. Silva. "Infinitesimal Dipole Model Using Space Mapping Optimization for Antenna Placement." *IEEE Antennas and Wireless Propagation Letters* (IF 3.448).
  Status: Published in 2018

- **Igor A. Baratta**, and Elson J. Silva. "Multi-Domain Transmission Conditions for Domain Decomposition Methods Applied to Scattering Problems." *IEEE Transactions on Magnetics* (IF 1.626)
  Status: Published in 2018

- **Igor A. Baratta**, and Elson J. Silva. "A Simplified Equivalent Model for Predicting the Installed Performance of Circularly Polarized Antennas." *IEEE Antennas and Wireless Propagation Letters* (IF 3.448).
  Status: Published in 2019

- Lucas Amorim, Thiago Goveia, Renato Mesquita, **Igor A. Baratta** "GPU Finite Element Method Computation Strategy Without Mesh Coloring" *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*
  Status: Published in 2020

- **Igor A. Baratta**, and Elson J. Silva. "High-level Implementation of a Scalable Two-Level Domain Decomposition Method for Time-Harmonic Electromagnetic Problems." Submitted to *IEEE Transactions on Antennas and Propagation* (IF 4.13).
  Status: Under review.

1.3.0.2 Conference presentations

A number of formal and informal presentations were given at various conferences, workshops, and meetings. The most relevant, in chronological order, are:

- Compumag 2017 - 18-22 June, 2017 - Daejeon Metropolitan City, Korea
  - **Baratta, Igor A.**, and Elson J. Silva. "Multi-Domain Transmission Conditions for Domain Decomposition Methods Applied to Scattering Problems."
- FEniCS'18 - 21-23 March, 2018 - University of Oxford Mathematical Institute, Oxford, UK
  - **Baratta, Igor A.**, and Elson J. Silva. "A Domain Decomposition Preconditioning for the Time-Harmonic curl-curl Maxwell's Equations using FEniCS."
- EMF 2018 - 10-12 April, 2018 - TU Darmstadt, Darmstadt, Germany
  - **Baratta, Igor A.**, and Elson J. Silva. "A two-level domain decomposition preconditioner for the Helmholtz equation with plane wave enrichment at the interfaces."
- CEFC 2018 - October 28-31, 2018 - Hangzhou, China
  - **Baratta, Igor A.**, et al. "A Quasi-optimal Domain Decomposition Preconditioner for the Meshless-Local Petrov Galerkin"
  - **Baratta, Igor A.**, et al. "A simplified model for predicting performance of circular polarized antennas"
- FEniCS‘19 - 12-14 June 2019 - Washington DC, USA
  - **Baratta, Igor A.**, et al. " Two-level domain decomposition preconditioners with improved plane wave coarse spaces"
  - **Baratta, Igor A.**, et al. "Complex Number support in FEniCS"

1.3.0.3 Awards and Internal Scholarships

1. NAG Travel Award - for participation in the 2018 FEniCS Conference at the University of Oxford, Oxford, UK - 21-23 March 2018.

2. NumFOCUS Travel Award - for participation in the 2019 FEniCS Conference at the Carnegie Institution for Science, Washington DC, USA - 12-14 June 2019.

3. NumFOCUS New Contributor Recognition - for the "outstanding contributions to the FEniCS project" - November 2, 2019.
<https://numfocus.org/blog/2019-numfocus-awards>.

4. Coimbra Group Short-term Scholarship Programme for Young Professors and Researchers 2019 - grant to finance short-term research visit to the University of Tartu in Estonia.

1.3.0.4 Software

The research described in this thesis has led to the development of some software components that are either part of an existing library, within the FEniCSx library, or are currently under development for the new `odd_dolfinx` library:

<https://github.com/IgorBaratta/odd_dolfinx>

1.4 Text Structure

Chapter 2 provides an introduction to time-harmonic wave equations, their discretization using finite element methods and classical solution methods. It also gives a brief overview of the software stack and the libraries we use throughout the project. In Chapter 3, we study one-level domain decomposition methods with an emphasis on Optimized Restricted Additive Schwarz preconditioners. We study the limitations of such methods and propose new means of improving convergence rates. Chapter 4 concerns two-level domain decomposition methods for the scalar Helmholtz Equation. Our focus is on coarse space construction and techniques that can be adapted to Maxwell's equations. In Chapter 5 we review state-of-art preconditioners and propose a new coarse space for time-harmonic Maxwell's equation. Finally, conclusions and future research directions are presented in Chapter 6.

2 Numerical Solution of Time-Harmonic Wave Problems

Contents

2.1 Introduction

One problem of practical interest is quantitative inverse electromagnetic scattering, fundamental to distinct application areas such as medical imaging, geophysical exploration, and nondestructive testing. This class of problem is typically solved using non-linear optimization methods, requiring repeated solutions of the forward problem arising from the discretization of Maxwell's equations, and the efficiency of the adopted solver is critical. Direct solvers, such as LU-factorization, are not memory efficient and lack scalability on parallel architectures, while purely algebraic iterative solvers may experience convergence issues. An appealing alternative is the use of physics-informed preconditioned iterative methods.

Occasionally, to simplify the analysis, scalar approximations are made in microwave scattering problems, and we start with these simplified models.The scalar-wave model is useful as a first step toward understanding the vectorial wave model. It can also be used as a base for acoustic and elastic wave phenomena. Although our primary focus is on developing preconditioners to the time-harmonic curl–curl Maxwell's equations, we also study these techniques for the scalar Helmholtz equations. We have adopted a bottom-up strategy, starting from the more

Table 1 – Electromagnetic Field Quantities

	Quantity	Unit
$\mathcal{E}(\boldsymbol{x}, t)$	electric field intensity	V/m
$\mathcal{H}(\boldsymbol{x}, t)$	magnetic field intensity	A/m
$\mathcal{D}(\boldsymbol{x}, t)$	electric flux density	C/m^2
$\mathcal{B}(\boldsymbol{x}, t)$	magnetic flux density	W/m^2
$\mathcal{J}(\boldsymbol{x}, t)$	electric current density	A/m^2
$\rho(\boldsymbol{x}, t)$	electric charge density	C/m^3

developed area of scalar waves and investigating adaptations of the techniques and principles to the vectorial case.

This chapter first presents the underlying equations for a general open-region electromagnetic problem in the *strong form* followed by the *weak form* and gives an overview of the finite element method. It introduces key concepts about finite elements repeatedly used in this thesis, such as degrees of freedom, basis functions, and interpolation operator. The degrees of freedom for scalar Lagrange finite elements, for which the degrees of freedom are nodal evaluations, and Nédélec finite elements, for which the degrees of freedom are moments against particular test functions, are used as examples to first demonstrate these concepts. Then, motivated by a literature survey and numerical experiments, we justify the need for high order methods and effective preconditioners to solve time-harmonic wave equations.

2.2 Electromagnetic Fields and Waves

The classical electromagnetic field theory mainly describes and relates four time-and space-dependent vector fields, current densities, and charge densities at any point in space and time (BALANIS, 2012). Table 1 presents these quantities and their respective units using the International System of Units (SI). In its differential form, the set of Maxwell's equations can be written as

$$\nabla \times \mathcal{E} + \frac{\partial \mathcal{B}}{\partial t} = 0, \tag{2.1}$$

$$\nabla \times \mathcal{H} + \frac{\partial \mathcal{D}}{\partial t} = \mathcal{J}, \tag{2.2}$$

$$\nabla \cdot \mathcal{D} = \rho, \tag{2.3}$$

$$\nabla \cdot \mathcal{B} = 0. \tag{2.4}$$

Equation (2.1) is known as Faraday's Law, or the law of magnetic induction, and describes the effect of a changing magnetic field on the electric field. Equation (2.2) is Ampère's circuital law generalized by Maxwell, which details how electrical currents can generate magnetic and alternating electric fields (displacement currents). Equation (2.4) states that the magnetic field is a divergence-free or a rotational field, and Equation (2.3) describes the relationship between the electric field and the electric charges that cause it. The electric current density $\mathcal{J}$ and the electric charge density ρ are governed by the continuity law,

$$\nabla \cdot \mathcal{J} = -\frac{\partial \rho}{\partial t}, \tag{2.5}$$

which describes the transport of electric charge.

Maxwell's equations are not sufficient to describe the electromagnetic fields completely. By including relations that describe the characteristics of the medium in which the electromagnetic fields propagate, the system can be closed. These are the so-called constitutive relations, relating, for instance, $\mathcal{D}$ and $\mathcal{B}$ to $\mathcal{E}$ and $\mathcal{H}$ respectively

$$\begin{cases} \mathcal{D} = \varepsilon \mathcal{E} \\ \mathcal{B} = \mu \mathcal{H} \end{cases}$$

where ε is the electric permittivity in (F/m) and the magnetic permeability in (H/m). These material properties can be anisotropic and inhomogeneous. Moreover, they can also be functions of time and the fields themselves. However, in the scope of this thesis, we assume only linear isotropic time-independent materials.

Another important constitutive relation is the Ohm's Law

$$\mathcal{J} = \sigma \mathcal{E} + \mathcal{J}_c$$

where σ is the electrical conductivity measured in (S/m), and $\mathcal{J}_c$ is an external applied conduction current density.

Throughout the project, the time dependence is assumed to be harmonic. This assumption is valid for many practical systems and extensively applied in microwave imaging problems (CHEN, 2018; NIKOLOVA, 2017). In general, we can represent such time variations by $e^{+j\omega t}$ and relate the instantaneous electromagnetic field vectors to their complex forms in a straightforward manner (BALANIS, 2012). In microwave inverse scattering applications, the domains are well approximated by isotropic linear medium, in which case the constitutive relations simplify to scalar complex-valued functions of position and frequency (BALANIS, 2012).

Under this complex representation, and using the constitutive relations of isotropic linear media, the first order time-harmonic Maxwell's equations reduce to

$$\nabla \times \mathbf{E} + j\omega\mu \mathbf{H} = 0, \tag{2.6}$$

$$\nabla \times \mathbf{H} - j\omega\varepsilon \mathbf{E} - \sigma \mathbf{E} = \mathbf{J}_c, \tag{2.7}$$

where $\mathbf{E}$, $\mathbf{H}$ are the complex amplitudes (or phasor vectors), dependent only on position. Eliminating $\mathbf{H}$ in this system of equations, we obtain the second order Maxwell equation, or the curl-curl formulation

$$\nabla \times \left(\mu^{-1}\nabla \times \mathbf{E}\right) - \omega^2\varepsilon\mathbf{E} + j\omega\sigma\mathbf{E} = -j\omega\mathbf{J}. \tag{2.8}$$

Let the relative magnetic permeability $\mu_r = \mu/\mu_0$, the complex electric permittivity $\tilde{\varepsilon_r} = \varepsilon/\varepsilon_0 - j\frac{\eta}{k_0}\sigma = \varepsilon_r - j\frac{\eta}{k_0}\sigma$, and the source $\mathbf{F} = -j\omega\mu_0\mathbf{J}$, so we obtain a more compact curl-curl formulation

$$\nabla \times \left(\mu_r^{-1}\nabla \times \mathbf{E}\right) - k^2\mathbf{E} = \mathbf{F}, \tag{2.9}$$

where $k^2(\boldsymbol{x}) = k_0^2\tilde{\varepsilon_r}(\boldsymbol{x})$ and k_0 is the free-space wavenumber.

In the case of electromagnetic waves in non-magnetic materials ($\mu_r = 1$) and source free region, and using the vector identity $\nabla \times \nabla \times \mathbf{F} = \nabla\left(\nabla \cdot \mathbf{F}\right) - \nabla^2\mathbf{F}$, Equation (2.9) simplifies to vector Helmholtz equation

$$\nabla^2\mathbf{E} - k^2\mathbf{E} = 0. \tag{2.10}$$

From the mathematical point of view, if k^2 is not an eigenvalue of the homogeneous problem, the solution Equation (2.9) together with suitable boundary conditions, should in principle unequivocally determine the electric field intensity (NÉDÉLEC, 2001; IOANNIDIS; KRISTENSSON; STRATIS, 2012). In practice, however, finding and analytical solutions may be achieved only in a limited number of cases, and therefore one has to use numerical simulation techniques to obtain reliable solutions for the electromagnetic wave field problems.

2.2.1 The Scalar Helmholtz Equation

Although the scalar Helmholtz equation essentially models time-harmonic propagation of linear acoustic waves, it can be a good approximation for time-harmonic electromagnetic waves in some settings. Also, the Helmholtz equation is related to time-harmonic Maxwell's equation, in a sense that every component of the electric **E** and magnetic **H** fields satisfies a form of Helmholtz equation in piecewise homogeneous materials. Therefore, we consider the simpler scalar Helmholtz equation whenever possible, and then we turn to the more complicated vectorial setting. In this way, one understands the analogies and differences between the models, without introducing the technicalities too early in the process.

An extended version of the Helmholtz equation can be written as

$$\nabla \cdot \left(\frac{1}{p(\boldsymbol{x})}\nabla u(\boldsymbol{x})\right) + k_0^2 q(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{2.11}$$

where k_0 is again the free-space wavenumber and p and q are two parameters of the medium. For the TM^z propagation mode, u represents the z-component of the electric field E_z, q represents the relative electric permittivity ε_r and p represents the relative magnetic permeability μ_r. Similarly, for the TE^z mode u, p and q represent H_z (the z-component of the magnetic field), ε_r and μ_r respectively. We also refer to a more compact Helmholtz formulation

$$\Delta u + k^2 u = f, \tag{2.12}$$

where the position vector is omitted and $k(\boldsymbol{x}) = k_0^2 q(\boldsymbol{x})$ a space-dependent wavenumber.

2.2.2 Absorbing Boundary Conditions

Unless explicitly noted, we are referring to problems posed in unbounded domains with exterior incident waves. In those cases, Maxwell's system is completed with the Silver–Müller radiation condition, a generalization of the Sommerfeld radiation condition, expressing the fact that scattered waves must propagate toward infinity only.

In a three-dimensional setting, the Sommerfeld radiation condition for scalar fields reads (GIVOLI, 2008)

$$\lim_{||\boldsymbol{x}||\to\infty} ||\boldsymbol{x}|| \left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||} \cdot \mathbf{grad}\, u + jku \right) = 0, \tag{2.13}$$

and the Silver–Müller (MONK et al., 2003) as

$$\lim_{||\boldsymbol{x}||\to\infty} ||\boldsymbol{x}|| \left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||} \times \nabla \times \mathbf{E} + jk\mathbf{E} \right) = 0, \tag{2.14}$$

for electromagnetic fields.

Equations (2.13) and (2.14) are exactly valid at infinity (JIN, 2015). When computing numerical solutions, however, it is often desirable to truncate the infinite domain. This truncation can be accomplished by introducing an artificial surface ($\partial\Omega_\infty$), to enclose the region of interest. Typical approaches include the use of mathematical boundary conditions and the use of fictitious absorbing material layers. These mathematical boundary conditions are usually referred as absorbing boundary conditions (ABC). Formally, they link the scalar field with its normal derivative at $\partial\Omega_\infty$:

$$\mathbf{n} \cdot \nabla\, u + \mathcal{B}(u) = 0, \tag{2.15}$$

or the the magnetic (**M**) and the electric (**J**) surface currents at $\partial\Omega_\infty$:

$$\mathbf{n} \times \nabla \times \mathbf{E} + \mathcal{B}(\mathbf{E}) = 0, \tag{2.16}$$

where $\mathbf{n}$ is an outward unit normal vector to $\partial\Omega_\infty$, and $\mathcal{B}$ is a carefully chosen surface operator. One crucial point in these approaches is to build $\mathcal{B}$ as approximate representations of the Dirichlet-to-Neumann (TURKEL, 2008) and Magnetic-to-Electric (MtE) (BOUAJAJI; ANTOINE; GEUZAINE, 2014) operators for scalar and vectorial electromagnetic fields respectively. Due to its ease of use, low order absorbing boundary conditions (ABC) have experienced much success in the last decades (de-facto truncation method in distinguished FEM books (MONK et al., 2003; JIN, 2015)). In contrast, high-order absorbing boundary conditions (HABC) provide higher-fidelity solutions for little extra computational cost, with the downside of introducing additional complexity and sometimes auxiliary unknowns. Since ABCs are closely related to transmission conditions in general domain decomposition, we return to this topic in Chapter 3.

One example of low-order ABC for electromagnetics is the impedance boundary condition

$$\frac{1}{Z}\mathbf{n} \times \mathbf{E} - \mathbf{n} \times (\mathbf{H} \times \mathbf{n}) = 0 \tag{2.17}$$

where $Z = \sqrt{\frac{\mu}{\varepsilon}}$ is the wave impedance. Equation 2.17 can be simplified using Equation 2.6, to produce

$$\mathbf{n} \times \nabla \times \mathbf{E} + jk_0 \mathbf{n} \times (\mathbf{E} \times \mathbf{n}) = 0. \tag{2.18}$$

From (2.16) and (2.18) we can recognize the tangential trace operator

$$\mathcal{B}(\mathbf{E}) = jk_0 \mathbf{n} \times (\mathbf{E} \times \mathbf{n}) \tag{2.19}$$

fitting the framework of the finite element method.

Another possible approach encompasses introducing fictitious absorbing material layers around $\partial\Omega_\infty$. One can design these fictitious materials, so that incident waves do not reflect at the interface. Because of this property, they are called perfectly matched layers (PML) and ideally absorb outgoing waves. Introduced by Jean-Pierre Berenger (BERENGER, 1994) over 25 years ago, PMLs behave as arbitrary order absorbing boundary conditions, and its implementation in the frequency domain is straightforward. A good overview of PMLs can be found in this set of notes (NATAF, 2013; JOHNSON, 2008; BÉRENGER, 2007). PMLs have also been used as transmission conditions in DDM, but due to some restrictions in the geometry and convexity of the computational domain, its application was restricted to sweeping type DDM (VION; GEUZAINE, 2014; ENGQUIST; YING, 2011) with regular geometric partition.

2.3 The Finite Element Method

This section briefly reviews the finite element method (FEM) and summarizes some basic concepts and notations used throughout this text. Also, we report some of our developments on automated finite element computations, such as the introduction of complex support and new mesh partitioning schemes to the FEniCs library. To keep the text more compact, we follow the notation of (LOGG; MARDAL; WELLS, 2012) closely, and we recommend, whenever possible, literature associated with implementation aspects. For a formal treatment of the subject, the reader is invited to refer to (BRENNER; SCOTT, 2007).

A methodology for creating discrete algorithms to approximate the solutions of partial differential equations is provided by the finite element technique.The fundamental concept is to break down the computing domain into discrete, or "finite" subdomains, and then apply simple functions to approximate the unknown solution across each element. The mathematical abstraction used in the finite element helps to reason about the problem and provides systematic ways of deriving practical computer implementations. However, it has a price in terms of complexity, especially for vector fields, as discussed later in this chapter.

The first application of the FEM to electromagnetics was perhaps the solution of homogeneous waveguide problems nearly 50 years ago (SILVESTER, 1969). Since then, the method was successfully applied to a variety of electrostatic, magnetostatic, and two-dimensional scalar problems (WEBB, 1995). However, not until the 1980s, with the development of edge-based vector elements, (NÉDÉLEC, 1980; BOSSAVIT; VERITE, 1982; BARTON; CENDES, 1987), it became an effective numerical technique in computational electromagnetics for solving curl-conforming vector field problems, such as electromagnetic scattering problems.

2.3.1 Scalar Finite Element

We start our overview of the standard (scalar) finite element method with an illustrative example. To derive the weak form for the Helmholtz's equation (2.12), we first multiply both sides of the equation with the complex conjugate of a sufficiently smooth arbitrary test function v, integrate by parts in Ω, the domain of interest, and after applying the divergence theorem, we find

$$-\int_\Omega \nabla u \cdot \nabla \bar{v}\, dx + \int_\Omega k^2 u\, \bar{v}\, dx + \int_{\partial\Omega} (\nabla u \cdot \mathbf{n})\, \bar{v}\, ds = \int_\Omega f\, \bar{v}\, dx. \tag{2.20}$$

Assuming that u is a classical solution of (2.12) with suitable boundary conditions, it is also a solution of (2.20) for any $v \in C_0^1(\Omega)$, nevertheless with a reduced smoothness requirement. If $\Omega \in \mathbb{R}^d$, $d = 1, 2, 3$, then the natural space for the weak solution of (2.20) and the test functions v is the Sobolev space $\mathcal{H}^1(\Omega)$, given by

$$\mathcal{H}^1(\Omega) := \{u : \Omega \to \mathbb{C} |\, u \in L^2(\Omega),\, \partial_{x_i} u \in L^2(\Omega), 1 \le i \le d\}. \tag{2.21}$$

Assuming that the test function v vanishes on Γ_D, where the solution u is known, we arrive at the following variational problem:

Find $u \in V$ such that

$$-\int_\Omega \nabla u \cdot \nabla \bar{v}\, dx + \int_\Omega k^2 u\, \bar{v}\, dx + \int_{\partial\Omega/\Gamma_D} (\nabla u \cdot \mathbf{n})\, \bar{v}\, ds = \int_\Omega f\, \bar{v}\, dx \qquad \forall v \in \widehat{V}. \tag{2.22}$$

To complete (2.22) we can then define the test $\widehat{V}$ and trial V spaces as

$$\widehat{V} := \{v \in \mathcal{H}^1(\Omega) : \ v = 0 \text{ on } \Gamma_d\}, \tag{2.23}$$

$$V := \{u \in \mathcal{H}^1(\Omega) : \ u = u_0 \text{ on } \Gamma_d\}. \tag{2.24}$$

A central abstraction of weak formulations of PDEs are the integrals, also called *forms*. In order to use a more abstract formalism, we define the complex-valued sesquilinear form $a(.,.)$

as the map $V \times \widehat{V} \to \mathbb{C}$, that is linear in one argument and anti-linear (or conjugate-linear) in the other, and the complex-valued anti-linear form $L()$ as the map $L : \widehat{V} \to \mathbb{C}$. So our linear variational problem (2.22) can be re-written in the following canonical form:

Find $u \in V$ such that

$$a(u, v) = L(v), \quad \forall v \in \widehat{V}, \tag{2.25}$$

In 2.25, the sesquilinear and anti-linear forms are defined by:

$$a(u, v) = -\int_\Omega \nabla u \cdot \nabla \bar{v}\, dx + \int_\Omega k^2 u\, \bar{v}\, dx + \int_{\partial\Omega} (\nabla u \cdot \mathbf{n})\, \bar{v}\, ds \tag{2.26}$$

$$L(f; v) = \int_\Omega f v\, dx. \tag{2.27}$$

This notation is standard in the study of partial differential equations and finite element methods. Problem (2.25) however is stated in an infinite-dimensional space V. To derive the finite-dimensional (numerical) approximation of such problems, we use the Galerkin procedure. For this purpose, we restrict our function spaces to a pair of discrete trial and test spaces. So the Galerkin solution is the function $u_h \in V_h \subset V$ that satisfies a finite-dimensional version of (2.25):

$$a(u_h, v) = L(v), \quad \forall v \in \widehat{V}_h \subset \widehat{V}. \tag{2.28}$$

Strictly speaking, the finite element method does not discretize the equations but the solution space. Let $V_h \subset V$ be a finite-dimensional space, and let $\{\phi_i\}_{i=0}^{N-1}$ be its basis, where N is the dimension of the finite-dimensional space. For simplicity, let $u_0 = 0$, so $\hat{V}_h = V_h$. The solution $u \in V_h$ can be written as

$$u(x) = \sum_{i=0}^{N-1} u_i \phi_i(x) \tag{2.29}$$

where $\mathbf{u} = [u_0,\ u_1, \cdots,\ u_{N-1}]$ is a vector coefficient. Since (2.28) holds for any $v \in V_h$, we can choose $v(x) = \sum_{i=0}^{N-1} \phi_i(x)$. Substituting the expanded form of u and v in Equation (2.28), we get a linear system

$$\mathrm{A}\mathbf{u} = \mathbf{b}$$

with

$$\mathrm{A}_{ij} = a\left(\phi_i, \phi_j\right),$$
$$\mathbf{b}_i = L\left(\phi_i\right).$$

A is the bilinear form a assembled as a matrix, and $\mathbf{b}$ is the linear form L assembled as a vector. In this context, finite element assembly means the numerical evaluation of a matrix A or a vector $\mathbf{b}$ for some given discrete function space V_h and forms a and L. Then the solution u is obtained by "simply" solving the linear system $\mathrm{A} \cdot \mathbf{u} = \mathrm{b}$. The error of the obtained approximate solution is minimized in the energy norm and orthogonal to the solution space V_h, (BRENNER; SCOTT, 2007).

We have assumed that we could construct discrete subspaces $V_h \subset V$. The creation of such subspaces by joining local function spaces that are defined by a collection of finite elements is a key component of the finite element method. The solution of the discrete problem then relies on the definition of a finite element. In this work, we follow Ciarlet's definition (CIARLET, 2002), also used in (LOGG; MARDAL; WELLS, 2012; BRENNER; SCOTT, 2007).

Definition 2.3.1. *(Finite Element) A finite element is defined by a triple $(K, \mathcal{V}, \mathcal{L})$, where*

- *$K \in \mathbb{R}^d$ is the element domain, e.g. a polyhedron in 3d.*
- *$\mathcal{V} = \mathcal{V}(K)$ is a finite dimensional function space on K of dimension n. e.g. a polynomial of order p in K.*
- *$\mathcal{L} = \{\ell_0,\, \ell_1, \cdots \ell_{n-1}\}$ is the set of degrees of freedom, a basis for the dual space $\mathcal{V}'$.*

The basis functions $\{\phi_0, \ldots \phi_{n-1}\}$ of the space $\mathcal{V}$ are defined by

$$\ell_i\left(\phi_j\right) = \delta_{ij} \tag{2.30}$$

where δ_{ij} is the Kronecker delta. For disambiguation, we define the dimension global function space with a capital N and the dimension of the finite element with lower case n. To define the global discrete function space V_h, we first partition the domain Ω into a finite set of cells $\mathcal{T}_h = \{K\}$ with disjoint interiors such that

$$\bigcup_{K \in \mathcal{T}_h} K = \Omega. \tag{2.31}$$

The set of cells $\mathcal{T}_h$ is known as triangulation or, more commonly, mesh. By decomposing the evaluation of the forms to single cells of the mesh and later assembling the contribution of each element into a global linear system, the problem can be solved computationally, as we will show later in this chapter.

Returning to our weak form (2.20) or (2.28), we now choose a finite element with a suitable local basis function. One of the most critical parts is to find a set of basis functions that can be used to expand the unknown solution. The best-known and most widely used finite element for discrete conforming $V_h \subset \mathcal{H}^1$ is the classic Lagrange element on simplices. The Lagrange element of arbitrary degree q is defined by:

$$\begin{cases} K \in \{\text{interval, triangle, tetrahedron}\} \\ \mathcal{V} = \mathcal{P}_q(K) \\ \ell(v) = v(\boldsymbol{x}^i), \quad i = 0, \ldots, n-1 \end{cases} \tag{2.32}$$

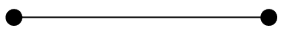
The dimension $n(q)$ of the Lagrange finite element depends on the order q of the polynomial on K. Basically, this element expands the coefficients with polynomials of order q, and the degrees of freedom are point evaluations. Examples of Lagrange finite elements, with their degrees of freedom coordinates on simplices can be found in Figure 2, Figure 3 and Figure 4, for the interval, triangle and tetrahedron cells respectively. Below we also give the example of a triangular Lagrange element based on Definition 2.3.1.

Example 2.3.1 (Lagrange element $q = 1$ on Tetrahedron)**.** *:*
Consider the reference tetrahedron cell K, as shown in 5, and let $\mathcal{P}_1(K)$ be the space of polynomials of order $q = 1$ on K. Let $\mathcal{L}$ be the set of bounded linear functionals representing point on the nodes, with coordinates x^i for $i = 1, \cdots, 4$, such that

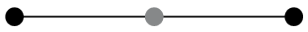
$$\mathcal{V} := \mathrm{Span}\,\{1,\ x_0,\ x_1\, x_2\}$$
$$\ell_i = \mathcal{P}_1 \to \mathbb{R}$$
$$\ell_i(v) = v(\boldsymbol{x}^i)$$

And the nodal basis for the linear Lagrange element with vertices at $\boldsymbol{x}^0 = (0,0,0)$, $\boldsymbol{x}^1 = (1,0,0)$, $\boldsymbol{x}^2 = (0,1,0)$, $\boldsymbol{x}^3 = (0,0,1)$ is given by

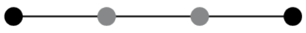
$$\begin{aligned}
\phi_0(\boldsymbol{x}) &= 1 - x_0 - x_1 - x_2 \\
\phi_1(\boldsymbol{x}) &= x_0 \\
\phi_2(\boldsymbol{x}) &= x_1 \\
\phi_3(\boldsymbol{x}) &= x_2
\end{aligned}$$



(a) $\mathcal{P}_1, n = 2$

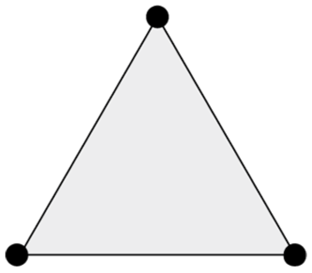


(b) $\mathcal{P}_2, n = 3$

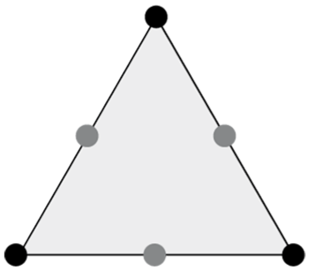


(c) $\mathcal{P}_3, n = 4$

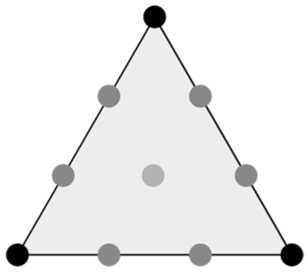
Figure 2 – Examples of the Lagrange $\mathcal{P}_q$ finite element on interval. $n(q) = (q + 1)$.



(a) $\mathcal{P}_1, n = 3$

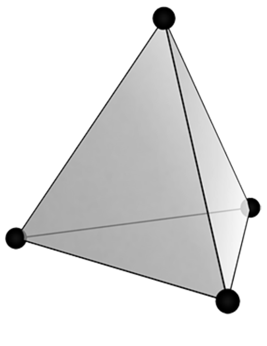


(b) $\mathcal{P}_2, n = 6$

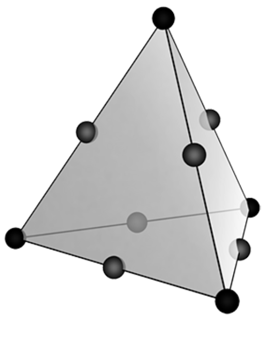


(c) $\mathcal{P}_3, n = 10$

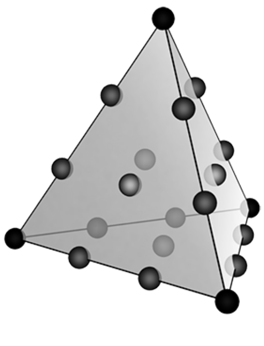
Figure 3 – Examples of the Lagrange $\mathcal{P}_q$ finite element on triangle. $n(q) = 1/2(q + 1)(q + 2)$



(a) $\mathcal{P}_1, n = 4$



(b) $\mathcal{P}_2, n = 10$



(c) $\mathcal{P}_3, n = 20$

Figure 4 – Examples of the Lagrange $\mathcal{P}_q$ finite element on tetrahedron. $n(q) = \frac{1}{6}(q + 1)(q + 2)(q + 3)$

2.3.1.1 Global Assembly

A fundamental step of implementing finite element methods is the computation of matrices and vectors from variational forms. Recall that the sesquilinear form is expressed as an

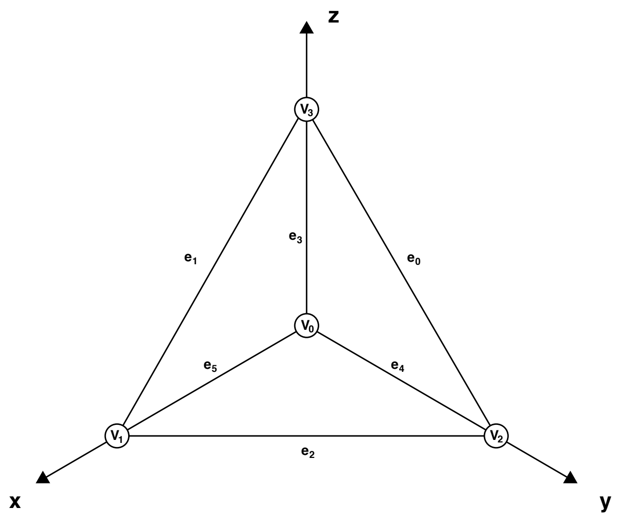


Figure 5 – Definition an ordering of a reference tetrahedron. The vertices are given by: $\mathbf{v}_0 = (0,0,0)$, $\mathbf{v}_1 = (1,0,0)$, $\mathbf{v}_2 = (0,1,0)$, $\mathbf{v}_3 = (0,0,1)$.

integral over Ω and so it may be naturally decomposed as a sum of local contributions from each cell. Each global basis function ϕ_i is supported only on a few adjacent cells, so the global matrix A is sparse, and every cell can be visited only once during the assembly process. Let $K \in \mathcal{T}_h$, and $\left\{\phi_i^K\right\}_{i=0}^{n-1}$ be the restriction K of the subset of $\{\phi_i\}_{i=0}^{N-1}$ supported on the cell K, so we can define a dense $n \times n$ local matrix A^K

$$\mathrm{A}_{ij}^K = a_K\left(\phi_i^K, \phi_j^K\right). \tag{2.33}$$

We may now compute the matrix A by accumulating these local contributions in the corresponding entries according to a local to global map, or *dofmap* $\iota(\cdot,\cdot) : [0, n-1] \to [0, N-1]$. The *dofmap* performs a mapping from the local $\left\{\phi_i^K\right\}_{i=0}^{n}$ to the global numbering of $\{\phi_i\}_{i=1}^{n}$, so that ϕ_i^K is the restriction to K of $\phi_{\iota(K,i)}$. This process called global assembly and it is described in Algorithm 1. An example of implementation in C++ is shown in Listing 1, which considers that a `kernel` for computing 2.33 is available.

Algorithm 1 Global assembly

```
1: $A \leftarrow 0$
2: **for** K in $\mathcal{T}_h$ **do**
3:     Compute $A^K$
4:     **for** $i$ in $[0,\ N_l)$ **do**
5:         **for** $j$ in $[0,\ N_l)$ **do**
6:             $\mathbf{A}_{\iota(K,i)\iota(K,j)} \leftarrow \mathbf{A}_{\iota(K,i)\iota(K,j)} + \mathbf{A}_{ij}^K$
7:         **end for**
8:     **end for**
9: **end for**
```

To manage a mesh and a *dofmap* in parallel, we use DOLFINx, the computational backend of the FEniCS Problem Solving Environment. In addition to a distributed mesh, DOLFINx needs

Listing 1 Snippet of an implementation of Algorithm 1 in C++

```cpp
for (int c = 0; c < Ncells; c++)
{
  // Get cell coordinates/geometry
  auto x_dofs = x_dofmap.links(c);  // cell vertices
  for (int i = 0; i < d; ++i)
    for (int j = 0; j < gdim; ++j)
      x_cell(i, j) = x_g(x_dofs[i], j);

  // Tabulate tensor - equation 2.33
  std::fill(Ae.begin(), Ae.end(), 0);
  kernel(Ae, coeffs.row(c), constants, x_cell);

  // Insert local tensor Ae into a CSR Matrix A
  // Lines 4 to 8 of Algorithm 1
  auto dofs = dofmap.links(c);
  for (int i = 0; i < Ae.shape(0); i++)
  {
    auto inds = A.row_indices(dofs[i]);
    auto&& data = A.row(dofs[i]);
    for (int j = 0; j < Ae.shape(1); j++)
    {
      auto it = std::lower_bound(inds.begin(), inds.end(), dofs[j]);
      auto pos = std::distance(inds.begin(), it);
      data[pos] += Ae(i, j);
    }
  }
}
```

a reference element to construct a function space and the corresponding *dofmap*. The reference element is a finite element on the reference cell $\hat{K}$, see for example Figure 5 for the reference tetrahedron and Example 2.3.1 for the basis functions of a $\mathcal{P}_1$ Lagrange element.

Let $\left\{\hat{\phi}_i\right\}_{i=0}^{n-1}$ be the basis of the reference element. We assume that each $K \in \mathcal{T}_h$ can be obtained by mapping $\hat{K}$ using an affine mapping $F_K : \hat{K} \to K$. Since we consider simplices the coordinate map can be written as:

$$\boldsymbol{x} = F_K \hat{\boldsymbol{x}} = J_K \hat{\boldsymbol{x}} + \mathbf{b}_K \tag{2.34}$$

where J_K is a $d \times d$ matrix, and $\mathbf{b}_K$ is a vector. Then the global basis is constructed so that its restriction on a given cell K amounts to transferring the reference basis to that cell. By this, we mean that we can write

$$\phi_i^K = \hat{\phi}_i \circ F_K^{-1} \tag{2.35}$$

for any $K \in \mathcal{T}_h$, where $\circ$ is a composition of functions.

2.3.1.2 Local Assembly

In the previous section, we considered the global assembly process that assumes the availability of a local tensor for each cell. In this section, we will investigate the computation of these local tensors. To exemplify local assembly we consider the original sesquilinear form (2.26) with Dirichlet boundary conditions on $\partial\Omega$. In this case, the last term vanishes since, in the definition of our function space, we choose $v = 0$ on Γ_d. Then for each local domain $K \in \mathcal{T}_h$, Equation (2.26) can be written as:

$$a_K(v, u) = -\int_K \nabla u \cdot \nabla \bar{v}\, dx + \int_K k^2 u\, \bar{v}\, dx, \tag{2.36}$$

and consequently as:

$$\mathrm{A}_{ij}^K = -\int_K \nabla \phi_i^K \cdot \nabla \phi_j^K \,\mathrm{d}x + \int_K k^2 \phi_i^K \cdot \phi_j^K \,\mathrm{d}x.$$

In many finite element codes, a common practice is to map the basis functions from the reference cell to each cell in the mesh and perform the integration on the reference domain. This implies scaling the integral accordingly by $|\det(J_K)|$, where $J_K = \hat{\nabla} F_K$ is the Jacobian of the transformation, and $\hat{\nabla}$ indicates differentiation with respect to $\hat{\boldsymbol{x}}$. Equation (2.35) shows the pullback of basis functions, which for Lagrange elements simplifies to

$$\phi_i^K(\boldsymbol{x}) = \hat{\phi}_i(\hat{\boldsymbol{x}}). \tag{2.37}$$

The pullback of the gradients can obtained via application of the chain rule:

$$\nabla \phi_i^K = J_K^{-T} \hat{\nabla} \hat{\phi}_i \circ F_K^{-1}. \tag{2.38}$$

Then we can write the local kernel in Equation (2.36) as:

$$\mathrm{A}_{ij}^K = -\int_{\hat{K}} \left(J_K^{-T} \hat{\nabla} \hat{\phi}_i \cdot J_K^{-T} \hat{\nabla} \hat{\phi}_j \right) |\det(J_K)| \,\mathrm{d}\hat{x} + \int_{\hat{K}} k^2 \hat{\phi}_i \hat{\phi}_j \,|\det(J_K)| \,\mathrm{d}\hat{x}. \tag{2.39}$$

Note that k can also be taken from a distinct finite element function space with an arbitrary reference basis. However, we restrict to piecewise constant coefficients, and k can be pulled from the integration in Equation (2.39). Also the integration over $\hat{K}$ in (2.39) is in general performed by numerical quadrature.On common finite element textbooks, as (SOLIN; SEGETH; DOLEZEL, 2003), one may find quadrature rules for a variety of reference elements. Let $\{\hat{\boldsymbol{x}}_q\}_{q=0}^{n_q-1}$ be a set of quadrature points taken on the reference cell, and let $\{w_q\}_{q=0}^{n_q-1}$ be the corresponding quadrature weights, defining a quadrature rule. Then each integration term in (2.39) can be approximated by a weighted sum:

$$\mathbf{A}_{ij}^K = \mathbf{B}_{ij}^K + k^2 \mathbf{M}_{ij}^K \tag{2.40}$$

$$\mathbf{B}_{ij}^K = \sum_{q=0}^{n_q-1} w_q \left(J_K^{-T}(\hat{\boldsymbol{x}}_q)\, \hat{\nabla} \hat{\phi}_i(\hat{\boldsymbol{x}}_q) \cdot J_K^{-T}(\hat{\boldsymbol{x}}_q)\, \hat{\nabla} \hat{\phi}_j(\hat{\boldsymbol{x}}_q) \right) |\det(J_K(\hat{\boldsymbol{x}}_q))| \tag{2.41}$$

$$\mathbf{M}_{ij}^K = \sum_{q=0}^{n_q-1} w_q \hat{\phi}_i(\hat{\boldsymbol{x}}_q) \cdot \hat{\phi}_j(\hat{\boldsymbol{x}}_q)\, |\det(J_K(\hat{\boldsymbol{x}}_q))|\,. \tag{2.42}$$

The local tensors $\mathbf{B}^K$ and $\mathbf{M}^K$ can be further simplified assuming an affine map, so that J_K is piecewise constant, and we can write:

$$\mathbf{B}_{ij}^K = \sum_{q=0}^{n_q-1} w_q \left(J_K^{-T} \hat{\nabla} \hat{\phi}_i \left(\hat{\boldsymbol{x}}_q \right) \cdot J_K^{-T} \hat{\nabla} \hat{\phi}_j \left(\hat{\boldsymbol{x}}_q \right) \right) \left| \det \left(J_K \right) \right| \tag{2.43}$$

$$\mathbf{M}_{ij}^K = \sum_{q=0}^{n_q-1} w_q \hat{\phi}_i \left(\hat{\boldsymbol{x}}_q \right) \cdot \hat{\phi}_j \left(\hat{\boldsymbol{x}}_q \right) \left| \det \left(J_K \right) \right| \tag{2.44}$$

In Equations (2.43) and (2.44), there is only reference to the basis functions of the reference element, so they can be precomputed and reused for all cells in the mesh. Basix is a finite element definition and tabulation run-time library that implements a wide range of arbitrary-order finite elements and can provide numerical tensors with the evaluations of basis functions and their derivatives at specified points (PROJECT, 2021). For our example Basix can provide the numerical tensors $\hat{\boldsymbol{\phi}}$ and $D\hat{\boldsymbol{\phi}}$ such that

$$\hat{\boldsymbol{\phi}}_{iq} := \hat{\phi}_i \left(\hat{\boldsymbol{x}}_q \right), \tag{2.45}$$

$$D\hat{\boldsymbol{\phi}}_{iqk} := \frac{\partial \hat{\phi}_i}{\partial \hat{x}_k} \left(\hat{\boldsymbol{x}}_q \right). \tag{2.46}$$

Listing 2 Complete C++ function to tabulate basis functions for order p Lagrange elements on tetrahedron using order q Gauss quadrature rule.

```cpp
#include <basix/finite-element.h>
#include <basix/quadrature.h>

using namespace basix;
// Tabulate order "p" Lagrange basis functions on a Tetrahedron
// with quadrature rule of order "q".
auto tabulate_basis(int p, int q) {
  // Define element family and cell type
  auto family = element::family::P; // Lagrange elements
  auto cell = cell::type::tetrahedron;

  // Tabulate quadrature points and weights
  auto [points, weights] = quadrature::make_quadrature("default", cell,
↪  q);

  // Define element and evaluate basis functions on quadrature points
  auto element = basix::create_element(family, cell, p);
  auto basis = element.tabulate(0, points);
  auto phi = view(basis, 0, all(), all(), 0);
  return phi;
}
```

For reference, we provide a complete function to tabulate basis functions for order p Lagrange elements on tetrahedron using order q Gauss quadrature rule on Listing 2. Since the coordinate map is also represented as a finite element function, the evaluation of F_K and the

Listing 3 Example of UFL representation of form (2.36) with first-order Lagrange elements on a triangle

```
element = FiniteElement("Lagrange", triangle, 1)
u = TrialFunction(element)
v = TestFunction(element)
k = Coefficient(FiniteElement("DG", triangle, 0))
a = -inner(grad(u), grad(v))*dx + k**2*inner(u, v)*dx
```

Jacobian J_K can be evaluated using the tools introduced for u and v. Let $\left\{\hat{\theta}_i\right\}_{i=0}^{n_c-1}$ be a basis for the coordinate space and $\mathbf{x}^K$ be the vector coefficients, the Jacobian can be expressed as:

$$J_K(\hat{\boldsymbol{x}}) = \sum_{i=0}^{n_c-1} \mathbf{x}_i^K \hat{\nabla}\hat{\theta}_i(\hat{\boldsymbol{x}}). \tag{2.47}$$

Using the tensor representation, and the fact that the $\hat{\nabla}\hat{\theta}_i(\hat{\boldsymbol{x}})$ is piecewise constant, J_K can be computed efficiently as a single small matrix-matrix product for each cell. For instance, let $D\hat{\boldsymbol{\theta}}$ be the tabulation matrix of the coordinate element for any point in the cell and let $\mathbf{x}^K$ be a vector coefficient expansion, the coordinates of cell K; then the Jacobian can be computed as:

$$J_K = \mathbf{x}^K \cdot D\hat{\boldsymbol{\theta}}. \tag{2.48}$$

Although it is crucial from a didactic point of view to go through all the formulation steps, we use automatic code generation for most of our development. Instead of writing (2.43), (2.43), and (2.47) explicitly we use a Domain-specific Language (DSL) specialized in the representation of variational forms. Listing 3 shows the representation of Equation (2.36) with first-order Lagrange elements on a triangle using the Unified Form Language (UFL) (ALNÆS et al., 2014). FFCx (FENICSPROJECT, 2019), the new version of the FEniCS Form Compiler, then takes this high-level UFL specification to generate efficient low-level C code that can be used to assemble the corresponding discrete operator. An excerpt of the code generated for this is shown in Listing 4.

In previous versions of FEniCS libraries, it was not possible to represent variational forms with complex numbers. During this Ph.D., to introduce complex support, we made contributions of varying degrees on diverse FEniCSx components, namely DOLFINx, FFCx, and UFL. The goal was to open up the possibility of the solution of large-scale complex-valued PDEs using FEniCS. A report with some of these changes, presented in the context of the Google Summer of Code, is in Appendix A.

Listing 4 Example of kernel generated by FFCx

```
void kernel(double complex* A,
            double complex* w,
            double complex* c,
            const double* x)
{
    // Quadrature rules
    static const double weights[3] =
            { 0.1666667, 0.1666667, 0.166667 };
    // Precomputed values of basis functions
    // The gradient of 1st order Lagrange is piecewise-constant
    static const double dphi_x[3] = { -1.0, 0.0, 1.0 };
    static const double dphi_y[3] = { { -1.0, 1.0, 0.0 } };
    // Three degrees of freedom an three quadrature points
    static const double phi[3][3] =
                 { { 0.666667, 0.166667, 0.166667 },
                  { 0.166667, 0.176666, 0.766665 },
                  { 0.166667, 0.666667, 0.166667 } };

    // Quadrature loop independent computations
    double sp[80] ...
    const double J_c0 = x[0] * dphi_x[0][0][0][0] + ...;
    const double J_c3 = x[1] * dphi_y[0][0][0][0] + ...;
    const double J_c1 = x[0] * dphi_y[0][0][0][0] + ...;
    const double J_c2 = x[1] * dphi_x[0][0][0][0] + ...;
    ...
    // Temporary variables
    double complex t0[3];
    double complex t1[3];
    double complex t2[3];
    for (int iq = 0; iq < 3; ++iq)
    {
        const double complex fw0 = sp[20] * weights[iq];
        const double complex fw1 = sp[19] * weights[iq];
        const double complex fw2 = sp[18] * weights[iq];
        const double complex fw3 = sp[17] * weights[iq];
        for (int i = 0; i < 3; ++i)
        {
            t0[i] = fw0 * phi[iq][i];
            t1[i] = fw1 * dphi_x[i] + ...;
            t2[i] = fw2 * dphi_x[i] + ...;
        }
        for (int i = 0; i < 3; ++i)
            for (int j = 0; j < 3; ++j)
                A[3 * i + j] += phi[iq][j] * t0[i]
                              + dphi_x[j] * t1[i]
                              + dphi_y[j] * t2[i];
    }
}
```

2.3.2 Vector-valued Finite Element

As for the scalar case, the weak formulation is obtained by taking the dot product of the original Equation (2.9) by the complex conjugate of a sufficiently smooth test function $\mathbf{v}$ and then integrating over the domain (Ω):

$$\int_\Omega \mu_r^{-1} (\nabla \times \mathbf{E}) \cdot (\nabla \times \bar{\mathbf{v}})\ dx - \int_\Omega k^2 \mathbf{E} \cdot \bar{\mathbf{v}}\ dx + \int_{\partial\Omega} \mathbf{n} \times (\mu_r^{-1} \nabla \times \mathbf{E}) \cdot \bar{\mathbf{v}}\ ds = \int_\Omega \mathbf{F} \cdot \bar{\mathbf{v}}\ dx \quad (2.49)$$

where again $\partial\Omega$ denotes the boundary of Ω. Under the assumption of a finite-energy system, the electric and magnetic fields should be square-integrable. Furthermore, together with (2.7), it also implies that $\nabla \times \mathbf{E}$ should be square-integrable. If $\Omega \in \mathbb{R}^3$, then the natural space for the weak solution of (2.49) and for the test functions $\mathbf{v}$ is the Sobolev Space of vector-valued functions $\mathcal{H}(\mathbf{curl}, \Omega)$, given by

$$\mathcal{H}(\mathbf{curl}, \Omega) := \{\mathbf{E} : \Omega \to \mathbb{C}^3 |\, \mathbf{E} \in [L^2(\Omega)]^3,\ \nabla \times \mathbf{E} \in [L^2(\Omega)]^3\}. \quad (2.50)$$

Assuming homogeneous Dirichlet boundary condition on $\Gamma_d \subset \partial\Omega$, that models Perfect Electrical Conductors (PEC), we can write our trial and test spaces as:

$$V = \{\mathbf{u} \in \mathcal{H}(\mathbf{curl}, \Omega) |\, \mathbf{n} \times \mathbf{u} = 0 \text{ on } \Gamma_d\} \quad (2.51)$$

and then we can then reformulate Equation (2.49) in the conventional abstract form, to subsequently apply the Galerkin procedure for discretization. The discrete formulation then reads

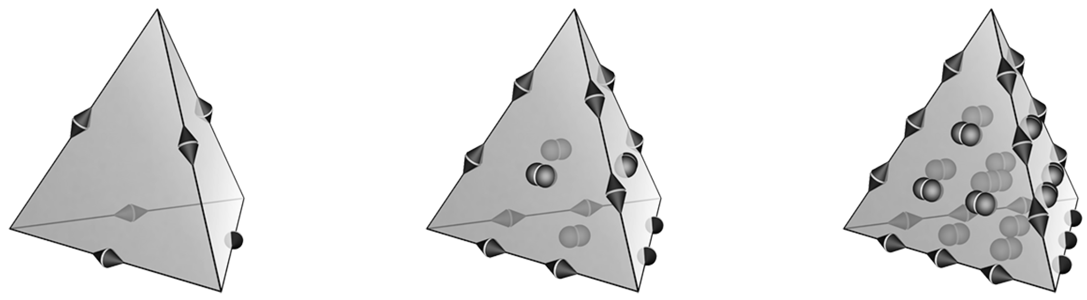
Find $\mathbf{E} \in V_h \subset V$, such that

$$a(\mathbf{E}, \mathbf{v}) = L(\mathbf{F}; \mathbf{v}), \quad \forall \mathbf{v} \in \widehat{V}_h. \subset V. \quad (2.52)$$

Again we assume that it is possible to create a discrete subspace V_h from V. However, the properties of $V \subset \mathcal{H}(\mathbf{curl})$ make this task more complicated. We discuss this briefly for the sake of completeness, but the reader is referred to (MONK et al., 2003) for a thorough discussion. The $\mathcal{H}(\mathbf{curl})$ can be understood as a space of finite energy solutions and belongs to a complex of functional spaces, often called *de Rham complex* (MONK et al., 2003; DEMKOWICZ, 2006):

$$\mathbb{R} \longrightarrow \mathcal{H}^1 \xrightarrow{\nabla} \mathcal{H}(\mathbf{curl}) \xrightarrow{\nabla\times} \mathcal{H}(div) \xrightarrow{\nabla\cdot} L^2. \quad (2.53)$$

In this sequence, the range of an operator corresponds to the null space of the following operator. The presence of $\mathbb{R}$ in the sequence means just that the null space of the gradient consists of constant fields. Also, the curl of a function is zero if and only if the function is the gradient of a scalar potential. The exact sequence property is essential in establishing the stability of the variational formulation (2.52) (DEMKOWICZ, 2006), which is beyond the scope of the project. However, this structure suggests constructing (piecewise) polynomial finite element discretizations of the $\mathcal{H}(\mathbf{curl})$ space so that the exact sequence property is also satisfied on



(a) $N1^e_1, n = 6$ (b) $N1^e_2, n = 20$ (c) $N1^e_3, n = 45$

Figure 6 – Examples of the Nédélec elements of the first kind $N1^e_q$ on tetrahedron. $n(q) = 1/2\, q(q+2)(q+3)$.

the discrete level. Electric and magnetic fields, electric and magnetic inductions, and electric charge density are all often discretized using finite element spaces in $\mathcal{H}(\textbf{curl})$, $\mathcal{H}(div)$, and L^2 respectively.

Nédélec introduced two families of elements that satisfy such a sequence, the elements of the first kind in 1980 (NÉDÉLEC, 1980), and the elements of the second kind six years later (NÉDÉLEC, 1986). These elements are constructed such that their tangential components are continuous across element facets, though their normal components are allowed to be discontinuous. In both papers, Nédélec considered only the three-dimensional case. Following Definition 2.3.1, the Nédélec element of the first kind of arbitrary order q can be described by

$$\begin{aligned}
K &= \text{tetrahedron} \\
\mathcal{V} &= [\mathcal{P}_{q-1}(K)]^3 + \mathcal{S}_q(K) \\
\mathcal{L} &= \left[\begin{array}{ll}
\int_e v \cdot t\, \phi\, dl & \phi \in \mathcal{P}_{q-1}(e) \text{ for each edge e} \\
\int_f v \times n \cdot \phi\, df & \phi \in [\mathcal{P}_{q-2}(f)]^2 \text{ for each face} f, \text{ for } q \geq 2 \\
\int_K v \cdot \phi\, dx & \phi \in [\mathcal{P}_{q-3}(K)]^3, \text{ for } q \geq 3.
\end{array}\right.
\end{aligned} \tag{2.54}$$

where $\mathcal{S}_q(K)$:

$$\mathcal{S}_q(K) = \{\mathbf{s} \in [\mathcal{P}(K)]^3 : \mathbf{s}(\boldsymbol{x}) \cdot \boldsymbol{x} = 0,\ \forall \boldsymbol{x} \in K\}.$$

Nédélec degrees of freedom are associated not only with the edges (e), but also with faces (f) and the cell (K) itself. We show some examples of first kind elements on tetrahedrons in Figure 6. For example, the functionals that define a first-order Nédélec element of the first kind on the reference tetrahedron K (defined in Figure 5) are the tangential integral moments on the edges.

The functionals that define a first-order Nédélec element on the reference tetrahedron K (defined in Figure 5) are the tangential integral moments on the edges $(i = 0, \cdots, 5)$

$$l_i : \boldsymbol{v} \mapsto \int_{e_i} \boldsymbol{v} \cdot (1)\boldsymbol{t}_i \tag{2.55}$$

and the corresponding basis functions are defined by:

$$\begin{aligned}
\hat{\boldsymbol{\phi}}_0(\boldsymbol{x}) &= \begin{pmatrix} 0 \\ -x_2 \\ x_1 \end{pmatrix} & \hat{\boldsymbol{\phi}}_1(\boldsymbol{x}) &= \begin{pmatrix} -x_2 \\ 0 \\ x_0 \end{pmatrix} \\
\hat{\boldsymbol{\phi}}_2(\boldsymbol{x}) &= \begin{pmatrix} -x_1 \\ x_0 \\ 0 \end{pmatrix} & \hat{\boldsymbol{\phi}}_3(\boldsymbol{x}) &= \begin{pmatrix} x_2 \\ x_2 \\ -x_0 - x_1 + 1 \end{pmatrix} \\
\hat{\boldsymbol{\phi}}_4(\boldsymbol{x}) &= \begin{pmatrix} x_1 \\ -x_0 - x_2 + 1 \\ x_1 \end{pmatrix} & \hat{\boldsymbol{\phi}}_5(\boldsymbol{x}) &= \begin{pmatrix} -x_1 - x_2 + 1 \\ x_0 \\ x_0 \end{pmatrix}.
\end{aligned} \tag{2.56}$$

The basis functions are also shown in Figure 7. Each basis is associated with one edge of the tetrahedron, and its tangential component tends to zero on the other edges, whereas the normal component is nonzero. Any function **p** in $\mathcal{V}_1$ can be written in terms of the basis:

$$\mathbf{p}(\hat{\boldsymbol{x}}) = \sum_{i=0}^{n-1} \alpha_i \hat{\boldsymbol{\phi}}_i(\hat{\boldsymbol{x}}) \tag{2.57}$$

where $\alpha_i \in \mathbb{C}$ is the expansion coefficients. These basis functions can then be used in finite element computations for assembling matrices and vectors.

Given that the degrees of freedom are connected to edges, facets, and volumes, high-order Nédélec elements have more difficult definitions. For instance, second-order Nédélec elements have 20 degrees of freedom, $\mathcal{L}_2 = \{l_0, ..., l_{19}\}$; two degrees of freedom associated with each edge (6 edges) and each face (4 faces). The two basis functions associated with face $\mathbf{f}_0$ $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ are shown in Figure 8. Once we define the basis functions for $\mathcal{V}_2$, , we can write a general finite element function in $\hat{K}$ using Equation (2.57).

As for the scalar case, we use DOLFINx to manage the distributed mesh and construct the *dofmap*. For Maxwell's case, we only consider tetrahedral meshes, where each $K \in \mathcal{T}_h$ is a tetrahedron and can be obtained by mapping $\hat{K}$ with an affine map. For any $K \in \mathcal{T}_h$ there is a map $F_K(\hat{K}) = K$ such that

$$\boldsymbol{x} = F_K \hat{\boldsymbol{x}} = J_K \hat{\boldsymbol{x}} + \mathbf{b}_K$$

where J_K is the Jacobian of the transformation, a cell-wise constant 3×3 matrix, and $\mathbf{b}_K$ is a vector. Because we are working in the $\mathcal{H}(curl)$ space, we need continuity-preserving mappings, such as the covariant Piola mapping. Let $\left\{\hat{\boldsymbol{\phi}}_i\right\}_{i=0}^{n-1}$ be the basis of the reference element, and let $\left\{\boldsymbol{\phi}_i^K\right\}_{i=0}^{n-1}$ be the basis functions of an arbitrary cell $K \in \mathcal{T}_h$ the covariant Piola

$$\boldsymbol{\phi}_i^K \circ F_K = J_K^{-T} \hat{\boldsymbol{\phi}}_i. \tag{2.58}$$

This mapping preserves the tangential component of basis functions on edges and facets. And the curl of $\boldsymbol{\phi}_i^K$ can be related to the curl of $\hat{\boldsymbol{\phi}}_i$ by

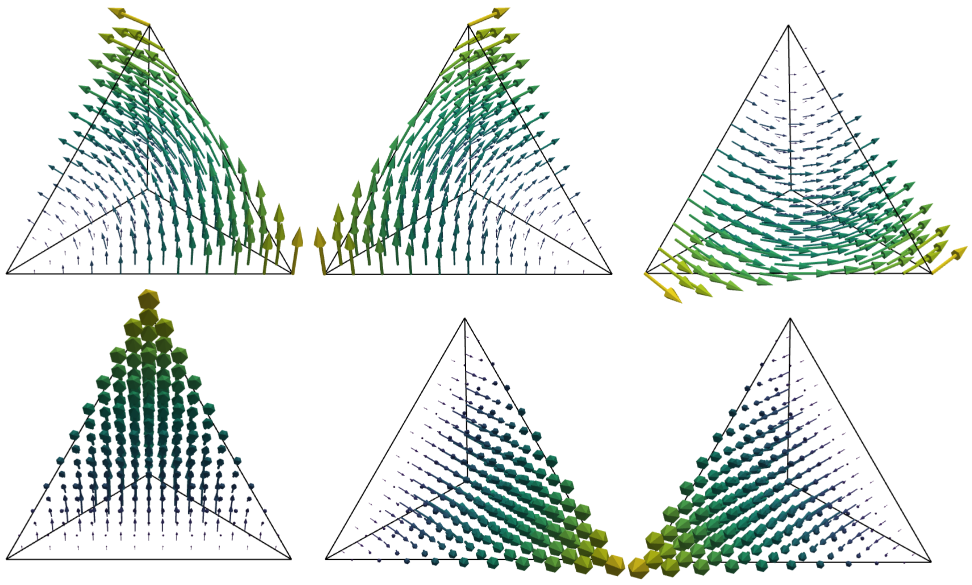


Figure 7 – Basis functions $\{\phi_i\}_{i=0}^{5}$, of the 1st order Nédélec element of first kind, on the reference tetrahedron T.

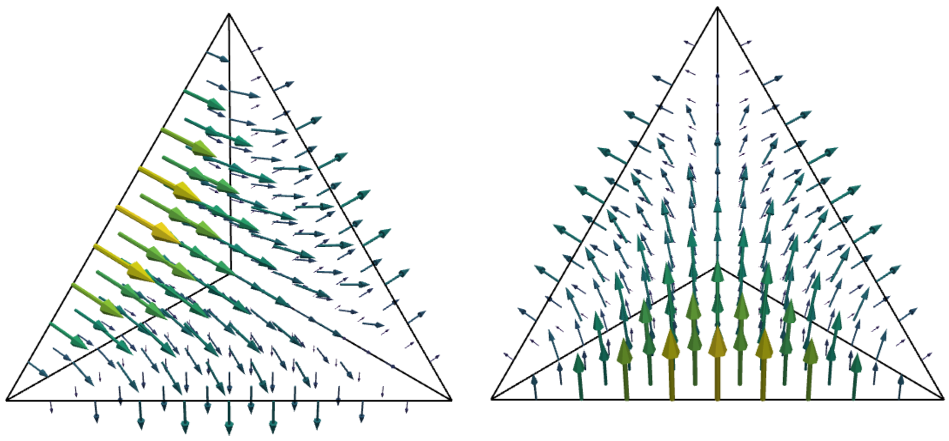


Figure 8 – Basis functions ϕ_{12} and ϕ_{13} of the 2nd order Nédélec element of first kind. Both basis functions are associated with facet 0.

$$\nabla \times \boldsymbol{\phi}_i^K = \frac{1}{|\det(J_K)|} J_K \hat{\nabla} \times \hat{\boldsymbol{\phi}}_i. \tag{2.59}$$

For discussing the local assembly, we consider homogeneous Dirichlet boundary conditions again on $\partial\Omega$, then for each $K \in \mathcal{T}_h$ we can rewrite the left-hand side of (2.49) as

$$a_K(\mathbf{E}, \mathbf{v}) = \int_K (\nabla \times \mathbf{E}) \cdot (\nabla \times \bar{\mathbf{v}})\ dx - k^2 \int_K \mathbf{E} \cdot \bar{\mathbf{v}}\, dx. \tag{2.60}$$

so then the local tensor becomes

$$\mathrm{A}_{ij}^K = \frac{1}{|\det(J_K)|} \int_{\hat{K}} J_K \hat{\nabla} \times \hat{\boldsymbol{\phi}}_i \cdot J_K \hat{\nabla} \times \hat{\boldsymbol{\phi}}_j\, \mathrm{d}x\ - \int_{\hat{K}} J_K^{-T} \hat{\boldsymbol{\phi}}_i \cdot J_K^{-T} \hat{\boldsymbol{\phi}}_j\ |\det(J_K)|\ \mathrm{d}x. \tag{2.61}$$

This process can produce somewhat complicated formulas with Piola mappings and exterior facet integrals (such as impedance boundary conditions). Fortunately, UFL is capable of

Listing 5 Example of UFL representation of form (2.60) with first-order Nédélec elements of the first-kind on a tetrahedron

```
element = FiniteElement("N1curl", tetrahedron, 1)
u = TrialFunction(element)
v = TestFunction(element)
k = Coefficient(FiniteElement("DG", tetrahedron, 0))
a = inner(curl(u), curl(v))*dx - k**2*inner(u, v)*dx
```

symbolic differentiation to derive the tensor expressions automatically. For instance, (2.60) can be easily expressed using UFL as shown in Listing 5.

The construction of global conforming $\mathcal{H}(curl)$ conforming spaces requires neighboring cells to agree on the layout and orientation of shared *dofs* on edges and faces. In general, finite element libraries require that the cells in a mesh are locally numbered to guarantee that entities are consistently ordered, and the method presented above would work. In FEniCSx, however, no assumption of mesh orientation is made, and instead, the approach of basis transformation developed in (SCROGGS et al., 2021) is used.

The transformation matrix $M \in \mathbb{R}^{n\times n}$ for a given cell can be constructed based on the orientation of each sub-entity to the reference orientation. This matrix can then be applied at cell level during the finite element assembly process, or interpolation (SCROGGS et al., 2021). Let $\{\hat{\phi}_i\}_{i=0}^{n-1}$ be a set of basis functions on the reference tetrahedron $\hat{K}$, that linear transformation of the basis can be computed as

$$\begin{pmatrix} \hat{\phi}_0^* \\ \vdots \\ \hat{\phi}_{n-1}^* \end{pmatrix} = M \begin{pmatrix} \hat{\phi}_0 \\ \vdots \\ \hat{\phi}_{n-1} \end{pmatrix}. \tag{2.62}$$

and the basis $\{\hat{\phi}_i^*\}_{i=0}^{n-1}$ have a consistent orientation, meaning that if this transformation is applied to each cell $K \in \mathcal{T}_h$, the required continuity is attained globally. For edge elements (first-order Nédélec elements of the first kind), this procedure is similar to multiplying the basis functions by minus one if the local orientation does not match the global orientation.

2.4 Solution of the linear system

As we saw in the previous section, the discretization of equations (2.12) or (2.9) using the finite element method with suitable boundary conditions, eventually lead to the problem of solving a linear system of equations

$$\mathrm{A}\mathbf{u} = \mathbf{b}, \tag{2.63}$$

where $\mathrm{A} \in \mathbb{C}^{N\times N}$, and $\mathbf{u}$, $\mathbf{b} \in \mathbb{C}^N$, and N is the global number of degrees of freedom.

Contrary to the case of elliptic problems where effective multigrid (TROTTENBERG; OOSTERLEE; SCHULLER, 2000) and domain decomposition methods (SMITH; BJORSTAD;

GROPP, 2004) are readily available (see for example (BALAY et al., 2019b)), the solution of (2.63) is less understood. The matrix inherits many characteristics from the original equations; it is symmetric unless non-reciprocal materials are used, and generally, it is not positive definite nor hermitian. For Maxwell's equation, (2.63) is indefinite for any $k > 0$, while for Helmholtz equations, it is indefinite when $k > \lambda_d$, where λ_d is the first eigenvalue of the Dirichlet problem on the computational domain. For moderate to high frequencies, this inequality holds, and the Helmholtz equation also becomes indefinite.

As the frequency grows, the wavelength becomes small compared to the computational domain, and the mesh density must increase to represent the oscillations of the solution and control the pollution effect (ERNST; GANDER, 2012; DIWAN; MOIOLA; SPENCE, 2019; BABUSKA; SAUTER, 1997). For example, to control phase errors with linear elements, the number of degrees of freedom in each direction increases $O(\sqrt{k})$ as $k \to \infty$ (MONK et al., 2003). It can be shown (IHLENBURG; BABUSKA, 1997) that the relative error e between the FE method and the exact solution is bounded by

$$e \leq C_1(p) \left(\frac{kh}{2p} \right)^p + C_2(p) k \left(\frac{kh}{2p} \right)^{2p} \tag{2.64}$$

where $C_1(p)$ and $C_2(p)$ are constants independent of k or h, but dependent on p. The second term dominates the error for high-frequency problems, so keeping $kh = const$ may lead to corrupted solutions. Figure 9 shows the finite element error computed using Listing 8, for a plane wave propagating in free space. We can note that h-refinement alone has trouble capturing features of the exact solution as we increase the frequency, especially for the lowest order approximation. Moreover, the lowest order approximation requires a highly refined mesh to reach the asymptotic region as predicted by theory.

In summary, the linear system inherits indefiniteness from the BVP, and it is potentially large. Direct solvers may require extensive memory and time resources. Moreover, iterative solvers alone may present slow convergence or diverge (ERNST; GANDER, 2012).

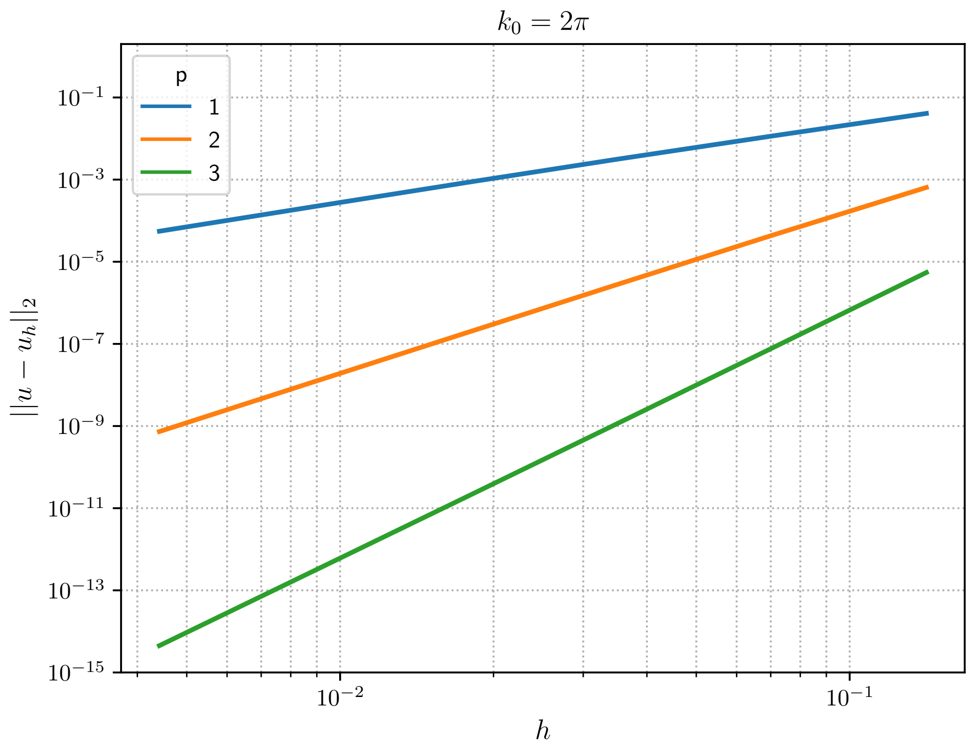
2.4.1 Sparse Direct Solvers

State-of-the-art sparse direct solvers comprise some variant of the classical LU factorization; they reduce A to U by row operations using multipliers in L ($A = LU$) (STRANG, 2007). The complexity of algorithms designed for sparse matrices is significantly higher than that of dense matrices. They need to reduce fill-in in the factors L and U to minimize any side effect on the sparsity of the matrix A. A typical sparse direct solver consists of four distinct steps

1. Reorder the rows and columns to reduce fill-in..

2. Create data structures for factors using the graph's structure.

3. Compute the L and U factors through a numerical factorization.



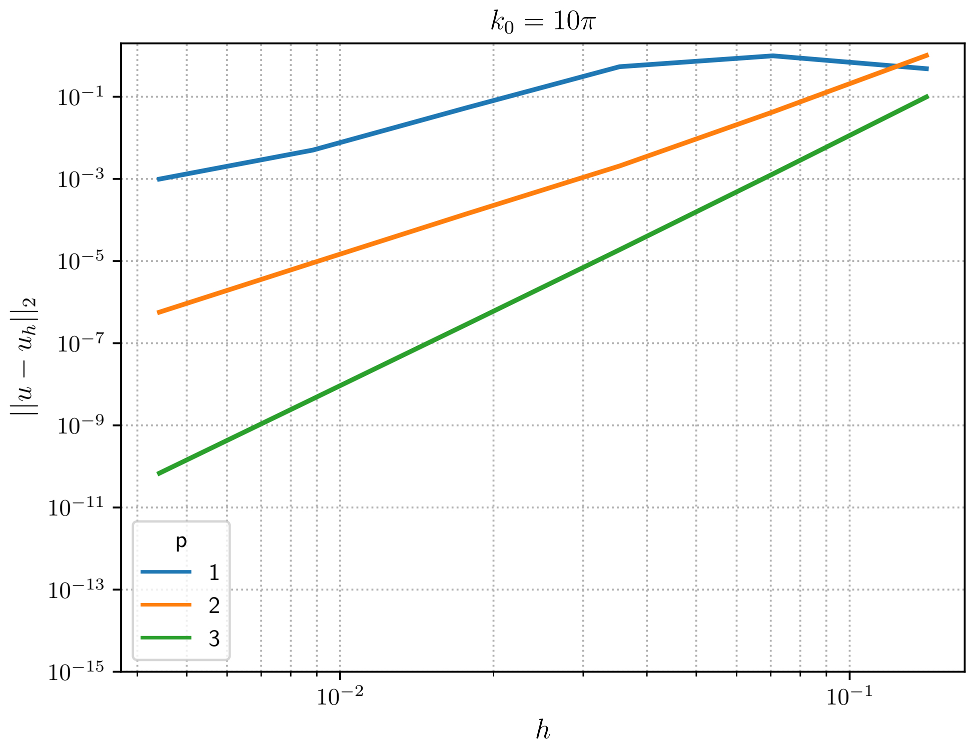


Figure 9 – Plane-wave propagation in free space: Finite element error for different polynomial approximations and uniform refinement.

4. Perform forward and back substitution using the L and U factors.

Steps 1 and 2 are exclusive to sparse matrices and involve only the graphs of the matrices (integer operations). Two different techniques stand out in open-source solvers for general sparse matrices: left-looking and multifrontal. While MUMPS(AMESTOY et al., 2001) and UMFPACK (DAVIS, 2004) use multifrontal algorithms, SuperLU (DEMMEL et al., 1999) uses left-looking. For an overview and distinctions of the techniques, we refer the reader to (ROTHBERG; GUPTA, 1993).

However, they all suffer from the same drawbacks. They are intrinsically sequential, and the speedup saturates quickly with the number of computing units. On structured meshes using optimal ordering, sparse direct solvers require $O(N^{3/2})$ operations and $O(N \log N)$ storage in two dimensions, and $O(N^2)$ operations and $O(N^{4/3})$ storage in three dimensions (RICHARDSON; SIME; WELLS, 2019). These numbers are even worse for unstructured meshes and high-order finite element methods. Recalling that the oscillatory nature of waves and the pollution effect entails very fine meshes and consequently very large systems of equations, the amount of work and memory the factorization demands makes this approach infeasible for practical three-dimensional problems.

2.4.2 Iterative Solvers

Since the matrix-vector product Av is a relatively cheap operation, iterative methods come into play. If A has at most m non-zeros in every row, then the *matvec* operation requires at most mN multiplications. Since $m \ll N$ for any practical problem using finite element discretizations, we can assume that the amount of work for a *matvec* is $O(N)$.

Iterative methods can be broadly classified into pure stationary iterations and Krylov subspace methods (STRANG, 2007; SAAD, 2003). Although Krylov methods are far superior in practice, studying stationary iterative methods provides many insights for creating preconditioners that can also be used with Krylov methods.

2.4.2.1 Stationary Iterations

We can write a simple procedure that generates a series of $\mathbf{u}^{k+1}$ that, under some conditions, converges to the solution of (2.63):

$$\mathbf{u}^{k+1} = \mathbf{u}^k + M^{-1}\mathbf{r}^k \tag{2.65}$$

where $\mathbf{r}^k = \mathbf{b} - A\mathbf{u}^k$, and M^{-1} is the preconditioner. The algorithm (2.65) is called a fixed-point (or pure iteration). When convergent, the iteration (2.65) will converge to the solution of the preconditioned system

$$M^{-1}A\mathbf{u} = M^{-1}\mathbf{b} \tag{2.66}$$

which has the same solution as the original system. The error vector at iteration k, $\mathbf{e}^k = \mathbf{u} - \mathbf{u}^k$, satisfies the relation:

$$\mathbf{e}^k = [\Psi]^k \mathbf{e}^0 = (I - M^{-1}A)^k \mathbf{e}^0. \tag{2.67}$$

where Ψ is the iteration matrix, and its expression does not change during the iteration, hence stationary method. A basic convergence result form linear algebra textbooks states that the algorithm (2.65) converges ($\mathbf{e}^k \to \mathbf{0}$ as $k \to \infty$) for an arbitrary initial error $\mathbf{e}_0$ if and only if $\rho(\Psi) < 1$. Where $\rho(\Psi)$ is the spectral radius of the iteration matrix and it is defined by

$$\rho(\Psi) = \max\{|\lambda_1|, |\lambda_2|, \cdots, |\lambda_N|\}. \tag{2.68}$$

If we use no preconditioner, $M = I$, all eigenvalues of A must lie inside the unit circle. The Galerkin matrices arising from (2.52) and (2.28) fail this test. In general, it is not possible to ensure that the spectral radius of the iteration matrix $\rho(\Psi)$ is smaller than 1 for general indefinite systems.

The solution of a fixed-point iteration is generated in a space spanned by powers of the iteration matrix Ψ applied to a given vector. The main computational cost is thus given by the *matvec* operation and application of M^{-1} to a vector. At nearly the same cost, Krylov methods provide faster convergences by sensibly searching solutions in this same space (DOLEAN; JOLIVET; NATAF, 2015).

2.4.2.2 Krylov Subspace Methods

For a given matrix A and a vector $\mathbf{v}$ we define the Krylov subspace of dimension n by

$$\mathcal{K}^n(A, \mathbf{b}) := \operatorname{Span}\left\{\mathbf{b}, A\mathbf{b}, \ldots, A^{n-1}\mathbf{b}\right\} = \operatorname{Span}\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n\}. \tag{2.69}$$

A sensible approach to compute a solution iteratively is to seek an optimal Krylov Space component. Krylov's methods differ by the way the "optimality" condition is imposed. There is no definitive choice for methods applied to generic matrices. The most appropriate choices for some types of complex matrices, following the guideline given in (BARRETT et al., 1994) are listed below.

- Hermitian Positive Definite (HPD): conjugate gradient (CG) method (SHEWCHUK et al., 1994).
- Hermitian Indefinite (HI) A: the minimal residual (MinRes) method
- Non-Hermitian: generalized minimal residual (GMRes) method (SAAD, 2003; SAAD; SCHULTZ, 1986) and stabilized bi-conjugate gradient squared (BiCGStab) method.

The generalized minimal residual (GMRes) method combined with appropriate preconditioners is the standard choice for solving time-harmonic propagation problems. The GMRES algorithm relies on the following steps:

1. Construct an orthonormal basis for the Krylov subspace $\mathcal{K}_n$ with ‘m‘ elements.
2. Minimize the norm of the residual in this basis:

$$\min_{\mathbf{q}_n \in \mathcal{K}_m} \|\mathbf{b} - \mathbf{A}\left(\mathbf{u}_0 + \mathbf{q}_n\right)\| = \min_{\mathbf{q}_n \in \mathcal{K}_n} \|\mathbf{r}_0 - \mathbf{A}\mathbf{q}_n\|$$

3. Evaluate the norm of residual at step n:
   - if it is small enough the algorithm has found the solution;
   - otherwise, increment ‘m‘ and return to step 1.

In practice, the GMRES alone stagnates when solving high-frequency problems, and it becomes intractable for problems that converge slowly due to its long vector recurrences. Unfortunately, no convergence estimates in terms of the condition number are available for GMRES. However, as for stationary iterative methods, insufficient convergence can be improved by preconditioning the linear system.

2.4.3 Preconditioners

An appealing alternative is the use of preconditioned iterative methods. Instead of solving the linear system $A\mathbf{u} = \mathbf{b}$, directly it might be preferable to solve $M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$. Where $A \in \mathbb{C}^{N\times N}$, and $\mathbf{u}$, $\mathbf{b} \in \mathbb{C}^N$, and N is the global number of degrees of freedom. In view of modern parallel computational architectures, a preconditioner is only effective if (1) the action of M^{-1} to a vector is cheap, (2) approximates the action of A^{-1}, and (3) can be efficiently applied in parallel. In other words, it should require far fewer operations than solving the original system and must be suitable to parallel computers.

From equation (2.67), one could think that the best preconditioner is $M = A$, which is equivalent to solving the linear system (2.63) with an exact method, however it violates items (1.) and (3.). Other straightforward choices of M induce well-known methods such as Jacobi, Gauss-Seidel, and SOR iterations (STRANG, 2007); however, such preconditioners seldom work with time-harmonic wave problems. Algebraic factorization preconditioners, such as incomplete LU factorization with reduced fill-in, ILU(`tol`), were successfully tested only for very-low frequencies in sequential machines (ERNST; GANDER, 2012). This approach becomes prohibitively expensive for high-frequency problems and/or parallel computers, and even decreasing the tolerance does not help as k increases.

For standard elliptic problems, like those modeled by Poisson’s equation, effective methods for constructing M include Multigrid methods and two-level Schwarz methods. However, for non-coercive Helmholtz and Maxwell equations, due to the intrinsic indefiniteness, even the coarse level must be fine enough to represent the oscillations of the solution; otherwise, the iterative method diverges. Non-standard methods such as shifted-Laplacian (GANDER; GRAHAM; SPENCE, 2015) and plane-wave (LIVSHITS, 2014) multigrid methods have been suggested and are revisited in 4. Furthermore, a new two-level method based on these principles is proposed later in Chapter 5.

2.5 Software Stack

This section gives an overview of some implementation aspects and describes how the libraries we rely on are interconnected. The schematic showing the implementation abstraction and the software stack is presented in Figure 10.

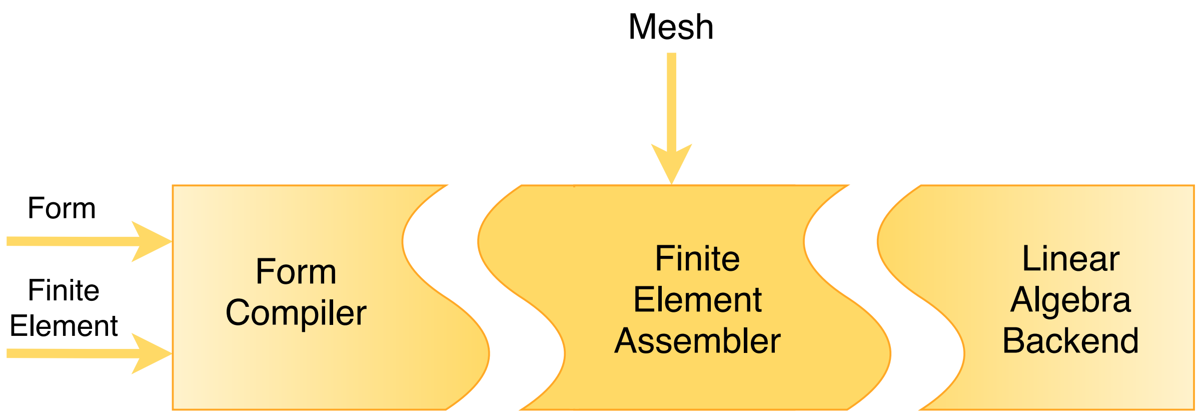


Figure 10 – A schematic overview of the computational tool-chain

Listing 6 Example of *UFL* description of a Helmholtz variational form

```
element = FiniteElement("Lagrange", triangle, degree)

v = TestFunction(element)
u = TrialFunction(element)
g = Coefficient(element)

a = inner(grad(u), grad(v)) * dx - k0**2 * inner(u, v) * dx + 1j * k0 *
↪  inner(u, v) * ds
L = inner(g, v) * ds
```

In Section 2.3, we have defined the concepts of forms and finite elements, we introduce some software components along the way. Now, these mathematical abstractions are directly used in our implementations. We specify the form (2.28) using Unified Form Language (*UFL*) (ALNÆS et al., 2014). To completely define the discrete abstract form, an element has to be specified. We use Basix (PROJECT, 2021), which implements the classical finite element abstraction of Ciarlet. These two components are the input for a form compiler as shown in Figure 10. In Listing 6, we show an example of the definition of variational forms using UFL. A near mathematical notation can be noted.

We use the form definition from UFL as an input to *FFCx*, the new version of the FEniCS Form Compiler (LOGG et al., 2012b), to generate a low-level C code. This low-level code can be used to assemble the corresponding discrete operator. During this project, we have extended the *FFCx* library to support complex-valued forms, and this feature is available since release 0.0.1.

In particular, one assembles the sesquilinear form into a complex-valued matrix and the anti-linear form into a complex-valued vector. For that purpose, we use DOLFINx, the computational back-end of FEniCS, that implements the finite element assembler in Python and C++. During this project, the DOLFINx library (ALNÆS et al., 2015) has also being extended to support complex numbers. A somewhat detailed report on the implementation aspects can be found at (BARATTA, 2018).

Although the *DOLFINx* library provides several elementary mesh constructors, for more complicated computational domains, we use GMSH (GEUZAINE; REMACLE, 2009) to generate the mesh and define the material distribution and boundaries (using mesh markers).

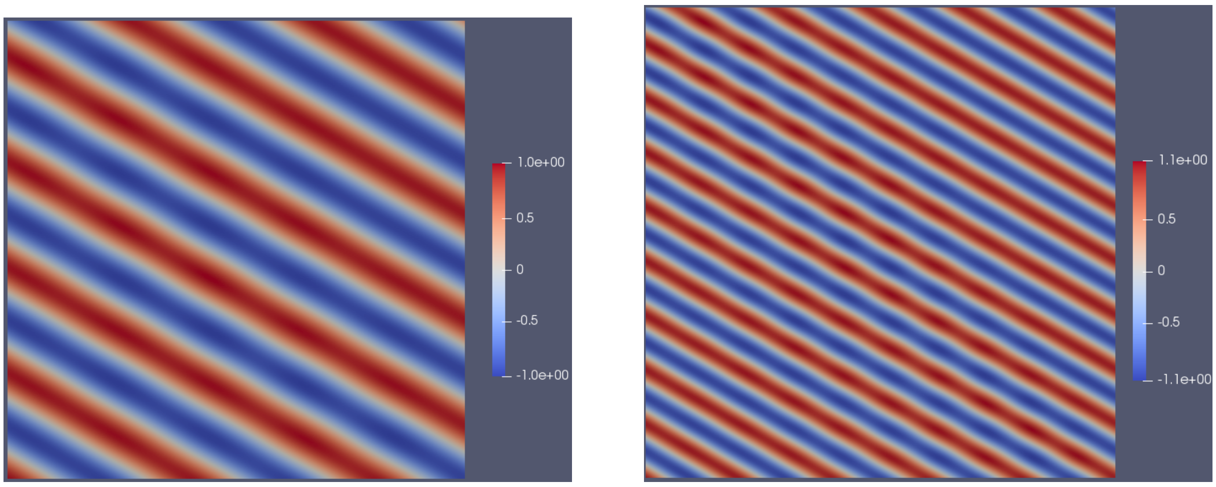
We use the PETSc library (BALAY et al., 2019b) as the linear algebra back-end. The vector and matrices are assembled into PETSc Matrix and Vector objects, respectively. It also provides several non-stationary iterative methods based on the Krylov spaces (KSP objects) and some standard preconditioners (PC objects).

In Listing 7 we present a complete program for solving a scalar plane wave propagation on a square domain to demonstrate the tool-chain presented in Figure 10. We use a first-order absorbing boundary condition to truncated the domain. We set $kh \approx 0.25$ and solve the problem for $k = 10\pi,\ 20\pi$. The solution can be seen on Figure 11. To execute the script using 12 MPI processes, simply run the following command:

```
mpiexec -n 12 python3 helmholtz.py
```

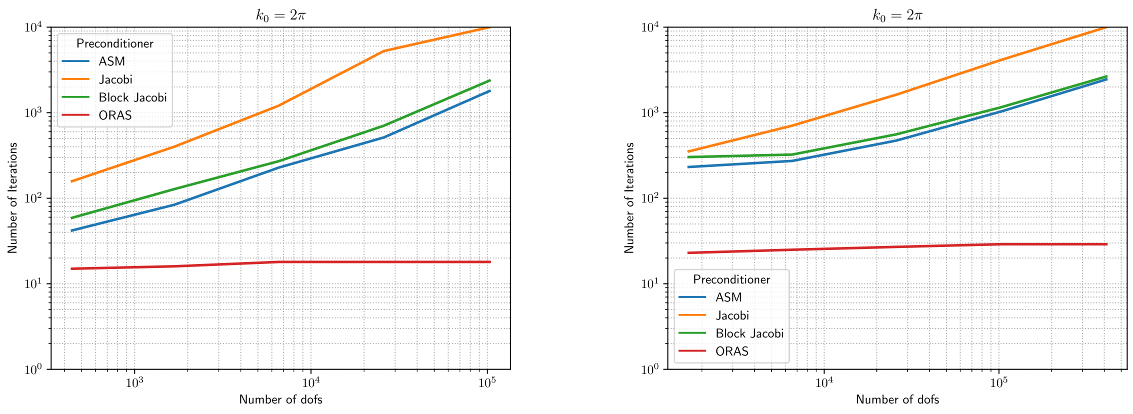
As a motivation for the following chapters, we show the iteration counts for the preconditioned GMRES with three algebraic preconditioners (readily available in PETSc) and the Optimized Restricted Additive Schwarz (ORAS) we develop in Chapter 3. A complete list of preconditioners available in PETSc can be found in (BALAY et al., 2019b). However, we apply only the preconditioners that work in parallel and need little parameter tweaking: Block Jacobi, Jacobi, and Additive Schwarz. We can see from Figures 12 (a) and (b), that even with a simple two-dimensional problem and moderate size, purely algebraic solvers struggle to converge. It can be seen that the number of iterations increases rapidly with the increase in the number of degrees of freedom. On the other hand, the number of iterations of ORAS remains steady with an increasing number of degrees of freedom.

For the problem of an electromagnetic wave propagating in free space in three dimensions, described in Listing 8, none of the algebraic preconditioned methods were able to converge to a solution with less than 1000 iterations. Example of solution for generated by Listing 8 is presented in Figure 13 using a direct solver.



(a) $k = 10\pi$ (b) $k = 20\pi$

Figure 11 – Solution of a scalar plane wave propagation problem using 1st order Lagrange elements



(a) $k = 2\pi$ (b) $k = 5\pi$

Figure 12 – Iteration count for the preconditioned GMRES with three different preconditioners: scalar plane wave propagating in free space.

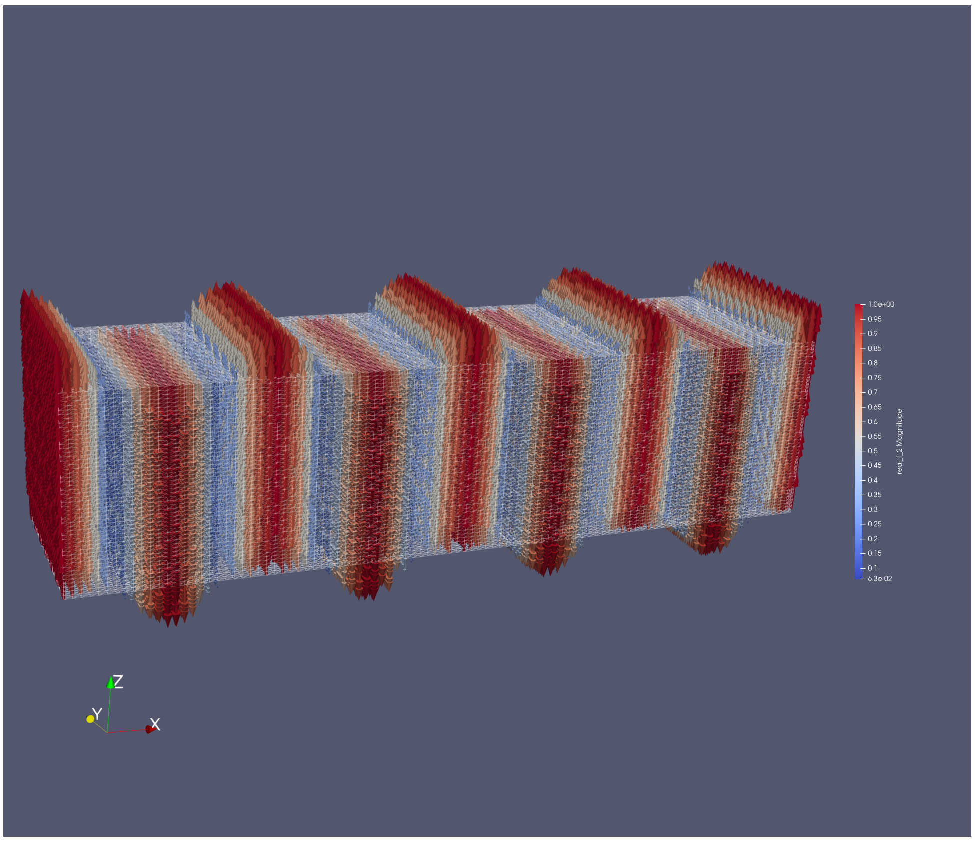


Figure 13 – Example of solution from Listing 8: electromagnetic plane wave propagating in free-space.

Listing 7 A complete program for solving the Helmholtz equation: plane wave propagating in free space.

```
import numpy as np
import dolfinx
import ufl
from mpi4py import MPI
from dolfinx.io import XDMFFile

p = 2
k0 = 10 * np.pi

mesh = dolfinx.UnitSquareMesh(MPI.COMM_WORLD, 100, 100)
element = ufl.FiniteElement("Lagrange", ufl.triangle, p)
n = ufl.FacetNormal(mesh)

# Definition of function space
V = dolfinx.FunctionSpace(mesh, element)


def incoming_wave(x):
    d = np.cos(theta) * x[0] + np.sin(theta) * x[1]
    return np.exp(1.0j * k0 * d)


# Incoming wave
theta = np.pi/4
ui = dolfinx.Function(V)
ui.interpolate(incoming_wave)
g = ufl.dot(ufl.grad(ui), n) + 1j * k0 * ui

# Define variational problem
u = ufl.TrialFunction(V)
v = ufl.TestFunction(V)

# Weak Form
a = ufl.inner(ufl.grad(u), ufl.grad(v)) * ufl.dx \
    - k0**2 * ufl.inner(u, v) * ufl.dx \
    + 1j * k0 * ufl.inner(u, v) * ufl.ds
L = ufl.inner(g, v) * ufl.ds

petsc_options = {"ksp_type": "gmres", "pc_type": "asm"}

u = dolfinx.Function(V)
solver = dolfinx.fem.LinearProblem(a, L, [], u, petsc_options)
solver.solve()

with XDMFFile(MPI.COMM_WORLD, "out.xdmf", "w") as file:
    file.write_mesh(mesh)
    file.write_function(u)
```

Listing 8 A complete program for solving the Maxwell's equation: plane wave propagating in free space.

```
import numpy as np
import dolfinx
import ufl
from mpi4py import MPI
from dolfinx.io import XDMFFile

p = 2
k0 = 2 * np.pi
theta = 0


mesh = dolfinx.BoxMesh(MPI.COMM_WORLD, [[0, 0, 0], [1, 1, 1]], [20, 20,
↪  20])
n = ufl.FacetNormal(mesh)

# Definition of function space
element = ufl.FiniteElement("N1curl", ufl.tetrahedron, p)
V = dolfinx.FunctionSpace(mesh, element)


def incoming_wave(x):
    d = np.cos(theta) * x[0] + np.sin(theta) * x[1]
    out = np.zeros(x.shape, dtype=np.complex128)
    out[2] = np.exp(1.0j * k0 * d)
    return out


# Incoming wave
Ei = dolfinx.Function(V)
Ei.interpolate(incoming_wave)
g = ufl.cross(ufl.curl(Ei), n) + 1j * k0 * \
    ufl.cross(n, ufl.cross(ufl.curl(Ei), n))


# Define variational problem
E = ufl.TrialFunction(V)
v = ufl.TestFunction(V)

# # Weak Form
a = ufl.inner(ufl.curl(E), ufl.curl(v)) * ufl.dx \
    - k0**2 * ufl.inner(E, v) * ufl.dx \
    + 1j * k0 * ufl.inner(ufl.cross(n, E), ufl.cross(n, v)) * ufl.ds
L = ufl.inner(g, v) * ufl.ds

petsc_options = {"ksp_type": "preonly", "pc_type": "lu"}

u = dolfinx.Function(V)
solver = dolfinx.fem.LinearProblem(a, L, [], u, petsc_options)
solver.solve()

with XDMFFile(MPI.COMM_WORLD, "out.xdmf", "w") as file:
    file.write_mesh(mesh)
    file.write_function(u)
```

3 Domain Decomposition Methods

Contents

Domain decomposition methods (DDM) frequently refer to the splitting of a partial differential equation, or its numerical approximation, into coupled sub-problems on smaller domains that are more amenable for computing (TOSELLI; WIDLUND, 2006). Thus, it consists of a divide-and-conquer strategy for the numerical solution of partial differential equations. The fundamental idea is that instead of solving one large problem on a single domain on a single process, it may be beneficial to solve many smaller problems on different processes a finite number of times.

We can loosely classify domain decomposition methods into overlapping and non-overlapping (or sub-structuring) methods. Sub-structuring methods are much less robust than overlapping methods when using automatic mesh partitioners, such as ParMETIS (KARYPIS, 2011), KaHIP (MEYERHENKE; SANDERS; SCHULZ, 2017), PT-SCOTCH (PELLEGRINI, 2012). These graph partitioners can introduce rough interfaces between subdomains that may significantly impair the convergence of non-overlapping methods (KLAWONN; RHEINBACH; WIDLUND, 2008). Moreover, the presence of cross-points and cross-edges (only in 3d), an inherent property of non-overlapping methods, may cause the iterative procedure to stagnate (GANDER; SANTUGINI, 2016). A special treatment for cross points in the context of time-harmonic wave problems has been proposed in (MODAVE et al., 2020); however, the technology is not mature and has only been tested for two-dimensional problems.

To avoid the issues mentioned above, we concentrate on overlapping domain decomposition methods. Overlapping methods trades off communication overhead and robustness

(GANDER; ZHANG, 2016). However, modern overlapping domain decomposition preconditioners use minimal overlap to decrease communication costs. Moreover, these methods possess a simple algorithmic structure because there is no need to solve special interface problems between neighboring subdomains (DOLEAN; JOLIVET; NATAF, 2015).

In this chapter, we give an overview of one-level domain decomposition methods. A fundamental feature of these methods is the transmission conditions that directly affect the convergence rates. We start with a simple mathematical formulation for the Helmholtz equation and give some convergence estimates. We then propose a new procedure to optimize transmission conditions tailored for the many subdomains case (BARATTA; SILVA, 2018). Finally, we extend these concepts to vectorial time-harmonic Maxwell's equation, which leads to an improved transmission condition optimization process.

3.1 Mathematical Formulation

We now introduce the mathematical formulation of overlapping domain decomposition methods. The mathematics is presented without great rigor, so we refer to the existing literature, e.g.: (TOSELLI; WIDLUND, 2006; SMITH; BJORSTAD; GROPP, 2004; DOLEAN; JOLIVET; NATAF, 2015; QUARTERONI; VALLI, 1999), for a more thorough treatment.

3.1.1 Additive Schwarz Method at Continuous Level

We start our overview of overlapping DDM with a description of a Helmholtz boundary value problem posed on an arbitrary domain. We consider, for instance, the classical domain Ω on the left of Figure 14.

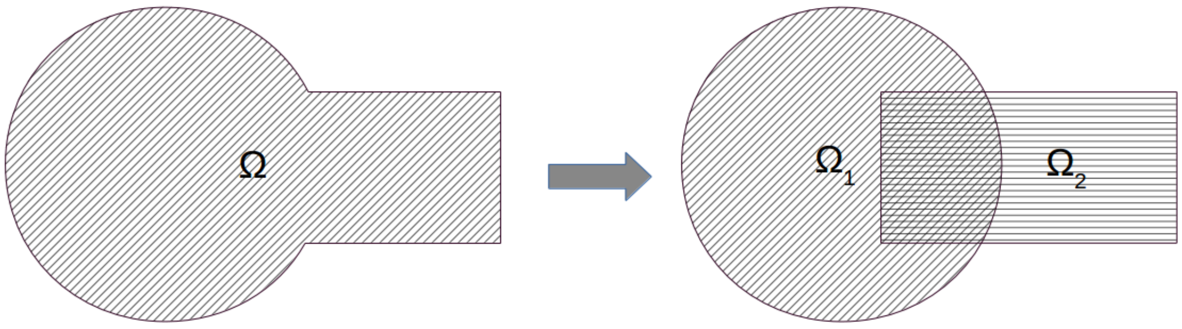


Figure 14 – Decomposition of the computational domain Ω into two overlapping subdomains, Ω_1 and Ω_2. This classical domain corresponds to the logo of the DDM community, see http://www.ddm.org/.

We seek the solution u, such that:

$$\left.\begin{aligned} \Delta u + k^2 u = f \quad &\text{in} \quad \Omega \\ \mathcal{C}(u) = 0 \quad &\text{on} \quad \partial\Omega \end{aligned}\right\} \tag{3.1}$$

where the boundary conditions, encapsulated by the operator $\mathcal{C}$ in (3.1), might be any combination of Dirichlet and absorbing boundary conditions.

The earliest known DDM was due to Schwarz in 1870. He formulated an iterative method for solving the Poisson problem placed on a union of simple geometries in order to prove the existence of the solution, see Figure 14. The original Schwarz algorithm is sequential by nature; each sub-domain is solved in a predetermined sequence, first Ω_1 then Ω_2. The extension to more subdomains is called the Multiplicative Schwarz Method. We are, however, interested in the parallel version, proposed by P. L. Lions in 1987 (GANDER et al., 2008). Instead of solving (3.1) directly, he proposed an iterative algorithm which solves all subdomains Ω_i for $i = 1, 2$ concurrently:

$$\left.\begin{aligned} \Delta u_i^{n+1} + k^2 u_i^{n+1} = f_i \quad &\text{in} \quad \Omega_i \\ \mathcal{C}(u_i^{n+1}) = 0 \quad &\text{on} \quad \partial\Omega_i \cap \partial\Omega \\ u_i^{n+1} = u_{3-i}^n \quad &\text{on} \quad \partial\Omega_i \cap \overline{\Omega}_{3-i} \end{aligned}\right\}. \tag{3.2}$$

Although the classical Schwarz algorithm converges for elliptic equations (but very slowly), it fails for non-coercive problems. Moreover, its convergence rates are very much dependent on the overlap size. These issues motivated the development of a new class of methods, the so-called Optimized Schwarz Methods (OSM) (GANDER, 2006). Such methods were introduced by Lions in 1990 (LIONS, 1990) for the Laplace equation and extended to the Helmholtz equation by Després in 1991 (DESPRÉS, 1991). The modification proposed by Després led to the first iterative method with proven convergence for indefinite operators.

OSM is based on classical domain decomposition methods, but they use more effective transmission conditions at the interfaces between subdomains. When compared to Classical DD algorithms, it has some distinct features:

- It converges faster, at (almost) the same cost per iteration,
- There are simple optimization procedures to determine the best transmission conditions,
- Only small changes in the implementation are required.

Let $\{\Omega_i\}$ be the set of P subdomains that completely covers the domain Ω. The OSM iteration is advanced by simultaneously solving:

$$\left.\begin{aligned} \Delta u_i^{n+1} + k^2 u_i^{n+1} = f_i \quad &\text{in} \quad \Omega_i \\ \mathcal{C}(u_i^{n+1}) = 0 \quad &\text{on} \quad \partial\Omega_i \cap \partial\Omega \\ \mathcal{B}_i(u_i^{n+1}) = \mathcal{B}_j(u_j^n) \quad &\text{on} \quad \Gamma_{ij}, j \in \mathcal{O}(i) \end{aligned}\right\}. \tag{3.3}$$

for $i = 0,\ 1, \cdots, P-1$. Where $\Gamma_{ij} = \partial\Omega_i \cap \Omega_j$ is the interface between subdomains Ω_i and Ω_j. For overlapping decompositions $\Gamma_{ij} \neq \Gamma_{ji}$. $\mathcal{O}(i)$ is set of neighbours of sub-domain Ω_i. In what follows, we assume that the linear operator $\mathcal{B}_i$ takes the form:

$$\mathcal{B}_i = \partial_{\hat{n}_i} + \mathcal{S}_i \tag{3.4}$$

where $\hat{n}_i$ is the unit outward normal to the boundary of the sub-domain Ω_i. Using the finite element abstraction presented in the previous chapter, we can write the algorithm (3.3) in the finite element framework:

$$a_i(u_i^{n+1}, v_i) = L(v_i), \quad \forall v_i \in \widehat{V}_i \tag{3.5}$$

where

$$\begin{aligned} a(u_i^{n+1}, v_i) = &- \int_{\Omega_i} \nabla u_i^{n+1} \cdot \nabla \bar{v}_i + \int_{\Omega_i} k^2 u_i^{n+1}\, \bar{v}_i \\ &+ \int_{\partial\Omega\cap\partial\Omega_i} \left(\nabla u_i^{n+1} \cdot \mathbf{n}\right) \bar{v} - \sum_{j\in\mathcal{O}(i)} \int_{\Gamma_{ij}} \mathcal{S}_i(u_i^{n+1})\bar{v} \end{aligned} \tag{3.6}$$

$$L(v_i) = \int_{\Omega_i} f_i \bar{v}_i + \sum_{j\in\mathcal{O}(i)} \int_{\Gamma_{ij}} g_j^n \bar{v}_i. \tag{3.7}$$

The subspace V_i is the space of restriction of functions in V to Ω_i. g_j accounts for the information from subdomain $j \in \mathcal{O}(i)$. An UFL representation of the sesquilinear form can be found in Listing 9. If a transmission operator $\mathcal{S}_i$ can be translated into a UFL form, we can use automatic code generation to test different operators.

Listing 9 UFL representation of 3.6 with 1st order absorbing boundary conditions and transmission operator $\mathcal{S}_i$ on the interface $dS(i) = \Gamma_i$

```
a = ufl.inner(ufl.grad(u), ufl.grad(v)) * ufl.dx \
    - k0**2 * ufl.inner(u, v) * ufl.dx \
    + 1j * k0 * ufl.inner(u, v) * ufl.ds
a+= ufl.inner(S(u), v)* ufl.dS(i)
```

3.1.2 Optimized Schwarz as a Preconditioner

Domain decomposition methods are seldom used as iterative solvers, and they perform much more effectively when used as preconditioners. Particularly in this project, we work with a variant called Optimized Restricted Additive Schwarz (ORAS) preconditioner (ST-CYR; GANDER; THOMAS, 2007). Before introducing the preconditioner, we review some basic concepts and describe our implementation strategy.

We discretize Ω into a finite set of cells, the finite element mesh $\mathcal{T}_h = \{K\}$. Using a mesh partitioning algorithm, we decompose Ω into P non-overlapping subdomains $(\Omega_i)_{0\leq i\leq P-1}$

so that the union of subdomains implies the union of all cells of the mesh $\mathcal{T}_h$:

$$\Omega := \bigcup_{i=0}^{P-1} \left\{\overline{\Omega}_i\right\} \quad \text{and } \Omega_i \cap \Omega_j = \emptyset, \quad i \neq j. \tag{3.8}$$

An example of a non-overlapping partition is shown in Figure 15a, where we highlight the first sub-domain and its interface. We obtain an overlapping partition by adding one or several mesh layers to each subdomain. In Figure 15b, we add a single mesh layer to each subdomain, producing a minimal overlap.

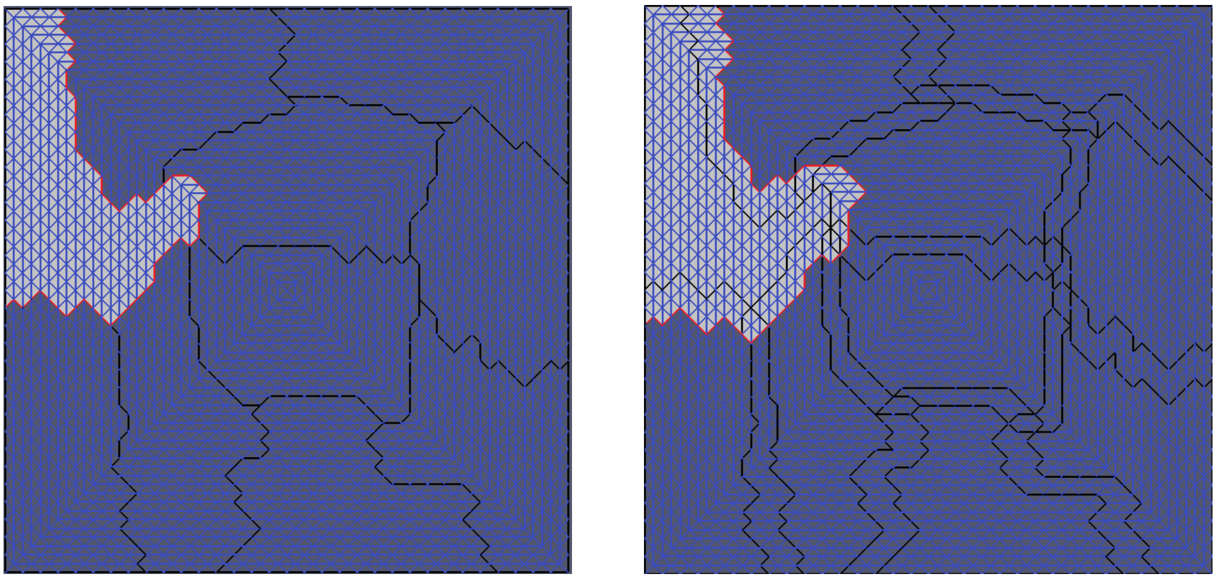

(a) Non-Overlapping partition (b) Overlapping partition

Figure 15 – Partition of the unit square mesh into 10 subdomains using a mesh partitioning algorithm.

The mesh partition induces a natural decomposition of the degrees of freedom (*dofs*) of the original finite element problem. Let $\mathcal{N}$ be the set indices of the N *dofs*, then

$$\mathcal{N} := \bigcup_{i=0}^{P-1} \mathcal{N}_i \tag{3.9}$$

where $\mathcal{N}_i$ is the set constrained to Ω_i. Recalling that $\mathbf{u}$ is the coefficient expansion vector, we can define a vector (function expanded in terms of the degrees of freedom)restricted to sub-domain Ω_i by:

$$\mathbf{u}_i = R_i \mathbf{u} \tag{3.10}$$

where $R_i \in \mathbb{R}^{N_i \times N}$ is the restriction matrix that maps coefficient vectors of functions in V to coefficient vectors of functions in V_i. By consequence, R_i^T is the extension matrix, that extends by zero the local coefficient vector from V_i to V. Finally, let D_i be the discrete partition of unit matrix that weights the solution from different subdomains at the overlap such that

$$\sum_{i=0}^{P-1} R_i^T D_i R_i = I. \tag{3.11}$$

where $I \in \mathbb{R}^{N\times N}$ is the identity matrix. Then the additive Schwarz method (ASM) proposed in (DRYJA; WIDLUND, 1987) can be defined as:

$$M_{ASM}^{-1} := \sum_{i=0}^{P-1} R_i^T A_i^{-1} R_i \quad \text{with } A_i = R_i A R_i^T. \tag{3.12}$$

The ASM does not converge as an iterative method and converges rather slow as a preconditioner in a Krylov iteration. With a small modification in (3.12), a new variant was introduced, the restricted additive Schwarz method (CAI; SARKIS, 1999):

$$M_{RAS}^{-1} := \sum_{i=0}^{P-1} R_i^T D_i A_i^{-1} R_i \quad \text{with } A_i = R_i A R_i^T. \tag{3.13}$$

Numerical experiments show that RAS converges faster and requires less communication than ASM. A thorough discussion about the reasons why this minor modification led to an improved algorithm can be found in (EFSTATHIOU; GANDER, 2003). Both ASM and ASM are the discrete counterparts of the classical parallel Schwarz method (3.2), which is known to fail for non-coercive problems. Also, since they operate only on the algebraic level, one cannot expect them to behave differently to block Jacobi or block Gauss-Seidel methods.

The discrete counterpart of the OSM, described in the last section, is the Optimized Restricted Additive Schwarz (ORAS) (ST-CYR; GANDER; THOMAS, 2007). It takes advantage of the enhancements introduced by the RAS and adds an enhanced transfer of information between adjacent subdomains. The ORAS preconditioner can be defined as

$$M_{ORAS}^{-1} := \sum_{i=0}^{P-1} R_i^T D_i \tilde{A}_i^{-1} R_i \tag{3.14}$$

where $\tilde{A}$ is the local matrix of the sub-problem i equipped with the appropriate transmission conditions. For instance, in our implementation, $\tilde{A}$ comes from the finite element assembly of (3.6). In practice, the global matrix A is not assembled, nor is the inverse of the preconditioner M_{ORAS} computed. Preferably, we provide operators to the linear algebra back-end so it can compute the action of the global finite element matrix applied to a vector $A\mathbf{v}$, and the solution to the auxiliary problem $M\mathbf{v} = \mathbf{c}$.

3.1.3 Operators' Implementation

Another example of a mesh and resulting partition is shown in Figure 16. The original subdomain interfaces are shown in black, and the new interfaces after the extension with a ghost layer are shown with the respective process color. The overlapping layer (ghost cell layer) can be obtained by incorporating into a subdomain all cells that are not owned by the process but have at least one vertex that touches the interface. In DOLFINx, this can be performed by using the functionality `mesh::add_ghosts`. Additional overlapping layers can be attached by executing this method recursively. A simple snippet showing how to read an XDMF mesh and

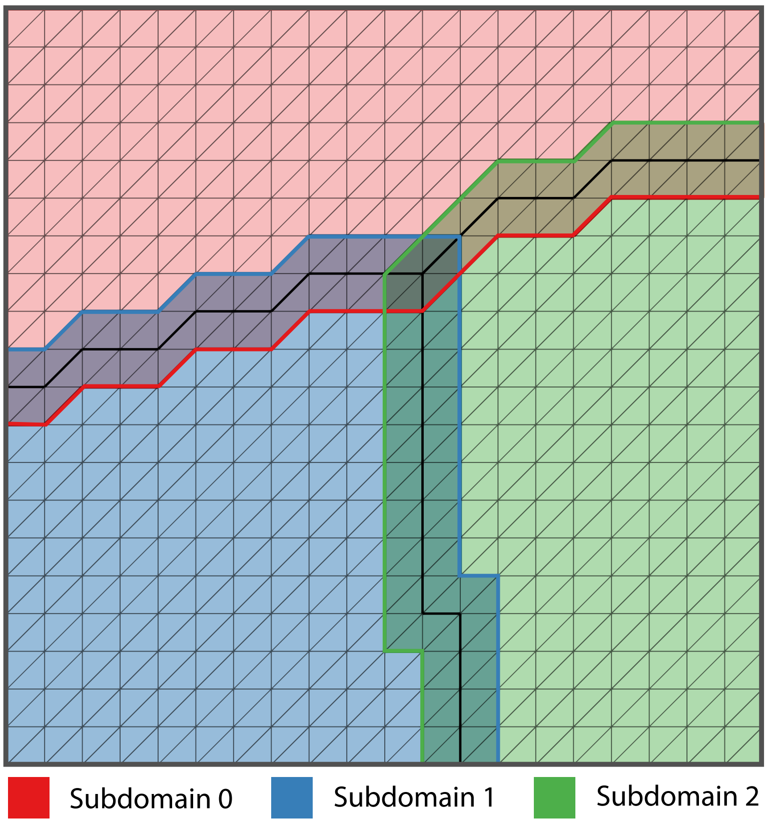


Figure 16 – A global mesh $\mathcal{T}$, associated with the global computational domain Ω, partitioned in 3 sub-meshes $\{\mathcal{T}_i\}_{i=0}^{2}$, now associated with $\{\Omega_i\}_{i=0}^{2}$. The initial interface between subdomains is marked in black.

add L layers of overlap is presented in Listing 10. Some extensions for partitioning meshes in parallel have been introduced to DOLFINx during this thesis and are described in the Appendix B.

Listing 10 Python code for adding L layers of ghosts to a generic mesh.

```python
import dolfinx
from mpi4py import MPI

comm = MPI.COMM_WORLD    # MPI communicator to read mesh
L = 2                    # number of layers

with XDMFFile(comm, filename, "r") as file:
    mesh = file.read_mesh()

for i in range(L):
    mesh = dolfinx.mesh.add_ghost_layer(mesh)
```

The decomposition of Ω into P overlapping subdomains $\{\Omega_i\}_{i=0}^{P-1}$ induces a natural decomposition of the global finite element space V on $\mathcal{T}_h$ into P local finite element spaces $\{V_i\}_{i=0}^{P-1}$ each of them defined on $\{\mathcal{T}_i\}_{i=0}^{P-1}$. Note that due to the overlap $N < \sum_{i=0}^{P-1} N_i$. However,

Listing 11 Restriction operator in DOLFINx

```
u.x.scatter_forward()
```

we can define $N = \sum_{i=0}^{P-1} N_i^o$, where N_i^o is the number of owned *dofs* by process i. In Figure 16, *dofs* of cells in the overlap layer are duplicated.

In DOLFINx, a vector in $\mathbb{C}^N$, for example the vector of expansion coefficients of a function $\mathbf{v} \in V$, is naturally distributed to the P MPI processes, such that $\mathbf{u}_i \in V_i$, and $\mathbf{x}_i \in \mathbb{C}^{N_i}$. The management of the parallel layout of a vector is carried out by an `IndexMap`. An illustration of an `IndexMap`, a parallel vector layout with ghosts, is shown in Figure 17. The color indicates the owning process, and the numbers indicate the local and global indices (global in brackets). For instance, each process p is aware of its forward and reverse neighbors and shared indices (owned and ghosts).

3.1.3.1 Restriction Operator

Since vectors in DOLFINx are naturally distributed, as explained above, the application of the restriction matrix R_i to a vector

$$\mathbf{x}_i = R_i\mathbf{x} \tag{3.15}$$

is obtained simply by updating the ghost region. The matrix R_i is not assembled, but instead, each process that owns a shared *dof* sends the corresponding values to the sharing processes, as shown in Figure 18. In DOLFINx this is implemented using asymmetric MPI neighbor communicators and neighborhood collectives. And we've made it available to the user as a simple function call shown in Listing 11.

The `IndexMap` associated with a vector (or with a `fem.Function`) has all information needed to create an asymmetric MPI neighbor communicator (owner to ghost). Notably, it is possible to attach information about the communication relationships between processes so the MPI implementation can perform optimizations, such as adjusting communication buffers, improving communication setup, or optimizing process layouts (GROPP et al., 2014). An example showing the use of Restriction Operator and access of the underlying data is shown in Listing 12. To get an idea of the weak scaling, we compared the time to update the ghost region (application of the restriction operator) using our methodology and using a PETSc VecScatter object. We fixed the number of owned cells in each process at 500.000 with one ghost layer and then called the code in Listing 12 ten times. From Figure 19 we can see that using asymmetric neighbor MPI communicators, we achieved an excellent weak scaling, while PETSc VecScatter is relatively inconsistent. It is worth noting that PETSc needs to build the communication pattern in its first iteration, but nevertheless, the best measured time for its VecScatter is slower than the average time of our implementation.

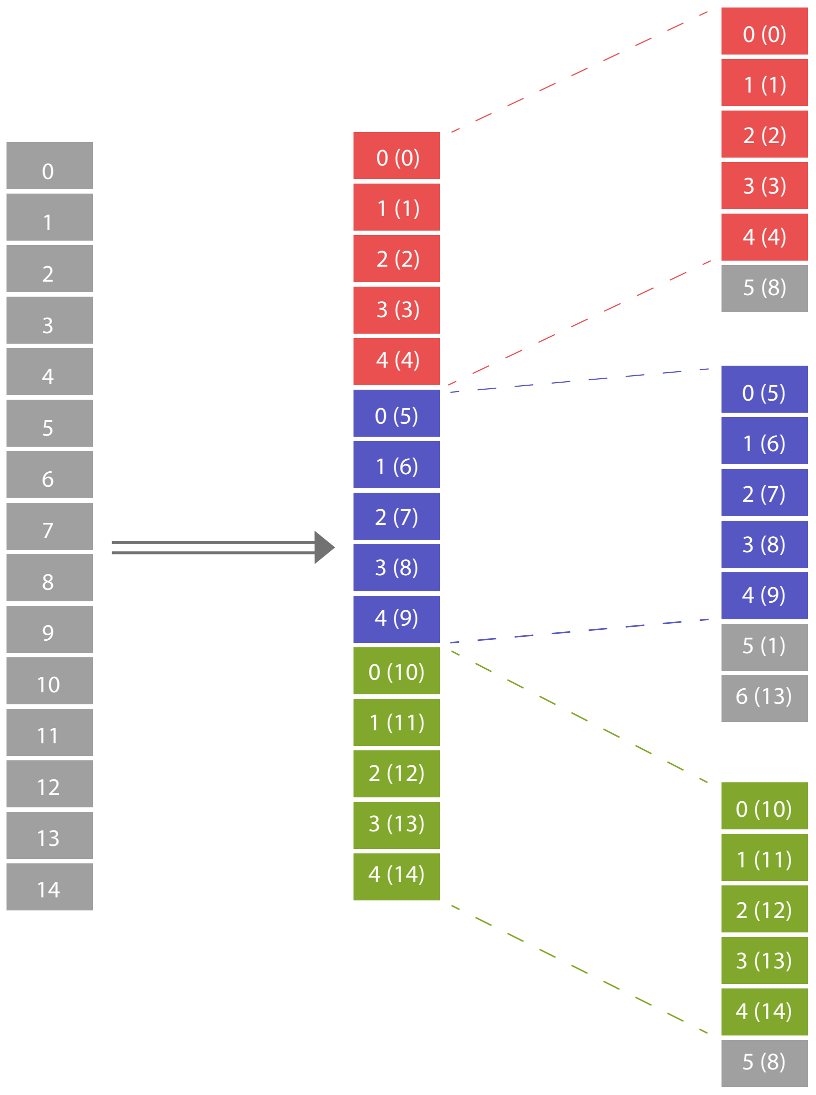


Figure 17 – Layout of a parallel vector of size 15 distributed to 3 processes. From left to right: Sequential vector, distributed vector, distributed vector with ghosting.

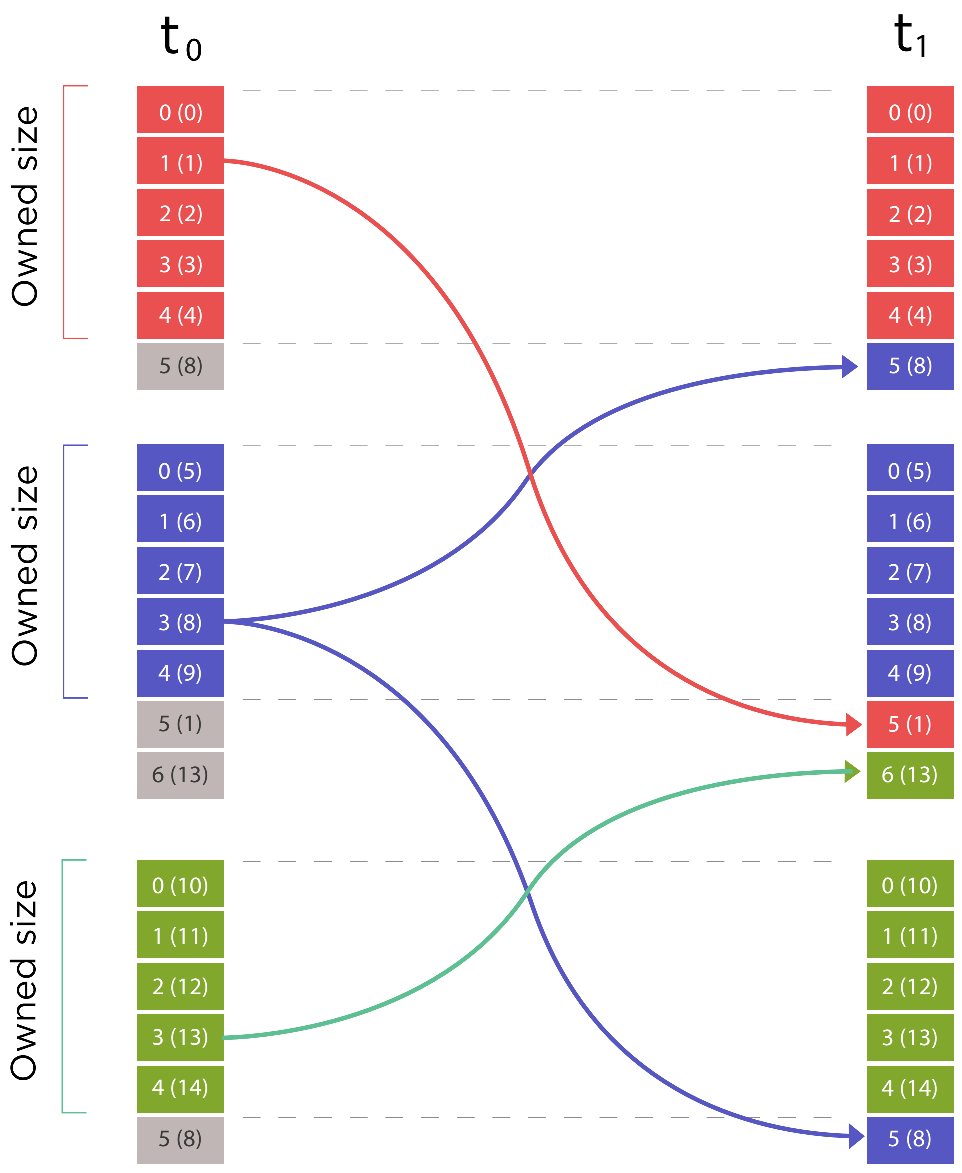


Figure 18 – Forward scattering of a vector in parallel. Communication direction is *dof* owner to ghost.

Listing 12 Accessing the local data of a distributed Finite Element Function.

```
# Given a Finite element "e" and a mesh from Listing 6
e = ufl.FiniteElement("Lagrange", ufl.tetrahedron, degree)

# A distributed finite Element function can be created
V = dolfinx.FunctionSpace(mesh, element)
u = dolfinx.Function(V)
imap = V.dofmap.index_map

assert u.x.array.size == imap.size_local + imap.num_ghosts

# Set local data to 1, and data on the overlap to 0
u.x.array[:imap.size_local] = 1
u.x.array[imap.size_local:] = 0

# Update ghost region (previously 0)
u.x.scatter_forward()

# Check if the ghosts values are 1
assert u.x.array[imap.size_local:] == 1
```

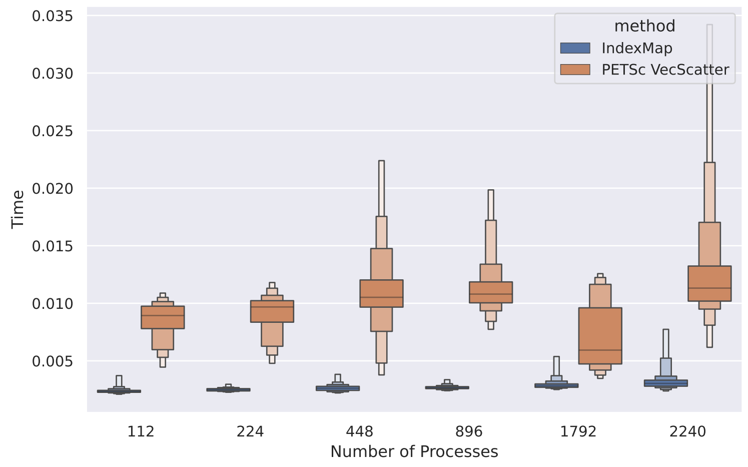


Figure 19 – Weak scaling of the proposed restriction operator compared to a PETSc VecScatter.

3.1.3.2 Prolongation and Partition of Unity Operators

Similar to the action of the Restriction Operator, the action of

$$\mathbf{y} = \sum_{i=1}^{P} R_i^T D_i \mathbf{x}_i \tag{3.16}$$

to a local vector $\mathbf{x}_i$ can be computed without forming R^T explicitly. It is computed by first weighting the local part vector with the discrete partition of unity matrix D_i of size N_i followed by a communication step. The weighted contributions are then accumulated as shown in Figure 20. In DOLFINx, this can be accomplished by calling Listing 13.

Note that the communication direction is now reversed (from ghost to owners), and a different neighbor MPI communicator is used. Also, if d processes share a *dof*, D_i can be defined by setting the corresponding diagonal entry to $1/d$. For instance, in Figure 16, most *dofs* on the overlap region are shared by two processes, and three processes share the *dofs* of the six cells in the center. Also, all *dofs* that are not in the interface or that are not in the overlap region are not shared. Interface *dofs* admit special treatment; the corresponding entry is 1 if the *dof* is owned and 0 otherwise.

Listing 13 Prolongation and partition of unity operators in dolfinx

```
u.x.array[:] = Di * u.x.array
u.x.scatter_reverse(ScatterMode.add)
```

3.1.3.3 Local matrix and local solve

The next ingredient we need to implement is the assembly of the local matrix with an appropriate transmission condition. Instead of assembling a global matrix, each process i assembles a local (to the process) sequential matrix (for instance, we use `MATSEQAIJ`) of size $N_i \times N_i$. The local matrix for the scalar Helmholtz problem can be assembled with the UFL form defined on Listing 9. We have created an extension to dolfinx that allows assembling matrices in the overlap region, which is available in the `odd` namespace. An excerpt of a code for assembling the local matrix and setting the local solver is shown in Listing 14.

This section established the necessary ingredients to define a one-level Restricted Additive Schwarz preconditioner using the FEniCSx framework. A rough sketch of a function that computes the action of M^{-1} to a distributed vector $\mathbf{x}$ using these ingredients is shown in Listing 15. Note that we assumed that the input $\mathbf{x}$ and output $\mathbf{y}$ are DOLFINx distributed vectors, but if we interface with PETSc, we must perform additional copy operations from and to the PETSc vectors. It remains for us now to establish a transmission operator that is suitable for a finite element framework. Hence, the definition of *transmission conditions* is the center of the following sections.

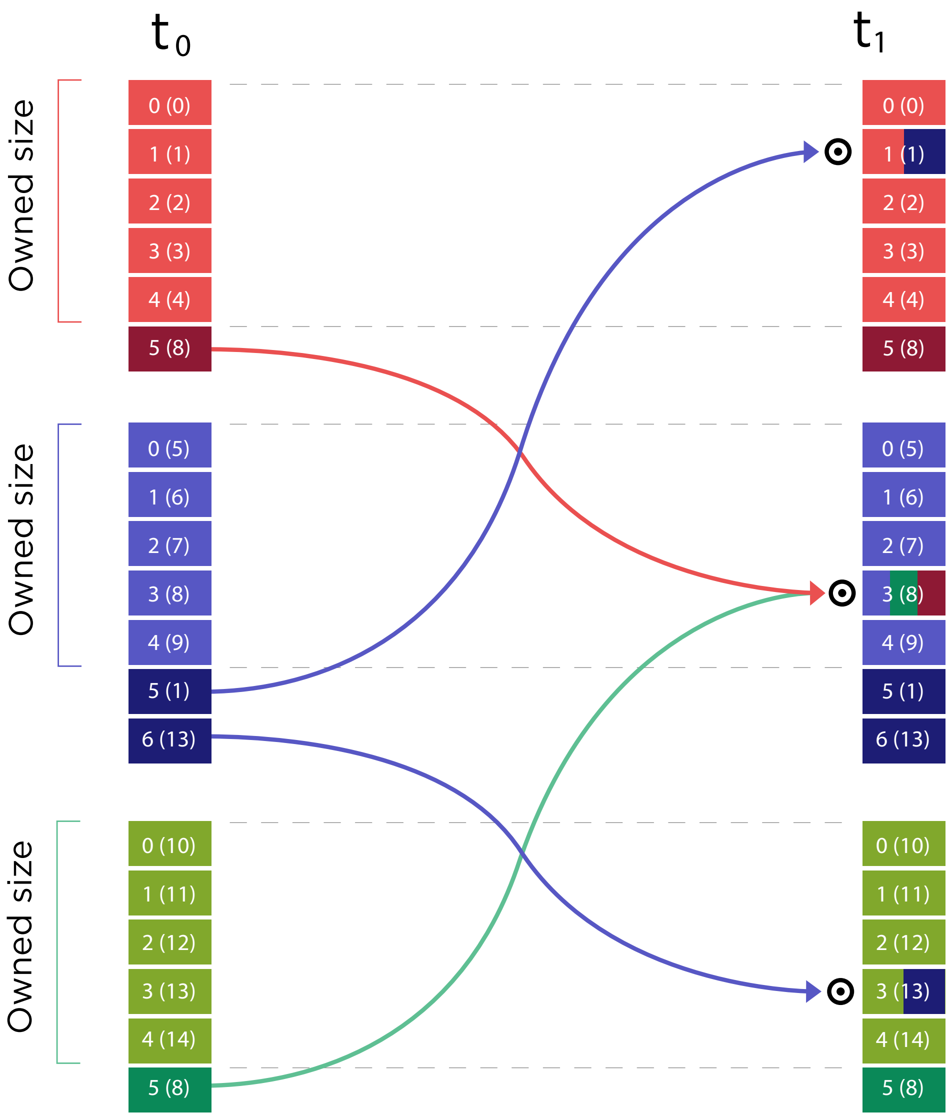


Figure 20 – Reverse scattering of a vector in parallel. Communication direction is ghost to *dof* owner.

Listing 14 Assembly of the Local Matrix and definition of the local solver.

```
seq_comm = MPI.COMM_SELF

# Assemble local form into a PETSc Sequential Matrix
Ai = assemble_local_matrix(seq_comm, a)

# Setup phase
solver = PETSc.KSP().create(seq_comm)
solver.setOperators(A)
solver.setType(PETSc.KSP.Type.PREONLY)
solver.pc.setType(PETSc.PC.Type.LU)
solver.pc.setFactorSolverType("mumps")
...

# Solver phase
solver.solve(x, y)
```

Listing 15 Assembly of the Local Matrix and definition of the local solver.

```
def apply(self, x, y):
    x.scatter_forward()
    solver.solve(x, y)
    y.array[:] = Di * y.array
    y.scatter_reverse(ScatterMode.add)
```

3.2 Transmission Conditions

In the context of time-harmonic wave propagation problems, to achieve satisfactory convergence rates, it is mandatory to use impedance-type transmission conditions for coupling adjacent subdomains (DESPRÉS, 1991; COLLINO; GHANEMI; JOLY, 2000). Over the last decades, optimized transmission conditions have been the subject of several works (DOLEAN; JOLIVET; NATAF, 2015; ANTOINE; GEUZAINE, 2017; GANDER; ZHANG, 2016). The optimal choice of $\mathcal{S}$ in (3.4) is the DtN (Dirichlet-to-Neuman) map (DOLEAN; JOLIVET; NATAF, 2015), as we shall see in the next section. Because of its pseudo-differential nature, the DtN is a non-local operator and hence not suited to the finite element framework. Optimized Schwarz methods, therefore, use some local approximation (DOLEAN; JOLIVET; NATAF, 2015; ANTOINE; GEUZAINE, 2017). In the following list, we present the four most prominent transmission conditions, $\mathcal{S}$, found in the literature for the scalar Helmholtz problem:

- $\mathcal{S}_{O0}(p, q)$ (DOLEAN; JOLIVET; NATAF, 2015; BENAMOU; DESPRÈS, 1997) - Zero-order polynomial approximation of the DtN symbol in the Fourier domain. It is a generalization of the transmission condition proposed by Després (DESPRÉS, 1991; BENAMOU;

DESPRÈS, 1997). The generalized operator can be written as:

$$\mathcal{S}_{O0}(p,q) = p + jq \qquad p, q \in \mathbb{R} \tag{3.17}$$

where p and q are two free parameters, and their choice strongly determines the convergence rate of the OSM algorithm.

- $\mathcal{S}_{O2}(\alpha, \beta)$ (GANDER; ZHANG, 2016; GANDER; MAGOULES; NATAF, 2002) - Second order polynomial approximation of the DtN symbol in the form:

$$\mathcal{S}_{O2}(\alpha,\beta) = \alpha + \beta\partial_{\tau\tau} \qquad \alpha, \beta \in \mathbb{C} \tag{3.18}$$

where α and β are obtained from the solution of a min-max optimization problem, involving the convergence rate of the algorithm (spectral radius of the iteration matrix).

- $\mathcal{S}_{sq}(N_p, \alpha, \epsilon)$ (GANDER; SCHÄDLE, 2011; BOUBENDIR; ANTOINE; GEUZAINE, 2012) - Instead of a polynomial approximation, a rational approximation was proposed in (BOUBENDIR; ANTOINE; GEUZAINE, 2012):

$$\mathcal{S}_{sq} = -jkC_0 - jk\sum_{l=1}^{N_p} A_l\nabla_\tau\left(\frac{1}{k_\epsilon^2}\nabla_\tau\right)\left(I + B_l\nabla_\tau\left(\frac{1}{k_\epsilon^2}\nabla_\tau\right)\right)^{-1} \tag{3.19}$$

where C_0, A_l e B_l are coefficients of the padé rational approximation of order N_p. This transmission condition leads to a DDM with near-optimal convergence for the evanescent modes and competitive for the propagating modes when compared to $\mathcal{S}_{O2}$. The implementation of this TC is more involving because it requires the solution of N_p auxiliary problems of the size of the interface.

- $\mathcal{S}_{pml}(\sigma)$ (VION; GEUZAINE, 2014; ENGQUIST; YING, 2011; STOLK, 2013; STOLK, 2017) - PMLs are also used as a volumetric approximation of the DtN map. They are constructed by attaching a layer Ω_{pml} to the transmission interfaces. For example, in Cartesian coordinates a transformation with absorption profile σ can be applied:

$$\sigma(\boldsymbol{x}_{pml}) = \frac{1}{k\left(\boldsymbol{x}_{pml} - \delta\right)}. \tag{3.20}$$

The quality of the operator $\mathcal{S}_{pml}$ is directly related to the number of layers. In the case of multiple subdomains this strategy can greatly increase the computational cost.

Both $\mathcal{S}_{pml}$ and $\mathcal{S}_{sq}$ involve additional degrees of freedom and are suited for regular partitions of the domain. Since we consider general domain partitions produced by automatic graph partitioners, in what follows, we assume polynomial transmission conditions.

3.2.1 Convergence analysis for a model problem

Consider the model domain, $\Omega = \mathbb{R}^2$, with Sommerfeld's radiation conditions at infinity. As it is usually done to study optimized Schwarz methods, we decompose the domain into two two overlapping subdomains (with $\delta > 0$), such that:

$$\begin{aligned} \Omega_1 &= (-\infty, \delta) \times \mathbb{R} \qquad \Gamma_{12}(x = \delta), \\ \Omega_2 &= (0, +\infty) \times \mathbb{R} \qquad \Gamma_{21}(x = 0). \end{aligned} \tag{3.21}$$

Due to the linearity of the problem, it is sufficient to analyze the convergence for the case where $f = 0$, which is equivalent to analyzing the error (DOLEAN; JOLIVET; NATAF, 2015). Omitting Sommerfeld's condition, the algorithm (3.3) for this two sub-domain case can be written as

$$\left.\begin{aligned} \Delta u_1^{n+1} + k^2 u_1^{n+1} = 0 \quad &\text{in} \quad \Omega_1 \\ (\partial_{\hat{n}_1} + \mathcal{S}_1)(u_1^{n+1}) = (\partial_{\hat{n}_1} + \mathcal{S}_1)(u_2^n) \quad &\text{on} \quad \Gamma_{12} \end{aligned}\right\}. \tag{3.22}$$

$$\left.\begin{aligned} \Delta u_2^{n+1} + k^2 u_2^{n+1} = 0 \quad &\text{in} \quad \Omega_2 \\ (\partial_{\hat{n}_2} + \mathcal{S}_2)(u_2^{n+1}) = (\partial_{\hat{n}_2} + \mathcal{S}_2)(u_1^n) \quad &\text{on} \quad \Gamma_{21} \end{aligned}\right\}. \tag{3.23}$$

We consider the following convention for the Fourier transform

$$\hat{f}(\nu) := \mathcal{F}\left[f(y)\right](\nu) = \int_{-\infty}^{\infty} f\left(y\right) e^{j\nu y} dy. \tag{3.24}$$

Applying the above mentioned Fourier transform in the y-direction to the algorithms in (3.22) and (3.23), we get an ordinary differential equation (ODE) in x for both Ω_1 and Ω_2:

$$\left.\begin{aligned} \partial_{xx}\hat{u}_1^{n+1} + (k^2 - \nu^2)\,\hat{u}_1^{n+1} = 0 \quad &\text{in} \quad \Omega_1 \\ (\partial_x + s_1)(\hat{u}_1^{n+1}) = (\partial_x + s_1)(\hat{u}_2^n) \quad &\text{on} \quad \Gamma_{12} \end{aligned}\right\}. \tag{3.25}$$

$$\left.\begin{aligned} \partial_{xx}\hat{u}_2^{n+1} + (k^2 - \nu^2)\,\hat{u}_2^{n+1} = 0 \quad &\text{in} \quad \Omega_2 \\ (-\partial_x + s_2)(\hat{u}_2^{n+1}) = (-\partial_x + s_2)(\hat{u}_1^n) \quad &\text{on} \quad \Gamma_{21} \end{aligned}\right\}. \tag{3.26}$$

The general solutions of these ODEs take the form of:

$$\hat{u}_i^{n+1}(x, \nu) = A_i^{n+1} e^{\lambda(\nu)x} + B_i^{n+1} e^{-\lambda(\nu)x} \tag{3.27}$$

where $\lambda(\nu) = \sqrt{\nu^2 - k^2}$ denotes the root of the characteristic equation $\lambda^2 + (k^2 - \nu^2) = 0$. Since Sommerfeld's radiation condition excludes growing solutions at infinity as well as incoming modes from infinity we can rewrite the solutions as:

$$\hat{u}_1^{n+1}(x, \nu) = \hat{u}_1^{n+1}(\delta, \nu) e^{\lambda(\nu)(x - \delta)}, \tag{3.28}$$

$$\hat{u}_2^{n+1}(x, \nu) = \hat{u}_2^{n+1}(0, \nu) e^{-\lambda(\nu)x}. \tag{3.29}$$

Application of the transmission conditions on Γ_{12} and Γ_{21} yields:

$$\hat{u}_1^{n+1}(\delta,\nu) = \left(\frac{s_1 - \lambda(\nu)}{s_1 + \lambda(\nu)} e^{+\lambda(\nu)\delta}\right) \hat{u}_2^{n}(\delta,\nu), \tag{3.30}$$

$$\hat{u}_2^{n+1}(0,\nu) = \left(\frac{s_2 - \lambda(\nu)}{s_2 + \lambda(\nu)} e^{-\lambda(\nu)\delta}\right) \hat{u}_1^{n}(0,\nu). \tag{3.31}$$

So we finally get the recursion relations

$$\hat{u}_1^{n+1}(\delta,\nu) = \left(\frac{s_1 - \lambda(\nu)}{s_1 + \lambda(\nu)} \frac{s_2 - \lambda(\nu)}{s_2 + \lambda(\nu)} e^{-2\lambda(\nu)\delta}\right) \hat{u}_1^{n-1}(\delta,\nu), \tag{3.32}$$

$$\hat{u}_2^{n+1}(0,\nu) = \left(\frac{s_1 - \lambda(\nu)}{s_1 + \lambda(\nu)} \frac{s_2 - \lambda(\nu)}{s_2 + \lambda(\nu)} e^{-2\lambda(\nu)\delta}\right) \hat{u}_2^{n-1}(0,\nu). \tag{3.33}$$

The convergence factor $\rho(\nu)$ for a double iteration is then defined by:

$$\rho(\nu) = \frac{s_1(\nu) - \lambda(\nu)}{s_1(\nu) + \lambda(\nu)} \frac{s_2(\nu) - \lambda(\nu)}{s_2(\nu) + \lambda(\nu)} e^{-2\lambda(\nu)\delta} \tag{3.34}$$

From (3.34), convergence in two steps is only achievable if $s = \lambda(\nu)$, which leads to non-local operators in real space (inverse Fourier transform of a square root operator). The standard choice is to use polynomial approximations of $\lambda(\nu)$ leading to local operators instead. In the Fourier domain the operators $\mathcal{S}_{O0}$, $\mathcal{S}_{O2}$ become:

$$\begin{aligned} s_{O0} &= p + jq \\ s_{02} &= \alpha + \beta\nu^2 \end{aligned} \tag{3.35}$$

hence the names 0th and 2nd order polynomial approximations. Substituting the transformed TCs into (3.34), the free parameters can be optimized to improve convergence. Similar procedure is used to design absorbing boundary conditions for domain truncation (DRUSKIN; GUTTEL; KNIZHNERMAN, 2016), except that the relative error is optimized and δ corresponds to the distance of the source to the truncation boundary.

3.2.2 Convergence analysis for the many sub-domain case

This subsection is based on our recently published paper (BARATTA; SILVA, 2018). We will concentrate on overlapping DDM again since the overlap can take care of the higher spatial frequencies (evanescent modes) due to the exponential decay on the convergence factor. However, the propagating modes are the main reason for the non-scalability concerning the number of subdomains because they can go much further than the evanescent modes (GANDER; ZHANG, 2016). Without loss of generality, the theoretical part will be restricted to the homogeneous problem (k is constant), but it is possible to extend the analysis to the heterogeneous case by incorporating materials in the computations (DOLEAN; GANDER; VENEROS, 2016).

Instead of following the conventional approach to analyze the algorithm convergence for different transmission conditions, as described in the previous section, we investigate the model problem described in Figure 21 and follow a procedure similar to the one presented in (CIARAMELLA; GANDER, 2017; CIARAMELLA; GANDER, 2018). The domain $\Omega = \mathbb{R}^2$ is now decomposed into P layered overlaping subdomains Ω_i, $i = 0 \dots P-1$. The overlap size is 2δ and the length of the domain is $L + 2\delta$. Since $\delta > 0$, $\Sigma_{i,j} \neq \Sigma_{j,i}$ and the boundaries of Ω_i are located at $\Sigma_{i,i-1} = x_i^l \times \mathbb{R}$ and $\Sigma_{i,i-1} = x_i^r \times \mathbb{R}$.

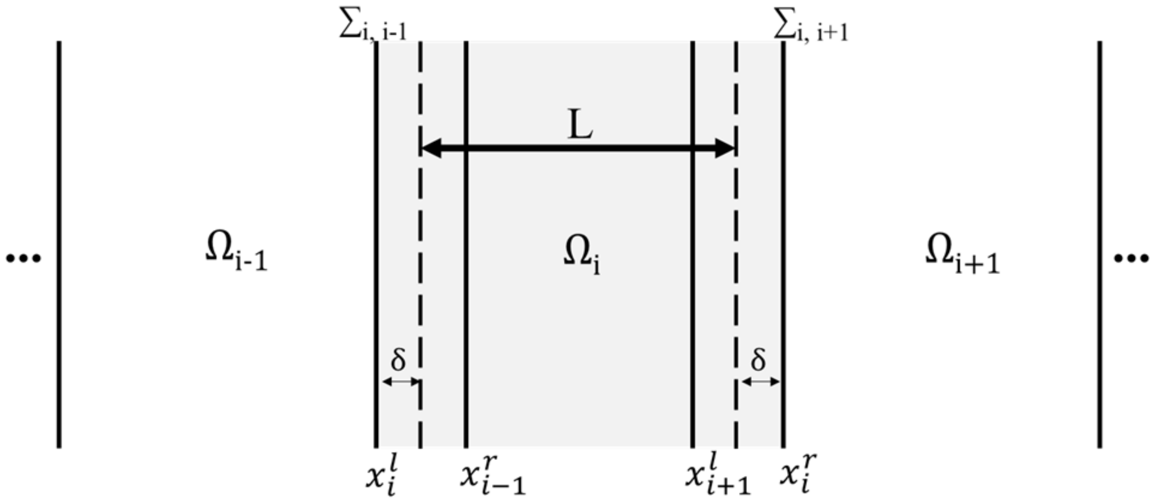


Figure 21 – Geometric description of subdomain Ω_i in a layered overlaping decomposition.

We can write the Fourier-transformed problem for sub-domain Ω_i as:

$$\left.\begin{array}{rll} \partial^2_{xx}\hat{u}_i^{n+1}(x,\nu) + (k^2-\nu^2)\hat{u}_i^{n+1}(x,\nu) = 0 & \textit{in} & \Omega_i \\ (-\partial_x + s)(\hat{u}_i^{n+1})(x_i^l,\nu) = (-\partial_x + s)(\hat{u}_{i-1}^n)(x_i^l,\nu) & \textit{on} & \Gamma_{i,i-1} \\ (\partial_x + s)(\hat{u}_i^{n+1})(x_r^r,\nu) = (\partial_x + s)(\hat{u}_{i+1}^n)(x_r^r,\nu) & \textit{on} & \Gamma_{i,i+1} \end{array}\right\} \tag{3.36}$$

The general solution of (3.36) takes the form:

$$\hat{u}_i^{n+1} = A_i^{n+1}e^{\lambda(x-x_i^m)} + B_i^{n+1}e^{-\lambda(x-x_i^m)}, \tag{3.37}$$

where $x_i^m = (x_r - x_i)/2$ is the mid-point of sub-domain i. Applying the TC on the left $(\Gamma_{i,i-1})$:

$$(s-\lambda)A_i^{n+1}e^{-\lambda a} + (s+\lambda)B_i^{n+1}e^{\lambda a} = (s-\lambda)A_{i-1}^n e^{\lambda b} + (s+\lambda)B_{i-1}^n e^{-\lambda b} \tag{3.38}$$

and on the right $(\Gamma_{i,i+1})$:

$$(s_i+\lambda)A_i^{n+1}e^{\lambda a} + (s_i-\lambda)B_i^{n+1}e^{-\lambda a} = (s_i+\lambda)A_{i+1}^n e^{-\lambda b} + (s_i-\lambda)B_{i+1}^n e^{\lambda b}, \tag{3.39}$$

Allows us to isolate B_i^{n+1} such that:

$$B_i^{n+1} = -\frac{s-\lambda}{s+\lambda}e^{-2\lambda a}A_i^{n+1} + \frac{s-\lambda}{s+\lambda}e^{\lambda(b-a)}A_{i-1}^n + e^{-\lambda(a+b)}B_{i-1}^n. \tag{3.40}$$

Substituting (3.40) on (3.39) we obtain an expression relating the current coefficient A_i^{n+1} to the coefficients of neighbour subdomains on the previous iteration

$$A_i^{n+1} = \frac{1}{\gamma}\left[\alpha_1 A_{i-1}^n + \alpha_2 B_{i-1}^n + \alpha_3 A_{i+1}^n + \alpha_4 B_{i+1}^n\right]. \tag{3.41}$$

Applying a similar procedure for B_i^{n+1}, we get:

$$B_i^{n+1} = \frac{1}{\gamma}\left[\alpha_4 A_{i-1}^n + \alpha_3 B_{i-1}^n + \alpha_2 A_{i+1}^n + \alpha_1 B_{i+1}^n\right]. \tag{3.42}$$

The parameters $\{\alpha_i\}$ and γ, are functions of the transmission condition (s), overlap (δ) and sub-domain width (L):

$$\begin{aligned}
\gamma &= (s+\lambda)^2 \cdot e^{\lambda(L+2\delta)} - (s-\lambda)^2 \cdot e^{-\lambda(L+2\delta)},\\
\alpha_1 &= -(s-\lambda)^2 \cdot e^{-2\lambda\delta},\\
\alpha_2 &= -(s-\lambda)\cdot(s+\lambda)\cdot e^{-\lambda L},\\
\alpha_3 &= +(s+\lambda)^2 \cdot e^{+2\lambda\delta},\\
\alpha_4 &= +(s-\lambda)\cdot(s+\lambda)\cdot e^{+\lambda L},
\end{aligned}$$

The scalar field $\hat{u}^{n+1}$ in the fourier domain can be fully described by the set of coefficients A_i^{n+1}, B_i^{n+1} for $i = 1 \cdots N_s$. If we define :

$$\hat{\mathbf{u}}^{n+1} = \begin{bmatrix} A_1^{n+1} & B_1^{n+1} & \cdots & A_{N_s}^{n+1} & B_{N_s}^{n+1}\end{bmatrix}^T, \tag{3.43}$$

We can develop a recurrence relation between the field at iteration $n+1$ and n through the iteration matrix $\Psi(s,\nu) \in \mathbb{C}^{2P\times 2P}$:

$$\hat{\mathbf{u}}^{n+1} = \Psi(s,\nu)\hat{\mathbf{u}}^n \tag{3.44}$$

A better convergence estimate, thus, can be obtained by calculating the spectral radius of the iteration matrix, $\rho(\Psi(s,\nu))$. Differently from the standard procedure, in our approach we solve the min-max problem numerically:

$$\min_s \left(\max_{\nu\in F} |\rho\left(\Psi(s,\nu)\right)|\right) \tag{3.45}$$

When $\nu^2 = k^2$, $|\rho(\Psi(s,\nu))| \geq 1$, for any value of p and q or α and β. For this reason, it is not possible to optimize the convergence factor over all Fourier modes (DOLEAN; JOLIVET; NATAF, 2015). So we choose a subset of modes $F \subset \mathbb{R}$, over which the optimization takes place, excluding modes near resonance.

We use a Stochastic-based optimization approach to obtain the free parameters in (3.35). More specifically, in this work, we apply the Differential Evolution algorithm (STORN; PRICE, 1997). The iteration matrix has a small size compared to the global number of degrees of freedom, and it is sparse (only four non-zeros per row). Therefore the computational cost for obtaining $\rho(\Psi(s,\nu))$ is rather small.

In Figure 22 and Figure 23 we compare the convergence estimates for TC obtained using the standard (3.34) and proposed (multi-domain) approaches. In the standard procedure, the TC is calculated only once for $P = 2$. In our approach, for each P value a different TC is obtained. For a given choice of free parameters, an increase in P affects only the propagative modes. So, we let the focus to be placed dynamically on the propagative modes as the number of subdomains grows, allowing a certain degradation in the convergence of the evanescent modes. By this means, we achieve a global improvement of the convergence that is dominated by the propagating modes. In the limit $P \to \infty$, the transmission conditions lead to Taylor expansion of order q of the DtN symbol.

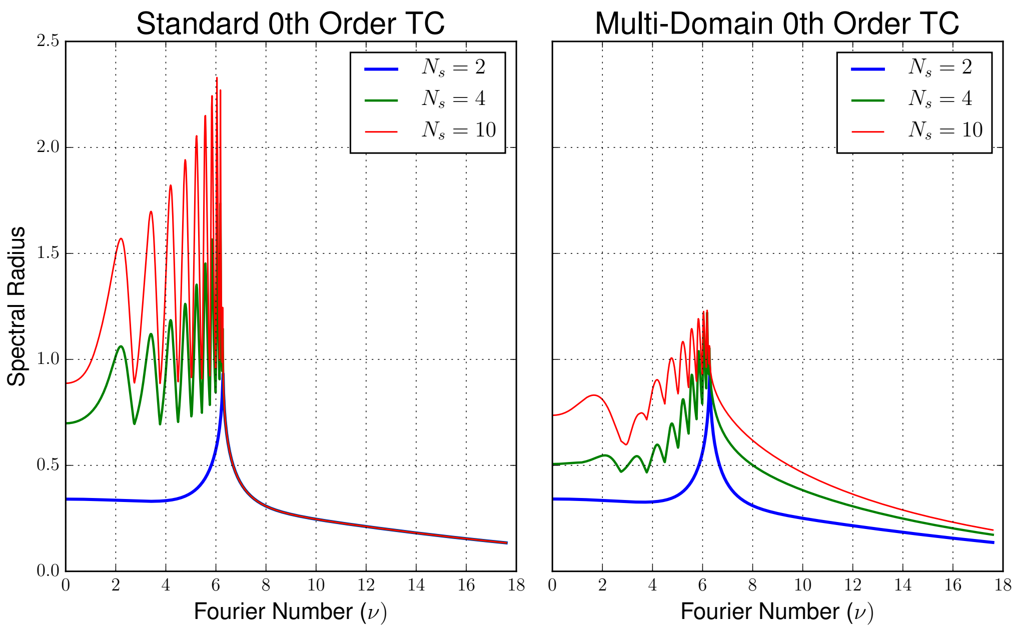


Figure 22 – Spectral Radius of the Schwarz iteration matrix: $k = 2\pi$, $L = 5$ and $\delta = 0.1$. Using 0th order transmission conditions.

3.2.3 Numerical Experiments

We consider a model problem with a known exact solution, a plane wave $p = e^{jk(\mathbf{d}\cdot\mathbf{x})}$ scattering by an infinitely long perfect conducting metallic cylinder with $r = 0.5m$. A first order absorbing boundary conditions (ABC) with a fictitious circular boundary Γ_∞ with radius $R = 1.5m$ is used to truncate the domain. For the implementation of the DDM, we use the volume formulation of the Optimized Restricted Additive Schwarz (ORAS) as described in (3.14).

In our experiments we set the discretization density to $n_\lambda = \frac{\lambda}{h} = 25$, using linear triangular finite elements. Each subdomain, Ω_i, is obtained by extending a non-overlapping partition by one element mesh layer. Each sub-system is factored only once, using a multifrontal LU factorization (DAVIS, 2004), and then in each iteration back-substitutions steps are performed.

Figure 24 shows the cyclic domain partition and absolute value of the solution (total electric field in z-direction) for $k = 5\pi$ and $N_s = 50$. Starting from a zero initial solution, on

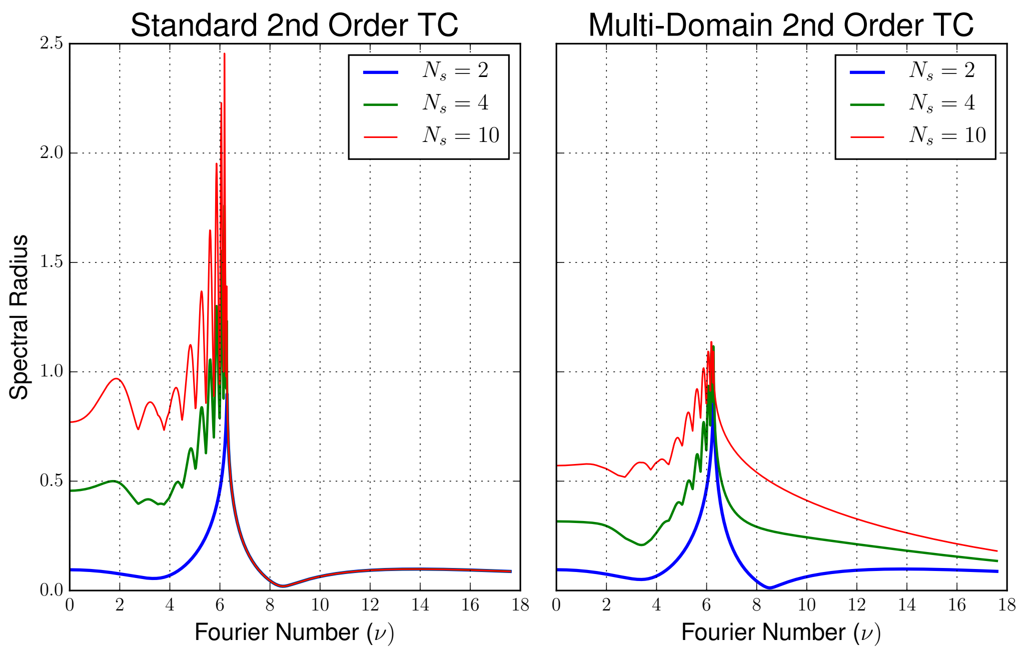


Figure 23 – Spectral Radius of the Schwarz iteration matrix: $k = 2\pi$, $L = 5$ and $\delta = 0.1$. Using 2nd order transmission conditions.

the left we show the solution after one iteration, and on the right after forty iterations, using multi-domain second order TC.

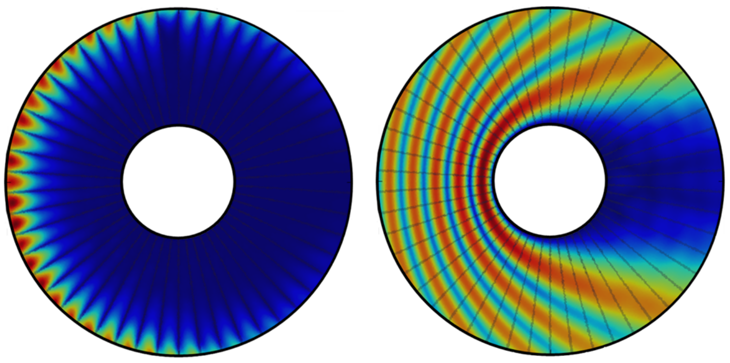


Figure 24 – Cyclic Domain Decomposition into 50 subdomains: Solution (left) after one iteration, (right) after forty iterations.

Emphasis will be given to the convergence of the methods, measured by the iteration count when a relative 10^{-4} decrease of the residual is reached. The relative error remained below 3.0% for all experiments. The ORAS is used as a preconditioner for the GMRES method. In addition to the two TC discussed previously, the standard OO_0 and OO_2 and our multi-domain version MO_0 and MO_2, we included in the experiments TC based on Taylor approximation of DtN symbol TO_0 and TO_2. Iteration counts for $k = 5\pi$ and an increasing number of subdomains are presented in Table 2.

This numerical experiment confirms the expected improvements in convergence. We obtain fewer iteration counts with our approach compared to other polynomial TC of the same order. With a non-zero overlap, as P increases, the real part of the multi-domain TC tends to zero and becomes similar to the TC based on Taylor expansion, but with an optimized imaginary part. However, differently from Taylor TC, our approach leads to a convergent non-overlapping DDM.

Table 2 – Iteration count of GMRES preconditioned with ORAS

# Subdomains	OO_0	TO_0	MO_0	OO_2	TO_2	MO_2
5	22	19	19	14	12	11
10	31	26	26	19	15	14
25	63	54	52	37	30	27
50	127	84	82	75	53	40

To further improve the convergence, DDM must be combined with a coarse space correction. So, we applied the multi-domain TC to an overlapping modified version of the Double Sweep preconditioner (VION; GEUZAINE, 2014). In Figure 25 we show the relative residual of GMRES using ORAS and Double Sweep as preconditioners, which indicates a promising combination between a coarse space correction and appropriate multi-domain TC.

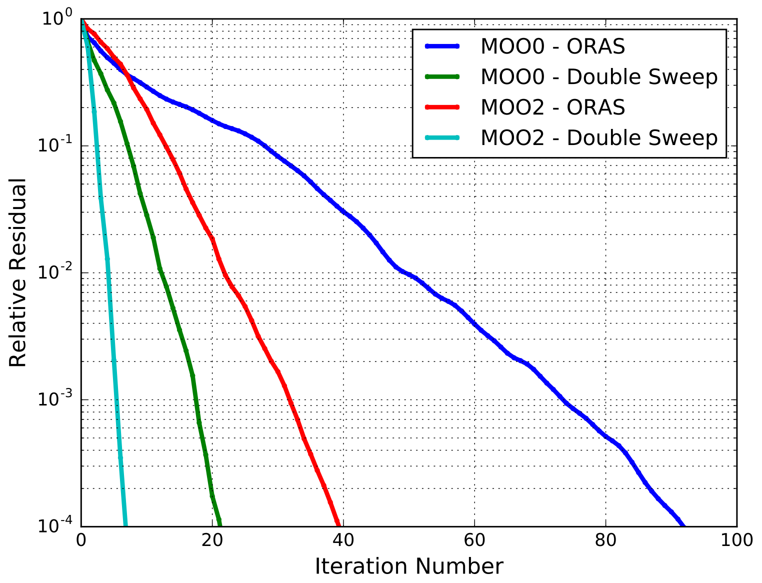


Figure 25 – Relative residual of GMRES for different preconditioners and TC: $k = 5\pi$.

3.3 Schwarz Methods for Maxwell's Equation

Although many authors have proposed optimized domain decomposition techniques for the first-order system (2.6) and (2.7), e.g.: (DOLEAN; GANDER; GERARDO-GIORDA, 2009; BOUAJAJI et al., 2012), the second-order formulation, which is more interesting in the computational point of view, has received limited consideration. However, a recently published manuscript reported that both formulations eventually lead to similar iterative methods in terms of convergence and transmission conditions (DOLEAN et al., 2015). Therefore the advances made to the first-order system can be directly applied to the second-order formulation.

As for the scalar case, many different transmission conditions have been proposed over the past years. A good overview in the context of finite element discretization can be found in (ANTOINE; GEUZAINE, 2017). Impedance transmission conditions were first proposed in (DESPRÉS, 1992) with pre-defined parameters. An extension to the optimized framework was presented and analyzed in (DOLEAN; GANDER; GERARDO-GIORDA, 2009). Second-order transmission conditions have also been the subject of research of different groups (DOLEAN et al., 2015; PENG; LEE, 2010; PENG; LEE, 2011). A perspective on the use of PMLs as transmission condition is given in (GANDER; ZHANG, 2014) and on the use of rational padé approximations, instead of the polynomial ones, is given in (BOUAJAJI; ANTOINE; GEUZAINE, 2014). All of these approaches share a crucial point. Their effectiveness stems from the quality of approximation of the operators linking the magnetic ($\mathbf{M}$) and the electric ($\mathbf{J}$) surface currents at the interfaces through the Magnetic-to-Electric (MtE) map (BOUAJAJI; ANTOINE; GEUZAINE, 2014).

Since analyzing the convergence of the Schwarz algorithm for Maxwell's equations is more involving than for the Helmholtz, we present only the optimization for the two-subdomain case. However, it was shown in (DOLEAN; GANDER; VENEROS, 2018) that convergence analysis considering only transverse magnetic (TM) and transverse electric (TE) modes in two dimensions can be extrapolated to arbitrary electromagnetic fields in three dimensions without loss of generality. This simplification allows the analysis of the many-subdomain case and heterogeneity using the method presented in the last subsection.

Apart from the technicalities stemming from the vectorial nature of electromagnetic fields, the analysis presented here is very similar to the one presented in Section 3.2.1. We proceed by decomposing $\Omega = \mathbb{R}^3$, now a three-dimensional domain, into two subdomains Ω_1 and Ω_2, with interfaces Γ_{12} and Γ_{21} aligned with the planes $x = \delta$ and $x = 0$ respectively. Again we advance the OSM iteration by computing concurrently

$$\left.\begin{aligned} \nabla\times\nabla\times\mathbf{E}_1^{n+1} - k^2\mathbf{E}_1^{n+1} = \mathbf{0} \quad &\text{in} \quad \Omega_1 \\ \mathcal{B}(\mathbf{E}_1^{n+1}) = \mathcal{B}(\mathbf{E}_2^{n}) \quad &\text{on} \quad \Gamma_{12} \end{aligned}\right\}. \tag{3.46}$$

$$\left.\begin{aligned} \nabla\times\nabla\times\mathbf{E}_2^{n+1} - k^2\mathbf{E}_2^{n+1} = \mathbf{0} \quad &\text{in} \quad \Omega_2 \\ \mathcal{B}(\mathbf{E}_2^{n+1}) = \mathcal{B}(\mathbf{E}_1^{n}) \quad &\text{on} \quad \Gamma_{21} \end{aligned}\right\}. \tag{3.47}$$

We consider here two different formulations for the linear operator $\mathcal{B}$. The 0th order formulation proposed by (DESPRÉS, 1992):

$$\mathcal{B}(\mathbf{E}) = (\nabla\times\ \mathbf{E})\times\mathbf{n} + jk\mathbf{n}\times(\mathbf{E}\times\mathbf{n}). \tag{3.48}$$

And the second order formulation proposed in (PENG; LEE, 2010; RAWAT; LEE, 2010):

$$\begin{aligned}\mathcal{B}(\mathbf{E}) &= \left(I + \left(\alpha_1 \mathcal{S}_{TM} + \alpha_2 \mathcal{S}_{TE}\right)\right)\left(\mathbf{n} \times \nabla \times \mathbf{E}\right) \\ &+ jk\left(I - \left(\alpha_3 \mathcal{S}_{TM} + \alpha_4 \mathcal{S}_{TE}\right)\right)\left(\mathbf{n} \times \left(\mathbf{E} \times \mathbf{n}\right)\right)\end{aligned} \tag{3.49}$$

where $\{\alpha_i\}$ are free parameters that can be optimized for improved convergence, and $\mathcal{S}_{TM} = \nabla_\tau \nabla_\tau$ and $\mathcal{S}_{TE} = \nabla_\tau \times \nabla_\tau \times$. We extend the convergence analysis to overlapping decomposition in the subsequent paragraphs, not available in the original papers. These transmission conditions are well adapted to variational formulations, see for example (2.49), and the presence of operators that act on TM and TE components separately dramatically simplifies the calculations. In what follows, we will use the following convention for the Fourier transform in y and z directions:

$$\hat{\mathbf{f}}(x;\nu) := \mathcal{F}\left[\mathbf{f}(y,z)\right](\nu) = \int_{-\infty}^{\infty} \mathbf{f}\left(x,y,z\right) e^{j(\nu_y y + \nu_z z)} dy\, dz \tag{3.50}$$

We take a Fourier transform of the curl-curl Maxwell's equations in (3.46) and (3.47), so that for $\hat{\mathbf{E}} = [\hat{E}_x, \hat{E}_y, \hat{E}_z]^T$ we get:

$$\begin{cases} -k^2 \hat{E}_x^1 + j\nu_y \frac{d\hat{E}_y^1}{dx} + j\nu_y \frac{d\hat{E}_z^1}{dx} + \left(\nu_y^2 + \nu_y^2\right) \hat{E}_x^1 = 0 \\ -k^2 \hat{E}_y^1 + j\nu_y \frac{dE_x}{dx} - \nu_y \nu_y \hat{E}_z^1 - \frac{d^2 \hat{E}_y^1}{dx^2} + \nu_y^2 \hat{E}_y^1 = 0 \\ -k^2 \hat{E}_z^1 + j\nu_y \frac{dE_x}{dx} - \nu_y \nu_y \hat{E}_y^1 - \frac{d^2 \hat{E}_z^1}{dx^2} + \nu_y^2 \hat{E}_z^1 = 0 \end{cases} \tag{3.51}$$

$$\begin{cases} -k^2 \hat{E}_x^2 + j\nu_y \frac{d\hat{E}_y^2}{dx} + j\nu_y \frac{d\hat{E}_z^2}{dx} + \left(\nu_y^2 + \nu_y^2\right) \hat{E}_x^2 = 0 \\ -k^2 \hat{E}_y^2 + j\nu_y \frac{dE_x}{dx} - \nu_y \nu_y \hat{E}_z^2 - \frac{d^2 \hat{E}_y^2}{dx^2} + \nu_y^2 \hat{E}_y^2 = 0 \\ -k^2 \hat{E}_z^2 + j\nu_y \frac{dE_x}{dx} - \nu_y \nu_y \hat{E}_y^2 - \frac{d^2 \hat{E}_z^2}{dx^2} + \nu_y^2 \hat{E}_z^2 = 0 \end{cases} \tag{3.52}$$

And for each ODE system we have the following general solution (SHAMPINE, 2018):

$$\hat{\mathbf{E}} = \begin{bmatrix} \dfrac{j\left(A_3 \nu_y + A_1 \nu_z\right) e^{-\lambda x}}{\lambda} - \dfrac{j\left(A_4 \nu_y + A_2 \nu_z\right) e^{\lambda x}}{\lambda} \\ A_3 e^{-\lambda x} + A_4 e^{\lambda x} \\ A_1 e^{-\lambda x} + A_2 e^{\lambda x} \end{bmatrix}. \tag{3.53}$$

Applying the Silver-Müller radiation condition radiation condition and referencing the solutions to their boundaries, we get

$$\hat{\mathbf{E}}_1 = e^{+\lambda(x-\delta)} \left[-\frac{j\left(A_2 \nu_z + A_4 \nu_y\right)}{\lambda},\ A_4,\ A_2 \right]^T, \tag{3.54}$$

$$\hat{\mathbf{E}}_2 = e^{-\lambda x} \left[\frac{j\left(A_1 \nu_z + A_3 \nu_y\right)}{\lambda}, A_3, A_1 \right]^T. \tag{3.55}$$

Electromagnetic fields in source-free regions are often represented in terms of transverse electric (TE) and transverse magnetic (TM) modes to simplify analysis (HARRINGTON, 1961). In this decomposition, we represent the waves as the sum of two waves having particular polarizations: TE, the component for which $\mathbf{E}_i$ is perpendicular to the plane of incidence (Γ_i), and the TM for which $\mathbf{H}_i$ is perpendicular to the plane of incidence. The local solutions (3.54) and (3.55) can be re-written using the TE-TM decomposition:

$$\hat{\mathbf{E}}_1 = A_{TM}\hat{\mathbf{E}}_1^{TM} + A_{TE}\hat{\mathbf{E}}_1^{TE}, \tag{3.56}$$

$$\hat{\mathbf{E}}_2 = A_{TM}\hat{\mathbf{E}}_2^{TM} + A_{TE}\hat{\mathbf{E}}_2^{TE}. \tag{3.57}$$

Applying the TE-TM decomposition to (3.54) and (3.55) yields:

$$\hat{\mathbf{E}}_1^{TM} = e^{+\lambda(x-\delta)}\left[0, -\tfrac{\nu_z}{\nu_y}, 1\right]^T, \tag{3.58}$$

$$\hat{\mathbf{E}}_1^{TE} = e^{+\lambda(x-\delta)}\left[-j\tfrac{|\nu|^2}{\nu_y\lambda}, 1, \tfrac{\nu_z}{\nu_y}\right]^T, \tag{3.59}$$

$$\hat{\mathbf{E}}_2^{TM} = e^{-\lambda x}\left[0, -\tfrac{\nu_z}{\nu_y}, 1\right]^T, \tag{3.60}$$

$$\hat{\mathbf{E}}_2^{TE} = e^{-\lambda x}\left[-j\tfrac{|\nu|^2}{\nu_y\lambda}, 1, \tfrac{\nu_z}{\nu_y}\right]^T. \tag{3.61}$$

Here we present only convergence estimates for second order transmission conditions, but similar calculations for zeroth order conditions follow directly. The first step is to compute the action of the interface operators from (3.49) on the local decomposed solutions. After a few calculations we obtain:

$$\begin{aligned}\hat{\mathcal{B}}_1\left(\hat{\mathbf{E}}_1\right) &= A_{TE}\left[\left(1+\alpha_1\hat{\mathcal{S}}_{TM}\right)\lambda + k\left(1-\alpha_3\hat{\mathcal{S}}_{TM}\right)\right]\left(\hat{\mathbf{E}}_1^{TE}\times\mathbf{n}_1\right)\\ &+ A_{TM}\left[\left(1+\alpha_2\hat{\mathcal{S}}_{TE}\right)\left(-\frac{k^2}{\lambda}\right) + jk\left(1-\alpha_4\hat{\mathcal{S}}_{TE}\right)\right]\left(\hat{\mathbf{E}}_1^{TM}\times\mathbf{n}_1\right)'\end{aligned} \tag{3.62}$$

$$\begin{aligned}\hat{\mathcal{B}}_2\left(\hat{\mathbf{E}}_2\right) &= A_{TE}\left[\left(1+\alpha_1\hat{\mathcal{S}}_{TM}\right)\lambda - k\left(1-\alpha_3\hat{\mathcal{S}}_{TM}\right)\right]\left(\hat{\mathbf{E}}_2^{TE}\times\mathbf{n}_2\right)\\ &+ A_{TM}\left[\left(1+\alpha_2\hat{\mathcal{S}}_{TE}\right)\left(-\frac{k^2}{\lambda}\right) - jk\left(1-\alpha_4\hat{\mathcal{S}}_{TE}\right)\right]\left(\hat{\mathbf{E}}_2^{TM}\times\mathbf{n}_2\right)\cdot\end{aligned} \tag{3.63}$$

It is possible to notice a block structure in (3.62) and (3.63), with different components of the transmission operator acting exclusively on the TE or TM modes. We thus obtain

$$\hat{\mathcal{B}}_1\left(\hat{\mathbf{E}}_1\right) = B_1\begin{bmatrix} A_{1,TE} \\ A_{1,TM}\end{bmatrix} \tag{3.64}$$

$$\hat{\mathcal{B}}_2\left(\hat{\mathbf{E}}_1\right) = B_2\begin{bmatrix} A_{2,TE} \\ A_{2,TM}\end{bmatrix} \tag{3.65}$$

where:

$$B_1 = (\lambda + jk)\begin{bmatrix} c_1 & \frac{\nu_z}{\nu_y}c_2 \\ \frac{\nu_z}{\nu_y}c_1 & -c_2\end{bmatrix}, B_2 = (\lambda - jk)\begin{bmatrix} -c_3 & -\frac{\nu_z}{\nu_y}c_4 \\ -\frac{\nu_z}{\nu_y}c_3 & c_4\end{bmatrix} \tag{3.66}$$

and c_i is function of α_j and the operators $\hat{\mathcal{S}}_{TE}$ and $\hat{\mathcal{S}}_{TM}$. Therefore, the Schwarz iteration, (3.46) and (3.47), can be re-written as

$$B_1\begin{bmatrix} A_{1,TE}^{n+1} \\ A_{1,TM}^{n+1}\end{bmatrix} = B_2\begin{bmatrix} A_{2,TE}^{n} \\ A_{2,TM}^{n}\end{bmatrix} e^{-\lambda\delta}. \tag{3.67}$$

Since the electromagnetic fields at each iteration can be fully described by $A^{n+1}_{i,TE}$ and $A^{n+1}_{i,TM}$, the convergence factor can be computed by the spectral radius of the iteration matrix Ψ:

$$\begin{bmatrix} A^{n+1}_{1,TE} \\ A^{n+1}_{1,TM} \end{bmatrix} = B_1^{-1} B_2 e^{-\lambda\delta} \begin{bmatrix} A^{n}_{2,TE} \\ A^{n}_{2,TM} \end{bmatrix} = \Psi \begin{bmatrix} A^{n}_{2,TE} \\ A^{n}_{2,TM} \end{bmatrix} \tag{3.68}$$

The iteration matrix has two eigenvalues that can be easily calculated with symbolic computing tools. In this project we use Sympy (MEURER et al., 2017), a Python open-source library for symbolic mathematics, and we obtain the following eigenvalues:

$$\lambda_1 = \left| \frac{1 + (\lambda - jk)(\alpha_1 \lambda - jk\alpha_3)}{1 + (\lambda + jk)(\alpha_1 \lambda + jk\alpha_3)} \right| \tag{3.69}$$

$$\lambda_2 = \left| \frac{1 + (\lambda - jk)(\alpha_2 \lambda - jk\alpha_4)}{1 + (\lambda + jk)(\alpha_2 \lambda + jk\alpha_4)} \right| \tag{3.70}$$

And we get the following convergence factor:

$$\rho(\Psi) = \left| \frac{\lambda + jk}{\lambda - jk} \cdot e^{-\lambda\delta} \right| \max\left\{ |\lambda_1|, |\lambda_2| \right\}. \tag{3.71}$$

again we solve the min-max optimization problem

$$\min_{\{\alpha\}_i} \left(\max_{\nu \in F} \left| \rho\left(\Psi(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \nu) \right) \right| \right) \tag{3.72}$$

to determine the best parameters in order to improve the convergence of the iterative method. We can note from Equations 3.71 and 3.72 that the first eigenvalue λ_1 only has parameters that act on the TM component (α_1 and α_3) and the second eigenvalue λ_2 components that act on the TE component (α_2 and α_4). This result means that we can optimize the parameters separately for TE and TM modes using the same optimization process we developed for Helmholtz Equation.

In the next section, we perform numerical experiments to confirm that the optimized second-order transmission condition optimized using the multi-domain formulation outperforms zeroth-order conditions in terms of convergence rates at the same cost per iteration and also outperforms the same order transmission condition using the standard optimization process.

3.3.1 Numerical Experiments

3.3.1.1 First example - *CubeSpheres* Target

The first example is taken from (GEFFRIN; SABOUROUX, 2009). We compute the scattering of a plane wave oscillating at $8GHz$ by a set of dielectric spheres. At this frequency, the wavelength is approximately $37.5mm$. Each sphere has a diameter of $15.9mm$ and a permittivity of $\epsilon_r = 2.6$. They are assembled to obtain a cube measuring $47.6mm$ on each side, see Figure 26. We truncate the domain $37.5mm$ apart from the target. We set the mesh size to $2.5mm$, which leads to approximately 750.000 tetrahedrons.

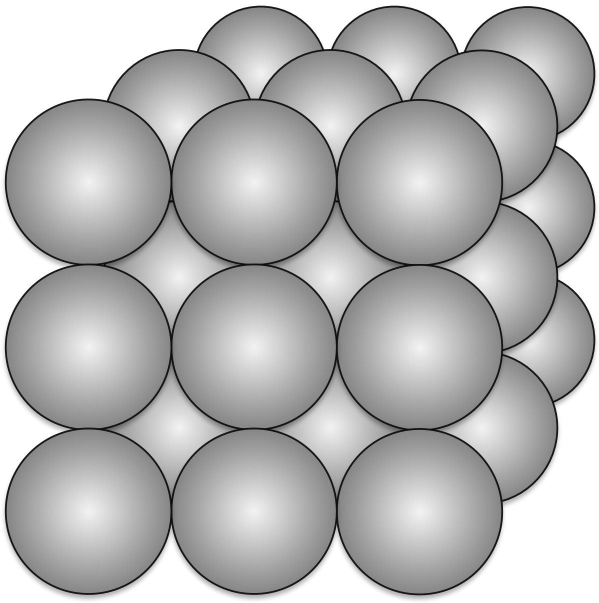


Figure 26 – Diagram of *CubeSpheres* target from (GEFFRIN; SABOUROUX, 2009)

We solve the problem on an Intel i7-10850H machine with 32 GB of RAM and 12 processing units. Using first-order Nédélec elements (edge elements) leads to a matrix that is already intractable with direct methods. Using iterative methods without preconditioning is no improvement; GMRES does not converge after 1000 iterations. The convergence history for the GMRES preconditioned with the ORAS using 12 MPI ranks (12 sub-domains) is shown in Figure 27. The second-order transmission condition MO2 performs much better than the zeroth order O0, leading to an iterative method three times faster. It is worth noting that while the second-order transmission condition has been optimized for this configuration, the zeroth-order has been used in its standard form (3.48). Also, we see that it performs better than the standard second-order formulation not optimized for multiple domains (O2) since it does not take the number of subdomains and the overlap into account.

3.3.1.2 Second Example - Microwave Field in a Loaded Cavity

The second example is taken from (BOSSAVIT; LAMAUDIÈRE; MAESTRALI, 1992; EHMANN et al., 1996). The simulated structure, Figure 28, consists of a PEC cylindrical cavity, which is closed except for a rectangular iris. The iris height is $43.18mm$, and its width can be adjusted. It connects the cavity to a standard rectangular $86.36mm \times 43.18mm$ waveguide. The cavity is loaded with a vertical rod of Plexiglas ($\epsilon_r = 2.7 - j0.1$) whose diameter is $7mm$. The waveguide is excited with the TE_{10} mode at 2.55 GHz. In Figure 29 we show the solution obtained using second-order Nédélec elements of the first kind.

In this experiment, we exclusively use second-order transmission conditions optimized with the multi-domain approach. We decompose the computational domain into P subdomains using ParMETIS (KARYPIS, 2011), and we show the obtained partitions in Figure 30. This experiment aims to demonstrate the impact of the number of sub-domains on the convergence of

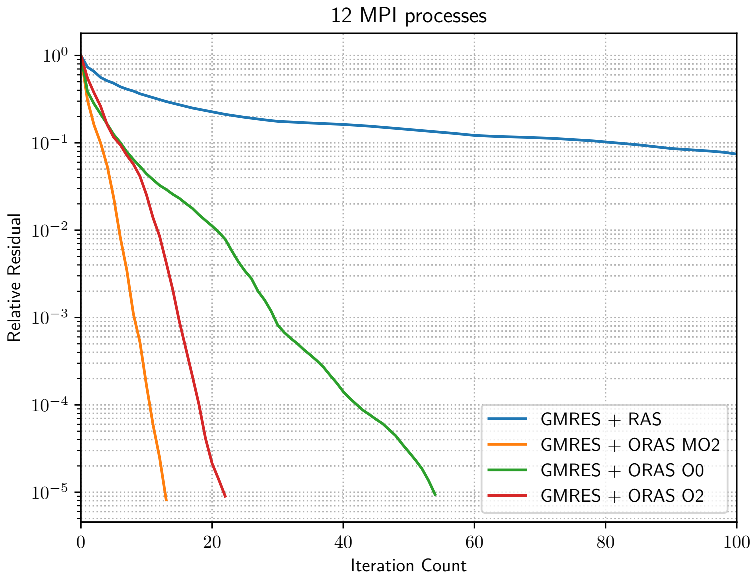


Figure 27 – Relative residual of GMRES using different transmission conditions.

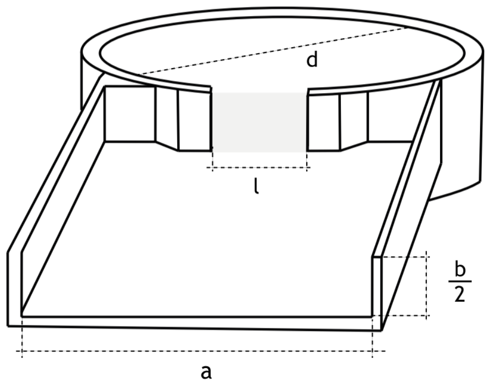
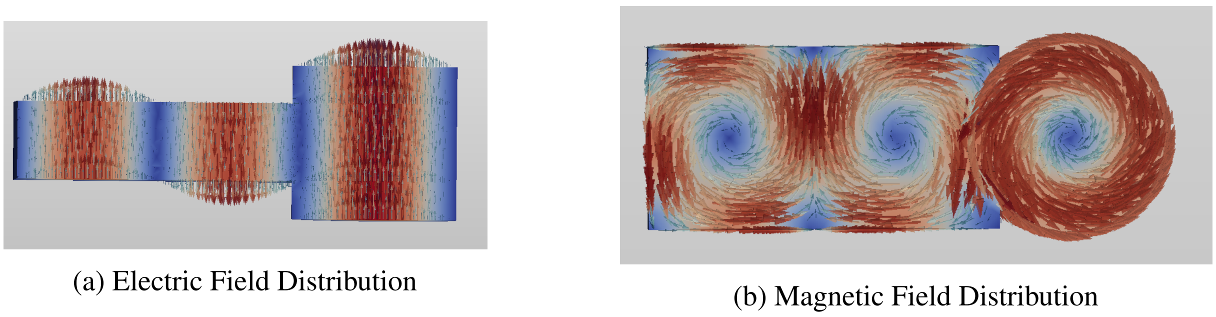


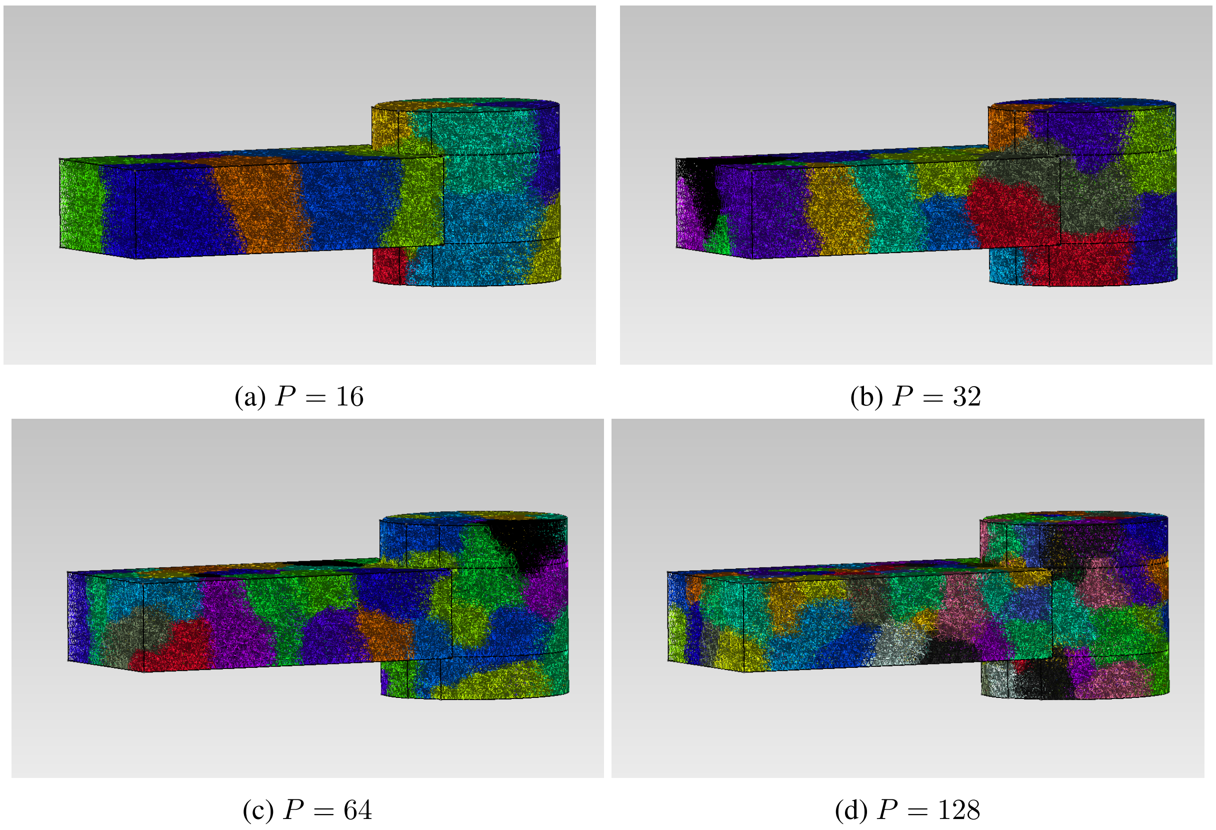
Figure 28 – Team Workshop 19 benchmark - Transversal cut showing the structure's lower half.

the preconditioned GMRES. For each domain decomposition, in Figure 31, we show the relative residual, and we notice the emergence of plateaus convergence history. We also remark that the plateau has a size proportional to the depth of the graph that connects all subdomains.


(a) Electric Field Distribution

(b) Magnetic Field Distribution

Figure 29 – Solution of microwave fields in a loaded cavity.


(a) $P = 16$

(b) $P = 32$

(c) $P = 64$

(d) $P = 128$

Figure 30 – Decomposition of the computational domain into P color-coded subdomains.

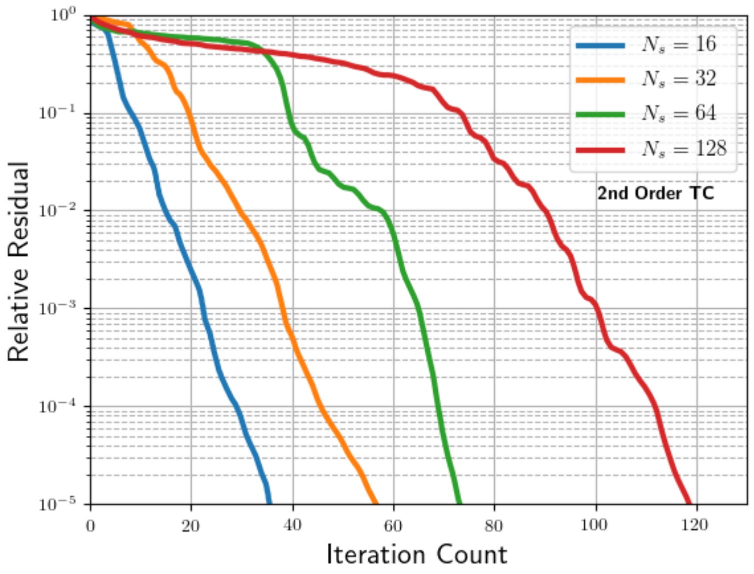


Figure 31 – Relative residual of GMRES using

3.4 Conclusions

For one-level domain decomposition preconditioners, based solely on local solves within subdomains, the iteration count increases with the number of subdomains (processes), and plateaus appear in the convergence history plots. However, up to a few hundred subdomains, one-level preconditioners remain the best alternative. They avoid the construction of a second-level and have less communication overhead. Therefore, improving the convergence of such methods is still of paramount importance.

In this chapter, we propose some improvements for one-level methods, given special attention to transmission conditions. We propose an optimization process for devising transmission conditions that consider the propagative modes dynamically, which are the root cause of weak convergence in one-level methods. For time-harmonic scalar waves, governed by the Helmholtz equation, we describe in detail this optimization process for devising transmission conditions. Numerical experiments show that the Schwarz preconditioner with our optimized transmission conditions leads systematically to faster convergence rates than using transmission conditions of the same order using the standard optimization procedure.

We show that we can optimize the transmission conditions for TE and TM modes separately for time-harmonic Maxwell's equations. Furthermore, for each electromagnetic mode, the optimization process reduces to the optimization of a scalar transmission condition, for which we can use the procedure developed for the Helmholtz equation. We conducted numerical experiments that confirm that, in fact, for a few hundred subdomains, the proposed optimized transmission conditions lead to iterative methods that converge better than the standard optimization process.

4 Two Level Methods For Scalar Valued Problems

Contents

4.1 Introduction

As remarked in the last section, one-level domain decomposition methods possess a severe drawback: even with optimal local transmission conditions, the convergence rates depend on the longest path of the graph defined by the connectivity between subdomains, preventing such methods from achieving good parallel scaling (VION; GEUZAINE, 2014; DOLEAN; JOLIVET; NATAF, 2015). This drawback is due to a lack of global communication between subdomains since they exchange information with their adjacent neighbors at each iteration. A promising solution consists of adding a second level that allows global communication, the so-called coarse space correction. One can then additively enrich the one-level preconditioner M^{-1} by

$$\mathcal{P}_{AD}^{-1} = Q + M^{-1}. \tag{4.1}$$

where $Q = ZA_c^{-1}Z^T$ is the coarse space problem, and Z is a full column rank matrix, also known as the interpolation operator or the deflation subspace matrix. In many methods, the coarse matrix A_c and the Galerkin operator $E = Z^TAZ$ are used interchangeably. There are several approaches to define an appropriate coarse operator Z for a particular situation. A simple option is to discretize the same problem on a coarse mesh. Z is the interpolation matrix from the coarse finite element space to the original, fine finite element function space.

Ultimately, once this operator is formed, there is more than one way to enrich one-level preconditioners. In addition to additive enrichment, other formulations could be considered, such as

$$\mathcal{P}_{A-DEF}^{-1} = M^1(I - AQ) + Q, \tag{4.2}$$

$$\mathcal{P}_{BNN}^{-1} = (I - QA)M^{-1}(I - AQ) + Q. \tag{4.3}$$

which have more desirable spectral properties for SPD problems (JOLIVET et al., 2012). In (TANG et al., 2009), the authors compared many algebraic formulae in terms of robustness and spectral properties. Nevertheless, whether they are amenable to the indefinite problems is an open question, and in this project, we restrict our efforts to additive enrichment.

For symmetric positive definite (SPD) problems, the convergence rates of the two-level methods have been extensively examined in (VUIK; NABBEN; TANG, 2005; TANG et al., 2009). However, for time-harmonic electromagnetic problems, applying standard coarse space approaches directly, such as coarse mesh coarse space correction, may lead to convergence deterioration.

In the following sections, we study some coarse spaces for the time-harmonic scalar wave equation and evaluate which would be more suitable for Maxwell's equation. In addition to convergence properties, one of the factors we consider is the scalability of the construction of the operator in parallel and its computational cost. The last two points seem to be overlooked in the preconditioner literature.

4.2 Two Level for SPD problems

For SPD problems, the convergence rates of the two-level methods have been extensively examined in (VUIK; NABBEN; TANG, 2005; TANG et al., 2009). Notably, the authors showed that the spectrum of the preconditioned system never deteriorates for any choice of the coarse (full column rank) matrix Z, meaning that the smallest non-zero eigenvalue does not decrease, nor the most significant eigenvalue increase. However, for indefinite problems, there are no guarantees that using a two-level method with an arbitrarily chosen coarse matrix Z accelerates convergence (QU; FISH, 2000; FISH; QU, 2000).

For homogeneous Laplace-type problems, such as electrostatic problems, an adequate coarse space is the Nicolaides method (NICOLAIDES, 1987). The constant function 1 belongs to the vector space spanned by its columns, and then the matrix Z is defined by vectors with local support in the subdomains (DOLEAN; JOLIVET; NATAF, 2015). The global structure of Z in its simplest form is thus

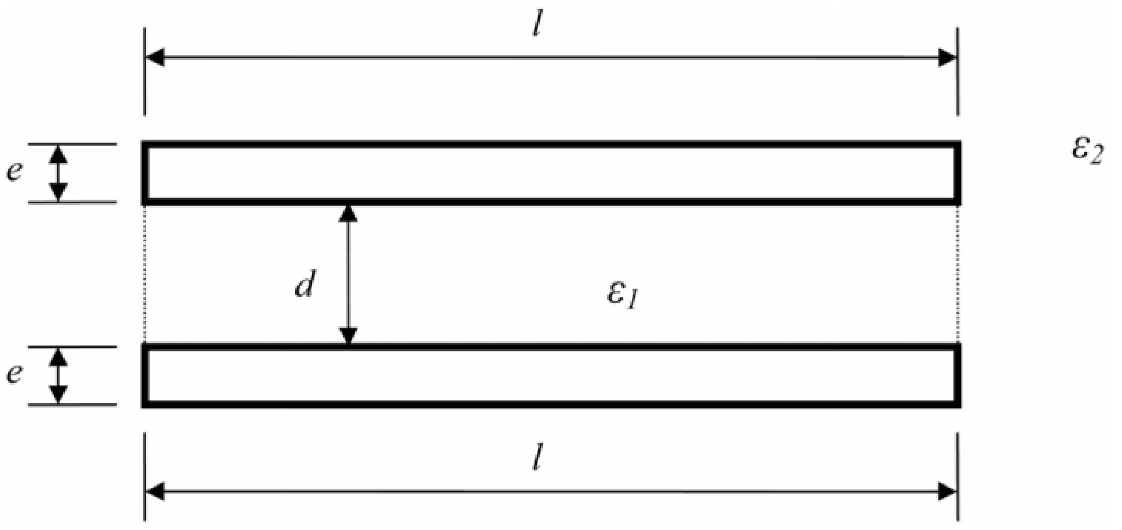


Figure 32 – Schematic of parallel plate capacitor problem. Example of SPD problem for which standard two-level methods work.

$$Z = \begin{bmatrix} 1_{\Omega_0} & 0 & \cdots & 0 \\ \vdots & 1_{\Omega_1} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1_{\Omega_{P-1}} \end{bmatrix} = \begin{bmatrix} D_0R_0\mathbf{1} & 0 & \cdots & 0 \\ \vdots & D_1R_1\mathbf{1} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & D_{P-1}R_{P-1}\mathbf{1} \end{bmatrix}. \tag{4.4}$$

The Nicolaides Galerkin $E = Z^TAZ$ operator has size $P \times P$, where P is the number of subdomains or, similarly in this context, the number of MPI processes. If correctly distributed, the extra cost of solving this global system is negligible, provided that the number of subdomains is not too large (which is valid for any pre-exascale machine). In our experiments solving a parallel plate capacitor problem, described in Figure 32, with $\epsilon_1 = \epsilon_2$, the addition of the Nicolaides coarse space removed the dependence on the number of subdomains, and the coarse problem can be easily solved in serial (no big overheads incurred in the total computational time). However, in the presence of heterogeneities, $\epsilon_1 \neq \epsilon_2$, the method stagnates. In this experiment, the number of subdomains is increased from 16 to 128.

Coarse mesh coarse space corrections have been proposed to problems with heterogeneities. These methods give good results when coefficient discontinuities are aligned with sub-domain interfaces (DRYJA; SARKIS; WIDLUND, 1996; DOHRMANN; WIDLUND, 2009), or remain inside the sub-domain and not near the boundaries (PECHSTEIN; SCHEICHL, 2009). However, when the discontinuities are along with sub-domain interfaces, such methods no longer work as expected. Avoiding such discontinuities along the interfaces may be possible for many problems, but it would restrict the use of automatic graph partitioners and the solution of highly heterogeneous problems.

A two-level domain decomposition that is robust to heterogeneous coefficients for arbitrary decompositions was proposed in (NATAF; XIANG; DOLEAN, 2010) and extensively tested numerically in (NATAF et al., 2011). The main idea behind the construction of the coarse space is the computation of the low-frequency modes associated with a generalized eigenvalue problem based on the Dirichlet-to-Neumann (DtN) map on the boundary of each sub-domain. Then the harmonic extensions of these low-frequency modes to the whole sub-domain are used to build the coarse space basis. The interpolation operator Z is very similar to the one proposed by Nicolaides and have the following block form

$$Z = \begin{bmatrix} W^1 & 0 & \cdots & 0 \\ \vdots & W^2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & W^P \end{bmatrix} \tag{4.5}$$

where W^i is a rectangular matrix whose columns are the harmonic extensions of the eigenvectors corresponding to small eigenvalues of the DtN map problem in each sub-domain. To construct the coarse space correction, the low-frequency modes of the DtN operator are obtained by solving the DtN eigenvalue problem:

$$\mathrm{DtN}_{\Omega_i}(u) = \lambda u. \tag{4.6}$$

This option is justified by the fact that a rapid drop in the error of additive Schwarz techniques correlates to big eigenvalues of the DtN map, whereas a slow decay corresponds to small eigenvalues. As a result, the small eigenvalues of the DtN map are responsible for the algorithm's convergence, and it is logical to include them in the coarse space correction.

This method has proved to scale very well with the number of subdomains (processes in an MPI setting). Numerical experiments using up to 8192 processes and 300 million unknowns have been performed, and the parallel efficiency remained around 90% (JOLIVET, 2014). An extension of this method to the Helmholtz equation was recently proposed in (CONEN et al., 2014), and will be further assessed in the next section.

4.3 Two Level for the Helmholtz Equation

The development of two-level domain decomposition preconditioners for sign-indefinite problems is a challenging open problem. For example, applying standard coarse-grid coarse spaces directly to Helmholtz's or Maxwell's equation may lead to convergence deterioration compared to one-level methods. Nevertheless, some very promising two-level variations have been proposed recently. We have identified four basic classes of two-level preconditioners, each one based on different construction principles. In this section, we review these classes of preconditioners, identify some open issues and possible research directions.

4.3.1 Sweeping type preconditioners

This class of two-level methods is based on a layered decomposition of the domain, inducing ordered sub-domain solutions (VION; GEUZAINE, 2014; STOLK, 2013; STOLK, 2017; GANDER; ZHANG, 2019; ENGQUIST; YING, 2011). The information propagates over the entire domain in one preconditioner application, and the number of iterations is strongly affected by the quality of the approximation of the DtN operator on the interface. High order transmission conditions localized with auxiliary unknowns and perfectly matched layers (PMLs) are frequently used. This method leads to nilpotent iterations in its optimal form, like an exact block LU factorization. It differs significantly from the others studied methods concerning the second level construction; the coarse matrix Z is never created, and global communication is performed through the sequential sweeping of subdomains solutions. The main drawback of these methods is the lack of parallelism and inflexibility in domain partitioning. As examined in depth in (GANDER; ZHANG, 2019), heterogeneities along and across boundaries can cause the method to stagnate.

4.3.2 DtN Coarse Space preconditioners

DtN Coarse Space preconditioners tailored for the Helmholtz equation were first proposed in (CONEN et al., 2014), adapting the successful idea from scalar elliptic problems (NATAF; XIANG; DOLEAN, 2010; NATAF et al., 2011). The coarse space is built by solving local eigenproblems involving the DtN operator on the subdomains interfaces. In this method, the interpolation matrix Z and the Galerkin operator E may be explicitly formed. On each interface $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$, one solves the following local DtN eigenproblem: find the eigenvalues λ and the eigenfunctions u_{Γ_i} such that

$$\mathrm{DtN}_{\Omega_i}\left(u_{\Gamma_i}\right) = \lambda u_{\Gamma_i}, \tag{4.7}$$

where the operator DtN_{Ω_i} is defined

$$\mathrm{DtN}_{\Omega_i}\left(u_{\Gamma_i}\right) = \left.\frac{\partial u}{\partial n}\right|_{\Gamma_i}. \tag{4.8}$$

One then extends a selected set of eigenvectors $\{v_{\Gamma_i}\}$ to the interior of sub-domain Ω_i by using the Helmholtz extension:

$$\left.\begin{array}{rcc} \Delta u_i + k^2 u_i = 0 & \text{in} & \Omega_i \\ \mathcal{C}(u_i) = 0 & \text{on} & \partial\Omega_i \cap \partial\Omega \\ u_i = u_{\Gamma_i} & \text{on} & \Gamma_i \end{array}\right\}. \tag{4.9}$$

An heuristic to select the m_i most appropriate eigenvectors was proposed in (CONEN et al., 2014). One selects, to constitute the coarse space, all eigenfunctions for which the associated eigenvalue λ satisfies

$$\mathrm{Re}(\lambda) < \max_{\boldsymbol{x}\in\Omega_i} k(\boldsymbol{x}). \tag{4.10}$$

This heuristic already considers local changes in the wavenumber and hence the method is naturally suited for heterogeneous problems. Now, the matrix Z of the DtN coarse space is a rectangular, block-diagonal matrix with blocks W_i, associated with the subdomain $\Omega_i, 0 \leq i \leq P-1$, as the one presented in (4.5). If m_i eigenvectors are selected and the sub-domain Ω_i has N_i dofs the block W_i matrix has dimensions $N_i \times m_i$, and the matrix Z has dimensions $N \times \sum_{j=0}^{P-1} m_i$.

In practice, instead of looking for the pair $\left(u_{\Gamma_j}, \lambda\right)$ and then computing the extension u_i of u_{Γ_i} from the interface Γ_i to the interior of the sub-domain Ω_i, it possible to directly compute the pair (u_i, λ) by solving the following eigenvalue problem: Find $(u_i, \lambda) \in (V_i, \mathbb{C})$ such that

$$\int_{\Omega_i} \nabla u_i \nabla v_i \mathrm{d}x - \int_{\Omega_i} k^2 u_i v_i \mathrm{d}x = \lambda \int_{\Gamma_i} u_i v_i \mathrm{d}s \quad \text{for all } v_i \in H^1\left(\Omega_i\right). \tag{4.11}$$

Since the construction of the interpolation operator is based only on local eigenvalue problems, it is possible to construct the coarse space in parallel; however, how it should be distributed is not clear in the literature. Furthermore, the sparsity of the Galerkin operator $E = Z^T A Z$ stems from the sparsity of Z. However, for high frequencies, the selection criterion (4.10), may lead to large coarse spaces, which are expensive to solve with direct methods.

Finally, for large subdomains, the solution of the eigenproblem (4.11) may also be expensive. Most of the algorithms for eigenvalue computations scale with (n^3) for dense matrices, where n is the number of rows of the symmetric matrix. The situation is not much better for sparse matrices, where iterative methods for solving the problem are used. For example, `scipy.eigs` provides a high-performance serial function to k eigenvalues and eigenvectors of the square matrix A. The function is a wrapper for double-precision floats to the ARPACK DNEUPD function, which uses the "Implicitly Restarted Arnoldi Method" to find the eigenvalues and eigenvectors. Computing the 100 first eigenpairs for 4.11 with 1M local *dofs* takes several hours on a consumer desktop. The ARPACK DNEUPD is very efficient for computing large eigenvalues (high frequencies) but less efficient for small ones (low-frequency modes). In this case, the cost of constructing the coarse space can easily dominate the cost of solving the original problem. Although it may lead to a computationally expensive construction, computational experiments showed that the method is robust to the number of subdomains and the presence of heterogeneities (CONEN, 2015; BONAZZOLI et al., 2017a).

4.3.3 Shifted-Laplacian coarse space preconditioners

Shifted Laplacian techniques have a successful track record on multigrid methods (ERLANGGA; VUIK; OOSTERLEE, 2004). However, the construction of coarse spaces for domain decompositions using this principle only appeared recently in (GRAHAM; SPENCE; VAINIKKO, 2017a; KIMN; SARKIS, 2013). The coarse space problem A_c is constructed by

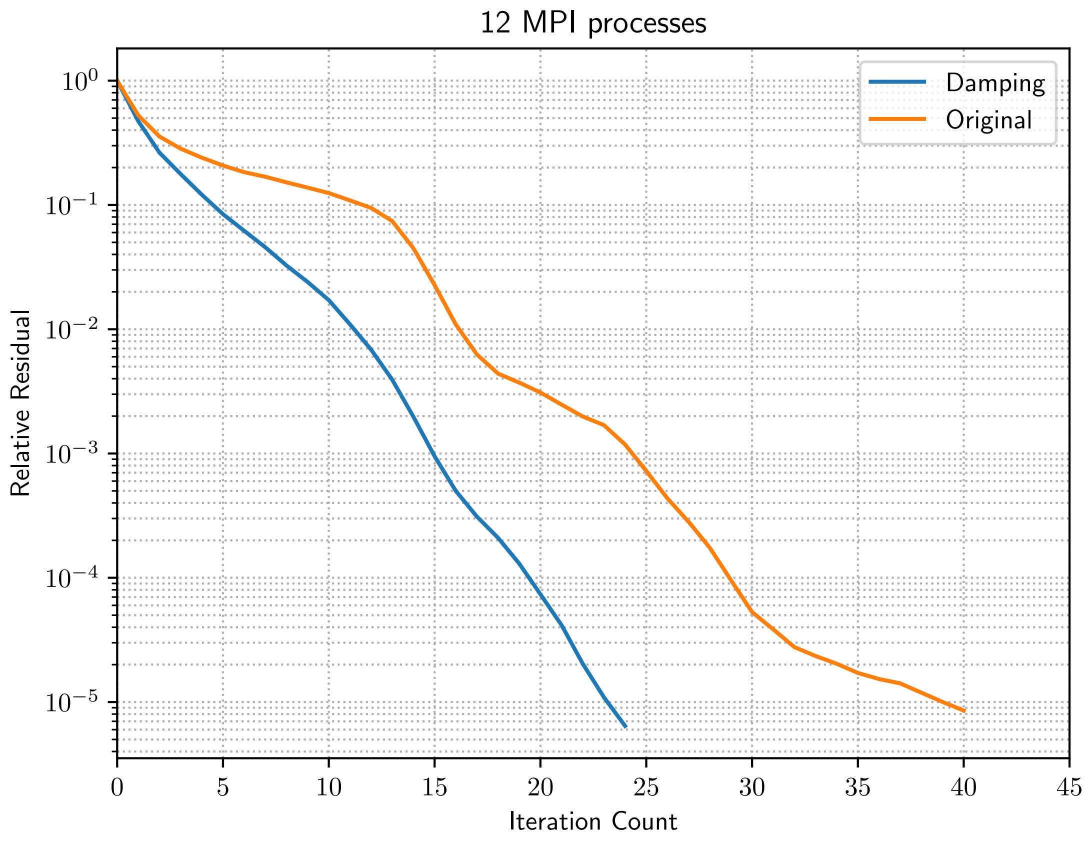


Figure 33 – The relative residual of GMRES using ORAS preconditioner for solving the original problem and the problem with damping.

discretizing:

$$\left.\begin{array}{rcc} \Delta u + (k^2 + j\beta)u = f & \text{in} & \Omega \\ \mathcal{C}(u) = 0 & \text{on} & \partial\Omega \end{array}\right\} \tag{4.12}$$

where $\beta \in \mathbb{R}$ is the absorption parameter. When damping is introduced, the decay of the Green's function is faster, and iterative methods more easily solve the problems. See for example Figure 33, where we can see that the preconditioned GMRES solves the problem with damping far more quickly than the original problem. Also, the problem in (4.12) can be efficiently discretized with a coarser mesh $\mathcal{T}_H$, with mesh size $H > h$ greater than the original problem. We can then define the corresponding finite element coarse space $V_H \subset V$. If we let $\mathcal{I}_0 : V_H \to V_h$ be the interpolation operator, the coarse matrix Z can be defined as the corresponding interpolation matrix.

Employing Fourier analysis, the authors of (COCQUET; GANDER, 2017) pointed out that one needs $|\beta| \approx k$ to obtain convergence independent of k for the multigrid method. This result was extended to coarse spaces for additive Schwarz methods in (GRAHAM; SPENCE; VAINIKKO, 2017b; GRAHAM; SPENCE; VAINIKKO, 2017a).

Extensive numerical experiments were carried out using this class of preconditioners in (GRAHAM; SPENCE; VAINIKKO, 2017b; GRAHAM; SPENCE; VAINIKKO, 2017a). In particular, it was compared with DtN coarse space preconditioners in (BONAZZOLI et al., 2017a). The DtN-based method generally gives fewer iterations than the shifted-Laplacian one, but the cost for its construction is less expensive, which is not considered in the articles. Although the method seems highly parallelizable, all experiments were performed on sequential codes, so execution time and parallel performance were not assessed.

4.3.4 Plane Wave Coarse Space

As well as Shifted-Laplacian preconditioners, Plane Wave methods were applied initially in the multigrid context (BRANDT; LIVSHITS, 1997) and only later to domain decomposition methods (FARHAT et al., 2005; FARHAT; MACEDO; LESOINNE, 2000; KIMN; SARKIS, 2007). For each sub-domain, a finite number of plane waves is chosen, and they are evaluated within the interior of subdomain (KIMN; SARKIS, 2007) or on a subset associated to boundary degrees of freedom (LEONG, 2008). Finally, the resulting local expansion vectors after extension to the interior of the subdomains are used to create the block matrix W^i (LEONG, 2008).

Plane-wave coarse spaces have many similarities with DtN Coarse Spaces. Instead of extending a selected set of eigenvectors to the interior of sub-domain Ω_i, we extend a set of plane waves. A plane wave is a function of the form:

$$p(\boldsymbol{x}) = p_0 e^{jk(\mathbf{d}\cdot\boldsymbol{x})}. \tag{4.13}$$

where $\mathbf{d}$ is the unitary direction vector.

There are infinitely many possible directions $\mathbf{d}$ in a plane (2d problems) or volume (3d problems). However, one needs a finite subset of linearly independent plane waves to enrich the coarse space properly. In two dimensions, see Figure 34, one generally uses a uniform discretization of the unit circle to get m plane wave directions $\mathbf{d}_j$

$$\mathbf{d}_j = \begin{pmatrix} \cos(\theta_j) \\ \sin(\theta_j) \end{pmatrix} \tag{4.14}$$

where

$$\theta_j = \frac{2\pi(j-1)}{m} \quad \text{for} \quad 1 \leq j \leq m. \tag{4.15}$$

The selection of plane waves in three dimensions is tricky, and uniform discretization of the unit sphere may lead to a huge number of directions. Most of the works present in the literature use the algorithm proposed in (TEZAUR; FARHAT, 2006). For $n_t \in \mathbb{N}$ construct the vectors

$$\mathbf{y}_{j,k,l} = \begin{bmatrix} \tan\left(\left(2\frac{j}{n_t} - 1\right)\frac{\pi}{4}\right) \\ \tan\left(\left(2\frac{k}{n_t} - 1\right)\frac{\pi}{4}\right) \\ \tan\left(\left(2\frac{l}{n_t} - 1\right)\frac{\pi}{4}\right) \end{bmatrix} \quad j,k,l = 0,\ldots,n_t. \tag{4.16}$$

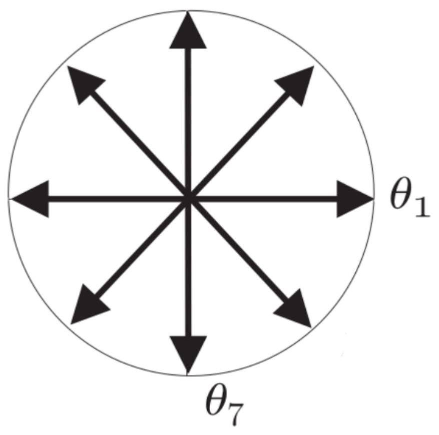


Figure 34 – Uniform discretization of the unit circle using eight directions.

Then, from the $(n_t + 1)^3$ vectors $\mathbf{y}_{j,kl}$, select $6n_t^2 + 2$ vectors that correspond to triplets $[j, k, l]$ such that at least one of the indexes j, k, l is equal to 0 or n_t. After the set of m_i plane waves per subdomain have been selected, they are extended to the interior of the sub-domain using the Helmholtz extension (4.9). From there, the plane wave construction algorithm follows the same steps as the DtN algorithm.

The main difference to the DtN algorithm is that the column space (or the image) of Z is now composed of plane waves instead of eigenvectors of the problem (4.11). An attractive feature of the DtN coarse space is that eigenvectors from distinct eigenvalues are linearly independent, and hence Z has a full column rank. However, the deflation subspace operator Z of the plane wave coarse space may be rank deficient. On the one hand, the number of plane waves per subdomain m_i has to be sufficiently large for the second level to be beneficial. On the other hand, if m_i is too large, some directions may be close to each other, making the corresponding plane waves almost linearly dependent. Thus, the rank deficiency of Z may cause the divergence of the whole iterative scheme.

A fundamental issue of this class of preconditioners is then how to select the m_i plane waves that should enter the coarse space. In (FARHAT et al., 2005), the authors proposed to use a filter based on a QR factorization of the blocks W_i. However, many numerical experiments have revealed that this filtering can remove essential modes, impairing the convergence of the method. A comparison of the plane wave and DtN coarse spaces can be found (CONEN, 2015). In the experiments, the authors noted that the convergence rates of both algorithms are very similar for most problems, but the construction of the DtN coarse space is much more expensive due to the need to solve the local eigenvalue problems. Also, the DtN coarse space converged reliably in all the experiments, while the plane wave method could not ensure convergence when the QR filter is used.

4.4 Improving the plane wave coarse space

The most evident weakness of the plane wave-based coarse space is the problematic construction of the interpolation operator Z, which may lead to stagnation in the iterative process. Some authors have proposed filtering to select which plane waves should enter the coarse space and generate a Z matrix with full column rank (FARHAT et al., 2005; FARHAT; MACEDO; LESOINNE, 2000; KIMN; SARKIS, 2007). However, a full column rank matrix is necessary but not sufficient conditioning for devising a convergent method, as shown in the numerical experiments presented in (CONEN, 2015). We, therefore, propose a new technique of enriching the coarse spaces with knowledge of the dominant plane waves. In the previous section, we gave an overview of the method, and now we present a pseudo-code describing it in more detail, see Algorithm 2. From line 3 onwards, the algorithm shares the steps with the DtN algorithm, as presented in (NATAF; XIANG; DOLEAN, 2010; CONEN et al., 2014; CONEN et al., 2015).

Algorithm 2 Plane Wave Coarse Space Construction Algorithm

```
1: for $i = 1$ to $P$ do
2:     Select $m_i$ dominant plane waves $\mathbf{p}^i = [p^i_1, p^i_2 \cdots p^i_{m_i}]$ at $\Gamma_i$
3:     for $j = 1$ to $m_i$ do
4:         Compute the extension $u^i_j$ of $p^i_j$ from $\Gamma_i$ to the sub-domain interior $\Omega_i$
5:     end for
6:     Define the block Matrix $W^i = \left[D_i u^i_1, D_i u^i_2, \cdots D_i u^i_{m_i}\right]$
7: end for
8: Define the Matrix $Z = \left[R^T_1 W_1, R^T_2 W_2, \cdots, R^T_N W_N\right]$
```

The critical point of the proposed improvement is selecting the dominant plane waves to enrich the coarse space, line 2 of the algorithm. We assume that a wave field is a weighted superposition of plane waves with very minor changes in direction with frequency. Then, using the second-order numerical micro-local analysis (NMLA) technique, we solve a lower-frequency problem to extract the directions and weights (BENAMOU; COLLINO; RUNBORG, 2004; BENAMOU; COLLINO; MARMORAT, 2011). The dominating plane waves are then carefully chosen to enrich the interfaces of each subdomain. Dominant plane waves in this context mean that they contain more energy and hence more information about the actual solution of the problem.

We assume, thereof, that the solutions of the Helmholtz equations take the form of the geometrical optics ansatz

$$u(\boldsymbol{x}) \approx \text{ superposition of } \left\{A_n(\boldsymbol{x})e^{jk\phi_n(\boldsymbol{x})}\right\}_{n=0}^{N-1} \tag{4.17}$$

where the amplitude $A_n(\boldsymbol{x})$ and phase $\phi_n(\boldsymbol{x})$ are dependent on the medium, boundaries and sources but independent of the frequency. As long as the medium is smooth, the asymptotic expansion (4.17) is a good approximation of the exact solution of the Helmholtz provided that

N is sufficiently large (ENGQUIST; RUNBORG, 2003). The amplitude and phase satisfy the eikonal/transport equations, but calculating them directly would require extensive computational resources (BENAMOU, 2003). Instead, we use the NMLA signal processing algorithm to estimate which plane waves contribute the most to $u(\boldsymbol{x})$.

Based on the above geometric optics ansatz, one can derive a local plane wave approximation at any point using Taylor expansions around an observation point $\boldsymbol{x}_0$ by

$$u(\boldsymbol{x}) = \left(A_n\left(\boldsymbol{x}_0\right) + \nabla A_n\left(\boldsymbol{x}_0\right)\left(\boldsymbol{x}-\boldsymbol{x}_0\right)\right) e^{i\omega\left(\phi_n(\boldsymbol{x}_0)+\nabla\phi(\boldsymbol{x}_0)\cdot(\boldsymbol{x}-\boldsymbol{x}_0)\right)} + \mathcal{O}\left(h^2+\omega h^2+\frac{1}{\omega}\right) \tag{4.18}$$

for $|\boldsymbol{x}-\boldsymbol{x}_0| < h \ll 1$.

Now let

$$\mathbf{d}_n := \frac{\nabla\phi_n\left(\boldsymbol{x}_0\right)}{\left|\nabla\phi_n\left(\boldsymbol{x}_0\right)\right|} = c\left(\boldsymbol{x}_0\right)\nabla\phi_n\left(\boldsymbol{x}_0\right) \tag{4.19}$$

be the wave direction of the n_{th} wave front passing through the point $\boldsymbol{x}_0, k\left(\boldsymbol{x}_0\right) = \omega/c\left(\boldsymbol{x}_0\right),$ and

$$B_n(\boldsymbol{x}) = \left(A_n\left(\boldsymbol{x}_0\right) + \nabla A_n\left(\boldsymbol{x}_0\right)\left(\boldsymbol{x}-\boldsymbol{x}_0\right)\right) e^{i\omega\left(\phi_n(\boldsymbol{x}_0)-\nabla\phi(\boldsymbol{x}_0)\cdot\boldsymbol{x}_0\right)} \tag{4.20}$$

the respective complex amplitude. Then, $u(\boldsymbol{x})$ can be approximated locally by a superposition of plane waves known at its vicinity:

$$u(\boldsymbol{x}) = \sum_{n=1}^{N} B_n e^{jk(\boldsymbol{x}-\boldsymbol{x}_0)\cdot\mathbf{d}_n}, \quad |\mathbf{d}_n| = 1 \tag{4.21}$$

The aim of NMLA is to extract the directions $\mathbf{d}$ and the complex weights B_n by probing and processing the wave field locally (BENAMOU; COLLINO; MARMORAT, 2011; BENAMOU; COLLINO; RUNBORG, 2004). We suppose that we can sample the wave field, $u(\boldsymbol{x})$, and its derivative on a circle $S_r\left(\boldsymbol{x}_0\right)$ centered at $\boldsymbol{x}_0$ with radius r, such that:

$$u\left(\boldsymbol{x}_0 + r\hat{\mathbf{s}}\right) = \sum_{n=1}^{N} B_n e^{i\alpha\widehat{\mathbf{s}}\cdot\mathbf{d}_n}, \quad \alpha = kr, \widehat{\mathbf{s}} \in \mathbb{S}^1 \tag{4.22}$$

If we define the angle variables $\theta = \theta(\widehat{\mathbf{s}})$ and $\theta_n = \theta\left(\mathbf{d}_n\right)$, $\widehat{s} = (\cos\theta, \sin\theta)$ and $\boldsymbol{x}(\theta) = \boldsymbol{x}_0 + r\widehat{\mathbf{s}}(\theta)$. The sampled impedance quantity on the circle $S_r\left(\boldsymbol{x}_0\right)$ can be written as:

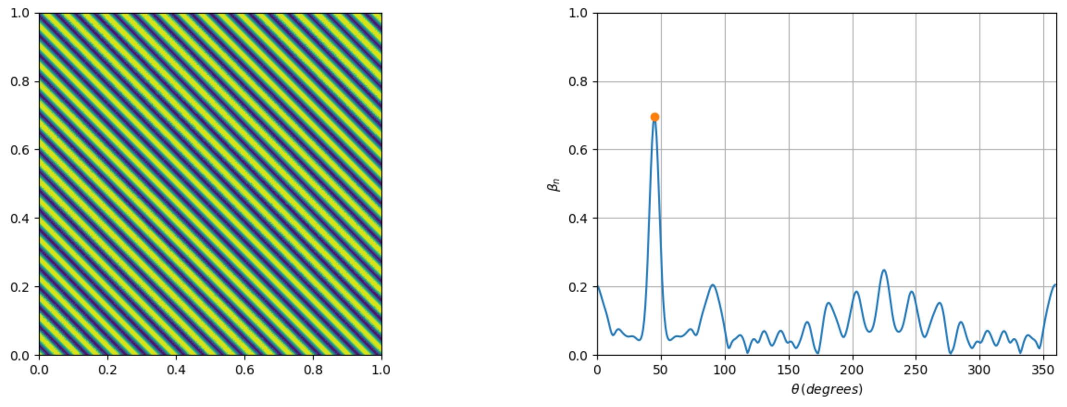
$$U(\theta) := \frac{1}{ik}\partial_r u(\boldsymbol{x}(\theta)) + u(\boldsymbol{x}(\theta)) \tag{4.23}$$

Then we apply the filtering operator $\mathcal{B}$ to the impedance quantity

$$\mathcal{BU}(\theta) := \frac{1}{2L_\alpha+1}\sum_{l=-L_\alpha}^{L_\alpha} \frac{(\mathcal{F}U)_l e^{il\theta}}{(-i)^l\left(J_l(\alpha) - iJ_l'(\alpha)\right)} \tag{4.24}$$

where $L_\alpha = \max\left(1, [\alpha], \left[\alpha + (\alpha)^{\frac{1}{3}} - 2.5\right]\right)$, J_l is the Bessel function of order l, J_l' is its derivative, and $(\mathcal{F}U)_l$ is the l -th Fourier coefficient of U. It is then shown in (BENAMOU; COLLINO; RUNBORG, 2004) that if $\alpha = kr \to \infty$ then

$$\lim_{\alpha\to\infty} \mathcal{BU}(\theta) = \begin{cases} B_n & \text{if } \theta = \theta_n \\ 0, & \text{otherwise.} \end{cases} \tag{4.25}$$



(a) Plane wave - $k = 50$, $\theta = 45^o$ (b) Filtered data $\mathcal{BU}(\theta)$, $\theta_n = 44.94^o$

Figure 35 – Extraction of dominant wave direction for a plane wave.

where the first condition is equivalent to $\widehat{\mathrm{s}} = \mathbf{d}_n$. Then, in practice, it is possible to obtain the dominant directions by selecting the peaks of the filtered data in 4.24. As long as the perturbation is relatively small with respect to the actual field, the estimation error is $\mathcal{O}\left(\frac{1}{kr}\right)$. That is, the estimation is more accurate the bigger the circle's radius is in relation to its wavelength. However, we sample a lower frequency problem in our method, and it is thus difficult to estimate whether the perturbation is small.

4.4.1 Numerical Experiments

First we test our NMLA implementation for a simple problem; we estimate the dominant wave direction for a single plane wave propagating with $\mathbf{d} = [\cos\pi/4, \sin\pi/4]$. In this case, for all points in the domain, the direction of propagation is the same. The real part of the wavefield in a unit square is presented in Figure 35. We also present the filtered data 4.24, and the estimated direction angle. A small relative error was obtained, about 0.3%.

In the second example, we estimate the dominant wave directions for the problem of a point source inside a unit square domain. In this case, instead of probing the solution of the original problem, we probe the fields at a lower frequency, say $\tilde{k} \approx \sqrt{k}$. We refer to Figure 36 for the solution of the actual and low-frequency problems. Fixing $(kh/2p)^p$, where p is the degree of the finite element approximation, the low-frequency problem has 33 times fewer degrees of freedom than the original one, and on a sequential computer using the direct method, its solution is approximately 1000 times faster. We estimate, for each node of the coarse mesh, the dominant wave direction for the low-frequency problem, and we compare it with the directions obtained for the original problem. See Figure 37 for the dominant wave directions arrow glyph plots. Good results have also been obtained for heterogeneous and complex geometry problems, although visualization is not straightforward.

Finally, we apply Algorithm 2 to construct the coarse space using the plane waves

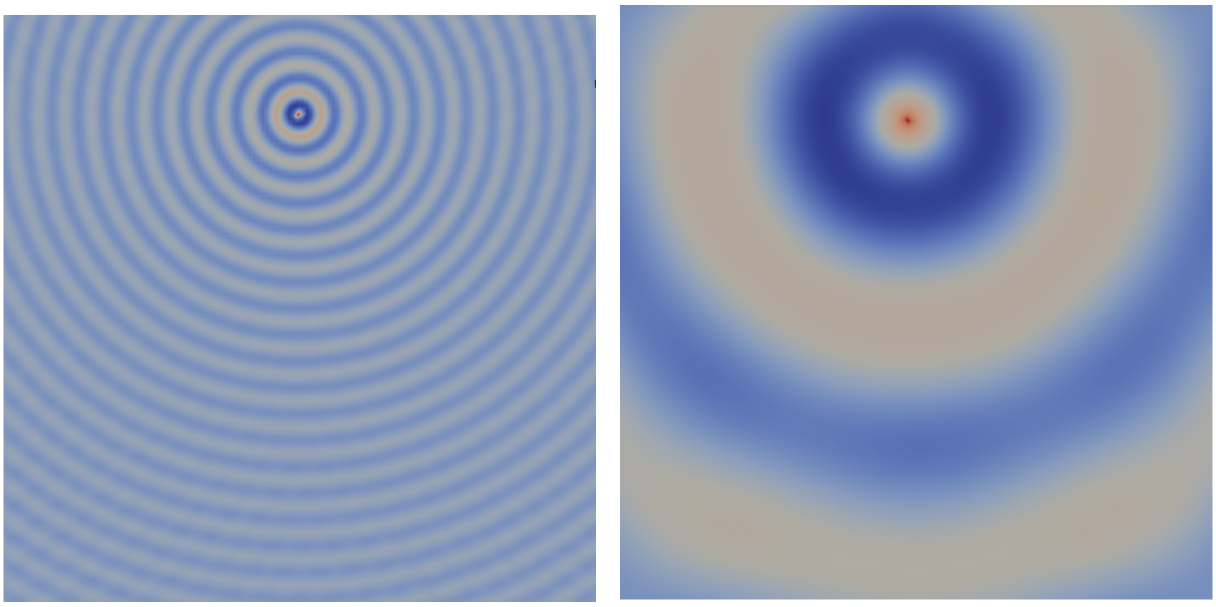

(a) $k = 50\pi$, $\#dofs \approx 1.6M$ (b) $\hat{k} = \sqrt{50\pi}$, $\#dofs \approx 40k$

Figure 36 – Point Source Inside Unit Square Mesh

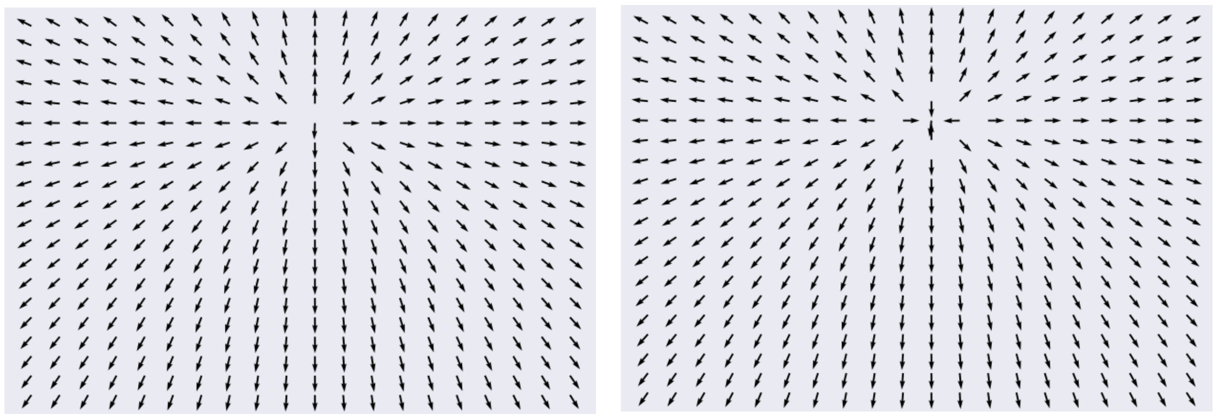

(a) $k = 50\pi$, $\#dofs \approx 1.6M$ (b) $\hat{k} = \sqrt{50\pi}$, $\#dofs \approx 40k$

Figure 37 – Extracted dominant wave directions

obtained from a lower-frequency problem. The process of extracting the dominant waves on the interfaces is better described in Algorithm 3. We solve the same problem as presented in Section ; a plane wave scattering by a circular cylinder. Previously we have used a cyclic decomposition, see Figure 24, but now we use an arbitrary decomposition using an automatic graph partitioner algorithm, see Figure 38. The proposed algorithm provides an automatic means of selecting the plane waves, and the issue of defining a priori the number of plane waves is avoided. The convergence history is presented in Figure 39. Compared with the previous chapter's results, which use just one-level preconditioners, we reduce the iteration count and relax the dependency on the number of subdomains.

Algorithm 3 Plane Wave Direction Extraction

```
1: Solve the Helmholtz equation $(\mathcal{T}_H, \widetilde{k} \approx \sqrt{k}) \to \widetilde{u}$
2: for $i = 1$ to $N_s$ do
3:     for $\mathbf{x}_v$ at the Interface(i) do
4:         Calculate $\mathbf{d}_v^i$ = NMLA($\widetilde{u}$, $\mathbf{x}_v$)
5:     end for
6:     $\mathbf{d}^i$ = Filter Directions
7: end for
```

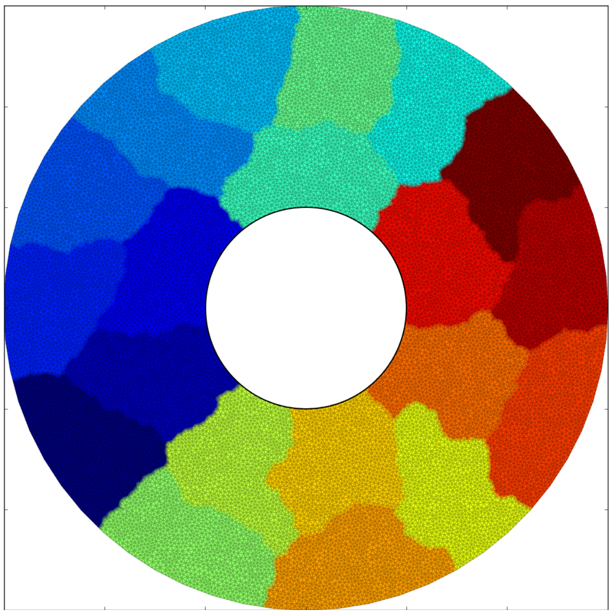


Figure 38 – Arbitrary decomposition of the scattering problem using ParMETIS.

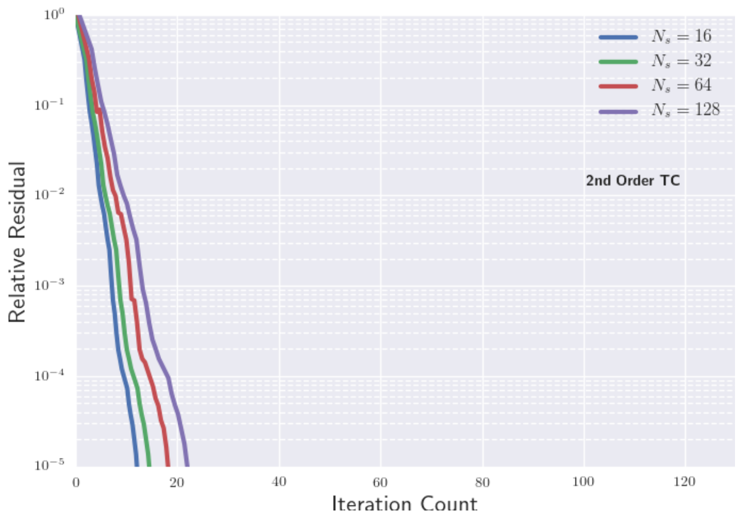


Figure 39 – Relative residual of GMRES preconditioned with the ORAS enriched with the plane wave coarse space.

4.4.2 Conclusions: Issues with the Plane-Wave Coarse Space

For problems where there are a few obvious dominant wave directions, like the plane wave propagation, waveguide, or scattering by simple objects, the proposed method can capture these plane waves, and the construction of the coarse space is straightforward. However, for complex heterogeneous problems, the number of wave-directions grows significantly, and so does the condition number of the coarse matrix $E = Z^{\dagger}AZ$. Also, we observe that the condition number varies with *find_peaks* algorithm used; if we require a large minimal horizontal distance between neighboring peaks, we can reduce the condition number, but we fail to capture the wave directions.

For a small number of subdomains, $P < 256$, and two-dimensional problems, this is not a problem, as the coarse space is small enough to be solved with a direct method in serial. However, for large P and three-dimensional problems, the coarse problem needs to be solved with an iterative solver, and the spectral properties become an issue. If m_i plane-waves are selected and the sub-domain Ω_i has N_i local dofs, the block W_i matrix has dimensions $N_i \times m_i$, and the matrix Z has dimensions $N \times \sum_{j=0}^{P-1} m_i$. Let $M = \sum_{j=0}^{P-1} m_i$, so E has dimension $M \times M$. In three dimensions, the number of plane waves per subdomain required for a converging coarse space can far exceed 1000, so the coarse problem becomes intractable. We argue then that the method can be effective for two-dimensional problems, nevertheless, it is not suitable for three-dimensional problems, and therefore we will not study its extension for Maxwell's equation.

5 Two-Level Preconditioners for Maxwell's Equations

Contents

5.1 Introduction

Although robust two-level methods are available for Maxwell's equation, it is worth mentioning the classical (HIPTMAIR; TOSELLI, 2000) and (TOSELLI, 2000), these works deal only with coercive problems stemming from discretization in the time domain. Besides, the literature on the construction of coarse spaces for time-harmonic Maxwell's equations is relatively scarce.

Sweeping-type preconditioners are perhaps the state-of-art two-level methods for time-harmonic vectorial problems (TSUJI; ENGQUIST; YING, 2012). The algorithms have some appealing characteristics, such as having convergence rates virtually independent of the wavenumber k and the number of subdomains P. However, they suffer from the same drawbacks as their scalar counterparts: they are inherently sequential and are not robust to heterogeneities. Recently partial sweeps (introduction of cuts in the solution sequence) to improve parallel efficiency have been proposed (VION; GEUZAINE, 2018). However, in practice, the parallel efficiency is still very low, and the methods are not suitable for highly heterogeneous problems, such as inverse electromagnetic problems.

Shifted Laplacian coarse spaces have a successful record on multigrid methods (ERLANGGA; VUIK; OOSTERLEE, 2004). However, the construction of coarse spaces for two-level Schwarz domain decomposition methods using this principle only appeared recently in (GRAHAM; SPENCE; VAINIKKO, 2017a) for Helmholtz problems and in (BONAZZOLI et al., 2019) for time-harmonic electromagnetic problems. The coarse space problem is constructed by adding damping to the problem, a complex shift in the wavenumber. When damping is introduced, the problem becomes less wave-like, the Green's function decays fast, so the iterative methods can efficiently solve the problem. Also, the problem with absorption can be discretized with a coarser mesh $\mathcal{T}_H$, with mesh diameter $H > h$ much larger than the original problem. From a careful parameter choice, the theory in (BONAZZOLI et al., 2019) indicates that it is possible to obtain convergence rates for the preconditioned GMRES independent of k.

The construction of Z for curl-conforming elements and coarse mesh coarse spaces is rather complicated since its entries are integral moments (edges, faces, and volumes) that need to be computed with accurate quadrature formulas. Some properties and implementation details of this operator for $\mathcal{H}_{curl}(\Omega)$ spaces can be found in (BOSSAVIT; RAPETTI, 2005). However, efficient implementations require nested meshes and hierarchical basis functions. The main drawback of hierarchical elements is the poor conditioning of the resulting matrix caused by the loss of linear independence as the polynomial order increases (BLUCK, 2012).

In this chapter, we present a scalable two-level preconditioner for time-harmonic electromagnetic wave problems based on the one-level Restricted Additive Schwarz (RAS) preconditioner (CAI; SARKIS, 1999). In the RAS preconditioner, an overlapping subdomain of the mesh is assigned to each MPI process, so each process approximately solves its local subdomain problem and combines its results with neighboring MPI processes in the overlapped regions. At the algebraic level, the RAS preconditioner reads:

$$M_{RAS}^{-1} := \sum_{i=0}^{P-1} R_i^T D_i \tilde{A}_i^{-1} R_i \tag{5.1}$$

where R_i is the restriction operator, R_i^T by consequence is the extension operator, D_i is the partition of unity matrix, and $\tilde{A}_i$ is the local matrix of the sub-problem i equipped with the appropriate transmission conditions (DOLEAN; JOLIVET; NATAF, 2015). The RAS preconditioner is relatively popular because of its easy implementation and availability on publicly accessible linear algebra software packages such as PETSc (BALAY et al., 2019a), and Trilinos (HEINLEIN et al., 2018). In these frameworks however the local matrix is generally computed algebraically as $A_i = \left(R_i A R_i^T\right)$, creating easy-to-use black box preconditioners.

One can additively enrich the one-level preconditioner M^{-1} by

$$\mathcal{P}_{AD}^{-1} = Q + M^{-1}. \tag{5.2}$$

where $Q = Z A_c^{-1} Z^T$ is the coarse space problem, and Z is a full column rank matrix also known as the interpolation operator. In this work, we use the shifted Laplacian approach to derive our coarse problem. The coarse space problem is constructed by adding damping to the problem, a complex shift in the wavenumber. However, instead of using a coarser mesh, the coarse problem is built using lower-order polynomial spaces. This choice simplifies the interpolation between spaces greatly; the interpolation operator is strictly local to the cell, and most operations can be performed only once on the reference element, avoiding the need of explicitly computing Z and Z^T. In addition, it maintains neighboring relationships between the subdomains, enabling a natural distribution of the coarse problem.

5.2 Formulation and discretization

This section presents the formulation of our model problem and reviews some finite element concepts that will be used in the subsequent sections. Most of these details were presented in Chapter 2; however, we repeat some concepts to make the chapter more self-contained and easier to read. Also, the reader is referred to (MONK et al., 2003) for an in-depth treatment of finite element for Maxwell's equations.

We are interested in solving the problem of electromagnetic scattering from a non-magnetic bounded object. Given a time-harmonic incoming wave $\mathbf{E}^i$ and a complex bounded volumetric source $\mathbf{F}$, we seek to find the total electric field $\mathbf{E}$ so that

$$\nabla \times \nabla \times \mathbf{E} - k(\boldsymbol{x})^2\mathbf{E} = \mathbf{F} \quad \text{in} \quad \Omega \tag{5.3}$$

where $k(\boldsymbol{x}) = k_0^2\tilde{\varepsilon_r}(\boldsymbol{x})$ and k_0 is the free-space wavenumber, $\tilde{\varepsilon_r} = \varepsilon_r - j\frac{\eta_0}{k_0}\sigma$ is the complex relative permittivity.

Equation (5.3) should be completed with the Silver–Müller radiation condition, a generalization of the Sommerfeld's radiation condition, expressing the fact that scattered waves must propagate toward infinity only. The Silver–Müller is strictly valid at infinity (JIN, 2015); however, it is often desirable to truncate the infinite domain artificially.This truncation can be achieved by enclosing the area of interest with an artificial surface. For simplicity, we are going to use the impedance boundary condition:

$$(\nabla \times \mathbf{E}) \times \mathbf{n} + jk\,(\mathbf{n} \times \mathbf{E}\times)\,\mathbf{n} = \mathbf{g} \quad \text{on} \quad \partial\Omega \tag{5.4}$$

where $\mathbf{n}$ is the unit outward normal to $\partial\Omega$ and $\mathbf{g}$ is a given tangential field on $\partial\Omega$. For example, $\mathbf{g}$ can be computed from an incident field by

$$\mathbf{g} = \left(\nabla \times \mathbf{E}^i\right) \times \mathbf{n} + jk\left(\mathbf{n} \times \mathbf{E}^i\right) \times \mathbf{n}. \tag{5.5}$$

The weak formulation can be obtained using the standard Galerkin procedure of taking the inner product of the original Equation (5.3) by the complex conjugate of a smooth vector test function $\mathbf{v}$ followed by integration by parts over Ω. After some algebraic manipulations and substitution of Equation (5.4), it reads

$$\int_\Omega \nabla \times \mathbf{E} \cdot \nabla \times \bar{\mathbf{v}} \,\mathrm{d}x - \int_\Omega k^2 \mathbf{E} \cdot \bar{\mathbf{v}} \,\mathrm{d}x + j \int_{\partial\Omega} k\mathbf{E}_t \cdot \bar{\mathbf{v}}_t \,\mathrm{d}s = \int_\Omega \mathbf{F} \cdot \bar{\mathbf{v}} \,\mathrm{d}x + \int_{\partial\Omega} \mathbf{g} \cdot \bar{\mathbf{v}}_t \,\mathrm{d}s \tag{5.6}$$

where

$$\mathbf{E}_t = (\mathbf{n} \times \mathbf{E}|_{\partial\Omega}) \times \mathbf{n}.$$

For $\Omega \in \mathbb{R}^3$, the natural space for the weak solution of (5.6) and for the test functions **v** is the Sobolev space of vector-valued functions $\mathcal{H}(curl, \Omega)$, given by

$$\mathcal{H}(curl, \Omega) := \{\mathbf{E} \in [L^2(\Omega)]^3 \text{ , } \nabla \times \mathbf{E} \in [L^2(\Omega)]^3\}. \tag{5.7}$$

A central aspect of the finite element method is the construction of discrete subspaces $V \subset \mathcal{H}(curl, \Omega)$ from infinite-dimensional function spaces by patching together local function spaces defined by a set of finite elements. Supposing that we can construct such a subspace, our discrete formulation reads: Find $\mathbf{E} \in V \subset \mathcal{H}(curl, \Omega)$, such that

$$a(\mathbf{E}, \mathbf{v}) = L(\mathbf{v}; \mathbf{F}, \mathbf{g}), \quad \forall \mathbf{v} \in V \subset \mathcal{H}(curl, \Omega). \tag{5.8}$$

where $a : V \times V \to \mathbb{C}$ is the bilinear form and L is the linear form, the left-hand side and right-hand side of Equation (5.6) respectively. Discretizing (5.8), one obtains a linear system

$$\mathbf{Au} = \mathbf{f}$$

where $\mathbf{A} \in \mathbb{C}^{N \times N}$, and $\mathbf{u}, \mathbf{f} \in \mathbb{C}^N$. Where $\mathbf{u}$ is the expansion vector of degrees of freedom of the discrete solution **E**, and N is the global number of degrees of freedom (*dofs*).

5.2.1 Finite Element Definition

Therefore, the construction of the finite element system and consequently the discrete spaces relies on the definition of a finite element. We follow Ciarlet's definition (CIARLET, 2002), also used in(LOGG; MARDAL; WELLS, 2012; BRENNER; SCOTT, 2007), and defined in 2, Definition 2.3.1.

Nédélec introduced two fundamental families of elements that are suitable for discretizing the $\mathcal{H}(curl, \Omega)$ space, the first kind in 1980 (NÉDÉLEC, 1980), and the elements of the second kind six years later (NÉDÉLEC, 1986). These elements are constructed such that the tangential components are continuous across element facets, though their normal components are allowed to be discontinuous. Thus they can be used in the presence of discontinuous electromagnetic properties while maintaining the required continuity of electromagnetic fields.

For the sake of brevity, in the rest of the chapter, we use only Nédélec elements of the first kind, and we call it simply Nédélec elements thereof. This, however, is not a requirement neither a limitation of our framework. Following Definition 2.3.1, the Nédélec element of arbitrary order q on a tetrahedron (T), can be described by

$$\mathcal{V} = [\mathcal{P}_{q-1}(T)]^3 + \mathcal{S}_q(T)$$
$$\mathcal{L} = \begin{cases} \int_e v \cdot t\, \psi \, dl & \psi \in \mathcal{P}_{q-1}(e) \text{ for each } e \\ \int_f v \times n \cdot \psi \, df & \psi \in [\mathcal{P}_{q-2}(f)]^2 \text{ for each } f \\ \int_T v \cdot \psi \, dx & \psi \in [\mathcal{P}_{q-3}(T)]^3. \end{cases}$$

And the non-standard subspace $\mathcal{S}_q(J)$ of homogeneous vector polynomials of degree q is defined by:

$$\mathcal{S}_q(T) = \{\mathbf{s} \in [\mathcal{P}(T)]^3 : \mathbf{s}(\boldsymbol{x}) \cdot \boldsymbol{x} = 0, \forall \boldsymbol{x} \in T\}.$$

For example, the functionals that define a first-order Nédélec element on the reference tetrahedron T (defined in Figure 5) are the tangential integral moments on the edges ($i = 0, \cdots, 5$)

$$\ell_i : \boldsymbol{v} \mapsto \int_{e_i} \boldsymbol{v} \cdot (1)\boldsymbol{t}_i \tag{5.9}$$

Any function $\mathbf{p}$ in $\mathcal{V}_1$ can be written in terms of the basis:

$$\mathbf{p}(\boldsymbol{x}) = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{\phi}_i(\boldsymbol{x}) \tag{5.10}$$

where $\alpha_i \in \mathbb{C}$ is the expansion coefficients. These basis functions can then be used in finite element computations for assembling matrices and vectors. High-order Nédélec elements have more complicated definitions since the degrees of freedom are associated with edges, facets, and volumes. For instance, second-order Nédélec elements have 20 degrees of freedom, $\mathcal{L}_2 = \{\ell_0, ..., \ell_{19}\}$; two degrees of freedom associated with each edge (6 edges) and each face (4 faces).The two basis functions associated with face $\mathbf{f}_0$ ($\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$) are shown in Fig. 8. Once we define the basis functions for $\mathcal{V}_2$, , we can write a general finite element function in $\hat{K}$ using Equation (5.10). Let $\{\hat{\boldsymbol{\psi}}_i\}^m$ be the basis functions associated with the finite dimensional $\mathcal{V}_2$, any function $\mathbf{f}$ in $\mathcal{V}_2$ can be written in terms of the basis:

$$\mathbf{p}(\boldsymbol{x}) = \sum_{i=0}^{m-1} \beta_i \boldsymbol{\psi}_i(\boldsymbol{x}) \tag{5.11}$$

where $\boldsymbol{\beta} \in \mathbb{C}^m$, even though the basis functions are real-valued.

5.2.2 Affine Mapping

The first step in most finite element software is to create/read a finite element mesh $\mathcal{T}_h = \{K\}$ covering the domain Ω. With a mesh and a definition of a local finite element space, it is straightforward to define a global finite element space over the mesh $\mathcal{T}_h$. The global space consists of functions whose restrictions to each $K \in \mathcal{T}_h$ lie in the local space $V(K)$ and that satisfy the required continuity requirements. However, it is customary to create a unique reference finite element $(\hat{K}, \hat{\mathcal{V}}, \hat{\mathcal{L}})$ and map it to each cell in the mesh. In order to maintain desired properties when mapping finite elements from a reference element to an actual element on a mesh, an appropriate mapping must be used (ROGNES; KIRBY; LOGG, 2009).

As we only consider tetrahedral meshes, each $K \in \mathcal{T}_h$ is a tetrahedron and can be obtained by mapping $\hat{K}$ with an affine map. For any $K \in \mathcal{T}_h$ there is a map $F_K(\hat{K}) = K$ such that

$$\boldsymbol{x} = F_K\hat{\boldsymbol{x}} = J\hat{\boldsymbol{x}} + \mathbf{b}_K$$

where J is the Jacobian of the transformation, a constant 3×3 matrix and $\mathbf{b}_K$ is a vector. Our main interest is the mapping between functions defined on $\hat{K}$ and K, and vice versa. The simplest mapping used to map scalar functions p is defined by $p \circ F_K = \hat{p}$, where $\circ$ denotes composition of functions. Because we are working in the $\mathcal{H}(curl)$ space, we need continuity-preserving mappings, such as the co-variant Piola mapping. If $\hat{\mathbf{p}} \in \hat{\mathcal{V}}$ we can define $\mathbf{p} \in \mathcal{V}$ as

$$\mathbf{p} \circ F_K = J^{-T}\hat{\mathbf{p}}. \tag{5.12}$$

This mapping preserves the tangential component of basis functions on edges and facets. We describe this process in more detail in Chapter 2.

5.2.3 Base Transformation

The construction of conforming $\mathcal{H}(curl)$ conforming spaces requires neighboring cells to agree on the layout and orientation of shared *dofs* on edges and faces. In general, finite element libraries require that the cells in a mesh are locally numbered to guarantee that entities are consistently ordered. In FEniCSx, however, no assumption of mesh orientation is made, and instead, the approach of basis transformation developed in (SCROGGS et al., 2021) is used.

The technique is local to a cell and uses the unique global indices for each mesh entity. A transformation matrix $M \in \mathbb{R}^{n \times n}$ can be compute by comparing the orientation of each sub-entity to its orientation on the reference cell. This transformation matrix can then be used at the cell level during the finite element assembly process (SCROGGS et al., 2021). Let $\{\hat{\phi}_i\}_{i=0}^{n-1}$ be a set of basis functions on the reference tetrahedron $\hat{T}$, that linear transformation of the basis can be computed as

$$\begin{pmatrix} \hat{\phi}_0^* \\ \vdots \\ \hat{\phi}_{n-1}^* \end{pmatrix} = M \begin{pmatrix} \hat{\phi}_0 \\ \vdots \\ \hat{\phi}_{n-1} \end{pmatrix}. \tag{5.13}$$

and the basis $\{\hat{\phi}_i^*\}_{i=0}^{n-1}$ have a consistent orientation, meaning that if this transformation is applied to each cell $T \in \mathcal{T}$, the required continuity is attained globally.

5.2.4 Interpolation

Associated with a finite element is also a local interpolation operator. Let $\hat{\mathcal{L}} = \{l_i\}^n$ and $\{\hat{\boldsymbol{\phi}}_i\}^n$ be the *dofs* and basis functions associated with the finite dimensional function space $\hat{\mathcal{V}}$ in $\hat{K}$. Given some vector function $\hat{\mathbf{f}}$ in $\hat{K}$, the local interpolation operator can be defined by

$$\Pi_{\hat{K}}\hat{\mathbf{f}}(\boldsymbol{x}) = \sum_{i=0}^{n-1} l_i(\hat{\mathbf{f}}(\boldsymbol{x}))\hat{\boldsymbol{\phi}}_i. \tag{5.14}$$

Now, let $\{\ell_i\}^m$ and $\{\hat{\boldsymbol{\psi}}_i\}^m$ be the *dofs* and basis functions associated with the finite dimensional function space $\hat{\mathcal{W}}$ on the same reference element $\hat{K}$. So any function $\hat{\mathbf{p}} \in \hat{\mathcal{W}}$ can be written using the expansion of coefficients as $\hat{\mathbf{p}}(\boldsymbol{x}) = \sum_{i=0}^{m-1} \hat{\alpha}_i\hat{\boldsymbol{\psi}}_i(\boldsymbol{x})$. The interpolation operator $\Pi_{\hat{K}} : \hat{\mathcal{W}} \to \hat{\mathcal{V}}$ can then be written as

$$\Pi_{\hat{K}}\hat{\mathbf{p}}(\boldsymbol{x}) = \sum_{i=0}^{n-1} l_i \underbrace{\left(\sum_{j=0}^{m-1} \hat{\alpha}_j\hat{\boldsymbol{\psi}}_j(\boldsymbol{x})\right)}_{\mathbf{p}\in\mathcal{W}} \hat{\boldsymbol{\phi}}_i(\boldsymbol{x}). \tag{5.15}$$

Using the fact that the functionals $\{l_i\}^n$ are linear, Equation (5.15) can be rearranged as

$$\Pi_{\hat{K}}\hat{\mathbf{p}}(\boldsymbol{x}) = \sum_{j=0}^{m-1} \hat{\alpha}_j \underbrace{\left[\sum_{i=0}^{n-1} l_i\left(\hat{\boldsymbol{\psi}}_j(\boldsymbol{x})\right)\hat{\boldsymbol{\phi}}_i(\boldsymbol{x})\right]}_{\Pi\hat{\boldsymbol{\psi}}_j(\boldsymbol{x})} \tag{5.16}$$

and consequently as

$$\Pi_{\hat{K}}\hat{\mathbf{p}}(\boldsymbol{x}) = \sum_{j=0}^{m-1} \hat{\alpha}_j\Pi_{\hat{K}}\hat{\boldsymbol{\psi}}_j(\boldsymbol{x}). \tag{5.17}$$

From Equation (5.17), we can reason that if we precompute the interpolation of the basis $\{\hat{\boldsymbol{\psi}}_j\}^m \in \hat{\mathcal{W}}$ to $\hat{\mathcal{V}}$ then the interpolation of any function $\hat{\mathbf{p}} \in \hat{\mathcal{W}}$ can be computed as linear combination of the transformed basis $\Pi_{\hat{K}}\hat{\boldsymbol{\psi}}_j$. This local interpolation operator will be used in the following section to compute the action of the interpolation matrix $Z : V_1 \subset \mathcal{H}(curl) \to V_2 \subset \mathcal{H}(curl)$ without explicit construction of the matrix.

5.3 Preconditioner Construction

Some preconditioners are strictly algebraic since they only use the information available from entries of the discretized matrix A. On the one hand, they can be straightforward to use (black-box preconditioners), and some are readily available in open source packages. On the other hand, they are not suited for large-scale time-harmonic electromagnetic problems attaining poor convergence rates. For the class of preconditioners we are interested in, special knowledge about the differential equation being solved is required, such as the application of transmission conditions and interpolation between spaces. Also, the methods require new matrices modeling related to physical processes. Therefore, it requires a closer integration connecting the discretization framework and the linear algebra back-end.

For developing our preconditioner, we use and extend the open-source finite element library FEniCSx, the new version of the FEniCS (ALNÆS et al., 2015; LOGG et al., 2012a), integrated with PETSc as the linear algebra back-end (BALAY et al., 2019a). In FEniCSx, forms are specified in the Unified Form Language (UFL) (ALNÆS et al., 2014), a domain-specific language for variational forms embedded in Python. The specification of the weak form of the equations, together with the discrete function space, is sufficient to define the problem completely. For example, Equation (2.49) can be expressed in a near mathematical notation in UFL as shown in Listing 16. The integrals are expressed through multiplication with a measure, representing an integral over either the interior of the domain Ω (dx, cell integral) or the boundary $\partial\Omega$ of Ω (ds, exterior facet integral). Note that we assume that $k(\boldsymbol{x})$ is piecewise-constant and can be discretized using a Discontinuous Galerkin space of order 0.

Listing 16 UFL representation of 3.6 with 1st order absorbing boundary conditions and transmission operator

```
mesh = Mesh(VectorElement("Lagrange", tetrahedron, 1))

element = FiniteElement("N1curl", tetrahedron, 2)
V = FunctionSpace(mesh, element)
E = TrialFunction(V)
v = TestFunction(V)

DG = FunctionSpace(mesh, FiniteElement("DG", tetrahedron, 0))
k = Coefficient(DG)

n = FacetNormal(mesh)

a = inner(curl(E), curl(v)) * dx
    - pow(k, 2) * inner(E, v) * dx
    + 1j * k * inner(cross(n, E), cross(n, v)) * ds
L = inner(F, v) * dx + inner(g, v) * ds
```

The UFL forms can then be compiled with FFCx, the new version of the FEniCS Form Compiler (FENICSPROJECT, 2019). FFCx generates efficient low-level C code that can be used to assemble the corresponding discrete operator. The generated code is not used directly but via DOLFINx to assemble a matrix, using the generated code for the bilinear form and a vector, using the generated code for the linear form. A simple Python snippet for assembling a distributed matrix and vector is presented below in Listing 17.

Thus, the DOLFINx assembly algorithm may call the generated code on each cell of the mesh to compute the element tensor. The entries of the local tensor are inserted into the global tensor using the local-to-global mapping $\iota_T : [0, n_T - 1] \rightarrow [0, N - 1]$. The global matrix A is

Listing 17 Python snippet that assembles UFL forms into PETSc matrices

```
A = dolfinx.fem.assemble_matrix(a)
b = dolfinx.fem.assemble_vector(L)
```

not assembled in practice, nor is the inverse of M is computed. Instead, we provide support to the linear algebra back-end for computing the action of the global finite element matrix applied to a distributed vector $A\mathbf{v}$, and for solving the auxiliary problem

$$\mathbf{y} = \mathcal{P}_{AD}^{-1}\mathbf{x} \tag{5.18}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{C}^N$ are complex distributed vectors of size N, the total number degrees of freedom. In C++ this is accomplished by creating a `PCSHELL` and in Python using `PC.Type.PYTHON` (BALAY et al., 2019a).

$$\mathcal{P}_{AD}^{-1} := \underbrace{Z A_c^{-1} Z^T}_{\text{Q}} + \underbrace{\sum_{i=0}^{P-1} R_i^T D_i \tilde{A}_i^{-1} R_i}_{M^{-1}} \tag{5.19}$$

Next, we will examine the preconditioner expressed by equation (4.1) and expanded in equation (5.19). The implementation of the action of M^{-1} to a distributed vector is described in Chapter 3, so we will focus only on the construction of Q and $\tilde{A}$.

One ingredient of M^-1 we still need to implement is the assembly of the local matrix with an appropriate transmission condition. Instead of assembling a global matrix, each process i assembles a local (to the process) sequential matrix (for instance, we use `MATSEQAIJ`) of size $N_i \times N_i$. We extend the variational form of Equation 2.49 with an impedance transmission condition applied at the interface between adjacent subdomains Γ_i, such that

$$\begin{aligned} a_i(\mathbf{E}_i, \mathbf{v}_i) = &\int_{\Omega_i} \nabla \times \mathbf{E}_i \cdot \nabla \times \bar{\mathbf{v}}_i \, \mathrm{d}x \\ &- \int_{\Omega_i} k_i^2 \mathbf{E}_i \cdot \bar{\mathbf{v}}_i \, \mathrm{d}x \\ &+ j \int_{\partial\Omega_i \cap \partial\Omega} k_i (\mathbf{E}_i \times \mathbf{n}) \cdot (\bar{\mathbf{v}}_i \times \mathbf{n}) \, \mathrm{d}s \\ &+ j \int_{\Gamma_i} k_i (\mathbf{E}_i \times \mathbf{n}) \cdot (\bar{\mathbf{v}}_i \times \mathbf{n}) \, \mathrm{d}s \end{aligned} \tag{5.20}$$

where $\mathbf{E}_i \in V_i \subset \mathcal{H}(curl, \Omega_i)$, and V_i is the restriction of V to Ω_i. If the subdomain Ω_i is strictly internal $\partial\Omega_i \cap \partial\Omega = \emptyset$, however the interface is always $\Gamma_i \neq \emptyset$.

The transmission condition need not match the absorbing boundary condition, but we use the same expression for both boundary conditions for simplicity. In this case, we can easily mark the facets where the impedance condition will be applied (all facets that are connected to only one cell). Once the local matrix $\tilde{A}_i$ has been assembled we can compute the action of

$$\mathbf{y}_i = \tilde{A}_i^{-1}\mathbf{x}_i$$

in two phases. In the setup phase, we compute the LU factorization of A_i, for example, interfacing with mumps (AMESTOY et al., 2000) via PETSc. And during each iteration of the iterative solver, solution phase, the action $\tilde{A}^{-1}$ is computed by forward-backward substitutions using its precomputed LU factorization.

5.3.1 Coarse Space Problem

In the one-level method presented in the previous session, the sub-domains only communicate with their immediate neighbors. Hence the convergence rates deteriorate with an increase in the number of subdomains P. As a result, we add an additional coarse problem to the one-level preconditioner, which links all subdomains at each iteration and is reportedly inexpensive to compute. Later, we will show that the construction of the proposed coarse space problem is indeed cheap compared to the original problem. Despite adversely affecting the time of each iterative method iteration, the number of iterations is considerably reduced and is independent of the number of processes.

In this section, we define the ingredients to enrich additively M^{-1} with a coarse problem Q that couples all subdomains. In effect, we will define the ingredients to apply Q to a distributed vector **x**, such that

$$\mathbf{y} = ZA_c^{-1}Z^T\mathbf{x} \tag{5.21}$$

where $A_c \in \mathbb{C}^{N_c \times N_c}$ is the coarse problem matrix, and $Z \in \mathbb{R}^{N \times N_c}$ is the interpolation operator from $V_c \to V_f$. By consequence $Z^T \in \mathbb{R}^{N_c \times N}$ is the interpolation operator from $V_f \to V_c$. In the following subsections, we will describe how to compute the action of these operators Z, Z^T, and A_c^{-1} and define the coarse space problem.

5.3.1.1 Interpolation Operator

Let $V_f \subset \mathcal{H}(curl, \Omega)$ be the fine discrete function space defined on a mesh $\mathcal{T}$, discretized using order q Nédélec elements. We use V_f to discretize the original problem, so we seek $\mathbf{E} \in V_f$ thath satisfies 2.52. Furthermore, let $V_c \subset \mathcal{H}(curl, \Omega)$ be the coarse function space defined on the same mesh $\mathcal{T}$, but using first-order Nédélec elements. Finally, we define $Z : V_c \to V_f$ to be the interpolation operator between the fine and coarse spaces and consequently, $Z^T : V_f \to V_c$. Instead of computing the matrix Z explicitly, we provide functionality in DOLFINx to perform the interpolation between different function spaces on the same mesh. The listing 18 shows how to use such functionality.

In Section 5.2.4, we showed that it is possible to precompute the interpolation of the basis functions from $\hat{\mathcal{W}}$ to $\hat{\mathcal{V}}$ on the reference element. We assumed that $\hat{\mathcal{W}}$ and $\hat{\mathcal{V}}$ are two discrete

Listing 18 Interface for interpolation in DOLFINx

```
Vf = dolfinx.FunctionSpace(mesh, ("N1curl", 2))
Vc = dolfinx.FunctionSpace(mesh, ("N1curl", 1))

uf = dolfinx.Function(Vf)
uc = dolfinx.Function(Vc)
...
# uf = Z * uc
uf.interpolate(uc)

# uc = Z^T * uf
uc.interpolate(uf)
```

spaces of a different order, defined on the same reference element. Since we define a unique reference element, $(\hat{K}, \hat{\mathcal{V}}, \hat{\mathcal{L}})$ for each element family, more steps are needed to map a function defined in $T \in \mathcal{T}$ to $\hat{K}$. This steps are summarized in Algorithm 4.

In Algorithm 4, the function `MapPullBack` pulls the physical data, $\boldsymbol{\alpha}$, back to the reference element, $\hat{\boldsymbol{\alpha}}$. The function `MapPushForward`, however, performs the inverse operation by pushing data from the reference element to the physical element using the map defined in Equation 2.58. M_c and M_f are the *dof* transformation matrices on the cell T for the fine and coarse elements, respectively, to ensure that tangential fields on edges and facets are continuous. A more general function that supports different types of elements and cell types has been implemented in DOLFINx.

Algorithm 4 Interpolation from $\mathbf{u}_c \in V_c$ to $\mathbf{u}_f \in V_f$

```
1: Precompute $\Pi\boldsymbol{\psi}$
2:
3: for K in $\mathcal{T}_h$ do
4:     $J$ = ComputeJacobian($K$, $\hat{K}$)
5:     $\boldsymbol{\alpha} = \mathbf{u}_c|K$
6:     $\hat{\boldsymbol{\alpha}}$ = MapPullBack($J$, $\boldsymbol{\alpha}$)
7:     $\hat{\boldsymbol{\alpha}} = M_c * \hat{\boldsymbol{\alpha}}$
8:     $\hat{\boldsymbol{\beta}} = \sum_{j=0}^{m-1} \hat{\alpha}_j \Pi\psi_j$
9:     $\boldsymbol{\beta}$ = MapPushForward($J$, $\hat{\boldsymbol{\beta}}$)
10:    $\mathbf{u}_f|K = \boldsymbol{\beta}$
11: end for
```

5.3.1.2 Coarse Problem Matrix

Just discretizing the original problem with a coarse mesh or lower-order polynomial approximation is not helpful for our model problem. The discretization must be fine enough to capture the wave's nature and coarse enough to be effective (cheap). Therefore, we add a complex shift to the wavenumber and define the auxiliary problem

Listing 19 UFL representation of the coarse problem

```
Vc = FunctionSpace(mesh, ("N1curl", 1))
Ec = TrialFunction(Vc)
vc = TestFunction(Vc)

ac = inner(curl(Ec), curl(vc)) * dx
    - (pow(k, 2) + 1j*xi) *  inner(Ec, vc) * dx

Ac = dolfinx.fem.assemble(ac)
```

$$\nabla \times \nabla \times \mathbf{E}_c - \left(k(\mathbf{x})^2 + j\xi\right)\mathbf{E}_c = \mathbf{F}_c \tag{5.22}$$

where $\xi \in \mathbb{R}$. The artificial source $\mathbf{F}_c$ is not assembled, but it is generated during the iterative solution process. Let $\mathcal{I}_c$ be the set of indices associated with the *dofs* of $V_c \subset \mathcal{H}(curl, \Omega)$ and let

$$\begin{aligned} a_c(\mathbf{E}_c, \mathbf{v}_c) = &\int_\Omega \nabla \times \mathbf{E}_c \cdot \nabla \times \bar{\mathbf{v}}_c \,\mathrm{d}x \\ &- \int_\Omega (k^2 + j\xi)\mathbf{E}_c \cdot \bar{\mathbf{v}}_c \,\mathrm{d}x \qquad \forall \mathbf{v}_c \in V_c \end{aligned} \tag{5.23}$$

be the bilinear form associated with the poblem with absorption, Equation 5.22. The coarse matrix A_c, then can be assembled by

$$(A_c)_{i,j} = a_c(\boldsymbol{\phi}_j, \boldsymbol{\phi}_i) \qquad i, j \in \mathcal{I}_c. \tag{5.24}$$

If V_c is discretized using first-order Nédélec elements, the snippet in Listing 19 shows how the coarse matrix can be assembled using DOLFINx.

For Nédélec elements, the number of *dofs* per cell is $n_q = q(q+2)(q+3)/2$, where q is the order of the element. So a first approximation of the time t_q to assemble a matrix of order q as function of t_p is

$$t_q \approx \frac{n_q^2}{n_p^2} t_p. \tag{5.25}$$

For $q = 2$ and $p = 1$, we have that $t_2 \approx 11t_1$. And $q = 3$ and $p = 1$, $t_3 \approx 56t_1$. Figure 40 shows the time for assembling a global matrix using form (2.52) and Nédélec elements of order 1, 2 and 3, for an increasing number of cells. These results show that the coarse space's construction based on approximation with low-order polynomials is cheap and has a predictable cost.

5.4 Numerical Experiments

The numerical experiments were performed on CSD3, the supercomputer of the University of Cambridge HPC Service. Each node consists of two 28-core Intel Xeon Platinum 8276

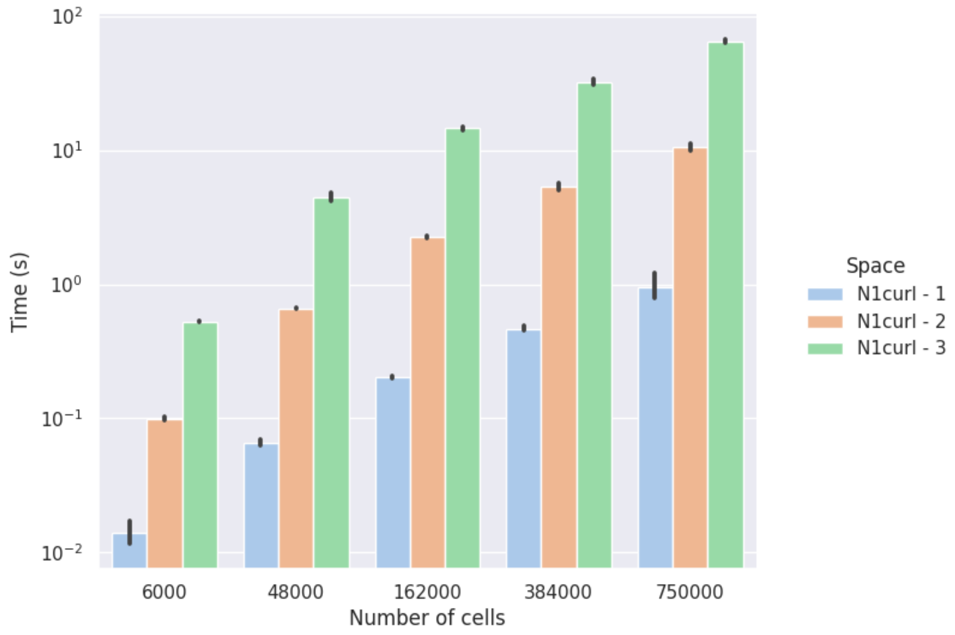


Figure 40 – Average assembling time of the local bilinear form (2.52) on a Unit Cube Mesh with varying number of cells and approximation order. The time is the average of each process assembling its local tensor, using 2240 MPI Processes.

Listing 20 Command to build DOLFINx and dependencies.

```
spack install py-fenics-dolfinx@0.0.1
              ^petsc@3.15.0+mumps+complex+int64
              ^intel-mpi@2019.10.317
              ^intel-mkl@2020.4.304
              cflags="-Ofast -march=native"
```

processors with 192GB of RAM, totalizing 56 cores per node and 3420 MB per core. The MPI programming model was used to execute each run on a fully populated node. For all numerical experiments, we use version 9.3.0 of the GNU Compiler Collection (released on March 12, 2020) and the Spack package manager (GAMBLIN et al., 2015) to build DOLFINx and its dependencies. The commands specifying versions and dependencies to reproduce the software stack used in this chapter can be found in Listing 20.

In the following subsections, we present two numerical experiments that highlight the effectiveness and scalability of the proposed preconditioner and implementation. In the first example, we evaluate the effect of increasing the number of subdomains, and we provide a breakdown of the computational cost at each step. In the second example, we compute the scattering of a plane wave incident upon the aperture of the COBRA cavity, and for a fixed mesh, we evaluate the impact of increasing the number of processes.

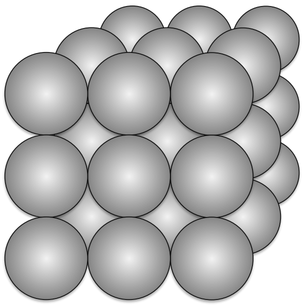


Figure 41 – Diagram of *CubeSpheres* target from (GEFFRIN; SABOUROUX, 2009).

Order	dofs	Time: Local Matrix Assembly (s)	Time: Local LU Factorization (s)	Time: Local solve (s)
1	0.2×10^6	0.225	12.10	0.31
2	1×10^6	2.24	272.87	2.36

Table 3 – Time for each step of the setup phase using 2240 MPI Processes.

5.4.1 Weak Scaling

The first numerical example is an adaptation of one of the benchmarks presented in (GEFFRIN; SABOUROUX, 2009). We compute the scattering of a plane wave oscillating at $16GHz$ by a set of dielectric spheres. At this frequency, the wavelength λ is approximately $18.75mm$. Each sphere has a diameter of $15.9mm$ and a permittivity of $\epsilon_r = 2.6$. They are assembled to obtain a cube measuring $47.6mm$ on each side, see Figure 41.

We set the mesh size to $1/\lambda \approx 1mm$, and increase the distance from the truncation layer to the scatter as we increase the number of subdomains, so each process has approximately 1'000'000 *dofs* using second-order Nédélec elements of the first kind. In this configuration, every subdomain is composed of approximately 162'000 cells. We use *gmsh* to generate a coarse mesh (GEUZAINE; REMACLE, 2009). The coarse mesh is then refined uniformly in parallel so to obtain the target mesh size.

We solve the problem described above using the GMRES, as implemented in PETSc, with one and two-level Optimized Restricted Additive Schwarz (ORAS) preconditioners. The one-level preconditioner is described in section Chapter 3. In the two-level preconditioner described in section here, we solve the coarse problem using a one-level ORAS, and we set the absorption parameter to $\xi \approx k(\boldsymbol{x})$. Finally, the actual problem is discretized using second-order Nedelec elements and the coarse problem using first-order elements.

Results for the setup phase are reported in Table 3. The values shown in the table show the average over ten runs using 2240 processes. Since the setup phase is embarrassingly parallel and there is no communication, we only report the timing for a fixed number of processors (2240) and approximately 2 billion dofs. As predicted in the previous section, the matrix assembly time is approximately ten times faster for the coarse space. On the other hand, in the LU factorization,

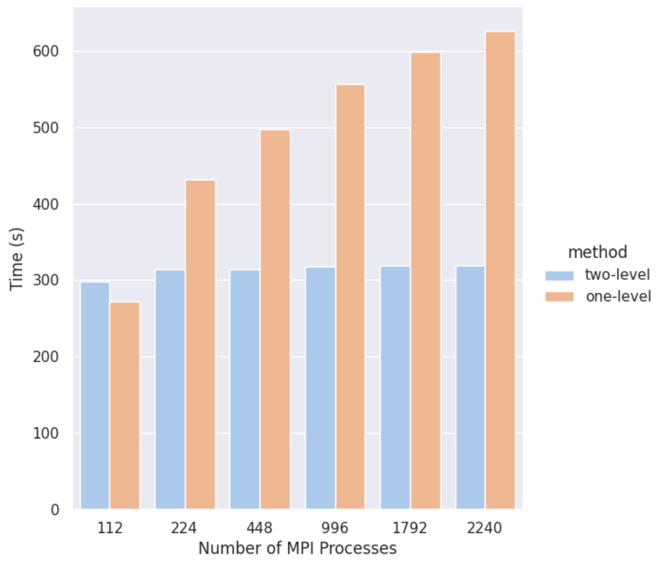


Figure 42 – Weak Scaling Experiment: Total time to solution.

which is the dominant step in the setup phase, this difference increases to 20 times. The setup phase for the two-level method is only 4% slower than the one-level method; however, the solution phase is much faster due to the reduced number of iterations.

A plot of the time-to-solution including both the setup and solution phases, on up to 2240 subdomains, is shown in Figure 42. We can recognize that the time-to-solution using the one-level preconditioner grows as we increase the number of processors. This can be explained by the increase in iterations, an inherent property of methods where communication is restricted to adjacent subdomains.

Figure 42 also shows a slight increase in the time of solution using the two-level preconditioner, but much less pronounced than in the one-level method. However, in this case, this increase cannot be justified by an increase GMRES iterations, as it remained at around 30 ± 2 iterations. The anticipated cause is the increase in inner iterations for solving the coarse space using a one-level method. We will evaluate how accurate the coarse space solution must be and the effects of an early stop in future works.

5.4.2 Strong Scaling

To demonstrate the strong scaling of the proposed preconditioner and implementation, we consider the well-known "COBRA" cavity problem. This problem has also been evaluated in the context of domain decomposition in (DOLEAN et al., 2015) and (BONAZZOLI et al., 2019). A more detailed description of the geometry can be found in (LIU; JIN, 2003). The lateral cavity walls are modeled as Perfect Electrically Conducting (PEC) surfaces. Also, a PEC plate terminates the final straight segment (100 mm long) of the cavity. PEC plates can be modeled as Dirichlet boundary conditions using curl-conforming elements.

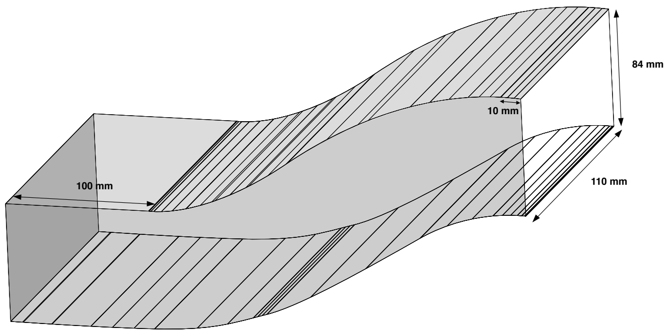


Figure 43 – Illustration of the "COBRA" cavity.

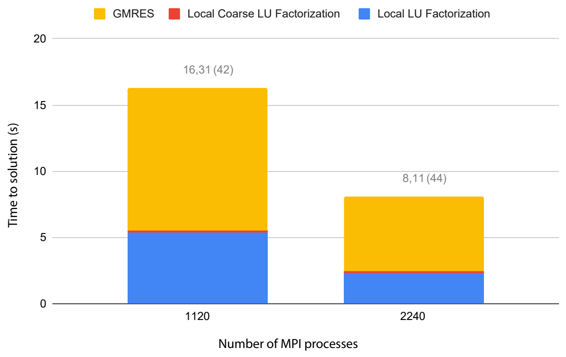


Figure 44 – Breakdown of the timings of the preconditioned GMRES, $f = 10GHz$ problem.

We compute the scattering of plane waves oscillating at $10GHz$ and $16GHz$ incident upon the cavity aperture. A box whose sides are four wavelengths away in each direction from the scatterer is used to truncate the domain. Unlike the previous problem, we use third-order Nédélec elements to discretize the actual problem and first-order Nédélec elements to discretize coarse problem. Then, the coarse problem is solved with GMRES preconditioned with the one-level ORAS, and the number of inner iterations is limited to 50.

For the $10GHz$ problem, we use approximately 6.8 million cells and 13.4 for the $16GHz$ problem. The dimensions N and N_c of the fine and coarse space respectively are shown in Table 4. We use 20 and 40 supercomputing nodes for both configurations, equaling 1120 and 2240 MPI processes, respectively.

The breakdown of the time to solution of the $10GHz$ problem is shown in Figure 44. The speedup going from 1120 (20 nodes) to 2240 (40 nodes) nodes is 2.1, which would indicate a superlinear speedup. This happens because the factorization and subsequent forward-backward substitutions are not linear in the number of *dofs*. If we only consider the speedup of the setup phase, the speedup would be 2.33. The number of GMRES iterations is shown in brackets and indicates good independence of the number of subdomains. It can also be noted that coarse space problem local factorization is essentially free compared to the local factorization of the first level (approximately 2%).

The same effect can be observed in Figure 45. Again, a speedup of approximately 2.4

Freq	#Cells	N	N_c
10 Ghz	6.8×10^6	136×10^6	9×10^6
16 GHz	13.4×10^6	262×10^6	17.7×10^6

Table 4 – Dimension of discretization spaces: number of *dofs*

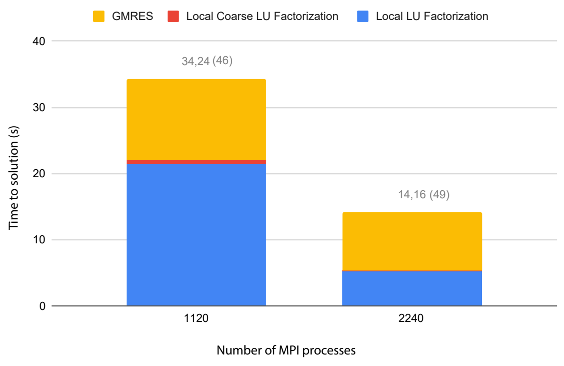


Figure 45 – Breakdown of the timings of the preconditioned GMRES, $f = 16GHz$ problem.

can be noticed, and the super-linear speedup is due to the LU factorization of the local problem.

5.5 Conclusions

This chapter presented a two-level domain decomposition preconditioner whose coarse space is based on the shifted Laplacian approach targeting electromagnetic scattering problems. The coarse space construction using low-order discrete spaces is cheap, and the computational cost is predictable. Furthermore, by using the same mesh for the original and coarse problem, we simplify the distribution of data in parallel, minimize communication, and implement an optimized interpolation operator (that is strictly local to the cell). Finally, we showed experimentally that the proposed preconditioner and corresponding implementation are well suited for simulations of large-scale electromagnetic scattering problems, scaling up to 2240 cores and more than 2 billion *dofs*. The building blocks needed to adapt the proposed method to specific problems were incorporated in the DOLFINx library and are freely available as open-source software.

6 Conclusions and Perspectives

This work investigated and proposed new techniques for solving time-harmonic electromagnetic problems discretized by the finite element method. We have shown through numerical experiments that strictly algebraic solving methods do not work correctly for such problems, and instead, an appealing alternative is to use physics-informed preconditioned iterative methods. This approach, however, requires a strong integration between the discretization and solution methods. As the current trend in high-performance computing trends toward increased parallelism, domain decomposition preconditioners come into play, and it is the class of method we consider in this thesis.

The implementation of effective domain decomposition preconditioners for time harmonic electromagnetic problems is remarkably challenging. However, we have shown that the implementation can be simplified significantly by employing a finite-element framework that allows each algorithm to be expressed at a suitable abstraction level. To manage all the abstractions and their interrelationships, we use and extend the FEniCS Problem Solving Environment. We have extended the Unified Form Language to support complex-valued partial differential equations. It is possible to express standard differential equations plus domain decomposition specific terms, such as transmission conditions and coarse spaces, for time-harmonic electromagnetic problems using a near mathematical representation. We also worked on automatic code generation and form compilers, so for a given sesquilinear form expressed using UFL, we can automatically generate optimized low-level element kernels with FFCx. These kernels are then used to assemble the local matrix and local vector on each process of an MPI program.

Furthermore, we improved the communication pattern of the computational back-end, DOLFINx, by introducing MPI Neighborhood Collectives and new functionalities for distributed vector gathers and scatters. Effective MPI communication is essential for high-performance domain decomposition methods since adjacent subdomains exchange information via MPI messages at each iteration. All extensions are now freely available as part of the FEniCS project.

In addition to the availability of suitable software infrastructure, the effectiveness of domain decomposition methods depends on two extra factors that cannot be obtained purely algebraically: the transmission conditions applied at the interface between adjacent subdomains and the coarse space correction that allows a global transfer of information. In Chapter 3, we described in detail a new optimization process for devising transmission conditions that automatically takes into account the propagative nature of waves. Numerical experiments showed that the Schwarz preconditioner with our optimized transmission conditions leads systematically to fast convergence rates. Furthermore, this optimization process and analysis help to reason about the non-convergence reason of traditional domain decomposition methods.

Even with optimal local transmission conditions, the convergence rates of one-level preconditioners depend on the number of processes, preventing such methods from achieving weak scalability on large-scale machines. After evaluating different two-level preconditioners, in Chapters 4 and 5, we propose a new two-level domain decomposition preconditioner for time-harmonic electromagnetic problems that is efficient and scalable in parallel: the construction of the coarse space problem is cheap, and the preconditioned GMRES depends weakly on the number of processes. The coarse space is constructed using the Shifted Laplacian approach, but instead of using coarse mesh, we discretize the coarse space with low order finite element function space. Consequently, the interpolation between spaces is a local operation, and no extra data structures are required for parallel communication and mesh management. We provide two numerical examples stemming from practical applications to assess both weak and strong scalability. Using the FEniCSx library, the numerical experiments highlight our approach's scalability on up to 2240 processes and 2×10^9 unknowns.

Perspectives

- Comprehensive numerical experiments have been carried out to evaluate the performance of the proposed preconditioners, but the available geometries are manufactured and less complex than those of industrial applications. We plan to extend the performance tests using realistic meshes and material distribution.

- The presented domain decomposition methods are synchronous by nature; at each iteration, there are distinct computation and communication phases with synchronization points. However, this approach is unsuitable for heterogeneous architectures, such as those composed of multiple GPUs and CPUs, given that each process has a different computation power. Future research should consider how to adapt the developed methods to these heterogeneous architectures using asynchronous communication progress.

- A multilevel extension of the current preconditioners is necessary for achieving high performance at exascale, e.g., over tens of thousands of MPI processes. However, such extensions in the domain decomposition community tend to lead to poor algorithmic performances, and profiling code at this scale requires different techniques and the availability of computational resources. Another alternative is to investigate hybrid parallel programming models with more control over fine-level parallelism.

- In Chapter 5, we recognized that one of the limitations of the proposed methods is the high computational incurred in exactly solving local and coarse problems. As a future research direction, we plan to study the effect of inexact solves on two-level Schwarz domain decomposition preconditioners for time-harmonic Maxwell's equations, aiming to determine how accurate the coarse space solution should be to devise more effective iterative solvers.

Bibliography

ALNÆS, M. S. et al. The fenics project version 1.5. *Archive of Numerical Software*, v. 3, n. 100, p. 9–23, 2015. Cited on pages: 49 and 107.

ALNÆS, M. S. et al. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 40, n. 2, p. 9, 2014. Cited on pages: 36, 48, and 107.

AMESTOY, P. R. et al. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, SIAM, v. 23, n. 1, p. 15–41, 2001. Cited on page 45.

AMESTOY, P. R. et al. Mumps: a general purpose distributed memory sparse solver. In: SPRINGER. *International Workshop on Applied Parallel Computing*. [S.l.], 2000. p. 121–130. Cited on page 109.

ANTOINE, X.; GEUZAINE, C. Optimized schwarz domain decomposition methods for scalar and vector helmholtz equations. In: *Modern Solvers for Helmholtz Problems*. [S.l.]: Springer, 2017. p. 189–213. Cited on pages: 67 and 76.

BABUSKA, I. M.; SAUTER, S. A. Is the pollution effect of the fem avoidable for the helmholtz equation considering high wave numbers? *SIAM Journal on numerical analysis*, SIAM, v. 34, n. 6, p. 2392–2423, 1997. Cited on page 43.

BALANIS, C. A. *Advanced engineering electromagnetics*. [S.l.]: John Wiley & Sons, 2012. Cited on pages: 22 and 23.

BALAY, S. et al. Petsc users manual. Argonne National Laboratory, 2019. Cited on pages: 101, 107, and 108.

BALAY, S. et al. *PETSc Users Manual*. [S.l.], 2019. Disponível em: <http://www.mcs.anl.gov/petsc>. Cited on pages: 43 and 49.

BARATTA, I. A. *Complex Number support in FEniCS*. 2018. <https://summerofcode.withgoogle.com/archive/2018/projects/5117785856278528/>. Accessed: 2019-05-01. Cited on page 49.

BARATTA, I. A.; SILVA, E. J. Multi-domain transmission conditions for domain decomposition methods applied to scattering problems. *IEEE Transactions on Magnetics*, IEEE, v. 54, n. 3, p. 1–4, 2018. Cited on pages: 55 and 70.

BARKA, A. Integration of antennas onboard vehicles and diffraction by large and complex structures with multiple-domain–multiple-methods techniques. *Proceedings of the IEEE*, IEEE, v. 101, n. 2, p. 280–297, 2013. Cited on page 14.

BARRETT, R. et al. *Templates for the solution of linear systems: building blocks for iterative methods*. [S.l.]: Siam, 1994. v. 43. Cited on page 46.

BARTON, M.; CENDES, Z. New vector finite elements for three-dimensional magnetic field computation. *Journal of Applied Physics*, AIP, v. 61, n. 8, p. 3919–3921, 1987. Cited on page 27.

BENAMOU, J.-D. An introduction to eulerian geometrical optics (1992–2002). *Journal of Scientific Computing*, Springer, v. 19, n. 1-3, p. 63–93, 2003. Cited on page 95.

BENAMOU, J.-D.; COLLINO, F.; MARMORAT, S. *Numerical microlocal analysis revisited*. Tese (Doutorado) — INRIA, 2011. Cited on pages: 94 and 95.

BENAMOU, J.-D.; COLLINO, F.; RUNBORG, O. Numerical microlocal analysis of harmonic wavefields. *Journal of Computational Physics*, Elsevier, v. 199, n. 2, p. 717–741, 2004. Cited on pages: 94 and 95.

BENAMOU, J.-D.; DESPRÈS, B. A domain decomposition method for the helmholtz equation and related optimal control problems. *Journal of Computational Physics*, Elsevier, v. 136, n. 1, p. 68–82, 1997. Cited on pages: 67 and 68.

BERENGER, J.-P. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics*, Elsevier, v. 114, n. 2, p. 185–200, 1994. Cited on page 26.

BÉRENGER, J.-P. Perfectly matched layer (pml) for computational electromagnetics. *Synthesis Lectures on Computational Electromagnetics*, Morgan & Claypool Publishers, v. 2, n. 1, p. 1–117, 2007. Cited on page 26.

BLUCK, M. Conforming hierarchical basis functions. *Communications in Computational Physics*, Cambridge University Press, v. 12, n. 4, p. 1215–1256, 2012. Cited on page 101.

BONAZZOLI, M. et al. Domain decomposition preconditioning for the high-frequency time-harmonic maxwell equations with absorption. *Mathematics of Computation*, v. 88, n. 320, p. 2559–2604, 2019. Cited on pages: 101 and 114.

BONAZZOLI, M. et al. Two-level preconditioners for the helmholtz equation. In: SPRINGER. *International Conference on Domain Decomposition Methods*. [S.l.], 2017. p. 139–147. Cited on pages: 90 and 92.

BONAZZOLI, M. et al. Parallel preconditioners for high-order discretizations arising from full system modeling for brain microwave imaging. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, Wiley Online Library, 2017. Cited on page 14.

BOSSAVIT, A.; LAMAUDIÈRE, J.; MAESTRALI, B. Team workshop problem 19. *Électricité de France*, 1992. Cited on page 80.

BOSSAVIT, A.; RAPETTI, F. A prolongation/restriction operator for whitney elements on simplicial meshes. *SIAM Journal on Numerical Analysis*, SIAM, v. 43, n. 5, p. 2077–2097, 2005. Cited on page 101.

BOSSAVIT, A.; VERITE, J.-C. A mixed fem-biem method to solve 3-d eddy-current problems. *IEEE Transactions on Magnetics*, IEEE, v. 18, n. 2, p. 431–435, 1982. Cited on page 27.

BOUAJAJI, M. E.; ANTOINE, X.; GEUZAINE, C. Approximate local magnetic-to-electric surface operators for time-harmonic maxwell's equations. *Journal of Computational Physics*, Elsevier, v. 279, p. 241–260, 2014. Cited on pages: 25 and 76.

BOUAJAJI, M. E. et al. Optimized schwarz methods for the time-harmonic maxwell equations with damping. *SIAM Journal on Scientific Computing*, SIAM, v. 34, n. 4, p. A2048–A2071, 2012. Cited on page 75.

BOUBENDIR, Y.; ANTOINE, X.; GEUZAINE, C. A quasi-optimal non-overlapping domain decomposition algorithm for the helmholtz equation. *Journal of Computational Physics*, Elsevier, v. 231, n. 2, p. 262–280, 2012. Cited on page 68.

BRANDT, A.; LIVSHITS, I. Wave-ray multigrid method for standing wave equations. *Electron. Trans. Numer. Anal*, v. 6, n. 162-181, p. 91, 1997. Cited on page 92.

BRENNER, S.; SCOTT, R. *The mathematical theory of finite element methods*. [S.l.]: Springer Science & Business Media, 2007. v. 15. Cited on pages: 26, 28, 29, and 103.

CAI, X.-C.; SARKIS, M. A restricted additive schwarz preconditioner for general sparse linear systems. *Siam journal on scientific computing*, SIAM, v. 21, n. 2, p. 792–797, 1999. Cited on pages: 59 and 101.

CHEN, X. *Computational Methods for Electromagnetic Inverse Scattering*. [S.l.]: Wiley Online Library, 2018. Cited on page 23.

CIARAMELLA, G.; GANDER, M. J. Analysis of the parallel schwarz method for growing chains of fixed-sized subdomains: Part i. *SIAM Journal on Numerical Analysis*, SIAM, v. 55, n. 3, p. 1330–1356, 2017. Cited on page 71.

CIARAMELLA, G.; GANDER, M. J. Analysis of the parallel schwarz method for growing chains of fixed-sized subdomains: Part ii. *SIAM Journal on Numerical Analysis*, SIAM, v. 56, n. 3, p. 1498–1524, 2018. Cited on page 71.

CIARLET, P. G. *The finite element method for elliptic problems*. [S.l.]: Siam, 2002. v. 40. Cited on pages: 29 and 103.

COCQUET, P.-H.; GANDER, M. J. How large a shift is needed in the shifted helmholtz preconditioner for its effective inversion by multigrid? *SIAM Journal on Scientific Computing*, SIAM, v. 39, n. 2, p. A438–A478, 2017. Cited on page 91.

COLLINO, F.; GHANEMI, S.; JOLY, P. Domain decomposition method for harmonic wave propagation: a general presentation. *Computer methods in applied mechanics and engineering*, Elsevier, v. 184, n. 2-4, p. 171–211, 2000. Cited on page 67.

CONEN, L. *Domain decomposition preconditioning for the Helmholtz equation*. Tese (Doutorado) — Università della Svizzera italiana, 2015. Cited on pages: 90, 93, and 94.

CONEN, L. et al. A coarse space for heterogeneous helmholtz problems based on the dirichlet-to-neumann operator. *Journal of Computational and Applied Mathematics*, Elsevier, v. 271, p. 83–99, 2014. Cited on pages: 88, 89, and 94.

CONEN, L. et al. Addendum to" a coarse space for heterogeneous helmholtz problems based on the dirichlet-to-neumann operator"'[j. comput. appl. math. 271 (2014) 83–99]. *Journal of Computational and Applied Mathematics*, v. 290, p. 670–674, 2015. Cited on page 94.

DAVIS, T. A. Algorithm 832: Umfpack v4. 3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 30, n. 2, p. 196–199, 2004. Cited on pages: 45 and 73.

DEMKOWICZ, L. *Computing with hp-ADAPTIVE FINITE ELEMENTS: Volume 1 one and two dimensional elliptic and maxwell problems*. [S.l.]: Chapman and Hall/CRC, 2006. Cited on page 38.

DEMMEL, J. W. et al. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, SIAM, v. 20, n. 3, p. 720–755, 1999. Cited on page 45.

DESPRÉS, B. Méthodes de décomposition de domaine pour les problemes de propagation d'ondes en régime harmonique. *These, Université Paris IX Dauphine, UER Mathématiques de la Décision*, 1991. Cited on pages: 56, 67, and 68.

DESPRÉS, B. A domain decomposition method for the harmonic maxwell equations. *Iterative methods in linear algebra*, Elsevier Science Publishers BV (North-Holland), Amsterdam, p. 475–484, 1992. Cited on page 76.

DIWAN, G. C.; MOIOLA, A.; SPENCE, E. A. Can coercive formulations lead to fast and accurate solution of the helmholtz equation? *Journal of Computational and Applied Mathematics*, Elsevier, v. 352, p. 110–131, 2019. Cited on page 43.

DOHRMANN, C. R.; WIDLUND, O. B. An overlapping schwarz algorithm for almost incompressible elasticity. *SIAM Journal on Numerical Analysis*, SIAM, v. 47, n. 4, p. 2897–2923, 2009. Cited on page 87.

DOLEAN, V.; GANDER, M. J.; GERARDO-GIORDA, L. Optimized schwarz methods for maxwell's equations. *SIAM Journal on Scientific Computing*, SIAM, v. 31, n. 3, p. 2193–2213, 2009. Cited on pages: 75 and 76.

DOLEAN, V. et al. Effective transmission conditions for domain decomposition methods applied to the time-harmonic curl–curl maxwell's equations. *Journal of computational physics*, Elsevier, v. 280, p. 232–247, 2015. Cited on pages: 76 and 114.

DOLEAN, V.; GANDER, M. J.; VENEROS, E. Schwarz methods for second order maxwell equations in 3d with coefficient jumps. In: *Domain Decomposition Methods in Science and Engineering XXII*. [S.l.]: Springer, 2016. p. 471–479. Cited on page 70.

DOLEAN, V.; GANDER, M. J.; VENEROS, E. Asymptotic analysis of optimized schwarz methods for maxwell's equations with discontinuous coefficients. *ESAIM. Mathematical Modelling and Numerical Analysis*, EDP Sciences, v. 52, n. 6, 2018. Cited on page 76.

DOLEAN, V.; JOLIVET, P.; NATAF, F. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. [S.l.]: SIAM, 2015. v. 144. Cited on pages: 46, 55, 67, 69, 72, 85, 86, and 101.

DRUSKIN, V.; GUTTEL, S.; KNIZHNERMAN, L. Near-optimal perfectly matched layers for indefinite helmholtz problems. *SIAM Review*, SIAM, v. 58, n. 1, p. 90–116, 2016. Cited on page 70.

DRYJA, M.; SARKIS, M. V.; WIDLUND, O. B. Multilevel schwarz methods for elliptic problems with discontinuous coefficients in three dimensions. *Numerische Mathematik*, Springer, v. 72, n. 3, p. 313–348, 1996. Cited on page 87.

DRYJA, M.; WIDLUND, O. *An additive variant of the Schwarz alternating method for the case of many subregions*. [S.l.]: Ultracomputer Research Laboratory, Univ., Courant Inst. of Mathematical . . . , 1987. Cited on page 59.

EFSTATHIOU, E.; GANDER, M. J. Why restricted additive schwarz converges faster than additive schwarz. *BIT Numerical Mathematics*, Springer, v. 43, n. 5, p. 945–959, 2003. Cited on page 59.

EHMANN, R. et al. Electromagnetic simulation of a loaded cavity. *IEEE transactions on magnetics*, IEEE, v. 32, n. 3, p. 922–925, 1996. Cited on page 80.

ENGQUIST, B.; RUNBORG, O. Computational high frequency wave propagation. *Acta numerica*, Cambridge University Press, v. 12, p. 181–266, 2003. Cited on page 95.

ENGQUIST, B.; YING, L. Sweeping preconditioner for the helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, SIAM, v. 9, n. 2, p. 686–710, 2011. Cited on pages: 26, 68, and 89.

ERLANGGA, Y. A.; VUIK, C.; OOSTERLEE, C. W. On a class of preconditioners for solving the helmholtz equation. *Applied Numerical Mathematics*, Elsevier, v. 50, n. 3-4, p. 409–425, 2004. Cited on pages: 90 and 101.

ERNST, O. G.; GANDER, M. J. Why it is difficult to solve helmholtz problems with classical iterative methods. In: *Numerical analysis of multiscale problems*. [S.l.]: Springer, 2012. p. 325–363. Cited on pages: 14, 43, and 47.

FARHAT, C. et al. Feti-dph: a dual-primal domain decomposition method for acoustic scattering. *Journal of Computational Acoustics*, World Scientific, v. 13, n. 03, p. 499–524, 2005. Cited on pages: 92, 93, and 94.

FARHAT, C.; MACEDO, A.; LESOINNE, M. A two-level domain decomposition method for the iterative solution of high frequency exterior helmholtz problems. *Numerische Mathematik*, Springer, v. 85, n. 2, p. 283–308, 2000. Cited on pages: 92 and 94.

FENICSPROJECT. *FFC-X: The FEniCS Form Compiler*. [S.l.]: GitHub, 2019. <https://github.com/FEniCS/ffcx>. Cited on pages: 36 and 107.

FISH, J.; QU, Y. Global-basis two-level method for indefinite systems. part 1: convergence studies. *International Journal for Numerical Methods in Engineering*, Wiley Online Library, v. 49, n. 3, p. 439–460, 2000. Cited on page 86.

GAMBLIN, T. et al. The spack package manager: bringing order to hpc software chaos. In: IEEE. *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. [S.l.], 2015. p. 1–12. Cited on page 112.

GANDER, M. J. Optimized schwarz methods. *SIAM Journal on Numerical Analysis*, SIAM, v. 44, n. 2, p. 699–731, 2006. Cited on page 56.

GANDER, M. J.; GRAHAM, I. G.; SPENCE, E. A. Applying gmres to the helmholtz equation with shifted laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed? *Numerische Mathematik*, Springer, v. 131, n. 3, p. 567–614, 2015. Cited on page 47.

GANDER, M. J.; MAGOULES, F.; NATAF, F. Optimized schwarz methods without overlap for the helmholtz equation. *SIAM Journal on Scientific Computing*, SIAM, v. 24, n. 1, p. 38–60, 2002. Cited on page 68.

GANDER, M. J. et al. Schwarz methods over the course of time. *Electron. Trans. Numer. Anal*, v. 31, n. 5, p. 228–255, 2008. Cited on page 56.

GANDER, M. J.; SANTUGINI, K. Cross-points in domain decomposition methods with a finite element discretization. *Electronic Transactions on Numerical Analysis*, Institute of Computational Mathematics, v. 45, p. 219–240, 2016. Cited on page 54.

GANDER, M. J.; SCHÄDLE, A. The pole condition: a padé approximation of the dirichlet to neumann operator. In: *Domain Decomposition Methods in Science and Engineering XIX*. [S.l.]: Springer, 2011. p. 125–132. Cited on page 68.

GANDER, M. J.; ZHANG, H. Algorithmic perspective of pml transmission conditions for domain decomposition methods. In: IEEE. *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*. [S.l.], 2014. p. 1–4. Cited on page 76.

GANDER, M. J.; ZHANG, H. Optimized schwarz methods with overlap for the helmholtz equation. *SIAM Journal on Scientific Computing*, SIAM, v. 38, n. 5, p. A3195–A3219, 2016. Cited on pages: 55, 67, 68, and 70.

GANDER, M. J.; ZHANG, H. A class of iterative solvers for the helmholtz equation: factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized schwarz methods. *SIAM Review*, SIAM, v. 61, n. 1, p. 3–76, 2019. Cited on page 89.

GEFFRIN, J. M.; SABOUROUX, P. Continuing with the fresnel database: experimental setup and improvements in 3d scattering measurements. *Inverse Problems*, IOP Publishing, v. 25, n. 2, p. 024001, 2009. Cited on pages: 8, 79, 80, and 113.

GEUZAINE, C.; REMACLE, J.-F. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, Wiley Online Library, v. 79, n. 11, p. 1309–1331, 2009. Cited on pages: 49 and 113.

GIVOLI, D. Computational absorbing boundaries. In: *Computational Acoustics of Noise Propagation in Fluids-Finite and Boundary Element Methods*. [S.l.]: Springer, 2008. p. 145–166. Cited on page 25.

GRAHAM, I.; SPENCE, E.; VAINIKKO, E. Domain decomposition preconditioning for high-frequency helmholtz problems with absorption. *Mathematics of Computation*, v. 86, n. 307, p. 2089–2127, 2017. Cited on pages: 90, 91, 92, and 101.

GRAHAM, I. G.; SPENCE, E. A.; VAINIKKO, E. Recent results on domain decomposition preconditioning for the high-frequency helmholtz equation using absorption. In: *Modern Solvers for Helmholtz Problems*. [S.l.]: Springer, 2017. p. 3–26. Cited on pages: 91 and 92.

GROPP, W. et al. *Using advanced MPI: Modern features of the message-passing interface*. [S.l.]: MIT Press, 2014. Cited on page 61.

HARRINGTON, R. F. *Time-harmonic electromagnetic fields*. [S.l.]: McGraw-Hill, 1961. Cited on page 77.

HARRINGTON, R. F. *Field computation by moment methods*. [S.l.]: Wiley-IEEE Press, 1993. Cited on page 16.

HEINLEIN, A. et al. Frosch: a fast and robust overlapping schwarz domain decomposition preconditioner based on xpetra in trilinos. In: SPRINGER. *International Conference on Domain Decomposition Methods*. [S.l.], 2018. p. 176–184. Cited on page 101.

HIPTMAIR, R.; TOSELLI, A. Overlapping and multilevel schwarz methods for vector valued elliptic problems in three dimensions. In: *Parallel solution of partial differential equations*. [S.l.]: Springer, 2000. p. 181–208. Cited on page 100.

IHLENBURG, F.; BABUŠKA, I. Finite element solution of the helmholtz equation with high wave number part i: The h-version of the fem. *Computers & Mathematics with Applications*, Elsevier, v. 30, n. 9, p. 9–37, 1995. Cited on page 14.

IHLENBURG, F.; BABUSKA, I. Finite element solution of the helmholtz equation with high wave number part ii: the hp version of the fem. *SIAM Journal on Numerical Analysis*, SIAM, v. 34, n. 1, p. 315–358, 1997. Cited on pages: 14 and 43.

IOANNIDIS, A. D.; KRISTENSSON, G.; STRATIS, I. G. On the well-posedness of the maxwell system for linear bianisotropic media. *SIAM Journal on Mathematical Analysis*, SIAM, v. 44, n. 4, p. 2459–2473, 2012. Cited on page 24.

JIN, J.-M. *The finite element method in electromagnetics*. [S.l.]: John Wiley & Sons, 2015. Cited on pages: 16, 25, and 102.

JOHNSON, S. G. Notes on perfectly matched layers (pmls). *Lecture notes, Massachusetts Institute of Technology, Massachusetts*, v. 29, 2008. Cited on page 26.

JOLIVET, P. *Méthodes de décomposition de domaine. Application au calcul haute performance*. Tese (Doutorado) — Grenoble, 2014. Cited on page 88.

JOLIVET, P. et al. High performance domain decomposition methods on massively parallel architectures with freefem++. *Journal of Numerical Mathematics*, De Gruyter, v. 20, n. 3-4, p. 287–302, 2012. Cited on page 86.

KARYPIS, G. Metis and parmetis. *Encyclopedia of parallel computing*, Springer, p. 1117–1124, 2011. Cited on pages: 54 and 80.

KIMN, J.-H.; SARKIS, M. Restricted overlapping balancing domain decomposition methods and restricted coarse problems for the helmholtz problem. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 196, n. 8, p. 1507–1514, 2007. Cited on pages: 92 and 94.

KIMN, J.-H.; SARKIS, M. Shifted laplacian ras solvers for the helmholtz equation. In: *Domain decomposition methods in science and engineering XX*. [S.l.]: Springer, 2013. p. 151–158. Cited on page 90.

KLAWONN, A.; RHEINBACH, O.; WIDLUND, O. B. An analysis of a feti–dp algorithm on irregular subdomains in the plane. *SIAM Journal on Numerical Analysis*, SIAM, v. 46, n. 5, p. 2484–2504, 2008. Cited on page 54.

LEONG, A. *Extension of Two-level Schwarz Preconditioners to Symmetric Indefinite Problems TR2008-914*. Tese (Doutorado) — New York University, 2008. Cited on page 92.

LIONS, P.-L. On the schwarz alternating method. iii: a variant for nonoverlapping subdomains. In: SIAM PHILADELPHIA, PA. *Third international symposium on domain decomposition methods for partial differential equations*. [S.l.], 1990. v. 6, p. 202–223. Cited on page 56.

LIU, J.; JIN, J.-M. Scattering analysis of a large body with deep cavities. *IEEE Transactions on Antennas and Propagation*, IEEE, v. 51, n. 6, p. 1157–1167, 2003. Cited on page 114.

LIU, V.; MILLER, D. A.; FAN, S. Highly tailored computational electromagnetics methods for nanophotonic design and discovery. *Proceedings of the IEEE*, IEEE, v. 101, n. 2, p. 484–493, 2013. Cited on page 14.

LIVSHITS, I. A scalable multigrid method for solving indefinite helmholtz equations with constant wave numbers. *Numerical Linear Algebra with Applications*, Wiley Online Library, v. 21, n. 2, p. 177–193, 2014. Cited on page 47.

LOGG, A.; MARDAL, K.-A.; WELLS, G. *Automated solution of differential equations by the finite element method: The FEniCS book*. [S.l.]: Springer Science & Business Media, 2012. v. 84. Cited on pages: 26, 29, and 103.

LOGG, A. et al. *Automated Solution of Differential Equations by the Finite Element Method*. [S.l.]: Springer, 2012. ISBN 978-3-642-23098-1. Cited on page 107.

LOGG, A. et al. Ffc: the fenics form compiler. In: *Automated Solution of Differential Equations by the Finite Element Method*. [S.l.]: Springer, 2012. p. 227–238. Cited on page 48.

MEURER, A. et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, PeerJ Inc., v. 3, p. e103, 2017. Cited on page 79.

MEYERHENKE, H.; SANDERS, P.; SCHULZ, C. Parallel graph partitioning for complex networks. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 28, n. 9, p. 2625–2638, 2017. Cited on page 54.

MODAVE, A. et al. An optimized schwarz domain decomposition method with cross-point treatment for time-harmonic acoustic scattering. *HAL-02432422*, 2020. Cited on page 54.

MOIOLA, A.; SPENCE, E. A. Is the helmholtz equation really sign-indefinite? *siam REVIEW*, SIAM, v. 56, n. 2, p. 274–312, 2014. Cited on page 14.

MONK, P. et al. *Finite element methods for Maxwell's equations*. [S.l.]: Oxford University Press, 2003. Cited on pages: 25, 38, 43, and 102.

NATAF, F. *Absorbing boundary conditions and perfectly matched layers in wave propagation problems*. [S.l.]: de Gruyter, 2013. Cited on page 26.

NATAF, F.; XIANG, H.; DOLEAN, V. A two level domain decomposition preconditioner based on local dirichlet-to-neumann maps. *Comptes Rendus Mathematique*, Elsevier, v. 348, n. 21-22, p. 1163–1167, 2010. Cited on pages: 88, 89, and 94.

NATAF, F. et al. A coarse space construction based on local dirichlet-to-neumann maps. *SIAM Journal on Scientific Computing*, SIAM, v. 33, n. 4, p. 1623–1642, 2011. Cited on pages: 88 and 89.

NÉDÉLEC, J.-C. Mixed finite elements in r 3. *Numerische Mathematik*, Springer, v. 35, n. 3, p. 315–341, 1980. Cited on pages: 27, 39, and 103.

NÉDÉLEC, J.-C. A new family of mixed finite elements in r 3. *Numerische Mathematik*, Springer, v. 50, n. 1, p. 57–81, 1986. Cited on pages: 39 and 103.

NÉDÉLEC, J.-C. *Acoustic and electromagnetic equations: integral representations for harmonic problems*. [S.l.]: Springer Science & Business Media, 2001. v. 144. Cited on page 24.

NICOLAIDES, R. A. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, SIAM, v. 24, n. 2, p. 355–365, 1987. Cited on page 86.

NIKOLOVA, N. K. *Introduction to microwave imaging*. [S.l.]: Cambridge University Press, 2017. Cited on page 23.

PECHSTEIN, C.; SCHEICHL, R. Scaling up through domain decomposition. *Applicable Analysis*, Taylor & Francis, v. 88, n. 10-11, p. 1589–1608, 2009. Cited on page 87.

PELLEGRINI, F. Scotch and pt-scotch graph partitioning software: an overview. *Combinatorial Scientific Computing*, Chapman and Hall/CRC, p. 373–406, 2012. Cited on page 54.

PENG, Z.; LEE, J.-F. Non-conformal domain decomposition method with second-order transmission conditions for time-harmonic electromagnetics. *Journal of Computational Physics*, Elsevier, v. 229, n. 16, p. 5615–5629, 2010. Cited on pages: 76 and 77.

PENG, Z.; LEE, J.-F. Non-conformal domain decomposition method with mixed true second order transmission condition for solving large finite antenna arrays. *IEEE Transactions on Antennas and Propagation*, IEEE, v. 59, n. 5, p. 1638–1651, 2011. Cited on page 76.

PROJECT, T. F. *Basix*. [S.l.]: GitHub, 2021. <https://github.com/FEniCS/basix>. Cited on pages: 35 and 48.

QU, Y.; FISH, J. Global-basis two-level method for indefinite systems. part 2: computational issues. *International Journal for Numerical Methods in Engineering*, Wiley Online Library, v. 49, n. 3, p. 461–478, 2000. Cited on page 86.

QUARTERONI, A.; VALLI, A. *Domain decomposition methods for partial differential equations*. [S.l.]: Oxford University Press, 1999. Cited on page 55.

RAWAT, V.; LEE, J.-F. Nonoverlapping domain decomposition with second order transmission condition for the time-harmonic maxwell's equations. *SIAM Journal on Scientific Computing*, SIAM, v. 32, n. 6, p. 3584–3603, 2010. Cited on page 77.

RICHARDSON, C. N.; SIME, N.; WELLS, G. N. Scalable computation of thermomechanical turbomachinery problems. *Finite Elements in Analysis and Design*, Elsevier, v. 155, p. 32–42, 2019. Cited on page 45.

ROGNES, M. E.; KIRBY, R. C.; LOGG, A. Efficient assembly of h(div) and h(curl) conforming finite elements. *SIAM Journal on Scientific Computing*, SIAM, v. 31, n. 6, p. 4130–4151, 2009. Cited on page 105.

ROTHBERG, E.; GUPTA, A. An evaluation of left-looking, right-looking and multifrontal approaches to sparse cholesky factorization on hierarchical-memory machines. *International Journal of High Speed Computing*, World Scientific, v. 5, n. 04, p. 537–593, 1993. Cited on page 45.

SAAD, Y. *Iterative methods for sparse linear systems*. [S.l.]: siam, 2003. v. 82. Cited on pages: 45 and 46.

SAAD, Y.; SCHULTZ, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, SIAM, v. 7, n. 3, p. 856–869, 1986. Cited on page 46.

SCROGGS, M. W. et al. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *arXiv preprint arXiv:2102.11901*, 2021. Cited on pages: 42 and 105.

SHAMPINE, L. F. *Numerical solution of ordinary differential equations*. [S.l.]: Routledge, 2018. Cited on page 77.

SHEWCHUK, J. R. et al. *An introduction to the conjugate gradient method without the agonizing pain*. [S.l.]: Carnegie-Mellon University. Department of Computer Science, 1994. Cited on page 46.

SILVESTER, P. Finite element solution of homogeneous waveguide problems. *Alta Frequenza*, N. Speciale, v. 38, n. 1, p. 313–317, 1969. Cited on page 27.

SMITH, B.; BJORSTAD, P.; GROPP, W. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. [S.l.]: Cambridge university press, 2004. Cited on pages: 43 and 55.

SOLIN, P.; SEGETH, K.; DOLEZEL, I. *Higher-order finite element methods*. [S.l.]: CRC Press, 2003. Cited on page 34.

ST-CYR, A.; GANDER, M. J.; THOMAS, S. J. Optimized multiplicative, additive, and restricted additive schwarz preconditioning. *SIAM Journal on Scientific Computing*, SIAM, v. 29, n. 6, p. 2402–2425, 2007. Cited on pages: 57 and 59.

STOLK, C. C. A rapidly converging domain decomposition method for the helmholtz equation. *Journal of Computational Physics*, Elsevier, v. 241, p. 240–252, 2013. Cited on pages: 68 and 89.

STOLK, C. C. An improved sweeping domain decomposition preconditioner for the helmholtz equation. *Advances in Computational Mathematics*, Springer, v. 43, n. 1, p. 45–76, 2017. Cited on pages: 68 and 89.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer, v. 11, n. 4, p. 341–359, 1997. Cited on page 72.

STRANG, G. *Computational science and engineering*. [S.l.]: Wellesley-Cambridge Press Wellesley, 2007. v. 791. Cited on pages: 43, 45, and 47.

STROHMAIER, E. et al. The top500 list and progress in high-performance computing. *Computer*, IEEE, v. 48, n. 11, p. 42–49, 2015. Cited on page 17.

TAFLOVE, A.; HAGNESS, S. C. *Computational electrodynamics: the finite-difference time-domain method*. [S.l.]: Artech house, 2005. Cited on page 16.

TANG, J. M. et al. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *Journal of scientific computing*, Springer, v. 39, n. 3, p. 340–370, 2009. Cited on page 86.

TEZAUR, R.; FARHAT, C. Three-dimensional discontinuous galerkin elements with plane waves and lagrange multipliers for the solution of mid-frequency helmholtz problems. *International journal for numerical methods in engineering*, Wiley Online Library, v. 66, n. 5, p. 796–815, 2006. Cited on page 92.

TOSELLI, A. Overlapping schwarz methods for maxwell's equations in three dimensions. *Numerische Mathematik*, Springer, v. 86, n. 4, p. 733–752, 2000. Cited on page 100.

TOSELLI, A.; WIDLUND, O. *Domain decomposition methods-algorithms and theory*. [S.l.]: Springer Science & Business Media, 2006. v. 34. Cited on pages: 54 and 55.

TROTTENBERG, U.; OOSTERLEE, C. W.; SCHULLER, A. *Multigrid*. [S.l.]: Elsevier, 2000. Cited on page 42.

TSUJI, P.; ENGQUIST, B.; YING, L. A sweeping preconditioner for time-harmonic maxwell's equations with finite elements. *Journal of Computational Physics*, Elsevier, v. 231, n. 9, p. 3770–3783, 2012. Cited on page 100.

TURKEL, E. Boundary conditions and iterative schemes for the helmholtz equation in unbounded regions. *Computational Methods for Acoustics Problems*, Saxe-Coburg Publication UK, p. 127–158, 2008. Cited on page 25.

VION, A.; GEUZAINE, C. Double sweep preconditioner for optimized schwarz methods applied to the helmholtz problem. *Journal of Computational Physics*, Elsevier, v. 266, p. 171–190, 2014. Cited on pages: 26, 68, 75, 85, and 89.

VION, A.; GEUZAINE, C. Improved sweeping preconditioners for domain decomposition algorithms applied to time-harmonic helmholtz and maxwell problems. *ESAIM: Proceedings and Surveys*, EDP Sciences, v. 61, p. 93–111, 2018. Cited on page 100.

VUIK, C.; NABBEN, R.; TANG, J. Deflation acceleration for domain decomposition preconditioners. In: *Proceedings of the 8th European Multigrid Conference on Multigrid, Multilevel and Multiscale Methods, The Hague, The Netherlands*. [S.l.: s.n.], 2005. Cited on page 86.

WANG, W.-J. et al. Massively parallel simulation of large-scale electromagnetic problems using one high-performance computing scheme and domain decomposition method. *IEEE Transactions on Electromagnetic Compatibility*, IEEE, 2017. Cited on page 14.

WEBB, J. Application of the finite-element method to electromagnetic and electrical topics. *Reports on Progress in Physics*, IOP Publishing, v. 58, n. 12, p. 1673, 1995. Cited on page 27.

Appendix

APPENDIX A – Complex Number support in FEniCSx

A.1 Project description

This report contains an overview of the work done for the Google Summer of Code 2018. The goal of this project was to open up the possibility of the solution of large-scale complex-valued PDEs using FEniCS. To achieve this goal, contributions of varying degrees we made on some FEniCSx components, namely DOLFINx, FFCx, and UFL. A chronological list (oldest first) of the Pull Requests (PR) created during the project is provided below. A summary will accompany each PR, but more information can be found on the respective links. The full report can be found at the following link:

<https://summerofcode.withgoogle.com/archive/2018/projects/5117785856278528>

A.2 Pull Requests List

A.2.1 DOLFINx

- PR #72: Make `la::PETScMatrix` and `la::PETScVector` compatible with PETSc in complex mode - first Pull Request created during the GSoC. The purpose of this PR was to make the classes `la::PETScMatrix` and `la::PETScVector` from the linear algebra (`la`) module compatible with PETSc compiled with complex number support.

- PR #81: Add some complex compatibility changes - This PR can be considered an extension of the previous one, with modifications on the class `VectorSpaceBasis` and addition of the MPI definition for complex double (`MPI_DOUBLE_COMPLEX`).

- PR #87: Add complex support to `la::SLEPcEigenSolver`: Although SLEPc separates the real and imaginary part of eigenvalues and eigenvectors, when PETSc is compiled with complex support this behavior changes. In this PR, changes were made to create a common interface between real and complex modes.

- PR #93: Update python `la` module to handle complex numbers - With the previous three PRs, the linear algebra module, within the C ++ layer, was already compatible with complex PETSc. In this PR, the `la::` python interface was also updated to handle complex numbers.

- PR #96: Define `ufc_scalar_t` for a complex UFC interface - This PR should be considered together with FFCx PR #31 (more information on the FFCx section).
- PR #104: Update the function module to handle complex numbers - The changes that have been made in this PR allow the creation of complex functions (functions, expressions and constants). In addition, the literal `j`, representing the imaginary unit, was defined to facilitate the creation of complex expressions using C ++ syntax.
- PR #112: Reorder complex values - The purpose of this PR is to allow the reordering of complex values according to explicit global indices. Later, these changes were used to represent complex values in XDMF.
- PR #120: Fix some tests and demos to work in both real and complex modes - Basically, the changes of this PR included the exchange of the operator `*` with `inner` in the forms expressed in the tests, to reflect the changes in UFL, since the emergence of complex support. Since now the operator `inner` takes the conjugate of the second term and and the former does not.
- PR #127: More changes in demos and tests to work in complex mode - This PR can be considered a sequence of the previous one, with the addition of some demos.
- PR #128: Update assembler to handle complex-valued equations - In this PR, the goal was to allow assembling of complex-valued forms and boundary conditions. A set of unit tests was also added.
- PR #134: Add Complex support to `XDMFFile` - This PR added support for writing and reading complex functions using the class `XDMFFile`. The real and imaginary components are split into two different data-sets to allow visualization using some visualization application, such as `paraview`.
- PR #141: Update XDMF tests to cover complex mode - The aim, in this PR, was to increase the coverage of the current XDMF tests to handle the complex case.
- PR #144: Get DOLFINx to compile with PETSc complex (remaining changes) - This PR involved the remaining modifications to get DOLFINx to compile with PETSc in complex mode, in preparation to run nightly tests for the complex mode on master branch.
- PR #146: Set nightly builds for complex mode - To test all changes to the code base considering the complex mode, `CircleCI` nightly builds were set. The complex build, was configured to run every day at 05:00 am UTC.
- PR #149: Update the scalar type according to the mode (real or complex) - In this PR, the scalar type parameter, to be passed to the form compiler, was update according to the mode. For example, if PETSc is compiled with complex support we pass a complex scalar to the form compiler.

- PR #153: Set nightly build branch to main - This is a "one-line" PR with small fix to set the nightly build branch to run only on the main branch.

A.2.2 FFCx: The FEniCSx Form Compiler

- PR #26: Add `#include<complex.h>` to the include list - This PR was closed. And its idea of including suitable headers for the complex mode in the generated code was incorporated on PR #31.
- PR #31: Create interface for complex scalars - the aim of this PR was to create a more flexible interface for the generated code, by using the `ufc_scalar_t` datatype that could be real or complex depending on the mode. This should apply to `tabulate_tensor` and `transform_values` functions. Also, the interface with the TSFC representation was updated, allowing the generation of code from complex-valued forms.
- PR #51: Add `scalar_type` to FFCx default parameters - This PR adds the `scalar_type` parameter to the list of FFCx default parameters. By default `scalar_type` is set to `double`.

A.2.3 UFL: Unified Form Language

- PR #95: Extend UFL to the complex domain - Fundamentally, the changes that allow the extension to complex values came from the Firedrake/UFL branch, with minor fixes to make it compatible with DOLFINx, and some improvements on the documentation.

APPENDIX B – Mesh Partitioning Improvements in DOLFINx

This report provides an overview of the work done for the Google Summer of Code 2019. The main goal of this project is to add the KaHIP partitioner to DOLFINx's graph wrappers and mesh partitioning and investigate whether the expected improvements obtained by KaHIP for mesh partitioning would reflect on DOLFINx's parallel tool-chain. This is related to Issue #116.

The second goal of this project is to add support for partitioning using a subset of processors. As currently implemented in DOLFIN-X, the mesh partitioners use all the available MPI processes to perform partition. Numerical experiments show that this can demand high memory usage (partitioning packages depend on the number of processes). Also, the running time increases significantly with the number of processes for a fixed size local mesh/graph per processor. This is related to Issue #9.

In the following sections, we will list the contributions and results demonstrating the improvements achieved during this project. The full report can be found at the following link:

<https://summerofcode.withgoogle.com/archive/2018/projects/5117785856278528>

B.1 KaHIP Partitioner

To test the parallel KaHIP partitioner, contributions were made to different open-source repositories.

B.1.1 List of Pull Requests and Commits - KaHIP repository

- PR #33: Use python3 print format in Scons - These simple modifications allow the use of the compile.sh script in systems that have only python3.
- PR #37: Add modified kahip lib to deploy/parallel - The aim of this PR is to simplify linking the parallel interface using cmake.
- Commit ad07d2d: bug fix in interface due to @IgorBaratta - This commit aims to fix the internal graph building process of KaHIP, more specifically, it sets the correct number of edges.

B.1.2 List of Pull Requests and Commits - DOLFINx repository

- PR #451: Solves Issue #116.
  - Create `FindKaHIP.cmake` file to link KaHIP and add a small test for interface in serial;
  - Define KaHIP as an optional package;
  - Create and interface for KaHIP following the code available for ParMETIS and PT-SCOTCH;
  - Add option to use the new interface in 'Partitioning.h';
  - Add some tests using the C++ interface.

B.1.3 Numerical Experiments

To test the interface and the performance of the new parallel partitioner, we performed some numerical experiments to evaluate its weak scalability. The run-time results were compared with those obtained using PT-SCOTCH, considered here as the benchmark.

These numerical experiments were performed using resources provided by the Cambridge Service for Data-Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk).

We consider the total mesh building time, which consists of three well-defined steps:

- **Read** - Read local mesh data (points and cells) from an XDMF file with HDF5 encoding.
- **Partition** - Build the distributed dual graph (cell-cell connections) from local mesh data; partition the mesh using one of the supported libraries (KaHIP or PT-SCOTCH), and perform halo exchange of cell partition data.
- **Distribute** - Distribute mesh from a set of points and cells on each local process with the precomputed partition data.

For this numerical experiment, the number of cells per processor is fixed at 100,000. Moreover, the number of processors increases from 128 to 1024. The mesh generation was performed in a pre-processing step and is not considered for run-time measurement purposes. The first tests were performed without the corrections on the KaHIP parallel interface, and the results are shown in Figure 46. After some minor bug fixes, we repeated the numerical experiments, and the results changed dramatically, as shown in Figure 47. Although the partition step still dominates the total time, the KaHIP partitioner becomes more scalable. Up to 1024 processors, the partitioning step using KaHIP behaved almost independently to the number of processors. This is a promising result; however, tests with higher core counts are still necessary for more definitive conclusions.

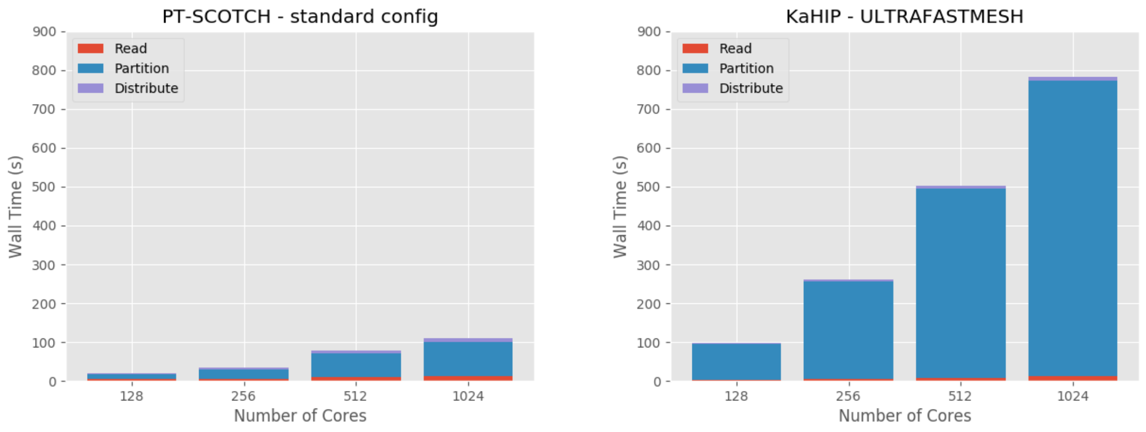


Figure 46 – Mesh creation time using two different mesh petitioners. Before modifications on KaHIP.

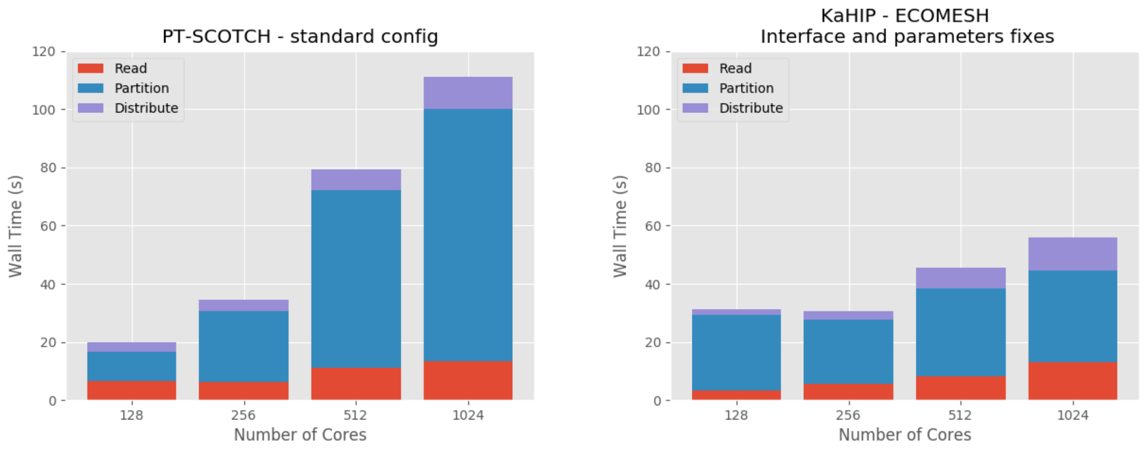


Figure 47 – Mesh creation time using two different mesh petitioners. After modifications on KaHIP.

B.2 Support partitioning on a subset of processors

B.2.1 List of Pull Requests - DOLFINx repository

- PR #459: This PR is a proposal for closing Issue #9, and related to Issue #358. This PR entail the following tasks:
  - Create `MPI::SubsetComm`, to extract a new communicator with only a subset of processes.
  - (XDMFFile) Separate read mesh data and mesh creation.
  - Move `Partitioning::partition_cells` to the public interface, and allow partitioning with a different number of parts and processes.
  - Provide a fine-grained control of mesh partition (different communicator for reading mesh data and partitioning, selection of partitioner), a small example is provided below.

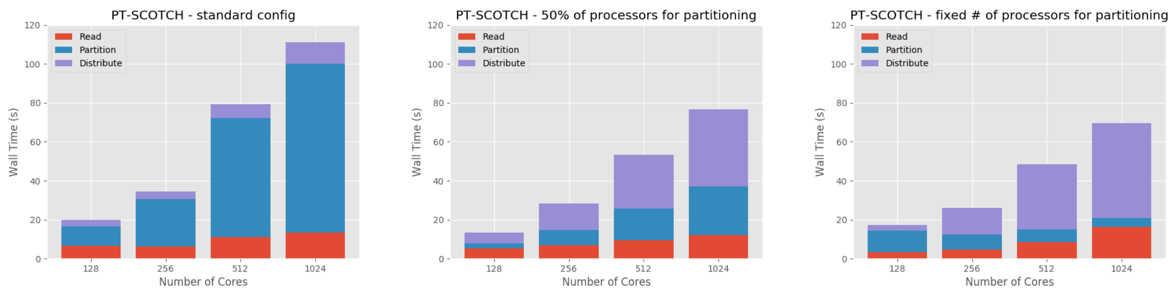


Figure 48 – Mesh creation time using a subset of available processes to partition the mesh.

- Add new c++ test for distributed meshes (generate, write, read).

B.2.2 Numerical Experiments

Again the wall-time considering the whole distributed mesh building process is measured for an increasing number of processors, and it is used as a performance metric. However, instead of using different partitioners, we compare the parallel performance using different numbers of processors for partitioning, and then we distribute the correspondent mesh data for each process of the primary communicator. We consider here three different settings:

- Use all available processes of the primary MPI communicator (the only option that was available on DOLFINx before PR).
- Use a fixed number of processes, fixed to 128 in the current numerical experiment.

It can be noted from this preliminary numerical experiment, that the total mesh building time strongly depends on the number of processors used in the mesh partitioning step. Using a smaller processor subset leads to a shorter total running time; however, the time of the distribution step (Distribute) is affected adversely.