Domain decomposition and multi-scale numerical methods for the modelling of urban floods

Miranda Boutilier

UNIVERSITÉ CÔTE D'AZUR | ÉCOLE DOCTORALE SCIENCES FONDAMENTALE ET APPLIQUÉES

THÈSE DE DOCTORAT

DÉCOMPOSITION DE DOMAINE ET MÉTHODES NUMÉRIQUES MULTI-ÉCHELLES POUR LA MODÉLISATION DES CRUES URBAINES

Miranda BOUTILIER

Université Côte d'Azur, Laboratoire J.A. Dieudonné (LJAD)

Présentée en vue de l'obtention du grade de docteur en Mathématiques d'Université Côte d'Azur

Dirigée par: Victorita Dolean-Maini / Konstantin Brenner

Soutenue le: 04 Octobre, 2024

Devant le jury, composé de:

Florence Hubert, Professeure, Université Aix Marseille

Roland Masson, Professeur, Université Côte d'Azur

Frédéric Valentin, Senior Researcher, LNCC

Martin Vohralik , Directeur de Recherche, INRIA

UNIVERSITÉ CÔTE D'AZUR

Décomposition de domaine et méthodes numériques multi-échelles pour la modélisation des crues urbaines

---

Domain decomposition and multi-scale numerical methods for the modelling of urban floods

Jury

Rapporteurs

Florence Hubert	Professeure, Université Aix Marseille
Frédéric Valentin	Senior Researcher, LNCC

Examinateurs

Roland Masson	Professeur, Université Côte d'Azur
Martin Vohralik	Directeur de Recherche, INRIA

Directrice de thèse

Victorita Dolean-Maini	Professeure, Eindhoven University of Technology

Codirecteur de thèse

Konstantin Brenner	Maître de conférences, Université Côte d'Azur

Abstract. The work of this thesis is dedicated to the simulation and numerical analysis of urban flood problems. While urban flooding caused by exceptional rainfall is particularly devastating in terms of economic and human damage, numerical modeling can be used to predict, anticipate and control such events. From the numerical perspective, the major challenge comes from a large contrast between a typical size of the simulation domain (10-100km) and the size of the relevant structural features (such as buildings or walls), which have to be represented at metric or infra-metric scales. This thesis addresses the multi-scale character of the urban flows by means of Domain Decomposition (DD) and Multi-scale (Ms) numerical methods.

The first part of the thesis focuses on linear diffusion problems posed in domains containing numerous polygonal perforations representing realistic structures in urban areas. We propose a low-dimensional coarse approximation space based on a coarse polygonal partitioning of the domain. The main theoretical contribution of this part is an error estimate regarding the H^1-projection over the coarse space; this error estimate is independent of the global regularity of the solution, which is expected to be low due to multiple corner singularities. Additionally, this part numerically explores the combination of the coarse space with overlapping Schwarz domain decomposition methods. This combination leads to an efficient two-level iterative linear solver and preconditioner for a Krylov method.

The second part of the thesis extends our methodology to nonlinear urban flow models. That is, we design DD and Ms methods to numerically solve the Diffusive Wave equation, which is obtained from Shallow Water systems by neglecting inertia terms. We show that the two-level preconditioner previously designed for linear diffusion problems performs well on the linearized Diffusive Wave model which arises at each iteration of Newton's method. Furthermore, we present nonlinear preconditioning techniques, including one and two-level RASPEN, which significantly reduce iteration counts when compared to Newton's method. We also propose the use of the Trefftz space for the coarse approximation of the numerical solution. Numerical experiments are conducted, with the main example being the numerical solution of the Diffusive Wave equation on a large urban area of Nice, France.

In the last part of the thesis, for nonlinear elliptic PDEs, we investigate a multi-scale method that combines tools from the classical Multi-scale Finite Element Method and Machine Learning. Our approach is based on the approximate substructured formulation in which the traces of the unknown function belong to a coarse finite element space. The substructured problem is solved by Newton's method, using local Dirichlet-to-Neumann (DtN) operators at each iteration. In order to reduce the computational cost associated with the evaluation of DtN operators, the latter are replaced by approximate models built on the basis of artificial neural networks. Numerical experiments on nonlinear p-Laplace and degenerate diffusion problems in 1D and 2D show promising results. With only a few training points per dimension of the DtN operator's domain, the approximate model achieves an accuracy of a few percent.

Keywords: Multi-scale methods, domain decomposition, nonlinear preconditioning, urban floods.

Résumé. Les travaux de cette thèse sont consacrés à la simulation et à l'analyse numérique des problèmes d'inondations urbaines. Les inondations urbaines provoquées par des précipitations exceptionnelles sont particulièrement dévastatrices en termes de dégâts économiques et humains. La modélisation numérique peut être utilisée pour prédire, anticiper et contrôler de tels événements. Du point de vue numérique, le défi majeur réside dans le grand contraste entre la taille typique du domaine de simulation (10-100 km) et la taille caractéristique des éléments structurels pertinents (tels que les bâtiments ou les murets), qui doivent être représentés à des échelles métriques ou inframétriques. Cette thèse aborde le caractère multi-échelle des inondations urbaines en mobilisant la décomposition de domaine (DD) et les méthodes numériques multi-échelles (Ms).

La première partie de la thèse se concentre sur un problème de diffusion linéaire posé dans des domaines contenant un grand nombre de perforations polygonales représentant des structures réalistes dans les zones urbaines. Nous proposons un espace d'approximation grossier de faible dimension basé sur un partitionnement polygonal grossier du domaine. La principale contribution théorique de cette section est une estimation de l'erreur concernant la projection H^1 de la solution sur l'espace grossier. L'analyse d'erreur est indépendante de la régularité globale de la solution, ce qui est un atout majeur compte tenu des singularités géométriques du domaine. En combinant la méthode de Schwarz avec recouvrement et la correction de l'espace grossier à la base de la méthode multi-échelle proposée, on parvient à un solveur itératif et un préconditionneur efficaces, dont les performances sont étudiées numériquement.

La deuxième partie de la thèse étend notre méthodologie numérique aux modèles d'écoulement non linéaires. En particulier, nous nous intéressons à l'équation de l'onde diffusive, obtenue à partir des équations de Barré de Saint-Venant en négligeant les termes d'inertie. Nous montrons que le préconditionneur à deux niveaux que nous avons conçu pour les problèmes de diffusion linéaire se prête également au problème de l'onde diffusive linéarisé qu'on obtient à chaque itération de la méthode de Newton. En outre, nous présentons des techniques de préconditionnement non linéaires, y compris la méthode RASPEN à un et deux niveaux, ce qui permet de réduire considérablement le nombre d'itérations par rapport à la méthode de Newton traditionnelle. Les exemples numériques illustrant les performances des méthodes proposées comprennent un cas test basé sur des données topographiques de la ville de Nice.

Dans la dernière partie, pour les EDP elliptiques non linéaires, nous étudions une méthode multi-échelle qui combine des outils de la méthode d'éléments finis multi-échelles classique avec ceux de l'apprentissage automatique. Notre approche repose sur la formulation sous-structurée approchée dans laquelle les traces de la fonction inconnue sont recherchées dans un espace d'éléments finis grossier. Le problème sous-structuré est résolu par la méthode de Newton, faisant appel à chaque itération à des opérateurs Dirichlet-to-Neumann (DtN) locaux. Dans le but de réduire le coût de calcul associé à l'évaluation des opérateurs DtN, ces derniers sont remplacés par des modèles approchés construits sur la base des réseaux de neurones artificiels. Les expériences numériques concernant les problèmes non linéaires de p-Laplace et de diffusion dégénérée en 1D et 2D aboutissent à des résultats prometteurs. Avec seulement quelques points d'entraînement par dimension du domaine des opérateurs DtN, le modèle approché atteint une précision de quelques pour cent.

Mots-clés : Méthodes multi-échelles décomposition de domaine, préconditionnement non linéaires, crues urbaines.

Remerciements

I am extremely grateful for the opportunity to complete this PhD thesis. This PhD has allowed me to grow in so many ways, and the knowledge I have gained as a researcher has been immeasurable. However, my growth has not been limited to my understanding of domain decomposition methods and urban flood modeling. The exposure to new cultures, people, and opportunities has been invaluable, and I will look back on this time of my life with intense gratitude.

I would like to first thank my advisors, Victorita Dolean and Konstantin Brenner. I thank Victorita for her consistent advice and support and for her commitment to advising my work even through a move to the Netherlands midway through my PhD. Through her busy schedule, she always made sure to give valuable feedback and comments on my work. Furthermore, I thank Konstantin for his unwavering support, both academically and otherwise. The constant ideas, guidance, and kindness are things I do not take for granted. I sincerely appreciate the help with my move and administrative bureaucracy, especially with my lack of French-speaking skills during the first parts of my PhD.

I offer my thanks to the members of my PhD examining committee: Frédéric Valentin, Florence Hubert, Martin Vohlarik, and Roland Masson for taking the time to review the thesis and participating in the defense. I would also like to thank Ronald Haynes for acting as my Masters' supervisor at Memorial University of Newfoundland and setting me on my current path.

I would also like to thank everyone at the Dieudonné laboratory at Université Côte d'Azur. To the administrative workers who have supported me through bureaucracy and the planning of various conference trips, and the members of faculty. As well to my fellow PhD and Post-doctoral researchers, to whom I wish good luck in their future endeavors. I would also like to thank the Agence Nationale de Recherche project Top-up for the funding throughout my PhD studies, without which the completion of this thesis would not have been possible. I am also grateful to the people I have met at various conferences and research schools, including the six-week CEMRACS hackathon at CIRM in Marseille.

On a personal note, I am eternally grateful for the friends I have made throughout my time in France. The expatriate community in Nice has allowed me to form friendships very dear to me, and I hope these friendships will continue when my time in Nice is over. I thank my family for supporting me from afar, and for understanding my choice to take this opportunity and be away from them for three years. Finally, to Jeremy, Maggie, and Molly- thank you for embarking on this journey with me.

Contents

Chapter 1

Introduction

In this thesis, we focus on the solution of linear and nonlinear partial differential equations (PDEs) posed on complex domains, which generally contain numerous polygonal perforations within them. The main real-world application we are concerned with involves the modeling of urban floods. Numerical modeling of overland flows plays an increasingly important role in predicting, anticipating, and controlling floods. Anticipating these flood events can aid in the positioning of protective systems including dams, dikes, or rainwater drainage networks. One of the challenges of this numerical modeling of urban floods is that small structural features (buildings, walls, etc.) may significantly affect the water flow [2, 110, 111].

Modern terrain survey techniques including photogrammetry and Laser Imaging, Detection, and Ranging (LIDAR) allow for the acquisition of high-resolution topographic data for urban areas. The data set used in this thesis has been provided by Métropole Nice Côte d'Azur (MNCA) and enables an infra-metric description of the urban geometries [4]. From the hydraulic perspective, these structural features can be assumed to be essentially impervious, and therefore represented as perforations (holes) in the model domain. We work within the assumption that the "buildings as holes" representation of the structures is relevant in many flood scenarios [42, 95]. However, we recognize that this approach is somewhat limited as it does not account for some flood processes such as flow entering the building. We remark, however, that the methods described in this thesis are not limited to perforated domains; for example, these techniques can be applied to buildings represented as low-permeability regions [95]. An example of the domains on which we work are given in Figure 1.1 for various data frames representing areas of Nice, France.

Depending on the geometrical complexity of the computational domain, the numerical solution of these models may become increasingly challenging to compute, as the typical model domain resulting from the realistic description of the urban environment will contain numerous perforations that are represented on different scales. Taking into account these structures can result in extremely small computational elements. Thus, in order to address the scale contrast, we wish to employ numerical strategies such as Multi-scale (Ms) and Domain Decomposition (DD) methods. Both Ms and DD methods presented in this work consider two levels of space discretization, referred to as the "fine" and "coarse" levels. This dual partitioning aims to achieve computational efficiency compared to classical fine-scale solution methods. Additionally, many of the proposed methods allow for parallel implementation to further improve efficiency. The first level of discretization, associated with a fine-scale triangular grid, is assumed to resolve all important small-scale features,

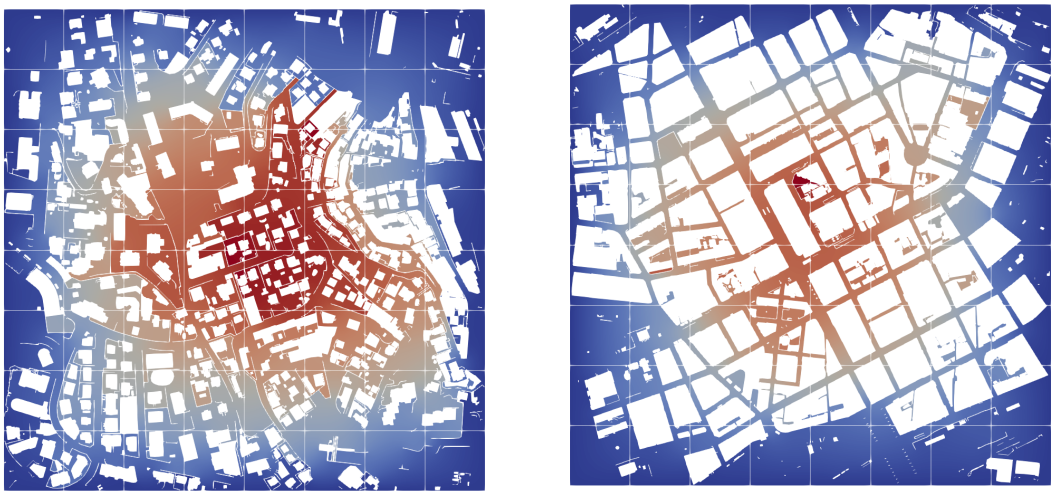


Figure 1.1: Example of computational domain divided in $N = 8 \times 8$ nonoverlapping subdomains.

but is likely to have many elements of highly variable size. The second (coarse) level of discretization is based on a coarse grid (or subdomain partitioning) having a characteristic element size much larger than that of the fine-scale partitioning. Both Ms and DD methods generally construct a coarse space associated with the coarse level of discretization. This coarse space plays the crucial role in both DD and Ms approaches. The hope is that by incorporating some fine-scale information, the coarse space can be constructed such that it approximates the fine-scale solution well.

With this, we introduce notations which will be used throughout the thesis. Let D be an open simply connected polygonal domain in $\mathbb{R}^2$. We denote by $(\Omega_{S,k})_k$ a finite family of perforations in D such that each $\Omega_{S,k}$ is an open connected polygonal subdomain of D. The perforations are mutually disjoint, that is $\overline{\Omega_{S,k}} \cap \overline{\Omega_{S,l}} = \emptyset$ for any $k \neq l$. We denote $\Omega_S = \bigcup_k \Omega_{S,k}$ and $\Omega = D \setminus \overline{\Omega_S}$, assuming that the family $(\Omega_{S,k})_k$ is such that Ω is connected. Note that the latter assumption implies that $\Omega_{S,k}$ are simply connected.

For illustration purposes, an example of the linear Poisson equation on such a perforated domain is given by

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{in} \quad \Omega, \\ \dfrac{\partial u}{\partial \mathbf{n}} & = & 0 & \text{on} \quad \partial\Omega \cap \partial\Omega_S, \\ u & = & g & \text{on} \quad \partial\Omega \setminus \partial\Omega_S, \end{array} \right. \tag{1.1}$$

where $\mathbf{n}$ is the outward normal to the boundary, g denotes Dirichlet boundary data, and $f \in L^2(\Omega)$.

1.1 Methodology

Domain decomposition methods are used throughout this thesis to efficiently solve model problems such as (1.1). For large linear systems, such as the systems focused

on in this thesis, one can generally use direct or iterative solvers. Direct solvers, while robust, have a large memory cost which can cause issues when the linear system becomes too large. On the other hand, while iterative methods are often more computationally efficient for large-scale problems and may require less memory, they may converge slowly or even fail to converge for poorly conditioned systems. For ill-conditioned problems, preconditioning can improve the convergence behavior of an iterative solver by improving the conditioning of a system. Here, we focus on domain decomposition (DD) preconditioners. The main idea behind DD methods is to break down a large problem domain into smaller subdomains, solve the problem on each subdomain independently, and glue the local solutions together to obtain a global solution. Typically, they solve the global problem iteratively, while the local solves are computed via a direct method. We remark as well that since the local problems are independent of each other, these methods naturally allow for parallel implementation.

Dual Partitionings are used throughout the thesis in the form of a "coarse" and "fine" level of discretization. We now detail the generation of both levels of discretization.

For the construction of the coarse level of discretization, consider a finite nonoverlapping polygonal partitioning of D into N subdomains, denoted by $(D_j)_{j=1,\dots,N}$. The induced nonoverlapping partitioning of Ω is denoted by $(\Omega_j)_{j=1,\dots,N}$ such that $\Omega_j = D_j \cap \Omega$. We refer to $(\Omega_j)_{j=1,\dots,N}$ as the coarse mesh over Ω. The fine triangulation is generated to be conforming with respect to the coarse mesh $(\Omega_j)_{j=1,\dots,N}$. The fine-scale Delaunay triangulation is generated using Triangle [97], a meshing tool which can mesh around given perforations. The coarse and fine levels of discretization for a perforated domain are shown in thick and thin blue lines, respectively in Figures 1.2 and 1.3. We observe that due to this coarse cell conforming mesh generation process, the fine-scale triangulation may vary slightly as we vary the number of subdomains. The stronger geometrical constraints due to a larger number of subdomains generally results in a higher number of fine-scale elements; for example, the left and right of Figure 1.2, representing the same flow domain, contain 6 718 and 11 228 triangles, respectively. Later, when performing numerical experiments, the consistency of the fine-scale triangulation across different numbers of coarse cells is enforced by generating a fine mesh that is conforming to the highest number of subdomains considered. For example, a fine mesh conforming to an 8×8 coarse grid will also conform to 2×2 and 4×4 coarse meshes. Figure 1.3 shows a matching background triangulation for different coarse meshes. We remark that a METIS partitioning [69] is a possible technique for the generation of coarse partitionings as well, but is not explored in this thesis.

The Diffusive Wave (DW) Equation is commonly used to model overland flows [91, 3]. While this thesis begins in Chapter 2 with a linear model problem for the design of a novel coarse space, this coarse space is then used for the DW equation in Chapter 3. The main real-world application of this work involves the efficient numerical solution of the DW model on an area of Nice, France in which there are many urban structures. We can think of the DW equation as a simplification of the 2D Shallow Water (SW) equations under zero-inertia assumptions; for this reason, the DWE equation is sometimes referred to as the Zero Inertia (ZI) equations [49]. For methodological purposes, we believe that the understanding of the multi-scale aspects of a simpler DW model has to be achieved before addressing the full system of SW equations.

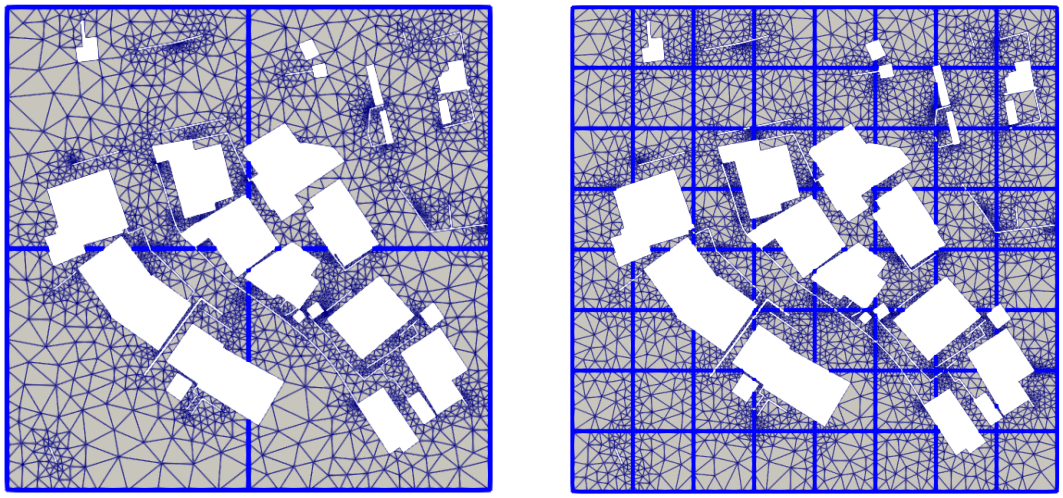


Figure 1.2: Coarse (thick blue lines) and fine (thin blue lines) discretizations for $N = 2 \times 2$ (left) and $N = 8 \times 8$ (right) subdomains for a given model domain.

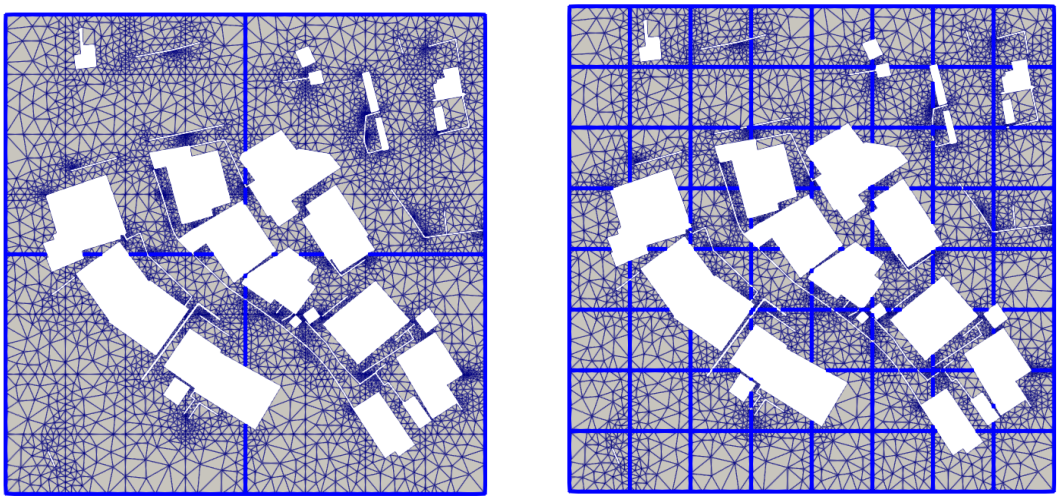


Figure 1.3: Coarse (thick blue lines) and fine (thin blue lines) discretization for $N = 2 \times 2$ (left) and $N = 8 \times 8$ (right) subdomains for a given model domain, with identical background fine-scale triangulations.

We now derive the DW equation from the 2D SW equations. Let $u(x, y)$ denote the total water surface elevation, $z_b(x, y)$ denote the underlying bathymetry, and $h(x, y) = u(x, y) - z_b(x, y)$ denote water depth. We consider the SW equations [104] with a zero source term in the mass conservation equation; in practice, a source term in the right-hand side of (1.2) would denote rainfall intensity. The SW equations are given by

$$\partial_t u + \partial_x q_x + \partial_y q_y = 0, \tag{1.2}$$

$$\partial_t q_x + \partial_y \left(\frac{q_x q_y}{h}\right) + \partial_x \left(\frac{q_x^2}{h} + \frac{1}{2} g h^2\right) = gh(S_{0x} - S_{fx}), \tag{1.3}$$

$$\partial_t q_y + \partial_x \left(\frac{q_x q_y}{h}\right) + \partial_y \left(\frac{q_y^2}{h} + \frac{1}{2} g h^2\right) = gh(S_{0y} - S_{fy}), \tag{1.4}$$

where $q_x = hv_x$ and $q_y = hv_y$ represent unit discharges and v_x, v_y denote the depth averaged x and y components of velocity. Additionally, $S_{0x} = -\partial_x z_b$ and $S_{0y} = -\partial_y z_b$ denote the respective slopes of the bathymetry $z_b(x, y)$. The friction slopes in both directions, denoted by S_{fx} and S_{fy}, are given by

$$S_{fx} = \frac{v_x \sqrt{v_x^2 + v_y^2}}{c_f^2 h^\beta} \qquad \text{and} \qquad S_{fy} = \frac{v_y \sqrt{v_x^2 + v_y^2}}{c_f^2 h^\beta}. \tag{1.5}$$

for some friction coefficient c_f. The choice of β in (1.5) can vary; common choices include $\beta = \frac{4}{3}$ for Manning friction and $\beta = 1$ for Chèzy friction [3], where the friction coefficients are chosen accordingly. Both choices are used to model friction in open channel flow, but for simplicity, we choose $\beta = 1$ corresponding to Chèzy friction throughout this thesis.

The Zero-Inertia model neglects the acceleration terms in the SW model; that is, the time derivatives of momentum components $\partial_t q_x, \partial_t q_y$ as well as the acceleration terms $\partial_x \left(\frac{q_x q_y}{h}\right)$, $\partial_y \left(\frac{q_x q_y}{h}\right)$, $\partial_x \left(\frac{q_x^2}{h}\right)$, and $\partial_y \left(\frac{q_y^2}{h}\right)$ in (1.2) – (1.4) are set to zero.

With this, (1.3) and (1.4) are modified such that the new system becomes

$$\begin{aligned} \partial_t u + \partial_x q_x + \partial_y q_y &= 0 \\ \partial_x h &= S_{0x} - S_{fx}, \qquad (1.6) \\ \partial_y h &= S_{0y} - S_{fy}. \qquad (1.7) \end{aligned}$$

From the definition of S_{0x} and S_{0y} combined with the fact that $h = u - z_b$, (1.6) and (1.7) result in

$$-\partial_x u = S_{fx} \qquad \text{and} \qquad -\partial_y u = S_{fy}. \tag{1.8}$$

Combining (1.5) with (1.8) and setting $\beta = 1$, we obtain

$$-\partial_x u = \frac{v_x \sqrt{v_x^2 + v_y^2}}{c_f^2 h} \qquad \text{and} \qquad -\partial_y u = \frac{v_y \sqrt{v_x^2 + v_y^2}}{c_f^2 h}. \tag{1.9}$$

The latter implies that

$$||\nabla u|| = \frac{v_x^2 + v_y^2}{c_f^2 h}$$

and

$$\sqrt{v_x^2 + v_y^2} = c_f h^{1/2} ||\nabla u||^{1/2}. \tag{1.10}$$

Substituting (1.10) into (1.9), we formally obtain

$$v_x = -\frac{c_f h^{1/2} \partial_x u}{||\nabla u||^{1/2}} \qquad \text{and} \qquad v_y = -\frac{c_f h^{1/2} \partial_y u}{||\nabla u||^{1/2}}.$$

Thus, recalling that $q_x = h v_x$ and $q_y = h v_y$, the original 2D SW system can be simplified to the scalar equation

$$\partial_t u - \partial_x \left(\frac{c_f h^{3/2} \partial_x u}{||\nabla u||^{1/2}} \right) + \partial_y \left(\frac{c_f h^{3/2} \partial_y u}{||\nabla u||^{1/2}} \right) = 0,$$

or equivalently,

$$\partial_t u - \operatorname{div} \left(c_f \frac{h^\alpha}{||\nabla u||^{1-\gamma}} \nabla u \right) = 0, \tag{1.11}$$

where $\gamma = \frac{1}{2}$ and $\alpha = \frac{3}{2}$.

Using similar notations from (1.1), the DW model on a perforated domain is given by

$$\begin{cases} \partial_t u - \operatorname{div} \left(c_f h^\alpha ||\nabla u||^{\gamma-1} \nabla u \right) &= 0 \quad \text{in} \quad \Omega \times (0,T], \\ u &= g \quad \text{on} \quad (\partial\Omega \setminus \partial\Omega_S) \times (0,T], \\ h^\alpha ||\nabla u||^{\gamma-1} \dfrac{\partial u}{\partial \mathbf{n}} &= 0 \quad \text{on} \quad (\partial\Omega \cap \partial\Omega_S) \times (0,T], \end{cases} \tag{1.12}$$

for a fixed time interval $(0,T]$, where (1.12) must be combined with some initial condition on u.

We remark that the choice of $\alpha = \frac{5}{3}$ in (1.12) would correspond to Manning friction. In fact, the authors of [101] concluded that (1.12) with variable ranges of $1 \leq \alpha \leq 2$ and $0 < \gamma < 1$ performs quite well, and one is not necessarily limited to specific choices of α and γ.

1.2 Summary and Contributions

The remainder of this thesis is structured as follows.

Chapter 2: Domain Decomposition and Coarse Approximations for Diffusion Models on Perforated Domains is based on the article [13], published in Applied Numerical Mathematics. It also includes portions taken from the conference proceedings papers [14, 12]. For the linear diffusion model problem (1.1), we propose a low-dimensional coarse approximation space in the spirit of Multi-scale Finite Element Methods (MsFEM). This coarse space can be used either as a coarse approximation or as a component of a two-level domain decomposition (DD) method. The coarse space is spanned by locally discrete harmonic basis functions. Along the subdomain boundaries, the basis functions are piecewise polynomial. The main theoretical contribution of this chapter is an error estimate regarding the H^1-projection over the coarse space; this error estimate is independent of the global regularity of the solution, which is expected to be low due to multiple corner singularities. For a

specific edge refinement procedure, the error analysis establishes superconvergence of the method even if the true solution has a low general regularity. This chapter also numerically explores the combination of the coarse space with overlapping DD methods. Generally, overlapping Schwarz methods can be employed either as a stationary fixed-point iteration or as a preconditioner for iterative Krylov methods such as Conjugate Gradient (CG) or Generalized Minimal RESidual (GMRES) methods [102]. The coarse space leads to an efficient two-level iterative linear solver which reaches the fine-scale finite element error in few iterations. Additionally, it bodes well as a domain decomposition preconditioner for Krylov methods; as expected, we find that the preconditioned Krylov method results in accelerated convergence when compared to the stationary iteration/fixed point method.

Chapter 3: Nonlinear Preconditioning Techniques for Urban Flood Models on Perforated Domains is based on the preprint [15], submitted to Computers and Mathematics with Applications. It focuses on the nonlinear Diffusive Wave equation (1.12) while also including numerical examples for a stationary porous medium equation on a perforated domain. For the DW model, we obtain topography data z_b from the 1 meter Digital Elevation Model available from [64]. This chapter tackles the dual challenge of multi-scale phenomena and nonlinearity in the model problem; addressing these points simultaneously increases the complexity and originality of our approach. We propose the linear Trefftz coarse space from [13] as a component of a two-level Restricted Additive Schwarz preconditioner for a linearized Newton system, which arises from the linearization of a nonlinear equation. Additionally, we use this coarse space as a component for multiple nonlinear preconditioning methods, including a Two-level RASPEN method. We provide a detailed complexity analysis of the proposed methods, discussing the benefits and downsides of each method. For these nonlinear methods applied to a time-dependent problem, we implement an adaptive local time-stepping method, which allows us to avoid a global time step reduction. Additionally, we propose the use of the Trefftz space for the approximation of the latter flood model. The main idea roughly consists in replacing the finite element space in the finite element discretization with the (conforming) coarse space. This coarse method is a very promising alternative, allowing us to use a small fraction of the fine-scale degrees of freedom to approximate the solution. This article also serves as a numerical application of nonlinear preconditioning methods applied to nonlinear problems, containing multiple numerical examples including the numerical solution of (1.12) on a large perforated domain representing a given area of Nice, France.

Chapter 4: Scientific Machine Learning for Nonlinear Elliptic PDEs with Rough Coefficients is based on the forthcoming article [16], submitted to the 2023 CEMRACS proceedings. It focuses on problems such as a porous medium equation and p-Laplace equation posed on heterogeneous domains. We employ a strategy which combines tools from Multi-scale Finite Element Methods (MsFEM) and Machine Learning. Our approach relies on a nonlinear approximate substructuring method which can be formulated based on local nonlinear Dirichlet-to-Neumann (DtN) operators. When solving the nonlinear substructured problem numerically by means of a fixed point method such as Newton's method, the DtN maps need to be computed repeatedly. To lower the computational cost caused by this repetitive evaluation of the DtN maps, we replace the latter by neural network-based surrogate models. The neural networks are trained to replicate the action of the local

nonlinear DtN maps on some coarse subset of the trace space. Once the training is completed, the surrogate DtN operators will be used within the nonlinear substructuring method. For training, we incorporate gradient information into the loss function; that is, we generate training data which contains both the local DtN maps and their (Fréchet) derivatives. Numerical experiments are performed in 1D and 2D and involve $p-$Laplace and degenerate diffusion equations. With just a few training points by dimension, the substitution model has an accuracy of a few percent when compared to the numerical solution.

The thesis concludes with a summary and suggestions for future work.

The undertaking of this thesis has led to the following articles, conference proceedings, and presentations (oral and poster).

Journal and conference papers/preprints

- [12] M. Boutilier, K. Brenner, V. Dolean. "A Trefftz-like coarse space for the two-level Schwarz method on perforated domains". International Conference on Domain Decomposition Methods, 2022.
- [14] M. Boutilier, K. Brenner, V. Dolean. "Trefftz approximation space for Poisson equation in perforated domains." International Conference on Finite Volumes for Complex Applications, 2023.
- [13] M. Boutilier K. Brenner, V. Dolean. "Robust methods for multiscale coarse approximations of diffusion models in perforated domains." Applied Numerical Mathematics, 2024.
- [16] M. Boutilier K. Brenner, V. Dolean. "Learning local Dirichlet-to-Neumann maps of nonlinear elliptic PDEs with rough coefficients." CEMRACS 2023 Proceedings (submitted), 2024.
- [15] M. Boutilier K. Brenner, V. Dolean. "Two-level Nonlinear Preconditioning Methods for Flood Models Posed on Perforated Domains." Computers and Mathematics with Applications (submitted), 2024.

Conference and seminar presentations

- Algoritmy 2024: Central-European Conference on Scientific Computing, "Multi-Domain Solutions of Urban Flood Models on Perforated Domains", High Tatra Mountains, Slovakia, March 2024.
- University of Strasbourg PDE Seminar, "Multi-Domain Solutions of Linear and Nonlinear PDEs Posed on Perforated Domains", Strasbourg, France, December 2023.
- International Conference on Finite Volumes for Complex Applications 10, "Trefftz Approximation Space for Poisson Equation Posed in Perforated Domains" (Poster), Strasbourg, France, November 2023.
- CEMRACS 2023: Scientific Machine Learning, "Learning Local Dirichlet-to-Neumann Maps for Multi-scale Urban Flood Modeling", Marseille, France, August 2023.
- IMPDE2023 : Research school on Iterative Methods for Partial Differential Equations, "Trefftz-like Coarse Space for Perforated Domains", Paris, France, May 2023.
- 27th International Domain Decomposition Conference, "A Trefftz-like Coarse Space for the Two-level Schwarz Method on Perforated Domains", Prague, Czech Republic, July 2022.
- 11th Conference on Parallel-in-Time Integration, "A Trefftz-like Coarse Space for the Two-level Schwarz Method on Perforated Domains" (Poster), Marseille, France, July 2022.

Chapter 2

Robust Domain Decomposition Methods and Coarse Approximations for Diffusion Models on Perforated Domains

This chapter is mainly based on the published paper [13] in Applied Numerical Mathematics. Additionally, preliminary studies were published in the proceedings papers [12, 14] of the 27th International Conference on Domain Decomposition Methods and the 10th International Conference on Finite Volumes for Complex Applications, respectively.

Contents

2.1 Introduction

In this chapter, we consider a linear diffusion model posed in a perforated domain. Recalling the notations from Chapter 1, let D be an open simply connected polygonal domain in $\mathbb{R}^2$. We denote by $(\Omega_{S,k})_k$ a finite family of perforations in D and denote $\Omega_S = \bigcup_k \Omega_{S,k}$ and $\Omega = D \setminus \overline{\Omega_S}$, assuming that the family $(\Omega_{S,k})_k$ is such that Ω is connected.

With this, an example stationary boundary value problem of interest is given by

$$\left\{\begin{array}{rcll} -\Delta u &=& f & \text{in} \quad \Omega, \\ \dfrac{\partial u}{\partial \mathbf{n}} &=& 0 & \text{on} \quad \partial\Omega \cap \partial\Omega_S, \\ u &=& 0 & \text{on} \quad \partial\Omega \setminus \partial\Omega_S, \end{array}\right. \tag{2.1}$$

where $\mathbf{n}$ is the outward normal to the boundary and $f \in L^2(\Omega)$.

Let us denote by $(\cdot,\cdot)_{L^2(\Omega)}$ the standard L^2 scalar product, that is,

$$(u,v)_{L^2(\Omega)} = \int_\Omega uv \,\mathrm{d}\mathbf{x}.$$

Setting

$$H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega) = \{u \in H^1(\Omega) \,|\, u|_{\partial\Omega\setminus\partial\Omega_S} = 0\}, \tag{2.2}$$

the weak solution of (2.1) satisfies the following variation formulation: Find $u \in H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$ such that

$$(\nabla u, \nabla v)_{L^2(\Omega)} = (f,v)_{L^2(\Omega)} \qquad \text{for all} \qquad v \in H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega). \tag{2.3}$$

As mentioned, in the context of urban flood modeling, u represents the flow potential (pressure head) and $(\Omega_{S,k})_k$ can be thought of as a family of impervious structures such as buildings, walls, or other similar structures. Although the linear model (2.1) is overly simplified to be directly used for urban flood assessment, the more general nonlinear elliptic or parabolic models are common in free surface flow simulations.

2.1.1 Finite Element Methods

As a discretization scheme, we choose finite element methods on a fine triangular mesh. This triangular mesh will be later referred to as the "fine-scale" discretization. In particular, we consider a conforming finite element method that solves the weak form of a model problem by replacing a function space with some finite-dimensional subspace.

Let us consider a triangulation $\mathcal{T}$ of Ω. We denote by V_h the space of functions that are continuous and triangle-wise polynomial on each triangle of $\mathcal{T}$, where h denotes the maximal element diameter, and by $V_{h,0}$ the space $V_h \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$. The finite element method for the variational problem (2.3) consists in finding $u_h \in V_{h,0}(\Omega)$ such that

$$a(u_h, v_h) = (f, v_h) \qquad \forall\, v_h \in V_{h,0}(\Omega). \tag{2.4}$$

Let us denote by $(\eta_s)_{s=1}^{N_{\overline{\Omega}}}$ the set of the standard finite element basis functions associated to degrees of freedom in $\overline{\Omega}$. The associated fine-scale finite element discretization results in the linear system

$$A\mathbf{u} = \mathbf{f}, \tag{2.5}$$

where A is a $N_\Omega \times N_\Omega$ matrix and N_Ω denotes the number of internal degrees of freedom in Ω. With this, the (i, j) entry of A is given by

$$A_{ij} = a(\eta_j, \eta_i) = \int_\Omega \nabla\eta_j \cdot \nabla\eta_i \, \mathrm{d}\mathbf{x},$$

and $\mathbf{f}$ is the vector from $\mathbb{R}^{N_\Omega}$ with components $f_j = \displaystyle\int_\Omega f\eta_j \, \mathrm{d}\mathbf{x}$. With this, $u_h = \displaystyle\sum_{k\in\mathcal{N}_\Omega} u_k\eta_k$ is the finite element approximation, where u_k is the kth entry of $\mathbf{u}$.

2.1.2 Classical Schwarz Methods

Domain decomposition (DD) methods are divide-and-conquer methods used to aid in the numerical solution of PDEs by partitioning the domain into multiple smaller subdomains, on which corresponding subproblems are solved. Depending on the DD method used, these subproblems can be solved independently, allowing us to harness the power of parallel computing. We focus here on overlapping domain decomposition, where each subdomain shares a portion if its interior with another. We remark, however, that an entire family of nonoverlapping domain decomposition methods exists [71, 108, 48]. In this chapter, we focus on a family of DD methods known as Schwarz methods.

The original Schwarz method was introduced in 1870 [96], not as a numerical method but as a tool to prove the Dirichlet principle. To recall, the Dirichlet principle states that if a function u is a solution to

$$\begin{cases} -\Delta u = 0 \text{ in } \Omega, \\ \quad u = g \text{ on } \partial\Omega, \end{cases} \tag{2.6}$$

on a bounded domain Ω, then u is the infimum of $\displaystyle\int_\Omega |\nabla v|^2 \, d\mathbf{x}$ over all sufficiently regular functions v satisfying $v = g$ on $\partial\Omega$. The original domain considered by Schwarz, shown in Figure 2.1, was composed of the union of an overlapping disk and rectangle.

Continuous Form

Thus, Schwarz proposed an iterative method which is widely referred to as the *alternating Schwarz method*. It consists in solving the problem in each subdomain sequentially, using the computed solution on the adjacent subdomain. Given an initial guess u_2^0 and the decomposition $\Omega = \Omega_1 \cup \Omega_2$ for two overlapping subdomains Ω_1 and Ω_2, the iterative method to solve (2.6) is given by the following: for iteration index $n = 0, 1, \ldots,$

$$\begin{aligned} -\Delta u_1^{n+1} &= 0 \text{ in } \Omega_1, & -\Delta u_2^{n+1} &= 0 \text{ in } \Omega_2, \\ u_1^{n+1} &= g \text{ on } \partial\Omega_1 \cap \partial\Omega, & u_2^{n+1} &= g \text{ on } \partial\Omega_2 \cap \partial\Omega, \\ u_1^{n+1} &= u_2^n \text{ on } \partial\Omega_1 \cap \overline{\Omega}_2, & u_2^{n+1} &= u_1^{n+1} \text{ on } \partial\Omega_2 \cap \overline{\Omega}_1. \end{aligned} \tag{2.7}$$

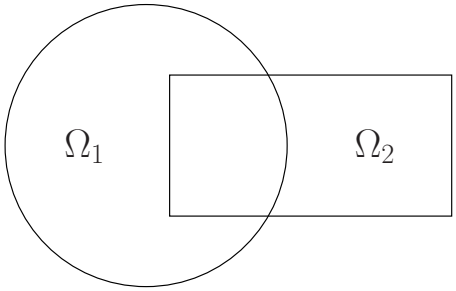


Figure 2.1: A visual of the domain $\Omega = \Omega_1 \cup \Omega_2$, where Ω_1 corresponds to a disk and Ω_2 corresponds to a rectangle. The first domain decomposition method was introduced by Schwarz on this domain.

Schwarz proved the convergence of the algorithm (2.7) using the maximum principle. Additionally, Schwarz deduced that if the sequence (2.7) converges, then u_1^∞ and u_2^∞ take the same values in the region of overlap $\Omega_1 \cap \Omega_2$. With this, u_1^∞ and u_2^∞ converge to the solution u of the Laplace equation on Ω_1 and Ω_2, respectively; we refer the reader to Chapter 1 of [40] and [52] for details.

One may note that in the iteration (2.7), the solution u_1^{n+1} must be computed before u_2^{n+1}, creating an iteration that is not parallel. An extension to the iteration was proposed over a century later by Lions [76]; this extension allows for parallel implementation of the iteration. The *parallel Schwarz method*, while appearing extremely similar to (2.7), involves a change to the transmission condition on the equation of the second subdomain. The parallel iteration is given by the following:for $n = 0, 1, \ldots,$

$$\begin{aligned} -\Delta u_1^{n+1} &= 0 \text{ in } \Omega_1, & -\Delta u_2^{n+1} &= 0 \text{ in } \Omega_2, \\ u_1^{n+1} &= g \text{ on } \partial\Omega_1 \cap \partial\Omega, & u_2^{n+1} &= g \text{ on } \partial\Omega_2 \cap \partial\Omega, \\ u_1^{n+1} &= u_2^n \text{ on } \partial\Omega_1 \cap \overline{\Omega}_2, & u_2^{n+1} &= u_1^n \text{ on } \partial\Omega_2 \cap \overline{\Omega}_1. \end{aligned} \tag{2.8}$$

We remark that for the parallel iteration, one requires an initialization of u_1^0 and u_2^0. With this, each iteration of (2.8) can be solved concurrently in Ω_1 and Ω_2.

We can think of the alternating and parallel Schwarz iterations as being analogous to the block Gauss-Seidel and block Jacobi iterations, respectively. As is true for the algebraic Gauss-Seidel and Jacobi methods, the alternating Schwarz methods will generally converge in fewer iterations that the parallel Schwarz iteration. However, the possibility for parallel computing must be considered.

We remark that the iterative methods (2.7) and (2.8) can be extended naturally to greater than two subdomains. Additionally, while the iterations (2.7) and (2.8) are written for the Laplace equation, they can be applied to a general linear PDE, given that the local Dirichlet problems are well posed. Consider the decomposition $(\Omega_j)_{j=1}^N$ into multiple overlapping subdomains Ω_j. The parallel Schwarz iteration for a general problem $\mathcal{L}u = f$, where $\mathcal{L}$ is a linear operator, is given by the following: for $n = 0, 1, \ldots,$

$$\begin{aligned} \mathcal{L}u_j^{n+1} &= f \quad \text{in} \quad \Omega_j, \\ u_j^{n+1} &= g \quad \text{on} \quad \partial\Omega_j \cap \partial\Omega, \\ u_j^{n+1} &= u_i^n \quad \text{on} \quad \partial\Omega_j \cap \overline{\Omega_i}, \end{aligned}$$

for $j = 1, \ldots, N$, where $i \neq j$ such that $\partial\Omega_i \cap \Omega_j$ is non-empty.

Together, the alternating and parallel Schwarz methods are often referred to as classical Schwarz methods. It is important to note that these classical Schwarz methods require overlapping subdomains for convergence. The natural question that arises is what one should take for the overlap between subdomains. Generally, increasing the overlap will decrease overall iteration count. However, this is at the cost of larger subproblems. Therefore, some sort of "sweet spot" must be chosen in terms of overlap size.

We remark that the family of optimized Schwarz methods [51] allow for subdomains that only share a common boundary without additional overlap. These optimized Schwarz methods impose Robin transmission conditions on the subdomain boundaries in order to increase the efficiency of the Schwarz algorithms and increase the convergence speed [86]. We refer the reader to [52] for a summary of Schwarz methods over time, and to [100, 40] for an introduction to DD methods.

Algebraic Form

We now introduce the algebraic form of the parallel Schwarz iteration. For this, consider the linear system $A\mathbf{u} = \mathbf{f}$ that arises from the finite element discretization of the linear PDE, where A is assumed to be symmetric positive definite.

Let N_j denote the number of degrees of freedom in Ω_j. Consider the following matrices:

- Restriction matrices R_j which restrict from the degrees of freedom in Ω to the degrees of freedom in subdomain Ω_j. These matrices are boolean matrices of size $N_j \times N_\Omega$, where $(R_j)_{i,k} = 1$ if the ith degree of freedom in the partitioning of Ω_j is the kth degree of freedom in the partitioning of Ω; otherwise, the matrices are full of zeros.

- Partition of unity matrices D_j, $N_j \times N_j$ diagonal matrices satisfying

$$\sum_{j=1}^{N} R_j^T D_j R_j = I, \tag{2.9}$$

  where I is the identity matrix of size N_Ω. Specifically, we define the partition of unity matrices such that $(D_j)_{i,i} = \frac{1}{n_i}$, where n_{ij} denotes the number of subdomains in which the ith degree of freedom in the partitioning of Ω_j is contained. Otherwise, the matrices are full of zeros.

Additive Schwarz Method: The Additive Schwarz method (ASM), analogous to the parallel Schwarz method (2.8), was introduced in [43] and is given by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + M_{AS}^{-1}(\mathbf{f} - A\mathbf{u}^n), \tag{2.10}$$

where the matrix M_{AS}^{-1} is given by

$$M_{AS}^{-1} = \sum_{j=1}^{N} R_j^T (R_j A R_j^T)^{-1} R_j. \tag{2.11}$$

We note that (2.10) is analogous to the continuous parallel Schwarz method. However, the iterative method (2.10) will not converge in the region of overlap, as the

solution on each subdomain in the overlap will be added multiple times, creating multiple contributions to the overlap.

Restricted Additive Schwarz Method: The issue with convergence in the overlap can be solved by the Restricted Additive Schwarz (RAS) method, given by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + M_{RAS}^{-1}(\mathbf{f} - A\mathbf{u}^n),$$

where the matrix

$$M_{RAS}^{-1} = \sum_{j=1}^{N} R_j^T D_j (R_j A R_j^T)^{-1} R_j, \tag{2.12}$$

contains partition of unity matrices which deal with multiple contributions in the overlapping region. The RAS method, introduced in [27], was discovered via the accidental modification of the additive Schwarz method. The RAS method converges in the overlapping region and forms a convergent fixed point iterative method.

So far, we have only discussed Schwarz methods as iterative methods. However, the Schwarz preconditions such as M_{RAS}^{-1} perform well as preconditioners for Krylov methods, which are typically faster than the fixed point iterations. However, one can accelerate these fixed point iterations by a Krylov method, using the Schwarz matrices as preconditioners. We introduce the left-preconditioned linear system given by

$$M_{AS}^{-1} A\mathbf{u} = M_{AS}^{-1}\mathbf{f}, \tag{2.13}$$

or

$$M_{RAS}^{-1} A\mathbf{u} = M_{RAS}^{-1}\mathbf{f}. \tag{2.14}$$

We note that the preconditioner M_{AS}^{-1} is symmetric as long as A is symmetric. Therefore, for the preconditioned system (2.13), one can use the Conjugate Gradient (CG) method as their Krylov solver of choice, given that the matrix A is symmetric positive definite. For the RAS preconditioned system (2.14), the preconditioner is non-symmetric, and one can use GMRES as a solver instead of CG. Generally, (2.14) will produce slightly less iterations when compared to (2.13), although (2.14) loses the symmetric property.

Multiplicative Schwarz Method The Multiplicate Schwarz (MS) Method, analogous to the alternating Schwarz method (2.7), is given by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + M_{MS}^{-1}(\mathbf{f} - A\mathbf{u}^n),$$

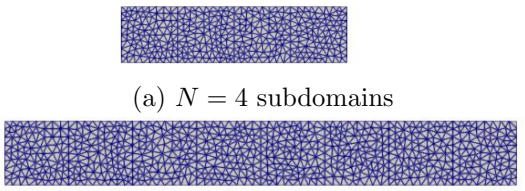
where the matrix M_{AS}^{-1} is given by

$$M_{MS}^{-1} = [I - \Pi_{j=1}^{N}(I - R_j^T (R_j A R_j^T)^{-1} R_j) A)] A^{-1}. \tag{2.15}$$

We remark that as this method is multiplicative, it is generally non-parallel in nature. However, as mentioned, the MS iterative method will generally converge in fewer iterations when compared to the RAS iterative method. As we are primarily concerned with possible parallel implementation, the MS iteration/preconditioner is not a focus of this thesis.

Two-level Schwarz Methods

The above methods presented, which we will refer to as one-level methods, tend to have a common problem that arises when the number of subdomains N grows very



(a) $N = 4$ subdomains

(b) $N = 8$ subdomains

Figure 2.2: Homogeneous domain for $N = 4$ (top) and $N = 8$ (right) subdomains, where each subdomain is a unit square.

large. We note that in the iterations from Section 2.1.2, the subdomains are limited to communicating with adjacent subdomains. Therefore, as N grows large, a lack of global communication results in a lack of scalability of the DD algorithms. In terms of scalability, there are generally two forms presented: *weak* and *strong* scalability. We note that as we consider domains containing multiple perforations within them, we also wish to achieve robustness with respect to the domain's geometry.

Weak scalability refers to how the runtime (or iteration count) varies with the number of subdomains as the problem size and the number of subdomains are increased proportionally; that is, N and the problem size N_Ω are increased while ensuring the subdomain size N_j is consistent. A method exhibits weak scalability if increasing both the problem size and the number of subdomains by a certain factor results in roughly constant solution time. For example, if a problem with 100 unknowns on 10 subdomains takes 10 seconds to run, a problem with 500 unknowns on 50 subdomains should take the same 10 seconds to run if the method is weakly scalable.

Strong scalability refers to how the solution time varies with the number of subdomains for a fixed total problem size. That is, the problem size N_Ω is kept consistent while N is increased. A method exhibits strong scalability if increasing the number of subdomains decreases the solution time proportionally for a fixed problem size. For example, if a problem with 100 unknowns on 10 subdomains takes 10 seconds to run, the same problem on 50 subdomains should take 2 seconds to run if the method is strongly scalable. However, we remark that perfect strong scalability is difficult to achieve in practice, as eventually, the subproblems become extremely small as the number of subdomains increases to infinity. See [29] for details on this topic, where the authors explore the scalabilities of classical one-level Schwarz methods.

We provide a visual regarding lack of weak scalability for the one-level RAS method in Figure 2.3. Weak scalability tests are done for the Laplace equation on N subdomains in one dimension, where each domain is a unit square (see Figure 2.2).

To combat this lack of scalability in the previously mentioned one-level Schwarz methods, we introduce the idea of two-level Schwarz methods which contain a coarse space to allow for global communication across all subdomains. We remark that coarse spaces can not only provide scalable DD methods, but also further accelerate the DD methods. Moreover, an optimal coarse space can lead to a DD method which converges in one iteration [53]. With some coarse matrix R_H of size $N_\mathcal{V} \times N_\Omega$ which will be defined/discussed later and M_H^{-1} given by

$$M_H^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H,$$

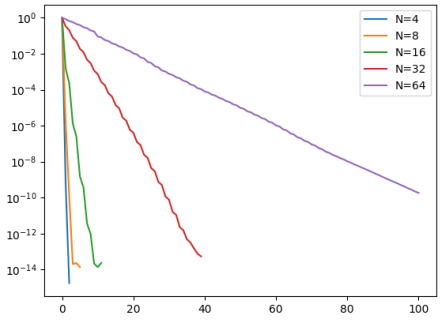


Figure 2.3: Krylov iteration count for the one-level RAS preconditioner used to solve the Laplace equation, for various numbers of subdomains.

the two-level preconditioned system is given by

$$M_{RAS,2}^{-1} A\mathbf{u} = M_{RAS,2}^{-1}\mathbf{f}, \tag{2.16}$$

or

$$M_{AS,2}^{-1} A\mathbf{u} = M_{AS,2}^{-1}\mathbf{f}, \tag{2.17}$$

where the two-level discrete RAS preconditioner is given additively by

$$M_{RAS,2}^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H + \sum_{j=1}^{N} R_j^T D_j (R_j A R_j^T)^{-1} R_j,$$

and the two-level discrete Additive Schwarz preconditioner is given additively by

$$M_{AS,2}^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H + \sum_{j=1}^{N} R_j^T (R_j A R_j^T)^{-1} R_j.$$

We expect the use of a well-constructed preconditioner to provide acceleration in terms of Krylov iteration count.

Additionally, the two-level iterative RAS method is given by

$$\begin{aligned} \mathbf{u}^{n+\frac{1}{2}} &= \mathbf{u}^n + M_{RAS}^{-1}(\mathbf{f} - A\mathbf{u}^n), \\ \mathbf{u}^{n+1} &= \mathbf{u}^{n+\frac{1}{2}} + M_H^{-1}(\mathbf{f} - A\mathbf{u}^{n+\frac{1}{2}}), \end{aligned}$$

which is equivalent to

$$\mathbf{u}^{n+1} = \mathbf{u}^n + (M_H^{-1} + M_{RAS}^{-1}(I - AM_H^{-1}))(\mathbf{f} - A\mathbf{u}^n). \tag{2.18}$$

We will rather use this "hybrid" iterative method (2.18) as the iterative version of the additive method will not converge. However, we remark that if a Krylov method is used, then the additive preconditioners (2.16) and (2.17) will converge as well. Generally, using preconditioned Krylov methods produces faster convergence than the iterative solver, both in terms of computation time and iterations.

2.1.3 Existing Families of Coarse Spaces

Our model problem can be thought of as the extreme limit case of the elliptic model containing highly contrasting coefficients with zero conductivity on the perforations. Two-level domain decomposition methods have been extensively studied for such heterogeneous problems. In the following subsections, we present families well-known classical coarse spaces.

MsFEM and Energy Minimizing Coarse Spaces

Multiscale Finite Element Methods (MsFEM) [60, 45] are a class of numerical techniques originally designed to tackle problems characterized by significant variations across multiple scales. MsFEM methods aim to efficiently capture the solution of problems involving multiscale phenomena, by incorporating information from different scales into the finite element framework. Unlike traditional finite element methods, which typically require fine meshes to resolve small-scale features, MsFEM methods seek to represent these features using coarse meshes, reducing the complexity of the problem while still preserving accuracy. In MsFEM methods, a Galerkin approximation of the variational formulation of a model PDE is computed on a lower dimensional, problem-dependent space. The basis functions of this lower-dimensional space are numerically computed; generally, these basis functions are the solution to the localized version of the original PDE with some boundary conditions imposed. Later in this chapter, we provide a coarse approximation space which, like MsFEM, is also composed of numerically computed harmonic basis functions. As we work on perforated domains, the basis functions will be computed based on specific coarse degrees of freedom that are based on the urban geometries in the model domain.

MsFEM methods can also be used as a coarse space for domain decomposition methods. Robust coarse spaces have been constructed using the ideas from MsFEM in [1, 55]. The authors of [94] discuss varied coefficient problems in the case where the coarse grid is not properly aligned with the heterogeneities. The combination of spectral and MsFEM methods can be found in [54], where the authors enrich the MsFEM coarse space with eigenfunctions along the edges. Additionally, the family of GDSW (Generalized Dryja, Smith, Widlund) methods [38] employ energy-minimizing coarse spaces and can be used to solve heterogeneous problems on less regular domains. These spaces involve a combination of edge and nodal basis functions. To deal with coefficient jumps in highly heterogeneous problems, an adaptive GDSW coarse space was introduced in [56].

Spectral Coarse Spaces

Spectral methods such as those given in [50, 87, 41, 98] obtain a robust coarse space via the local solutions of spectral problems in each subdomain. The authors of [87] proposed a coarse space for overlapping Schwarz methods whose construction is based on an eigenvalue problem for the Dirichlet-to-Neumann (DtN) operator on each subdomain. The Generalized Eigenproblems in the Overlaps (GenEO) coarse space [98] involves the solution of generalized eigenvalue problems in (originally) the overlapping zone of the partitioning. Like in the DtN coarse space, a number of low frequency eigenfunctions are selected; these local functions are converted into global coarse basis functions by being combined with partition of unity functions.

We proceed to the algebraic definition of a spectral space, similar to the GenEO space, which will be used in this chapter as a numerical comparison. The construction of this space involves the solution of a generalized eigenvalue problem in each subdomain and will be referred to as the GenEO-like coarse space.

Let $\bar{A}_j$ denote the local "Neumann" matrix on $\bar{\Omega}_j$; i.e., the stiffness matrix for the local Neumann problem. With this, we introduce the discrete eigenvalue problem given by

$$\bar{A}_j \mathbf{p}_k^j = \lambda_k^j (D_j A_j D_j) \mathbf{p}_k^j, \tag{2.19}$$

where $A_j = R_j A R_j^T$. The eigenvalue problem solves for the eigenpairs $(\lambda_k^j, \mathbf{p}_k^j)$. For the construction of the algebraic coarse space, we take $k = 1, \ldots, m_j$ eigenvectors corresponding to the m_j lowest eigenvalues of (2.19). The GenEO-like coarse matrix R_H^T is defined such that its columns are given by

$$R_j^T D_j \mathbf{p}_k^j \tag{2.20}$$

for $j = 1, \ldots, N$ and $k = 1, ..., m_j$, where there are $\sum_{j=1}^{N} m_j$ total columns in R_H^T.

Nicolaides Coarse Space

The Nicolaides coarse space [89] is a traditional coarse space made of piecewise constant functions per subdomain. We now present the algebraic form of the Nicolaides space, as it will be used in this chapter as a numerical comparison to the proposed coarse space. Given algebraic restriction, extension, and partition of unity matrices, the Nicolaides coarse matrix R_H is defined such that each column of R_H^T is given by

$$(R_H^T)_j := R_j^T D_j R_j \mathbf{1}, \tag{2.21}$$

for $j = 1, \ldots, N$, where $\mathbf{1}$ is a $(N \times 1)$ vector of ones.

We remark that the Nicolaides space can be thought of as a particular case of the GenEO-like coarse space. Specifically, if $m_j = 1$ and the subdomains Ω_j are connected, the lowest eigenvalue is zero such that the corresponding eigenvector is a constant. As this is also true for the DtN coarse space, the Nicolaides and spectral coarse spaces are identical in this case.

For scalability, the Nicolaides coarse space requires that the subdomains be connected. Otherwise, the Nicolaides space may not form a sufficient coarse space in the sense that it will not achieve weak/strong scalability as a component of a two-level DD method. With our realistic perforated domains and a coarse partitioning such as the ones shown in Figure 1.2, disconnected subdomains occur often with long, thin heterogeneities that can completely cut through a subdomain. Therefore, to obtain robustness and scalability, the Nicolaides coarse space with this geometric subdomain partitioning will be insufficient for our case; this is shown via numerical examples in Section 2.4.4. To combat this, we introduce a slightly modified Nicolaides coarse space that we expect to be scalable regardless of the initial coarse partitioning of the subdomains. Here, we begin with an initial regular rectangular partitioning, identify the disconnected regions of each subdomain from node and triangle connectivity, and treat each disconnected component as a separate subdomain to obtain our new subdomain partitioning. Then, a traditional Nicolaides coarse space is generated on this new partitioning.

In other words, let w_j denote the number of disconnected components in Ω_j and let $\Omega_{j,w}, w = 1, \ldots, w_j$ denote the corresponding nonoverlapping component. Then

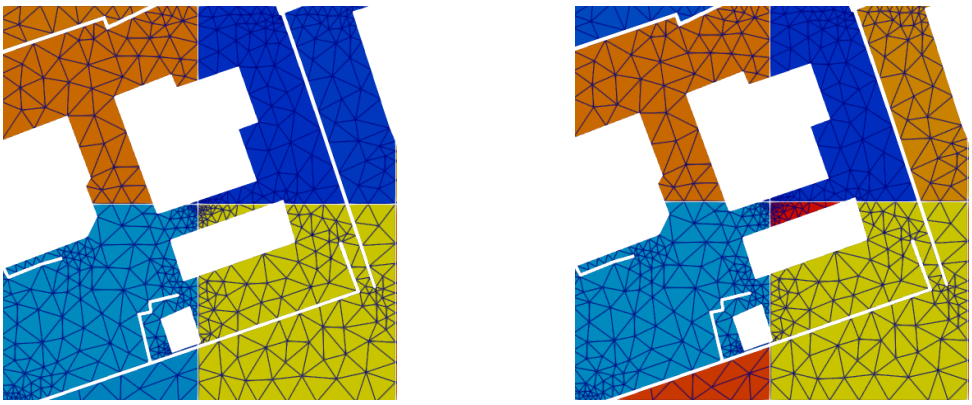


Figure 2.4: Possible partitioning of the domain without (left) and with (right) additional partitioning by disconnected component, with each subdomain/component depicted by a color block.

our new nonoverlapping partioning contains $w = \sum_{j=1}^{N} w_j$ total subdomains and is given by

$$(\hat{\Omega}_j)_{j\in\{1,\dots,w\}} = \left((\Omega_{j,w})_{w\in\{1,\dots,w_j\}}\right)_{j\in\{1,\dots,N\}},$$

and an associated overlapping partioning can be generated. Once this new partitioning is generated, the Nicolaides coarse space is generated normally, containing one piecewise constant function per "enriched" (new) subdomain. The resulting coarse space on this new partitioning will contain w total columns and will be referred to as the Enriched Nicolaides coarse space. This special partitioning in visualized in the right of Figure 2.4.

2.1.4 Numerical Methods for Multiscale Problems

Aside from the earlier mentioned domain decomposition and MsFEM, there exists extensive literature on problems involving highly oscillatory coefficients or multiscale geometrical features. This includes the Localized Orthogonal Decomposition (LOD) method [82], which decomposes the solution space into a low-dimensional space with good approximation properties and a high-dimensional "remainder" space. The low-dimensional space is approximated by locally supported basis functions which can be computed in parallel. Additionally, we note the existence of multiscale partition of unity methods such as the Generalized Finite Element Method (GFEM) [7] and the Multiscale Spectral Generalized Finite Element Method (Ms-GFEM) [8, 81]. In [8, 81], the optimal approximation spaces are constructed locally by solving generalized eigenvalue problems over a set of overlapping subdomains. Partition of unity functions are used to glue the local contributions into a globally conforming coarse approximation space. We note that these methods vary from the approach presented later in this chapter as we consider nonoverlapping coarse cells and do not rely on eigenproblems.

The method we choose to consider here is closer to the previously discussed MsFEM, as well as the Multiscale Hybrid-Mixed Method (MHM) [5] or polytopal methods such as Virtual Element Methods (VEM) [20]. In comparison to the classical MsFEM, our method leads to a larger coarse space; the size of the proposed coarse matrix is discussed in Section 2.4. Compared to VEM, the major difference is that we numerically compute the approximation of the locally harmonic basis. By

doing so, we manage to incorporate singular functions (corresponding to the corners of the domain) into the coarse space. We are also able to deal with very general polygonal cells (not star-shaped, not simply connected, etc.), which differs from VEM. Additionally, we willingly avoid using a method of fundamental solutions of any kind, because of our long-term motivation in problems more complex than the linear model problem. The MHM method combines ideas from the Mixed Finite Element Method (MFEM) and MsFEM. Compared to MHM, where Neumann traces are approximated by piecewise polynomials, our method is formulated in terms of approximate Dirichlet traces. We remark as well the existence of Generalized Multiscale Finite Element Methods (GMsFEM) [44], which were created to generalize MsFEM methods by constructing coarse trial spaces via a spectral decomposition of snapshot spaces.

Specifically on perforated domains with small and numerous perforations, the authors of [74] introduced an MsFEM method for diffusion problems with Dirichlet boundary conditions imposed on the perforation and domain boundaries, with an error estimate provided. In [36, 21], this method was extended to advection-diffusion problems, with [21] imposing both Dirichlet and Neumann boundary conditions on the perforation boundaries. In [74, 21, 36], Crouzeix-Raviart type boundary conditions were imposed on the local problems to provide robustness with respect to the position of the perforations. As well, the addition of bubble functions is included. The authors of [22] also provides an MsFEM method for perforated domains, providing numerical results and analysis for problems posed on domains with numerous small, regular perforations. In [22], Neumann boundary conditions are posed on the perforation boundaries, with the assumption that the coarse grid does not intersect with the perforations along the edges for analysis purposes (although this assumption is not necessary numerically). Aside from classical MsFEM methods, the authors of [58] introduced a Heterogeneous Multiscale method (HMM) [106] method to solve elliptic homogenization problems in perforated domains, with periodic perforations required for analysis purposes. Additionally, the authors of [33, 34, 99] proposed a GMsFEM method for perforated domains for small, regular perforations.

2.1.5 Chapter Outline

The remainder of this chapter is laid out as follows. In Section 2.2, we introduce the continuous Trefftz coarse space and discuss its approximation properties. The Galerkin method based on the coarse space is introduced and we provide an error estimate for the Trefftz space as a coarse approximation. In Section 2.3, we introduce the Trefftz coarse space in its discrete matrix form. With this, we provide the matrix forms of the coarse approximation and domain decomposition methods. The two-level domain decomposition methods are presented as both an iterative solver and a preconditioner for Krylov methods. In Section 2.4, we provide numerical results for the Trefftz space used as a coarse approximation, in an iterative Schwarz method, and as a preconditioner for Krylov methods. We provide numerical results for three different types of model domains; the first domain is a simplified domain with one perforation at the corner, the second type is used for scalability tests and is a combination of multiple unit squares, and the third type of model domain is a realistic urban domain with numerous perforations of various shape. Section 2.5 concludes with a summary.

2.2 Continuous Trefftz Approximation

In this section, we introduce a splitting of the weak formulation (2.3) into the locally harmonic component of the solution and the component of the solution which vanishes on the coarse skeleton Γ. With this, we propose the continuous finite-dimensional coarse space that can be used to efficiently approximate the locally harmonic component of the solution. We then perform the error analysis of the coarse Galerkin approximation; in this regard, the main results are Proposition 2.6 and Theorem 2.8. This error estimate establishes that the "coarse" solution accurately approximates the locally harmonic component of the solution, and therefore the solution itself.

2.2.1 Coarse Mesh and Space Decomposition

We begin with a coarse discretization of Ω which involves a family of polygonal cells $(\Omega_j)_{j=1,\dots,N}$, the so-called coarse skeleton Γ, and the set of coarse grid nodes that will be referred to by $\mathcal{V}$.

The construction is as follows. Consider a finite nonoverlapping polygonal partitioning of D denoted by $(D_j)_{j=1,\dots,N}$ and an induced nonoverlapping partitioning of Ω denoted by $(\Omega_j)_{j=1,\dots,N}$ such that $\Omega_j = D_j \cap \Omega$. We will refer to $(\Omega_j)_{j=1,\dots,N}$ as the coarse mesh over Ω. Additionally, we denote by Γ its skeleton, that is $\Gamma = \bigcup_{j=1,\dots,N} \partial\Omega_j \setminus \partial\Omega_S$.

We define $H^1_\Delta(\Omega)$ as a subspace of $H^1(\Omega)$ composed of piece-wise harmonic functions, weakly satisfying the homogeneous Neumann boundary conditions on $\partial\Omega \cap \partial\Omega_S$ such that

$$H^1_\Delta(\Omega) = \{\, u \in H^1(\Omega) \mid (\nabla u|_{\Omega_j}, \nabla v)_{L^2(\Omega_j)} = 0 \quad \text{for all} \quad v \in H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega_j)\}. \tag{2.22}$$

In other words, $H^1_\Delta(\Omega)$ can be defined as $u \in H^1(\Omega)$ such that for all subdomains Ω_j, the equations

$$\left\{ \begin{array}{rcll} -\Delta u|_{\Omega_j} & = & 0 & \text{in} \quad \Omega_j, \\ \dfrac{\partial u}{\partial \mathbf{n}} & = & 0 & \text{on} \quad \partial\Omega_j \cap \partial\Omega_S, \end{array} \right. \tag{2.23}$$

are satisfied in a weak sense. We further define the space $H^1_\Gamma(\Omega)$ as the subspace of functions vanishing on the coarse skeleton Γ such that

$$H^1_\Gamma(\Omega) = \{\, u \in H^1(\Omega) \mid u|_\Gamma = 0\}. \tag{2.24}$$

By definition, $H^1_\Delta(\Omega)$ is orthogonal to $H^1_\Gamma(\Omega)$. Since $H^1_\Gamma(\Omega)$ is a closed subspace of $H^1(\Omega)$, we deduce that $H^1(\Omega) = H^1_\Delta(\Omega) \oplus H^1_\Gamma(\Omega)$ (see e.g. [93]). In other words, a given function $v \in H^1(\Omega)$ admits a unique decomposition into $v_\Delta + v_b$, where $v_\Delta \in H^1_\Delta(\Omega)$, $v_b \in H^1_\Gamma(\Omega)$ and $(\nabla v_\Delta, \nabla v_b)_{L^2(\Omega)} = 0$. Although for simplicity we will call v_Δ the "locally harmonic" or "piece-wise" harmonic component of v, we wish to stress that the space $H^1_\Delta(\Omega)$ also contains information about the normal traces of v_Δ over $\partial\Omega_S$. The function v_b will be referred to as the local or "bubble" component of v.

Using the orthogonal decomposition of $H^1(\Omega)$ introduced above, we can express (2.3) as the following: Find $u = u_\Delta + u_b$ with $u_\Delta \in H^1_\Delta(\Omega) \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$ and

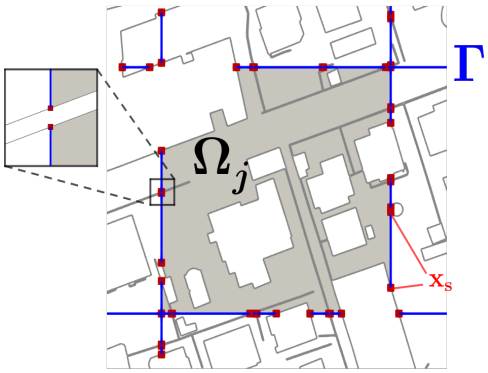


Figure 2.5: Coarse grid cell Ω_j, nonoverlapping skeleton Γ (blue lines), and coarse grid nodes $\mathbf{x}_s = (x_s, y_s) \in \mathcal{V}$ (red dots). Coarse grid nodes are located at $\overline{\Gamma} \cap \partial\Omega_S$.

$u_b \in H^1_\Gamma(\Omega)$ satisfying

$$(\nabla u_\Delta, \nabla v)_{L^2(\Omega)} = (f, v)_{L^2(\Omega)} \qquad \forall v \in H^1_\Delta(\Omega) \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega), \tag{2.25}$$

$$(\nabla u_b, \nabla v)_{L^2(\Omega)} = (f, v)_{L^2(\Omega)} \qquad \forall v \in H^1_\Gamma(\Omega). \tag{2.26}$$

We remark that the formulation (2.25)-(2.26) uncouples the local and the piece-wise harmonic components of u, moreover that the "bubble" component of the solution u_b can be computed from (2.26) locally (and in parallel) on each Ω_j, while the problem (2.25) remains globally coupled over the computational domain Ω.

2.2.2 Continuous Trefftz Space

We now proceed with the goal to approximate the locally harmonic component u_Δ of the solution. For this, we introduce the Trefftz coarse space, a finite-dimensional subspace of $H^1_\Delta(\Omega)$ that is spanned by functions that are piece-wise polynomial on the skeleton Γ.

Let $(e_k)_{k=1,\dots,N_e}$ denote a nonoverlapping partitioning of Γ such that each "coarse edge" e_k is an open planar segment, and we denote $H = \max_{k=1,\dots,N_e} |e_k|$. The set of coarse grid nodes is given by $\mathcal{V} = \bigcup_{k=1,\dots,N_e} \partial e_k$. Figure 2.5 illustrates the location of the nodal degrees of freedom that typically result from clipping $(D_j)_j$ with Ω_S. It is important to note that a straight segment of Γ may be subdivided into multiple edges. As we will show in Section 2.4, this subdivision can be intentional (see Figure 2.6) to achieve convergence of the coarse approximation.

We define

$$V^\Gamma_{H,p} = \{\, v \in C^0(\overline{\Gamma}) \mid v|_{e_k} \in \mathbb{P}_p(e_k) \quad \text{for all} \quad k = 1, \dots, N_e\},$$

where $\mathbb{P}_p(e_k)$ denotes the set of polynomials of order (at most p) over an edge e. We also define

$$V_{H,p} = \{\, v \in H^1_\Delta(\Omega) \mid v|_\Gamma \in V^\Gamma_{H,p}\}.$$

Let $(g_\alpha)_{\alpha=1,\dots,N_{H,p}}$ be some basis of $V^\Gamma_{H,p}$, where $N_{H,p}$ is the dimension of $V^\Gamma_{H,p}$. In practice, $(g_\alpha)_{\alpha=1,\dots,N_{H,p}}$ is set up by combining the set of the nodal piece-wise linear "hat" functions with the set of the higher order edge-based basis functions.

The function $\phi_\alpha \in V_{H,p}$ associated to g_α is computed by weakly imposing

$$\begin{cases} \Delta\phi_\alpha &= 0 \quad \text{in} \quad \Omega_j, \\ \dfrac{\partial\phi_\alpha}{\partial\mathbf{n}} &= 0 \quad \text{on} \quad \partial\Omega_j \cap \partial\Omega_S, \\ \phi_\alpha &= g_\alpha \quad \text{on} \quad \partial\Omega_j \setminus \partial\Omega_S, \end{cases} \tag{2.27}$$

against the tests functions in $V_{h,p}$.

The Galerkin method to approximate $u_\Delta \in H^1_\Delta(\Omega)$ based on the coarse space reads as follows: find $u_{\Delta,H} \in V_{H,p} \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$ such that

$$(\nabla u_{\Delta,H}, \nabla v)_{L^2(\Omega)} = (f, v)_{L^2(\Omega)} \qquad \forall v \in V_{H,p} \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega). \tag{2.28}$$

2.2.3 Error Analysis

We now provide an error estimate for the convergence of the Trefftz approximation space. We begin with a few preliminary results that will be necessary in the proof.

Lemma 2.1 (Polynomial interpolation over an edge)**.** *Let e_k be a coarse edge. We denote by $\mathcal{I}^p_k$ the Lagrange interpolator of order p defined with respect to some set of interpolation points containing the endpoints of e_k. Then, there exists $c_0 = c_0(p) > 0$ such that for every v of sufficient regularity, we have*

$$\|v - \mathcal{I}^p_k v\|_{L^2(e_k)} + |e_k||v - \mathcal{I}^p_k v|_{H^1(e_k)} \le c_0 |e_k|^{p+1} |v|_{H^{p+1}(e_k)}.$$

This directly implies the following L^2 and H^1 estimates:

$$\|v - \mathcal{I}^p_k v\|_{L^2(e_k)} \le c_0 H^{p+1} |v|_{H^{p+1}(e_k)} \qquad \text{and} \qquad \|v - \mathcal{I}^p_k v\|_{H^1(e_k)} \le c_1 H^p |v|_{H^{p+1}(e_k)},$$

where $c_1 = c_0 \left(1 + H^2\right)^{1/2}$ can be bounded e.g. as $c_1 \le c_0 \left(1 + \operatorname{diam}(\Omega)^2\right)^{1/2}$.

Proof. See Proposition 1.5 and 1.12 of [46]. See also Lemma 4.2 and Remark 4 of [11] for estimates with explicit dependency on p and higher dimension. □

Lemma 2.2 (Gagliardo-Nirenberg interpolation inequality)**.** *Let Ω be Lipschitz domain in $\mathbb{R}^d$ and $\|\cdot\|_{s,q}$ denote the Slobodeskii-Sobolev norm in $W^{s,q}(\Omega)$, $s \ge 0, q \ge 1$ (see the reference below). Let $s, s_1, s_2 \ge 0$, $1 \le q, q_1, q_2 \le \infty$ and $\theta \in (0,1)$ be such that*

$$s = \theta s_1 + (1-\theta) s_2, \quad \frac{1}{r} = \frac{\theta}{q_1} + \frac{1-\theta}{q_2}.$$

Then, there exists $c_{GN} = c_{GN}(s_1, q_1, s_2, q_2) > 0$ such that

$$\|u\|_{s,q} \le c_{GN} \|u\|^\theta_{s_1,q_1} \|u\|^{1-\theta}_{s_2,q_2},$$

for all $u \in W^{s_1,q_1}(\Omega) \cap W^{s_2,q_2}(\Omega)$ as long as the following condition fails: s_2 is an integer ≥ 1, $q_2 = 1$ and $s_2 - s_1 \le 1 - 1/q_1$.

Proof. See Theorem 1 of [19] for details. □

Corollary 2.3 (Interpolation in H^s)**.** *Let $\|\cdot\|_s$, $s \ge 0$ denote the $H^s(\Omega) = W^{s,2}(\Omega)$ norm, where Ω satisfies the assumptions of Lemma 2.2. Let $0 \le \theta \le 1$ and $s = \theta s_1 + (1-\theta) s_2$. There exists $c_{GN} = c_{GN}(s_1, s_2) > 0$ such that*

$$\|u\|_s \le c_{GN} \|u\|^\theta_{s_1} \|u\|^{1-\theta}_{s_2},$$

for all $u \in H^{max(s_1,s_2)}(\Omega)$.

Proof. The result follows from Lemma 2.2, setting $q_1 = q_2 = 2$ and using standard Sobolov embedding theory. □

Lemma 2.4. *Let $1/2 < s \leq 1$ and let $g \in H^s(\partial\Omega_j \setminus \partial\Omega_S)$ satisfy $g|_{\partial(\Omega_j \cap \Gamma)} = 0$. Then, the function $\widetilde{g} : \partial D_j \to \mathbb{R}$ defined by*

$$\widetilde{g}(x) = \begin{cases} g(x) & x \in \partial(\Omega_j \cap \Gamma), \\ 0 & x \in \partial D_j \setminus \partial(\Omega_j \cap \Gamma), \end{cases}$$

belongs to $H^s(\partial D_j)$.

Proof. The proof is quite basic, for the full proof we refer e.g. to Lemma 5 of [18]. □

Lemma 2.5. *Let $g \in H_0^1(\Gamma_j)$, then there exists $\phi \in H^1(\Omega_j)$ satisfying $\phi|_{\Gamma_j} = g$ such that*

$$|\phi|_{H^1(\Omega_j)} \leq c_{GN} C_j \|g\|^{1/2}_{L^2(\Gamma_j)} \|g\|^{1/2}_{H^1(\Gamma_j)},$$

where C_j depends only on D_j.

Proof. In view of Lemma 2.4, we can extend g on ∂D_j by zero so that the extension $\widetilde{g}$ is in $H^1(\partial D_j)$. Then, there exists $\widetilde{\phi} \in H^1(\Omega_j)$ such that $\widetilde{\phi} = \widetilde{g}$ on ∂D_j and

$$|\widetilde{\phi}|_{H^1(D_j)} \leq C_j \|\widetilde{g}\|_{H^{1/2}(\partial D_j)}.$$

Using Sobolev interpolation, we have

$$\begin{aligned} |\widetilde{\phi}|_{H^1(D_j)} &\leq c_{GN} C_j \|\widetilde{g}\|^{1/2}_{L^2(\partial D_j)} \|\widetilde{g}\|^{1/2}_{H^1(\partial D_j)}, \\ &= c_{GN} C_j \|g\|^{1/2}_{L^2(\Gamma_j)} \|g\|^{1/2}_{H^1(\Gamma_j)}. \end{aligned}$$

The result follows by setting $\phi = \widetilde{\phi}|_{\Omega_j}$. □

With preliminary lemmas established, we present the following proposition, which is a novel contribution of this article.

Proposition 2.6 (Interpolation error estimate)**.** *Let $(\gamma_l)_{l=1,\ldots,N_\gamma}$ be a finite nonoverlapping partitioning of Γ and v the element of $H^1_\Delta(\Omega)$ such that the traces of v belong to $H^{p+1}(\gamma_l)$ for all l and some $p = 1, 2, \ldots$. Assume in addition that the set of coarse edges $(e_k)_k$ is a subdivision of $(\gamma_l)_l$. Then, there exists $\phi \in H^1(\Omega)$ such that $\phi|_\Gamma \in V^\Gamma_{H,p}$ and satisfying*

$$|v - \phi|_{H^1(\Omega)} \leq C H^{p+\frac{1}{2}} \left(\sum_{l=1}^{N_\gamma} |v|^2_{H^{p+1}(\gamma_l)} \right)^{1/2}. \tag{2.29}$$

Proof. Since $v \in H^1(\gamma_l)$, it follows that $v \in C^0(\overline{\gamma_l})$. However, being in $H^{1/2}(\Gamma)$, v can not have discontinuities. Therefore $v \in H^1(\Gamma)$ and thus is continuous on $\overline{\Gamma}$.

Let $\mathcal{I}^p_\Gamma$ be an interpolation operator from $C^0(\Gamma)$ to $V^\Gamma_{H,p}$ such that $\mathcal{I}^p_\Gamma v|_{e_k} = \mathcal{I}^p_k v$ for all k, where $\mathcal{I}^p_k$ is defined in Lemma 2.1. Let $E_\Gamma = v|_\Gamma - \mathcal{I}^p_\Gamma(v|_\Gamma)$. Since $E_\Gamma(x) = 0$

for every $x \in \mathcal{V}$, it follows from Lemma 2.5 that there exists $\psi_j \in H^1(\Omega_j)$ satisfying $\psi_j|_\Gamma = E_\Gamma$ such that

$$|\psi_j|_{H^1(\Omega_j)} \leq c_{GN} C_j \|E_\Gamma\|^{1/2}_{L^2(\Gamma_j)} \|E_\Gamma\|^{1/2}_{H^1(\Gamma_j)}.$$

Let $\phi \in H^1(\Omega)$ be defined as $\phi|_{\Omega_j} = v|_{\Omega_j} - \psi_j|_{\Omega_j}$ for all Ω_j. We have

$$\begin{aligned} |v-\phi|_{H^1(\Omega_j)} &\leq c_{GN} C_j \|E_\Gamma\|^{1/2}_{L^2(\Gamma_j)} \|E_\Gamma\|^{1/2}_{H^1(\Gamma_j)}, \\ &\leq c_{GN} C_j \left(\sum_l ||E_{\Gamma,j}||^2_{L^2(\gamma_l \cap \Gamma_j)} \right)^{1/4} \left(\sum_l \|E_{\Gamma,j}\|^2_{H^1(\gamma_l \cap \Gamma_j)} \right)^{1/4}. \end{aligned}$$

Since $(e_k)_k$ is a subdivision of $(\gamma_l)_l$, we deduce from Lemma 2.1

$$|v-\phi|_{H^1(\Omega_j)} \leq c_{GN} c_0 c_1 C_j \, H^{p+1/2} \left(\sum_l |u|^2_{H^{p+1}(\gamma_l \cap \Gamma_j)} \right)^{1/2}. \tag{2.30}$$

By summing (2.30) over all Ω_j, we obtain (2.29) with $C = \sqrt{2} c_{GN} c_0 c_1 \max_j C_j$. □

Proposition 2.7 (Best approximation)**.** *The solution of* (2.28) *satisfies*

$$|u_\Delta - u_{\Delta,H}|_{H^1(\Omega)} \leq |u_\Delta - \phi|_{H^1(\Omega)},$$

for any $\phi \in H^1(\Omega)$ *such that* $\phi|_\Gamma \in V^\Gamma_{H,p}$.

Proof. We first remark that, because $u_{\Delta,H}$ is the projection of u_Δ on $V_{H,p}$, it satisfies

$$|u_\Delta - u_{\Delta,H}|_{H^1(\Omega)} \leq |u_\Delta - v_H|_{H^1(\Omega)}, \tag{2.31}$$

for any $v_H \in V_{H,p}$. Given some $\phi \in H^1(\Omega)$ such that $\phi|_\Gamma \in V^\Gamma_{H,p}$, we set $v_H \in V^\Gamma_{H,p}$ be such that $v_H|_\Gamma = \phi|_\Gamma$. Since $u_\Delta - v_H \in H^1_\Delta(\Omega)$, we have that

$$|u_\Delta - v_H|_{H^1(\Omega)} \leq |u_\Delta - \phi|_{H^1(\Omega)}, \tag{2.32}$$

that is to say that $u_\Delta - v_H$ is the minimizer of $\psi \mapsto |\psi|_{H^1(\Omega)}$ over the set $\{\psi \in H^1(\Omega_j) \,|\, \psi = u_\Delta - v_H \text{ on } \Gamma_i\}$. The result follows by combining (2.31) and (2.32). □

From the propositions 2.7 and 2.6, we deduce the following error estimate regarding the locally harmonic part of the solution.

Theorem 2.8. *Let* u *be a solution of* (2.3) *and* u_Δ *its* H^1 *projection on* $H^1_\Delta(\Omega)$. *Let* $u_{\Delta,H}$ *satisfy* (2.28). *Then, under the assumptions of Proposition 2.6 with* u_Δ *instead of* v,

$$|u_\Delta - u_{\Delta,H}|_{H^1(\Omega)} \leq C H^{p+\frac{1}{2}} \left(\sum_{l=1}^{N_\gamma} |u|^2_{H^{p+1}(\gamma_l)} \right)^{1/2}.$$

Remark 2.9. Theorem 2.8 expresses the approximation properties of $V_{H,p}$ in $H^1_\Delta(\Omega)$, or the approximation of u_Δ by $u_{\Delta,H}$. In addition, the estimate of the theorem holds for $u - (u_{\Delta,H} + u_b)$, where u_b satisfies (2.26). However, we remark that $u_{\Delta,H}$ also provides a low-order approximation of u. Indeed, in view of the orthogonality between $H^1_\Delta(\Omega)$ and $H^1_\Gamma(\Omega)$, we have

$$|u - u_{\Delta,H}|^2_{H^1(\Omega)} = |u_\Delta - u_{\Delta,H}|^2_{H^1(\Omega)} + |u_b|^2_{H^1(\Omega)}.$$

While the first term in the right-hand side can be estimated based on Theorem 2.8, the H^1 norm of u_b can be controlled using (2.26) and local Poincaré inequalities. More precisely, we have

$$|u_b|_{H^1(\Omega_j)} \leq C_{P,j}\mathrm{diam}(\Omega_j)\|f\|_{L^2(\Omega_j)},$$

where $C_{P,j}$ denote the Poincaré constants associated to Ω_j.

Remark 2.10. The broken H^{k+1} norm in the right-hand side of (2.29) involves only the traces of the solution along the sections of the coarse skeleton. Therefore, this estimate is valid for u having low general regularity that is due, for example, to corner singularities. As a matter of fact, the estimate (2.29) provides an a priori criterion for the adaptation of the coarse mesh: one has to ensure that the edge norm in the right-hand side is small. For a sufficiently regular right-hand side f, this can be achieved by moving the coarse edges away from the "bad" perforation corners.

We further note that Theorem 2.8 is especially valuable for a so-called space or edge refinement, which is a procedure that involves splitting the edges of an otherwise fixed coarse grid. In that case, one observes superconvergence of the error with a rate of $p + \frac{1}{2}$.

2.3 Discrete Trefftz Space and Two-level Schwarz Method

We begin with the algebraic formulation and implementation of the problem. Let us consider a triangulation $\mathcal{T}$ of Ω which is assumed to be conforming with respect to the polygonal partitioning $(\Omega_j)_{j=1,\dots,N}$. We denote by $\mathcal{N}_{\overline{\Omega}}$ the set of the triangulation nodes. In order to account for Dirichlet boundary condition imposed on $\partial\Omega \setminus \partial\Omega_S$, we introduce a set of internal nodes

$$\mathcal{N}_\Omega = \{\ \mathbf{x}_s \in \mathcal{N}_{\overline{\Omega}} |\ \ \mathbf{x}_s \notin \partial\Omega \setminus \partial\Omega_S\}.$$

The total number of nodes in $\mathcal{N}_\Omega$ is denoted by N_Ω.

We denote by V_h the space of functions that are continuous and triangle-wise polynomial on each triangle of $\mathcal{T}$, where h denotes the maximal element diameter, and by $V_{h,0}$ the space $V_h \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$. The finite element method for the variational problem (2.3) consists in finding $u_h \in V_{h,0}(\Omega)$ such that

$$(\nabla u_h, \nabla v_h)_{L^2(\Omega)} = (f, v_h)_{L^2(\Omega)} \qquad \forall\, v_h \in V_{h,0}(\Omega). \tag{2.33}$$

Let us denote by $(\eta_{\mathbf{s}})_{\mathbf{s}\in\mathcal{N}_{\overline{\Omega}}}$ the set of the standard finite element basis functions associated to the nodal degrees of freedom. The associated "fine-scale" finite element discretization results in the linear system

$$A\mathbf{u} = \mathbf{f}, \tag{2.34}$$

where A is a $N_\Omega \times N_\Omega$ matrix with elements

$$A_{ij} = a(\eta_{\mathbf{s}_j}, \eta_{\mathbf{s}_i}) = \int_\Omega \nabla\eta_{\mathbf{s}_j} \cdot \nabla\eta_{\mathbf{s}_i}\, \mathrm{d}\mathbf{x},$$

and $\mathbf{f}$ is the vector from $\mathbb{R}^{N_\Omega}$ with components $f_j = \int_\Omega f\eta_{\mathbf{s}_j}\, \mathrm{d}\mathbf{x}$.

Because the triangular mesh resolves the fine-scale structure of the domain, the system may be quite large and the size of the triangular elements may vary by several orders of magnitude. As a result, the matrix A is expected to be poorly conditioned.

2.3.1 Discrete Trefftz Approximation

In most practical situations, the coarse basis functions defined by (2.27) can not be computed analytically. Therefore, we consider the finite element approximation of $V_{H,p}$. The basis of the discrete Trefftz space is obtained through the finite element approximation of (2.27). The finite element discretization of (2.27) results in the system of the form

$$\tilde{A}_j \boldsymbol{\phi}_s^j = \mathbf{b}_s^j,$$

where $\tilde{A}_j$ is the local stiffness matrix and $\mathbf{b}_s^j$ accounts for the Dirichlet boundary data in (2.27). Let $\overline{R}_j$ denote the boolean restriction matrices corresponding to the nodes of $\overline{\Omega_j}$, and let $\boldsymbol{\phi}_s$ be a global vector which, when restricted to a subdomain Ω_j, returns $\boldsymbol{\phi}_s^j$. That is,

$$\boldsymbol{\phi}_s = \sum_{j \in \mathcal{N}_s} \overline{R}_j^T \overline{D}_j \boldsymbol{\phi}_s^j,$$

for $s = 1, \ldots, N_{\mathcal{V}}$, where $\overline{D}_j$ are partition of unity matrices corresponding to $\overline{\Omega_j}$ and $\mathcal{N}_s = \{\, j \mid \mathbf{x}_s \text{ is contained in } \Omega_j \,\}$. Discretely, the coarse transition matrix R_H is such that the kth row of R_H is given by $\boldsymbol{\phi}_k^T$ for $k = 1, \ldots, N_{\mathcal{V}}$.

The finite element counterpart of (2.28) and therefore the discrete coarse approximation from Section 2.2.2 can be expressed algebraically as

$$\mathbf{u}_{\Delta,H} = M_H^{-1} \mathbf{f}, \tag{2.35}$$

where

$$M_H^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H.$$

2.3.2 Discrete Two-level Schwarz Method

Now, we will show how the coarse approximation introduced in the previous section can be combined with the RAS method to construct a simple yet efficient iterative linear solver for the fine-scale finite element method.

Let $\left(\Omega_j'\right)_{j=1,\ldots,N}$ denote the overlapping partitioning of Ω such that $\Omega_j \subset \Omega_j'$.

We denote by R_j the boolean restriction matrices corresponding to the degrees of freedom of the overlapping subdomains Ω_j'. Here, the boolean matrices R_j are of size $(N_j \times N_\Omega)$, where N_j denotes the number of degrees of freedom in Ω_j' and N_Ω denotes the number of degrees of freedom in Ω.

Given this framework, recall the following iterative procedure. Generally, we take the coarse approximation as the initial iterate $\mathbf{u}^0 = \mathbf{u}_{\Delta,H}$. This allows us to begin with the accuracy of the coarse approximation and continue to iterate to finite element precision. With this, the iteration is given by

$$\begin{aligned} \mathbf{u}^{n+\frac{1}{2}} &= \mathbf{u}^n + M_{RAS}^{-1}(\mathbf{f} - A\mathbf{u}^n), \\ \mathbf{u}^{n+1} &= \mathbf{u}^{n+\frac{1}{2}} + M_H^{-1}(\mathbf{f} - A\mathbf{u}^{n+\frac{1}{2}}), \end{aligned} \tag{2.36}$$

where

$$M_{RAS}^{-1} = \sum_{j=1}^{N} R_j^T D_j (R_j A R_j^T)^{-1} R_j,$$

and D_j denote the partition-of-unity matrices satisfying

$$I_N = \sum_{j=1}^{N} (R_j)^T D_j (R_j).$$

Additionally, we introduce the preconditioned system

$$M_{RAS,2}^{-1} A\mathbf{u} = M_{RAS,2}^{-1}\mathbf{f}, \tag{2.37}$$

where the two-level discrete RAS preconditioner is given additively by

$$M_{RAS,2}^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H + \sum_{j=1}^{N} R_j^T D_j (R_j A R_j^T)^{-1} R_j.$$

We expect the use of a well-constructed preconditioner to provide acceleration in terms of Krylov iteration count. Generally, using preconditioned Krylov methods produces faster convergence than the iterative solver, both in terms of computation time and iterations.

We note that iteration (2.36) can be written as

$$\mathbf{u}^{n+1} = \mathbf{u}^n + (M_H^{-1} + M_{RAS}^{-1}(I - AM_H^{-1}))(\mathbf{f} - A\mathbf{u}^n), \tag{2.38}$$

a hybrid method. As mentioned in the introduction, we implement this hybrid iterative method as an additive iterative method will not converge. However, we choose to use the additive two-level RAS preconditioner (2.37) for Krylov methods as it is commonly used in domain decomposition literature and has an increased capacity for parallel computing. With this, we remark that we could use a hybrid method as a Krylov accelerator and that the Schwarz preconditioners presented here are not exhaustive.

In the context of domain decomposition, our coarse approximation is referred to as a "coarse space" for the Schwarz methods. However, we remark that the construction of the matrix form of the coarse approximation and coarse space are identical, with the difference being solely in application.

2.4 Numerical Results

In this section, we illustrate the performance of the discrete Trefftz space in three different scenarios involving either a standalone Galerkin approximation (2.28), an iterative approach (2.36), or a preconditioner approach (2.37).

2.4.1 Trefftz Coarse Approximation on L-Shaped Domain

To properly display the approximation properties, it is helpful to have an exact solution to the equation. However, the generation of this exact solution is difficult with multiple singularities. To combat this issue, we can use a model domain with one singularity/corner in the domain; an example of this would be an L-shaped domain with a reentering corner (Figure 2.6). The domain is defined by $D = (-1,1)^2$, $\Omega_S = (0,1)^2$ and $\Omega = D \setminus \overline{\Omega_S}$. We consider the problem (2.1) with zero

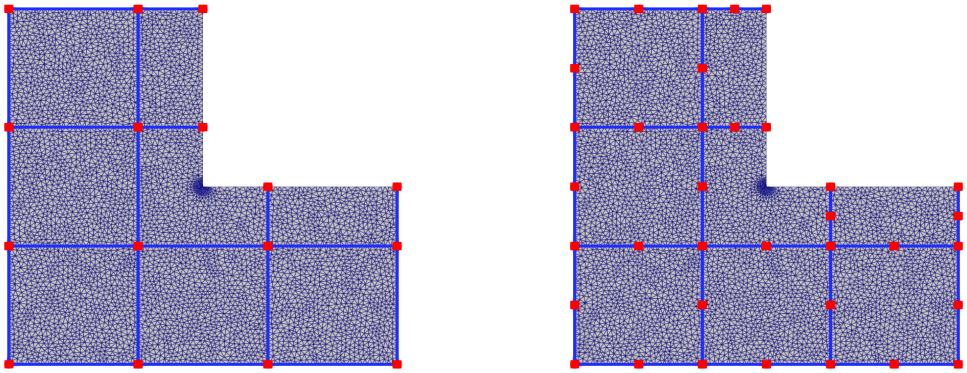


Figure 2.6: Coarse and fine discretizations of the L-shaped domain with 0 (left) and 1 (right) additional degree(s) of edge refinement.

right-hand side and a non-homogeneous Dirichlet boundary condition on $\partial\Omega \setminus \partial\Omega_S$ provided by the singular exact solution $u(r,\theta) = r^{\frac{2}{3}} \cos(\frac{2}{3}(\theta - \pi/2))$.

In order to assess the convergence of the discrete Trefftz method, we consider two strategies regarding the refinement of the coarse partitioning. The procedure involving the reduction of the diameter of the coarse cells will be referred to as *mesh refinement*. The sequence of such meshes will be constructed as follows: first the background domain D is partitioned into $N = (2k+1)^2$, $k \in \mathbb{N}$, squares, then the nonoverlapping coarse cells Ω_j are generated by excluding Ω_S. The choice of N being a square of an odd number ensures the consistency of the mesh sequence in terms of the shape of the elements. The second considered refinement strategy will be referred to as *edge refinement* procedure, which involves subdividing the edges of an original "3×3 grid". This edge refinement approach is illustrated by Figure 2.6 and is inspired by the Multiscale Hybrid-Mixed method [5]. We remark that with both refinement procedures, it is ensured that none of the coarse grids will have a degree of freedom located at the corner $(0,0)$. As a result, the corner singularity will be captured by the basis functions associated with the L-shaped domain. We also note that in order to improve the precision of the fine-scale finite element method, the size of the triangles is graded in the vicinity of the corner $(0,0)$.

Figure 2.7 reports, for both mesh and edge refinement strategies, the relative error in H^1 semi-norm and L^2 norm as functions of maximal coarse edge length H. The black dashed line represents the typical fine-scale finite element error; here, we use $\mathbb{P}_2$ finite elements. The relative error and the experimental order of convergence for edge refinement procedure are equally reported in Table 2.1. In accordance with the error estimate (2.29), for the edge refinement procedure, we observe superconvergence in the energy norm with rates of approximately $3/2$ and $5/2$ for $p = 1$ and $p = 2$, respectively. In the L^2 norm, the observed convergence rates are approximately 3 and $7/2$ for $p = 1$ and $p = 2$, respectively. In contrast to the edge refinement convergence, the convergence resulting from the mesh refinement seems to be controlled by the low global regularity of the solution. We observe the convergence rates typical for finite element methods on quasi-uniform meshes, that is, close to $2/3$ and $4/3$ in H^1 and L^2 norms, respectively.

	H^1 semi-norm				L^2 norm			
Ref. lvl.	$p=1$	eoc	$p=2$	eoc	$p=1$	eoc	$p=2$	eoc
0	-1.098	-	-1.812	-	-1.881	-	-2.864	-
1	-1.601	1.669	-2.665	2.833	-2.762	2.925	-4.026	3.858
2	-2.132	1.765	-3.35	2.274	-3.677	3.042	-4.993	3.215
3	-2.626	1.641	-4.043	2.305	-4.514	2.781	-6.014	3.389

Table 2.1: Relative error and the experimental order of convergence for the Trefftz approximation with $p = 1, 2$ and using the edge refinement. Ref. level refers to the degree of additional edge refinement. Eoc refers to error (rate) of convergence.

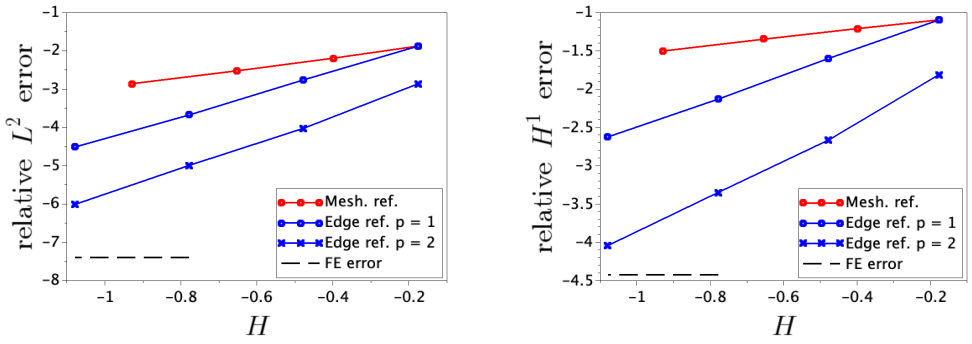


Figure 2.7: Coarse approximation error for L-shaped domain with edge (blue) and mesh (red) refinement in L^2 norm (left) and the energy norm (right). The black dashed line denotes the finite element error. For the edge refinement procedure, we observe convergence rates of approximately $3/2$ and $5/2$ for $p = 1$ and $p = 2$, respectively. In the L^2 norm, the observed convergence rates are approximately 3 and $7/2$ for $p = 1$ and $p = 2$, respectively. For the mesh refinement procedure, we observe convergence rates of approximately $2/3$ and $4/3$ in the H^1 and L^2 norms, respectively.

2.4.2 Iterative DD and Preconditioned Krylov methods on L-Shaped Domain

We report on Figure 2.8 the convergence history of the iterative methods for the L-shaped domain. More precisely, we report two different error metrics, referred to as algebraic error and full error. For a given iterate, algebraic error is defined as a distance (based on L^2 or H^1 norms) between a given iterate and the solution of the algebraic system (2.34), while full error reflects the distance to the exact solution of (2.1). Figure 2.8 reports the convergence histories for linear systems resulting from an increasingly accurate background finite element discretization. The overlap is set to $\frac{1}{20}\mathcal{H}_j$, where $\mathcal{H}_j = \max(x_{max,j} - x_{min,j}, y_{max,j} - y_{min,j})$ and $x_{min,j}, y_{min,j}, x_{max,j}, y_{max,j}$ denote the minimal and maximal x and y coordinates that are contained in Ω_j. We consider Trefftz spaces of order $p = 1$ and $p = 2$.

Because the width of the overlap is maintained constant, we observe on Figure 2.8 that the convergence the iterative methods is not sensitive to the accuracy of the background finite element method until the precision of the latter is reached. The convergence of preconditioned GMRES method is exponential, while the convergence curve of the algebraic error of the two-level fixed point method (2.36) exhibits two distinct slopes, with fast convergence at the initial stage. We note that the performance of both the stationary iteration and the preconditioned GMRES algorithm is improved by the higher order Trefftz space. We note that the initial slope of the stationary iteration method is approaching the slope of the preconditioned GMRES as the Trefftz order increases. In general, the results obtained in terms of the full error (right column of Figure 2.8) seem to point toward the following conclusion: the stationary iteration method appears to be an acceptable alternative to the preconditioned GMRES if very high accuracy is not required.

We further study the impact of edge refinement on the performance of the iterative methods. We report on Figure 2.9 the convergence of algebraic error in H^1 and L^2 norms for edge refinement up to degree three. This is reported for both the iterative and preconditioner approaches. We observe that reducing H not only improves the initial coarse approximation (the first iteration point), but also accelerates the convergence of the iterative methods. The most notable is the impact on the initial slope of the two-level iterative RAS method. Here, again, the initial slope of the stationary iteration method comes close to that of the preconditioned GMRES as we increase the order of edge refinement.

2.4.3 Iterative DD and Preconditioned Krylov Methods on Small Urban Domain

We examine the performance of the two-level stationary iteration and preconditioned GMRES method over a domain based on realistic urban geometries for which the data sets were kindly provided by Métropole Nice Côte d'Azur. We focus here on a relatively small spatial frame shown in Figure 2.10). The data frame is 160×160 meters and contains two kinds of structural features representing buildings (and assimilated small elevated structures) and walls in urban data. The number of perforations associated to buildings and walls is 63 and 77, respectively. This "small" model domain would take a minimum of 23 000 degrees of freedom to triangulate without imposing maximum triangle area or constraining the mesh to match the coarse partitioning. While the fine-scale mesh will change depending on N, the number of degrees of freedom in the mesh will be close to this number. Due to the

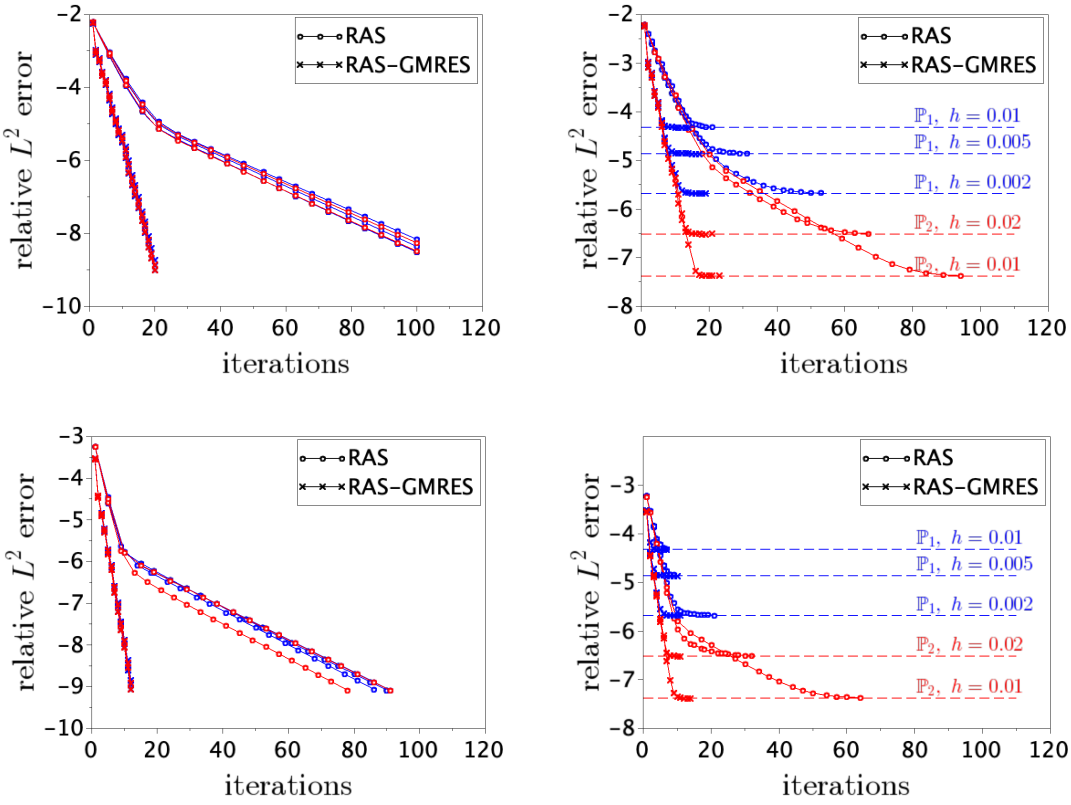


Figure 2.8: Convergence in L^2 of the two-level iterative (RAS) and preconditioned GMRES (RAS-GMRES) methods for the L-shaped domain on 5×5 subdomains with overlap of $\frac{1}{20}\mathcal{H}_j$ and using first (top row) and second (bottom row) order Trefftz approximation. The dashed horizontal lines show the error of the fine-scale finite element method. Left to right: algebraic error, full error. Results for finite elements of order 1 and 2 are shown in blue and red, respectively.

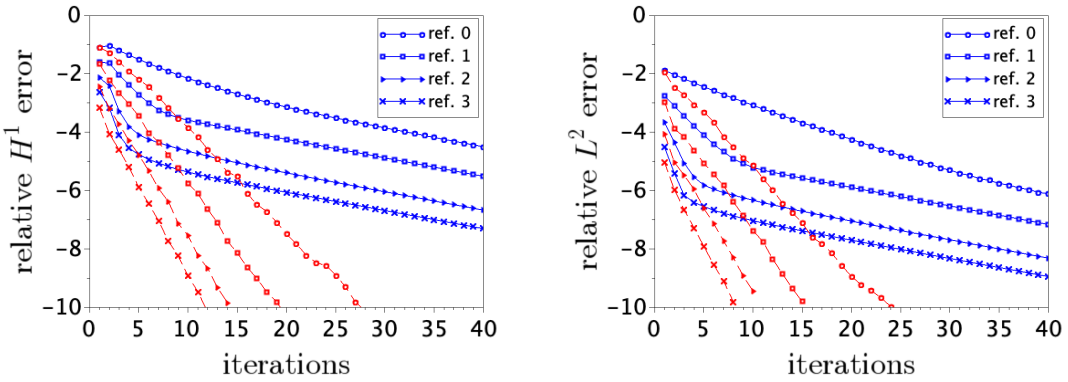


Figure 2.9: Convergence in H^1 (left) and L^2 (right) norms of the two-level iterative (blue) and preconditioned GMRES (red) methods for the L-shaped domain on 3×3 subdomains with overlap of $\frac{1}{20}\mathcal{H}_j$. Algebraic errors are shown for the first order Trefftz space with various degrees of edge refinement.

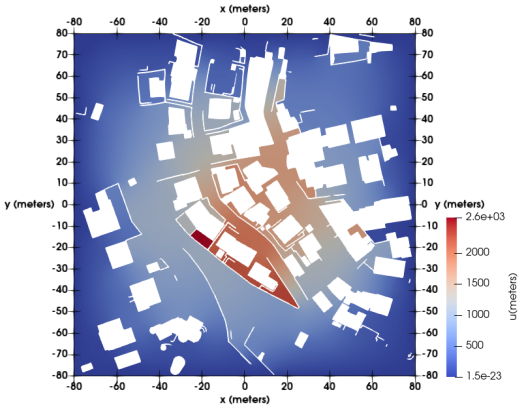


Figure 2.10: Finite element solution with $f = 1$ on a smaller urban model domain.

geometric complexity of the computational domain, the mesh includes a number of very small triangles; the minimal triangle area is given as 5.83×10^{-13}. We consider the model problem (2.1) with $f = 1$, which we discretize with continuous piece-wise affine finite elements; the finite element solution is reported in Figure 2.10.

We report on Figures 2.11 and 2.12 the convergence history of the stationary iterative method for different sizes of overlap. This figure shows results for both algebraic and full errors for various fine-scale discretization sizes h. The performance of the preconditioned GMRES is reported on Figures 2.13 and 2.14. In lieu of a true solution (which would exhibit multiple singularities), we provide a fine grid numerical solution to which the iterates are compared. The grid for the reference solution is generated via multiple levels of edge bisection ("red refinement"). We note that in contrast to the previous numerical experiment, we do not perform any mesh grading near the reentering corners of the domain; as a result, the finite element error is high.

We expose on Figure 2.11 the convergence history of the two-level stationary iteration method with overlap $\frac{1}{20}\mathcal{H}_j$. We observe that the convergence of the algebraic error is robust with respect to h and that the fine-scale finite element method (black lines) can be reached in a few steps. Further iterations do not improve the overall precision of the approximate solution even though the algebraic error may decrease. A stopping criterion which prevents excess iterations is warranted; such a posterioiri estimates are discussed for example in [6, 67]. As expected, the performance of the stationary iteration method considerably deteriorates (see Figure 2.12) in the case of minimal geometric overlap.

Figures 2.13 and 2.14 report the performance of the two-level preconditioned GMRES method for overlaps of $\frac{1}{20}\mathcal{H}_j$ and minimal geometric overlap, respectively. Compared to the stationary iterative method, a major improvement is achieved for the case of minimal geometric overlap. However, due to the low accuracy of the background finite element discretization, both the iterative and preconditioned GMRES methods perform comparably for the overlap of $\frac{1}{20}\mathcal{H}_j$.

2.4.4 Weak Scalability Tests for Preconditioned Krylov Method on Manufactured Perforations

We proceed with weak scalability tests for the two-level RAS method with the Nicolaides and spectral coarse spaces (given by (2.21) and (2.20)) compared to the Trefftz space. To keep each subdomain uniform for weak scalability tests, we

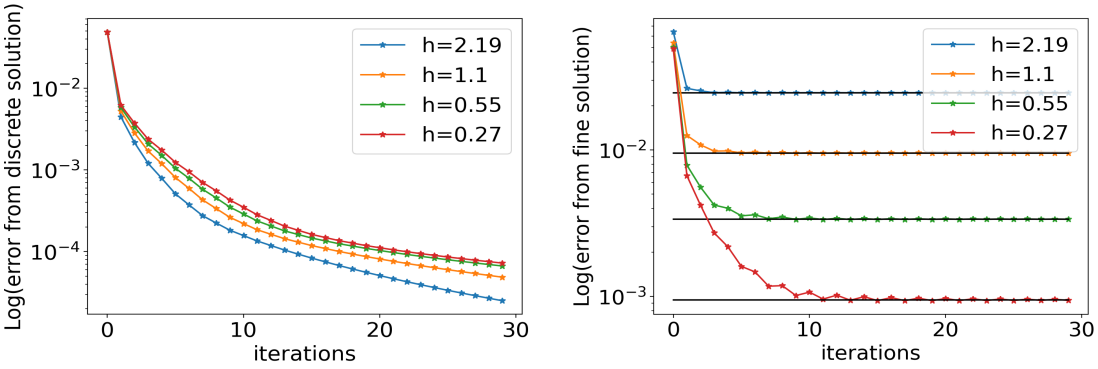


Figure 2.11: Convergence of the two-level iterative method for the urban data set on 8×8 subdomains with overlap $\frac{1}{20}\mathcal{H}_j$. The black horizontal lines show the error of the fine-scale finite element method. Left to right: algebraic error, full error.

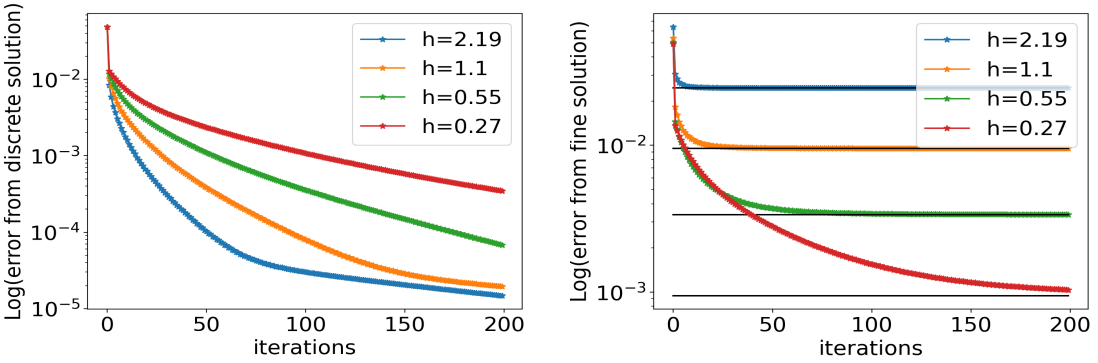


Figure 2.12: Convergence of the two-level iterative method for the urban data set on 8×8 subdomains with minimal geometric overlap. The black horizontal lines show the error of the fine-scale finite element method. Left to right: algebraic error, full error.

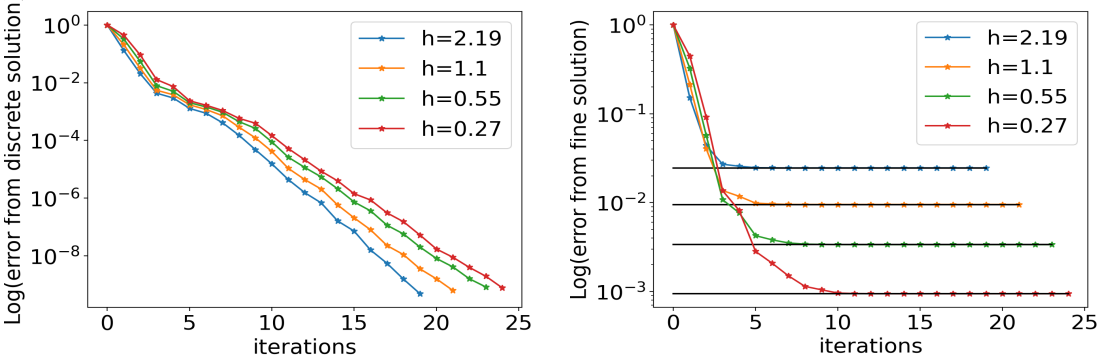


Figure 2.13: Convergence of the two-level preconditioned Krylov method for the urban data set on 8×8 subdomains with overlap $\frac{1}{20}\mathcal{H}_j$. The black horizontal lines show the error of the fine-scale finite element method. Left to right: algebraic error, full error.

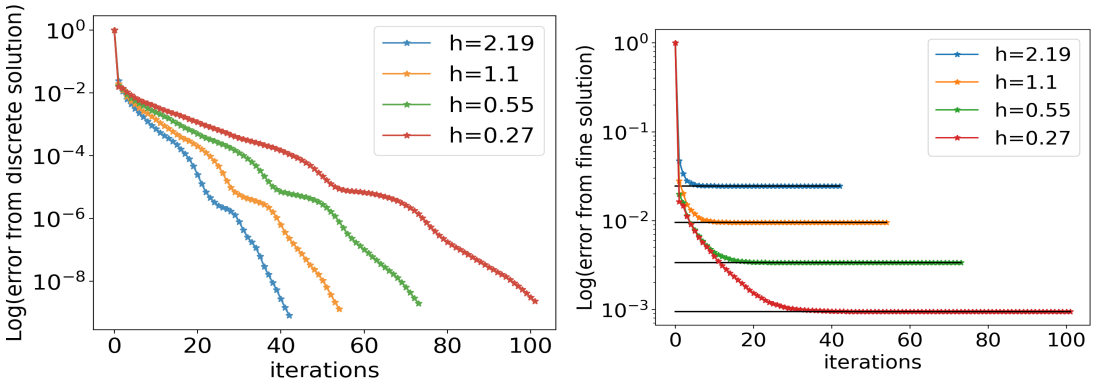


Figure 2.14: Convergence of the two-level preconditioned Krylov method for the urban data set on 8×8 subdomains with minimal geometric overlap. The black horizontal lines show the error of the fine-scale finite element method. Left to right: algebraic error, full error.

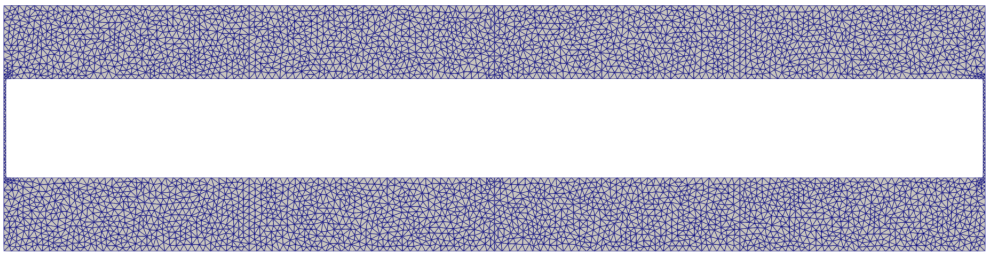


Figure 2.15: Visualization of manufactured T0 heterogeneity for $N = 4$ subdomains.

proceed with simple, manufactured perforated domains before moving to realistic, urban domains. This section serves to display the weak scalability of the Trefftz space and directly compare its scalability to that of other popular coarse spaces. Here, we begin with a strip of N unit square subdomains in one direction. Here, the subdomain size remains constant and as N increases the total problem size increases as well. We then include one long perforation that cuts through all but the two "end" (leftmost and rightmost) subdomains; see for example Figure 2.15. This type of perforation and model domain is referred to as a T0 heterogeneity. For this experiment, the overlap is again set to $\frac{1}{20}\mathcal{H}_j$. We note that in these examples, we solve the model Poisson problem but only pose Dirichlet boundary conditions on the along the leftmost and rightmost edges of Ω. Convergence results are shown in Table 2.2, recording preconditioned GMRES iterations and dimension for the two-level RAS preconditioner with both the Nicolaides and Trefftz coarse spaces; dimension refers to the number of columns in the coarse matrix R_H^T. For the Nicolaides space, we include results for both the "traditional" and "enriched" Nicolaides spaces, where traditional Nicolaides refers to Nicolaides without any further partitioning of the subdomains. With this, for a T0 heterogeneity, the enriched Nicolaides space will have a dimension which is about double that of the traditional Nicolaides space.

We see from Table 2.2 that in the case of a T0 heterogeneity, the traditional Nicolaides coarse space is unable to provide scalability with respect to iteration count; that is, the number of iterations increases significantly as N increases. We also remark that the while enriched Nicolaides is significantly better than traditional

N	Trad. Nic		Enr. Nic		Trefftz	
	it.	dim	it.	dim	it.	dim
4	15	4	15	6	12	16
8	40	8	38	14	12	32
16	70	16	53	30	11	64
32	115	32	59	62	11	128
64	194	64	59	126	10	256

Table 2.2: Preconditioned GMRES iterations (it.) and dimension (dim.) for the traditional Nicolaides, Enriched Nicolaides, and Trefftz-like coarse space. Each subdomain contains around 270 mesh points.

Nicolaides, the Trefftz space provides additional acceleration in terms of iteration count when compared to enriched Nicolaides.

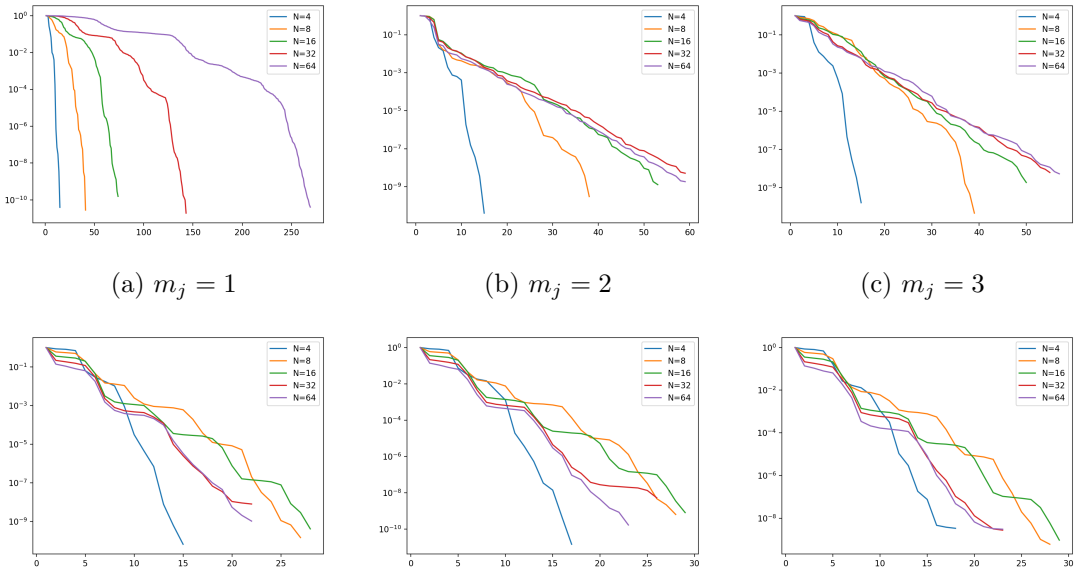
Furthermore, we provide results for the spectral GenEO-like coarse space for various values of eigenvalues taken in Figure 2.16. We see from Figure 2.16 that taking $m_j = 2$ eigenvalues results in robustness. We can think of the $m_j = 2$ case as equivalent to enriched Nicolaides, as this will take all eigenvalues clustered around 0. We also see that taking $m_j = 4$ eigenvalues, such that two eigenvalues are taken per connected component, results in an additional acceleration. However, the acceleration does not exceed the acceleration obtained from the Trefftz coarse space.

For reference, plots of the subdomain and its corresponding eigenvalues for the GenEO-like coarse space are shown in Figure 2.17. We can see clearly from the clustered eigenvalues that at least two eigenvalues (or one eigenvalue per connected component) is needed for robustness. The clustering of eigenvalues also matches with the numerical results which suggest we need two eigenvalues per connected component for an additional acceleration. Taking more than two eigenvalues per connected component does not appear to have any additional effect on the Krylov acceleration.

2.4.5 Scalability Tests for Preconditioned Krylov Method on Larger Urban Domains

As a final numerical experiment, we follow the experiment from the previous subsection and report the performance of the two-level preconditioned GMRES method on a larger data set shown on Figure 2.18. This larger data set contains 306 buildings and 477 walls of varying sizes, the dimensions of the domain are 640×640 meters. The "large" model domain (with walls) would take a minimum of 91 945 degrees of freedom to triangulate without imposing maximum triangle area or constraining the mesh to match the coarse partitioning. The minimal triangle area is observed to be 5.83×10^{-13}.

We note that depending on the partitioning of the domain, the perforations resulting from this data set (especially the wall data) can span across multiple



(a) $m_j = 1$ (b) $m_j = 2$ (c) $m_j = 3$

(d) $m_j = 4$ (e) $m_j = 5$ (f) $m_j = 10$

Figure 2.16: GMRES iterations for the GenEO-like coarse space for various numbers of eigenvalues m_j. Each subdomain contains around 270 mesh points.

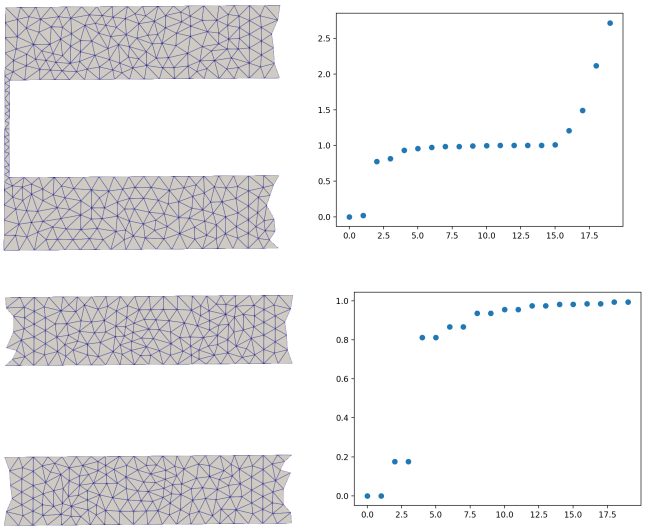
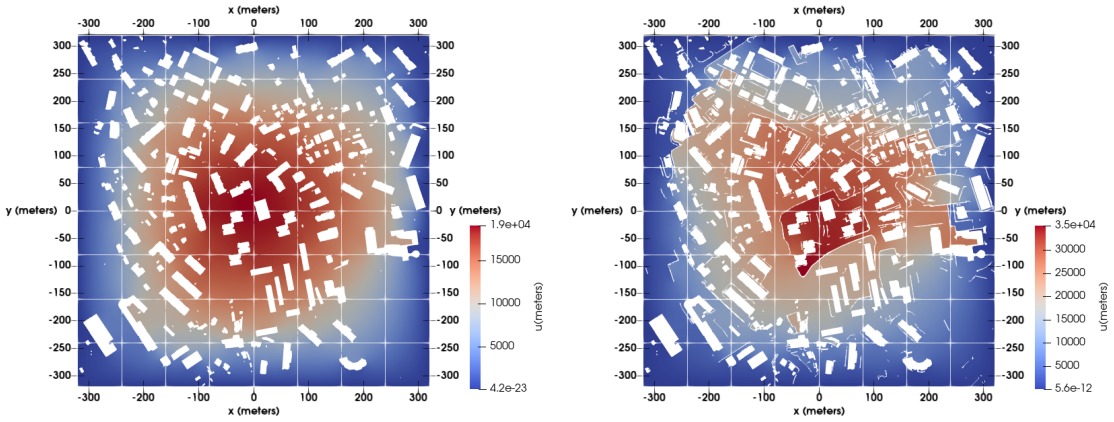


Figure 2.17: Subdomains (left) and their corresponding eigenvalues (right) for the GenEO-like coarse space.



(a) Without walls (b) With walls

Figure 2.18: Approximate finite element solution over an urban perforated domain divided in $N = 8 \times 8$ nonoverlapping subdomains.

coarse cells, which is a challenging situation for traditional coarse spaces. To further explore this observation, we will provide numerical results for data sets that do and do not include walls as perforations. Figure 2.18 reports the finite element solution obtained for a data set excluding and including walls. We can see from Figure 2.18 that the inclusion of walls noticeably affects the numerical solution.

For the sake of comparison, we provide numerical results for the Nicolaides coarse space with the additional connected partitioning enforced ("enriched" Nicolaides) in addition to the Trefftz space. Figure 2.19 reports convergence histories of the preconditioned GMRES method using the Nicolaides and Trefftz coarse spaces for the data set shown in the right of Figure 2.18. Table 2.3 summarizes the numerical performance for data sets shown in Figure 2.18 including or excluding walls. In particular, for both preconditioners, it reports the dimensions of the coarse spaces, as well as the number of GMRES iterations required to achieve a relative L^2 error of 10^{-8}. We refer to relative dimension as the would-be dimension of the coarse space in the case of a domain without perforations with $\Omega_S = \emptyset$; that is, the relative dimensions are computed as $\frac{\dim(R_H)}{(\sqrt{N}+1)^2}$ for the Trefftz space and as $\frac{\dim(R_H)}{N}$ for the Nicolaides space.

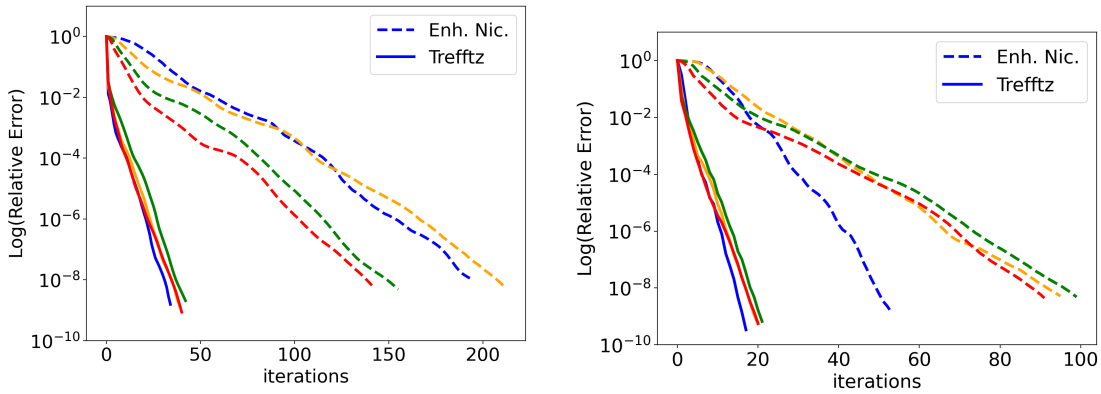
As we provide numerical results for various numbers of subdomains N and the computational domain Ω remains fixed independently of N, the results of this experiment should be interpreted in terms of a strong scalability. However, we wish to stress that the fine-scale triangulation is conforming to the nonoverlapping partitioning $(\Omega_j)_{j=1}^N$. Consequentially, the system (2.34) changes from one coarse partitioning to another. Nevertheless we ensure that the dimension of the system (2.34) is roughly constant throughout the experiment for a given N.

As an additional test to ensure robustness with respect to the data frame, we present Krylov scalability tests on a different data frame with and without walls. The new frame is shown in Figure 2.20 and is also of size 640×640 meters. This larger data set contains 552 buildings and 662 walls of varying sizes, and would take a minimum of 143 178 degrees of freedom to triangulate without imposing maximum triangle area or constraining the mesh to match the coarse partitioning.

Table 2.4 summarizes the numerical performance for data sets shown in Figure 2.20 including or excluding walls. In Table 2.4, we use the two-level ASM precon-

Table 2.3: GMRES iterations, dimension, and relative dimension for the Trefftz and Nicolaides coarse spaces. Results are shown for minimal geometric overlap and $\frac{1}{20}\mathcal{H}_j$, and the computational domain is given in Figure 2.18. As the dimension of the Nicolaides space will change with respect to the overlap, its dimension is given as the average dimension over the two overlap values.

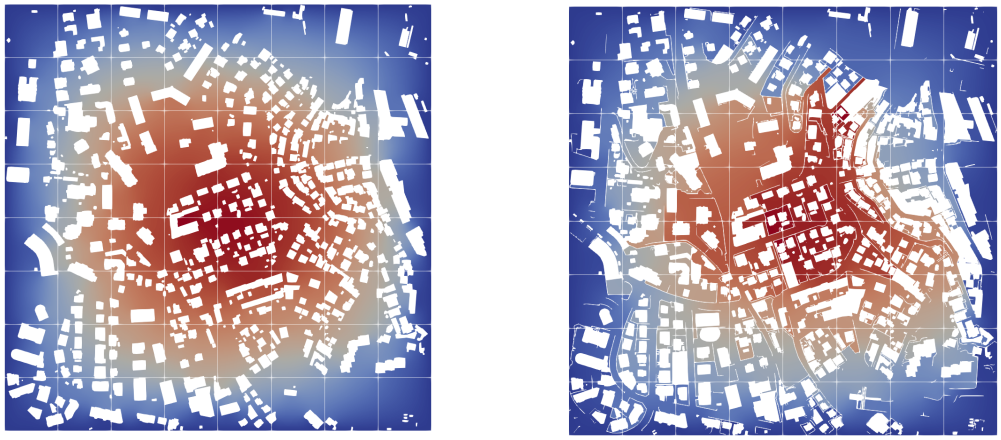
		Nicolaides			Trefftz		
		it.		dim. (rel)	it.		dim. (rel)
N		min.	$\frac{1}{20}\mathcal{H}_j$		min	$\frac{1}{20}\mathcal{H}_j$	
4	no walls	107	40	18 (1.1)	31	16	144 (5.8)
	walls	194	54	39 (2.4)	35	18	283 (11.3)
8	no walls	110	65	74 (1.2)	35	18	328 (4.0)
	walls	213	96	150 (2.3)	40	21	658 (8.1)
16	no walls	103	62	300 (1.2)	37	18	797 (2.8)
	walls	156	100	504 (2.0)	43	22	1499 (5.2)
32	no walls	94	53	1157 (1.1)	38	18	1997 (1.8)
	walls	142	92	1614 (1.6)	41	21	3319 (3.0)



(a) Minimal geometric overlap

(b) Overlap $\frac{1}{20}\mathcal{H}_j$

Figure 2.19: Convergence curves for the Trefftz (solid lines) and Nicolaides (dashed lines) coarse spaces for the larger data set shown by Figure 2.18 involving both buildings and walls and two overlap sizes. Colors correspond to the number of subdomains as follows: $N = 16$ (blue), $N = 64$ (orange), $N = 256$ (green), $N = 1024$ (red).



(a) Without walls (b) With walls

Figure 2.20: Approximate solution over an urban perforated domain divided in $N = 8 \times 8$ nonoverlapping subdomains.

ditioner (2.13), which allows us to have a symmetric preconditioner for which we can compute the condition number. Then, for the symmetric ASM preconditioner, we record preconditioned Conjugate Gradient iterations and record the condition number $\kappa(M_{ASM,2}^{-1}A)$ in Table 2.4 as well. Figure 2.21 reports convergence histories of the preconditioned CG method using the Nicolaides and Trefftz coarse spaces for the data set shown in the right of Figure 2.20.

The performance of the Trefftz coarse space appears to be very robust with respect to both N and the complexity of the computational domain. While the Nicolaides space is also fairly robust with respect to N (as expected), the Trefftz space provides an additional acceleration in terms of iteration count. The improvement with respect to the alternative Nicolaides approach is quite striking, particularly in the case of the minimal geometric overlap. As expected, increased overlap in the first level of the Schwarz preconditioner provides additional acceleration in terms of iteration count. However, for the Trefftz space, the results with minimal geometric overlap appear to already be quite reasonable.

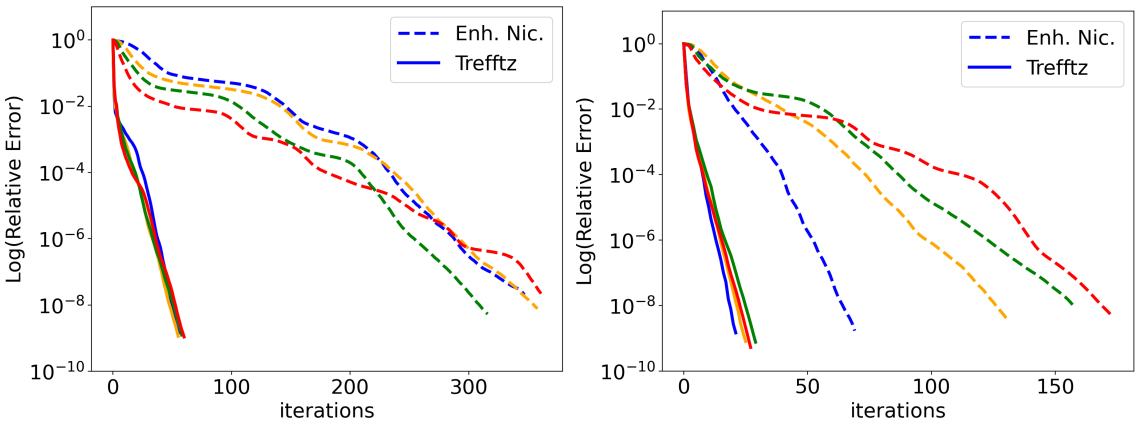
We observe form Tables 2.3 and 2.4 that the dimension of the Trefftz space is generally larger than Nicolaides, but we note that the contrast between the dimensions of two spaces reduces as N grows. In general, the dimension of the Trefftz coarse space appears reasonable given the geometrical complexity of the computational domain. We also note that the Trefftz space outperforms the Nicolaides space significantly for the domains including walls. The Trefftz space is robust with respect to data complexity, performing similarly with and without the addition of walls in the domain.

2.5 Conclusions

We presented a theoretical and numerical study of the Trefftz coarse space for the Poisson problem posed in domains that include a large number perforations, such as those encountered in the field of urban hydraulics. The main theoretical contribution concerns the error estimate regarding the H^1-projection over the coarse space. The error analysis does not rely on global regularity of the solution and is performed

Table 2.4: CG iterations, condition number, dimension, and relative dimension for the Trefftz-like and Nicolaides coarse spaces. Results are shown for minimal geometric overlap and $\frac{1}{20}\mathcal{H}_j$, and the computational domain is given in Figure 2.20. As the dimension of the Nicolaides space will change with respect to the overlap, its dimension is given as the average dimension over the two overlap values.

		Nicolaides					Trefftz				
		it.		cond.		dim. (rel)	it.		cond.		dim. (rel)
N		min.	$\frac{1}{20}\mathcal{H}_j$	min.	$\frac{1}{20}\mathcal{H}_j$		min	$\frac{1}{20}\mathcal{H}_j$	min.	$\frac{1}{20}\mathcal{H}_j$	
16	no walls	149	51	581	82	21 (1.3)	52	28	59	11	170 (6.8)
	walls	348	70	6826	133	96 (6.0)	56	22	136	7	400 (16.0)
64	no walls	164	78	567	119	85 (1.3)	50	28	50	12	433 (5.3)
	walls	359	132	5902	297	256 (4.0)	56	26	57	9	880 (10.9)
256	no walls	136	81	273	89	312 (1.2)	56	27	54	10	1010 (3.5)
	walls	317	159	4575	12	719 (2.8)	59	30	60	13	1912 (6.6)
1024	no walls	120	83	341	149	1204 (1.2)	56	28	76	13	2500 (2.3)
	walls	362	174	3895	1310	2044 (2.0)	61	28	97	13	4253 (3.9)



(a) Minimal geometric overlap (b) Overlap $\frac{1}{20}L$

Figure 2.21: Convergence curves for the Trefftz (solid lines) and Nicolaides (dashed lines) coarse spaces for the larger data set shown by Figure 2.20 involving both buildings and walls and two overlap sizes. Colors correspond to the number of subdomains as follows: $N = 16$ (blue), $N = 64$ (orange), $N = 256$ (green), $N = 1024$ (red).

under some very minimal assumptions regarding the geometry of the domain. In accordance to the presented error analysis, for a specific edge refinement procedure, the coarse approximation achieves superconvergence. This superconvergence is observed in the numerical experiment involving the L-shaped domain. Combined with RAS, the Trefftz space leads to an efficient and robust iterative solver or preconditioner for linear systems resulting from fine-scale finite element discretizations. The performance of the two-level RAS method is demonstrated through numerical experiments involving realistic urban geometries. We observe that, for finite element discretizations with moderate accuracy, the two-level RAS method reaches the precision of the fine-scale discretization in a few iterations. Used in combination with domain decomposition methods as a preconditioner for Krylov methods, the coarse space provides significant acceleration in terms of Krylov iteration counts when compared to a more standard Nicolaides coarse space. This improvement comes at the price of a somewhat larger coarse problem with a larger dimension.

Chapter 3

Two-level Nonlinear Preconditioning Methods for Urban Flood Models

Contents

This chapter is mainly based on the article submitted to Computers and Mathematics with Applications. The corresponding submitted preprint is available on arXiv [15].

3.1 Introduction

In this chapter, we apply domain decomposition and multi-scale approaches to solve stationary and time-dependent partial differential equations. The main focus of the chapter is the Diffusive Wave model [3],

$$\partial_t u - \operatorname{div}\left(c_f h(u, z_b(\mathbf{x}))^{\alpha} ||\nabla u||^{\gamma-1} \nabla u\right) = 0,$$

where $z_b(\mathbf{x})$ represents bathymetry/topography, u is total water level, $h(u, z_b(\mathbf{x})) = \max(u - z_b(\mathbf{x}, 0)$ is the depth of the water with respect to bathymetry, c_f denotes friction, $\gamma \leq 1$, and $1 \leq \alpha \leq 2$. As mentioned in the introduction, we choose $\alpha = \frac{3}{2}$ and $\gamma = \frac{1}{2}$. The Diffusive Wave Equation (DWE) is doubly nonlinear in the sense that there are nonlinearities both in the $(u - z_b)$ term and in the gradient term. With this model problem, we wish to model overland flows in urban areas, particularly the area of Nice, France. As in Chapter 2, the structural features such as buildings and walls in the model domain can be assumed to be essentially impervious, and therefore represented as perforations (holes) in the model domain.

With this, we recall the notations form Chapter 1. Let D be an open simply connected polygonal domain in $\mathbb{R}^2$. We denote by $(\Omega_{S,k})_k$ a finite family of perforations in D and denote $\Omega_S = \bigcup_k \Omega_{S,k}$ and $\Omega = D \setminus \overline{\Omega_S}$. With this, the model problem of focus is given by the following: for a fixed time interval $(0, T]$,

$$\begin{cases} \partial_t u - \operatorname{div}\left(k(x, u, \nabla u)\nabla u\right) = 0 & \text{in} \quad \Omega \times (0, T], \\ u = g & \text{on} \quad (\partial\Omega \setminus \partial\Omega_S) \times (0, T], \\ k(x, u, \nabla u)\dfrac{\partial u}{\partial \mathbf{n}} = 0 & \text{on} \quad (\partial\Omega \cap \partial\Omega_S) \times (0, T], \end{cases} \tag{3.1}$$

where $k(x, u, \nabla u) = h(u, z_b(\mathbf{x}))^\alpha ||\nabla u||^{\gamma-1}$, $\mathbf{n}$ is the outward normal to the boundary and $f \in L^2(\Omega)$. The problem (3.1) must be combined with some initial condition on u. We remark that (3.1) becomes degenerate, as $k(x, u, \nabla u) \to 0$ as $h \to 0$ and $k(x, u, \nabla u) \to \infty$ as $\nabla u \to 0$.

In the context of urban flood modeling, $(\Omega_{S,k})_k$ can be thought of as a family of impervious structures such as buildings, fences, or other similar structures. However, problems posed on perforated domains also arise in multiple real-world scenarios, such as fluid/groundwater flow in porous media, atmospheric models, battery models, and drug delivery systems. After some discretization to be detailed later, this nonlinear equation can be written as

$$F(\mathbf{u}) = 0,$$

where the numerical solution $\mathbf{u}$ approximates the exact solution u.

Our goal is to numerically solve the model problem on these perforated domains by using domain decomposition techniques. The latter use a divide and conquer approach to partition the domain into multiple smaller subdomains. Therefore, one has two "levels" of space discretization; the first level is based on a coarse polygonal partitioning of Ω, while the second one is associated with the fine-scale triangulation and is designed to resolve the small-scale details of the model domain. Domain decomposition (DD) approaches to solve nonlinear PDEs can generally split into two categories: Newton-Krylov Schwarz (NKS) and Schwarz Newton-Krylov (SNK) methods.

3.1.1 Newton-Krylov Schwarz Methods

In NKS methods [24], the nonlinear problem $F(\mathbf{u})$ is first linearized, typically by Newton's method or an inexact Newton method, which solve a linear system at each iteration. For exact Newton's method, one needs the the residual $F(\mathbf{u})$ as well as the Jacobian $\nabla F(\mathbf{u})$. The exact Newton's method is described in Algorithm 3.1.1.

Algorithm 3.1.1 Newton's method to solve $F(\mathbf{u}) = 0$

Require: $\mathbf{u}^0$, residual function F, Jacobian ∇F
  for outer iteration $k = 0, \ldots,$ to convergence **do**
    Compute Newton update $\boldsymbol{\delta} = [\nabla F(\mathbf{u})]^{-1} F(\mathbf{u})$
    Compute $\mathbf{u}^{k+1} = \mathbf{u}^k - \boldsymbol{\delta}$;
  end for

Often when implementing Newton's method, a backtracking line search is implemented to control the step size of Newton's method. This often aids in convergence for certain problems, as the line search prevents "overshooting" in the Newton step direction. Newton's method with backtracking line search is given in Procedure 1.

Procedure 1 Newton's Method with backtracking line search to solve $F(\mathbf{u}) = 0$

  for outer iteration $k = 0, \ldots,$ to convergence **do**
    $d = 1$
    Compute Newton update $\boldsymbol{\delta} = [\nabla F(\mathbf{u})]^{-1} F(\mathbf{u})$
    while $||F(\mathbf{u} - d\boldsymbol{\delta})|| > (1 - \frac{d}{4})||F(\mathbf{u})||$ **do**
      $d = \frac{d}{2}$
    end while
    Compute $\mathbf{u}^{k+1} = \mathbf{u}^n - d\,\boldsymbol{\delta}$;
  end for

We remark that each iteration of Newton's method requires the solution of a linear system. It is well-known that for large linear systems, a direct linear solve can become quite expensive and time-consuming. Therefore, we can solve the linear systems via iterative methods, which are often more computationally efficient for large-scale problems and may require less memory. As mentioned in the introduction, preconditioning can improve the convergence behavior of an iterative solver by improving the conditioning of a system. Here, we apply typical linear DD methods to the linear system in Algorithm 3.1.1 ; i.e., use standard domain decomposition preconditioners from Chapter 2 coupled to a Krylov method on the linear system to be solved at each Newton iteration. Therefore, in the case of NKS methods applied to a nonlinear problem $F(\mathbf{u}) = 0$, the system remains unchanged, and we use DD methods to make the linear solve more efficient.

3.1.2 Schwarz Newton-Krylov Methods

Stiff nonlinearities arise in many nonlinear problems, including problems such as heterogeneous problems with high-contrast coefficients. These stiff nonlinearities can cause a plateau in the norm of the Newton residual before/if the quadratic region of convergence is reached. In these cases in a global Newton's method, one may be forced to take very small time steps, or damp the Newton iteration to an extremely small damping factor. To combat this, more recently, SNK methods were introduced; these methods do not begin with a linearization and are commonly referred to as nonlinear preconditioning methods. In SNK methods, the nonlinear system itself is changed, unlike in NKS methods. The nonlinear preconditioners involve the solution of local nonlinear problems and can therefore localize stiff nonlinearities.

Specifically, given a nonlinear system $F(\mathbf{u}) = 0$, these SNK preconditioning

methods solve a new nonlinear system $\mathcal{F}(\mathbf{u})$ via Newton's or inexact Newton's method, where if $\mathcal{F}(\mathbf{u}) = 0$, then $F(\mathbf{u}) = 0$ (i.e., the new nonlinear system shares solutions with the original nonlinear system). This system $\mathcal{F}(\mathbf{u}) = 0$ is referred to as *the nonlinearly preconditioned system*. The latter is generated by applying Newton's or inexact Newton's method to a fixed point equation. The goal is for the nonlinearly preconditioned system to be solved faster than the original one; that is, to more quickly advance to the region of quadratic convergence for Newton's method and reduce the number of exact or inexact Newton iterations.

In the Additive Schwarz Preconditioned Inexact Newton (ASPIN) [25], the authors introduce a method where the nonlinearly preconditioned system $\mathcal{F}(u) = 0$ consists of more balanced/uniform nonlinearities. ASPIN involves the solution of local nonlinear problems, using similar restriction and extension operators that are used in the linear case. Then, a global linear problem is solved via inexact Newton. In theory, for difficult problems with intense nonlinearities, this technique can accelerate convergence when compared to a typical Newton or inexact Newton method. Additionally, it can aid in the occurrence of stagnation in Newton's method and desensitize Newton to the initial guess. We remark as well the existence of the MSPIN method [78], where generally, the partitioning of degrees of freedom is based on field type (field-splitting) instead of the typical subdomain partitioning. A main difference between the methods is that MSPIN can be thought of as analogous to the Multiplicative Schwarz (MS) method for linear systems, a generally non-parallel method, while ASPIN is analogous to the Additive Schwarz method (ASM). With this, one can also think of ASPIN and MSPIN as varying from each other similarly to Jacobi and Gauss-Seidel methods. In MSPIN and ASPIN, inexact Newton solves are computed for the nonlinearly preconditioned system.

The authors of [39] extended the ASPIN method to form Restricted Additive Schwarz Preconditioned Exact Newton (RASPEN), which can be thought of as analogous to the linear RAS method. Similarly to the linear case, RASPEN does not sum contributions in the overlap, using partition of unity operators to form a fixed point iteration to which Newton's method can be applied. The authors of [39] showed that by design, the nonlinearly preconditioned system $\mathcal{F}_{RASPEN}(\mathbf{u}) = 0$ can be solved via exact Newton, as an equation can be derived for the exact Jacobian matrix $\nabla\mathcal{F}_{RASPEN}(\mathbf{u})$.

The authors of [30] introduced SRASPEN, which is obtained similarly by applying Newton's method to the fixed point equation resulting from the nonlinear SRAS (Substructured Restricted Additive Schwarz) method. Unlike the aforementioned methods, the SRASPEN method works on nonoverlapping subdomains; these nonoverlapping domain decomposition methods are often referred to as substructuring methods. Within the same spirit, the authors of [31] introduced DNPEN, a nonlinear preconditioning method which applies Newton's method to the Dirichlet-Neumann (DN) method, which is a substructuring method based on matching fluxes along subdomain interfaces. Additionally, the authors of [79] propose an adaptive nonlinear preconditioning technique which "turns off" the nonlinear preconditioning for outer Newtons where it is not necessary.

The aforementioned techniques can be considered as being *left preconditioning methods*, as they are solving a new tangential system $\mathcal{F}(\mathbf{u}) = 0$ by exact or inexact Newton. We also remark that there are a family of *right nonlinear preconditioning techniques*; these methods change the unknowns of the original system. We can give as an example the following works [73, 88, 71, 72], where the authors introduced nonlinear preconditioners analogous to the Finite Element Tearing and Interconnecting

(FETI), Finite Element Tearing and Interconnecting Dual-Primal (FETI-DP), and Balancing Domain Decomposition by Constraints (BDDC) methods. Additionally, the authors of [77] introduced the Nonlinear Elimination Preconditioned Inexact Newton (NEPIN), where "problem" components of the residual are implicitly eliminated and a resulting modified Newton direction is computed.

As is the case for linear DD methods, two-level methods (methods which include a coarse correction/level), are often needed for scalability in nonlinear preconditioning methods. Particularly, the number of GMRES iterations for the global tangential solve will not be scalable without a coarse correction. The coarse component allows for global communication across all subdomains. In the original article introducing RASPEN [39], the authors proposed a Full Approximation Scheme (FAS) to apply the coarse level. For MSPIN, a multiplicative coarse correction counterpart was introduced. In [63], the author proposes a linear coarse level for ASPIN and in [57], the authors provide numerical evidence of scalability of two-level variants of ASPIN and RASPEN. In particular, they propose multiple approaches of adding the coarse space correction (both additively and multiplicatively) based on Galerkin projections, and a numerical comparison of the methods is reported for different types of coarse spaces.

3.1.3 Chapter Outline

The remainder of this chapter is laid out as follows. Section 3.2 provides a detailed description of the finite-element/ finite-volume hybrid discretization used for the Diffusive Wave Equation. Section 3.3 briefly recalls the coarse space from [13] and Chapter 2, which will be used in the two-level methods described in the chapter. Section 3.4 summarizes domain decomposition methods for linear problems, presenting the well-known linear Additive Schwarz and Restricted Additive Schwarz methods. Section 3.5 describes the proposed methods in detail and discusses the cost/complexity of each proposed algorithm. Section 3.6 provides numerical results, with experiments based on a porous medium equation posed on both an L-shaped domain and a large urban domain, followed by the Diffusive Wave equation posed on a large urban domain. Section 3.7 introduces the use of the Trefftz space for the coarse approximation of the numerical solutions of the nonlinear model problems. This section includes numerical experiments including a similar Diffusive Wave experiment from Section 3.6. Section 3.8 concludes with a summary.

3.2 Discretization of Diffusive Wave equation

We now proceed with a discretization of the Diffusive Wave model (3.1). Let $T_f > 0$ be the final flow simulation time and let $0 = t_0 < t_1 < \ldots < t_{N_T} = T_f$ be a family of real numbers such that $\Delta t_n = t_{n+1} - t_n$.

We consider the following semi-implicit time discretization of (3.1): for $u^{n+1} = u|_{t_{n+1}}$,

$$\begin{cases} \dfrac{u^{n+1} - u^n}{\Delta t_n} - \operatorname{div}\left(c_f \kappa(u^{n+1}, \nabla u^n) \nabla u^{n+1}\right) = 0 & \text{in} \quad \Omega, \\ u^{n+1} = g & \text{on} \quad \partial\Omega \setminus \partial\Omega_S, \\ c_f \kappa(u^{n+1}, \nabla u^n) \dfrac{\partial u^{n+1}}{\partial \mathbf{n}} = 0 & \text{on} \quad \partial\Omega \cap \partial\Omega_S, \end{cases} \tag{3.2}$$

where u^0 is a provided initial condition.

Let $\mathcal{T}$ denote the triangulation of Ω which is conforming with respect to the partitioning $(\Omega_j)_{j=1}^N$. We introduce the finite element space defined by

$$V_h = \{v \mid v \in C^0(\overline{\Omega}), \quad \text{s.t.} \quad v|_q \in \mathbb{P}_1, \quad \text{for all} \quad q \in \mathcal{T}\}.$$

Recalling the definition of the space $H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$ defined in (3.7), let $V_{h,0} = V_h \cap H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega)$ and let $g_h \in V_h$ be some approximation of the boundary data g. The finite element discretization of (3.2) reads as: Find $u_h^{n+1} \in V_h$ satisfying $u_h^{n+1}|_{\partial\Omega\setminus\Omega_S} = g_h$ such that

$$\frac{1}{\Delta t_n}\int_\Omega (u_h^{n+1}-u_h^n)\, v_h \,\mathrm{d}\mathbf{x} + \int_\Omega c_f \kappa(u_h^{n+1}, \nabla u_h^n)\nabla u_h^{n+1}\cdot\nabla v_h \,\mathrm{d}\mathbf{x} = 0 \qquad \text{for all} \quad v_h \in V_{h,0}. \tag{3.3}$$

Let $(\mathbf{x}_i)_{i=1}^{N_\Omega}$ denote the set of triangulation points; the set of point indices $\{1, 2, ..., N_\Omega\}$ is denoted by $\mathcal{N}$. We denote the nodal "hat" basis functions $(\eta^\ell)_{\ell=1}^{N_\Omega}$. Furthermore, we define u_i^{n+1} and u_ℓ^{n+1} as the nodal values of the discrete solution at nodes $\mathbf{x}_i$ and $\mathbf{x}_\ell$ at t_{n+1}, respectively. Additionally, we set $\mathcal{N}_T = \{i \in \mathcal{N} \mid \mathbf{x}_i \in \overline{T}\}$ for any triangle $T \in \mathcal{T}$ (T assumed to be open) and denote $\mathcal{T}_i$ as the subset of triangles connected to the node i such that $\mathcal{T}_i = \{T \in \mathcal{T} \mid \mathbf{x}_i \in \overline{T}\}$. Furthermore, we denote by $\mathcal{N}_i$ the set of nodes adjacent to node i such that $\mathcal{N}_i = \bigcup_{T\in\mathcal{T}_i} \mathcal{N}_T$. Finally, let $\mathcal{N}_D$ denote the set of Dirichlet nodes such that $\mathcal{N}_D = \{i \in \mathcal{N} \mid \mathbf{x}_i \in \partial\Omega \setminus \Omega_S\}$.

In order to enhance the stability of the numerical scheme, we modify the finite element scheme by means of discretization techniques from finite volume methods. In particular, we introduce mass lumping in the accumulation and upwinding in the diffusion term. This discretization technique is generally referred to as a Finite Volume- Finite Element (FV-FE) discretization [35], or it is often referred to as a Control Volume Finite Element method [28, 37].

Diffusion term: Since $\sum_{\ell\in\mathcal{N}_T} \nabla\eta_\ell = 0$ on each triangle T, we observe for the diffusive term that for $i \in \mathcal{N}$,

$$\begin{aligned}
\int_\Omega c_f \kappa(u_h^{n+1}, \nabla u_h^n)\nabla u_h^{n+1}\cdot\nabla\eta_i \,\mathrm{d}\mathbf{x} &= \sum_{T\in\mathcal{T}_i}\sum_{\ell\in\mathcal{N}_T} c_f u_\ell^{n+1}\int_T \kappa(u_h^{n+1}, \nabla u_h^n)\nabla\eta_\ell\cdot\nabla\eta_i \,\mathrm{d}\mathbf{x},\\
&= \sum_{\ell\in\mathcal{N}_i} c_f(u_\ell^{n+1}-u_i^{n+1})\sum_{T\in\mathcal{T}}\int_T \kappa(u_h^{n+1}, \nabla u_h^n)\nabla\eta_\ell\cdot\nabla\eta_i \,\mathrm{d}\mathbf{x},\\
&= \sum_{\ell\in\mathcal{N}_i} c_f(u_\ell^{n+1}-u_i^{n+1})\sum_{T\in\mathcal{T}_i\cap\mathcal{T}_\ell}\int_T \kappa(u_h^{n+1}, \nabla u_h^n)\nabla\eta_\ell\cdot\nabla\eta_i \,\mathrm{d}\mathbf{x},
\end{aligned}$$

In order to deal with the degeneracy of $\kappa(u, \boldsymbol{\xi}) = h(u, z_b)^\alpha \|\boldsymbol{\xi}\|^{\gamma-1}$ for $h(u, z_b) = 0$, we introduce upwinding in our discretization. Denoting

$$\tau_{i\ell,T}^n = -c_f\left|\nabla u_h^n|_T\right|^{1-\gamma}\int_T \nabla\eta_\ell\cdot\nabla\eta_i \,\mathrm{d}\mathbf{x} \qquad \text{and} \qquad \tau_{i\ell}^n = \sum_{T\in\mathcal{T}_i\cap\mathcal{T}_\ell}\tau_{i\ell,T}^n,$$

we introduce upstream water depth defined by

$$h_{i\ell}^{n+1} = \begin{cases} h(u_i^{n+1}, z_b(\mathbf{x}_i)) & \text{if } \tau_{i\ell}^n(u_i^{n+1}-u_\ell^{n+1}) \geq 0,\\ h(u_\ell^{n+1}, z_b(\mathbf{x}_\ell)) & \text{otherwise.}\end{cases}$$

Then, we approximate

$$\int_\Omega c_f \kappa(u_h^{n+1}, \nabla u_h^n) \nabla u_h^{n+1} \cdot \nabla \eta_i \, \mathrm{d}\mathbf{x} \approx \sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha (u_i^{n+1} - u_\ell^{n+1}) \qquad \text{for all} \quad i \in \mathcal{N} \backslash \mathcal{N}_D.$$

Accumulation term: Let $M_{i\ell}$ denote the (i,ℓ) entry of a classical finite element matrix, that is $M_{i\ell} = \int_\Omega \eta_i \eta_\ell \, \mathrm{d}\mathbf{x}$. Then, denoting $m_i = \sum_{\ell=1}^{N_\Omega} M_{i\ell}$, we approximate

$$\frac{1}{\Delta t_n} \int_\Omega (u_h^{n+1} - u_h^n) \cdot \eta_i \, \mathrm{d}\mathbf{x} \approx \frac{m_i}{\Delta t_n} \left(u_i^{n+1} - u_i^n\right) \qquad \text{for all} \quad i \in \mathcal{N} \setminus \mathcal{N}_D.$$

The resulting combined FV-FE scheme writes as

$$\begin{cases} \dfrac{m_i}{\Delta t_n} \left(u_i^{n+1} - u_i^n\right) + \displaystyle\sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha (u_i^{n+1} - u_\ell^{n+1}) = 0 & \text{for} \quad i \in \mathcal{N} \setminus \mathcal{N}_D, \\ u_i^{n+1} = g_h(\mathbf{x}_i), & \text{for} \quad i \in \mathcal{N}_D. \end{cases} \tag{3.4}$$

In matrix-vector notations for a given vector $\mathbf{u} = (u_i)_{i \in \mathcal{N} \backslash \mathcal{N}_D}$, the scheme can be summarized as follows: for time step $n = 0, 1, \ldots,$ solve

$$F(\mathbf{u}^{n+1}) = 0,$$

where the ith component of $F(\mathbf{u}^{n+1})$ is given by

$$F(\mathbf{u}^{n+1})_i = \frac{m_i}{\Delta t_n} \left(u_i^{n+1} - u_i^n\right) + \sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha (u_i^{n+1} - u_\ell^{n+1}) \quad \text{for } i \in \mathcal{N} \setminus \mathcal{N}_D, \tag{3.5}$$

and $u_i^{n+1} = g_h(\mathbf{x}_i)$ for $i \in \mathcal{N}_D$.

Additionally, the Jacobian matrix can be assembled such that the (i,i) and (i,ℓ) entries are given by

$$\nabla F(\mathbf{u}^{n+1})_{ii} = \left\{ \tfrac{m_i}{\Delta t_n} + \textstyle\sum_{\ell \in \mathcal{N}_i} \left[\tau_{i\ell}^n \alpha \left(h_{i\ell}^{n+1}\right)^{\alpha-1} H_{i\ell}^{n+1} + \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha \right] \quad \text{for } i \in \mathcal{N} \setminus \mathcal{N}_D, \right.$$

$$\nabla F(\mathbf{u}^{n+1})_{i\ell} = \left\{ \tau_{i\ell}^n \alpha \left(h_{i\ell}^{n+1}\right)^{\alpha-1} (1 - H_{i\ell}^{n+1}) - \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha \quad \text{for } i \in \mathcal{N} \setminus \mathcal{N}_D, \right.$$

where

$$H_{i\ell}^{n+1} = \begin{cases} 1 & \text{if } \tau_{i\ell}^n (u_i^{n+1} - u_j^{n+1}) \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

3.3 Multiscale Coarse space

We now briefly describe the coarse space that will be used for all introduced two-level algorithms; we refer the reader to Chapter 2 for a detailed description. The coarse space is of the multiscale type, composed of piecewise harmonic basis functions with respect to a polygonal partitioning of the domain. The latter are computed via the numerical solutions of linear Laplace problems.

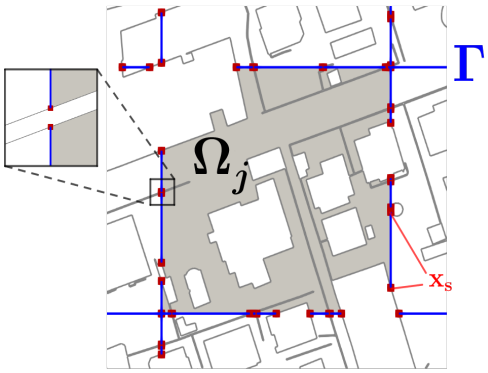


Figure 3.1: Coarse cell Ω_j, nonoverlapping skeleton Γ (blue lines), and coarse grid nodes $\mathbf{x}_s \in \mathcal{V}$ (red dots). Coarse grid nodes are located at $\overline{\Gamma} \cap \partial\Omega_S$.

Consider a nonoverlapping partitioning of Ω denoted by $(\Omega_j)_{j=1,\dots,N}$. We denote by Γ its skeleton, that is $\Gamma = \bigcup_{j=1,\dots,N} \partial\Omega_j \setminus \partial\Omega_S$. Let $(e_k)_{k=1,\dots,N_e}$ denote a nonoverlapping partitioning of the Γ such that each "coarse edge" e_k is an open planar segment. The set of coarse grid nodes is given by $\mathcal{V} = \bigcup_{k=1,\dots,N_e} \partial e_k$. Figure 3.1 illustrates the location of the coarse grid nodes that typically result from clipping $(D_j)_j$ with Ω_S.

Let $H^1_\Delta(\Omega)$ be a subspace of $H^1(\Omega)$ composed of piece-wise harmonic functions, weakly satisfying the homogeneous Neumann boundary conditions on $\partial\Omega \cap \partial\Omega_S$, that is

$$H^1_\Delta(\Omega) = \{\, u \in H^1(\Omega) \mid (\nabla u|_{\Omega_j}, \nabla v)_{L^2(\Omega_j)} = 0 \quad \text{for all} \quad v \in H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega_j)\}, \tag{3.6}$$

where

$$H^1_{\partial\Omega\setminus\partial\Omega_S}(\Omega) = \{u \in H^1(\Omega) \,|\, u|_{\partial\Omega\setminus\partial\Omega_S} = 0\}. \tag{3.7}$$

In other words, $u \in H^1_\Delta(\Omega)$ if and only if $u \in H^1(\Omega)$ and for all subdomains Ω_j, the equations

$$\left\{ \begin{array}{rcll} -\Delta u|_{\Omega_j} & = & 0 & \text{in} \quad \Omega_j, \\ \dfrac{\partial u}{\partial \mathbf{n}} & = & 0 & \text{on} \quad \partial\Omega_j \cap \partial\Omega_S, \end{array} \right. \tag{3.8}$$

are satisfied in a weak sense.

With this, we define

$$V_H^\Gamma = \{\, v \in C^0(\overline{\Gamma}) \mid v|_{e_k} \in \mathbb{P}_1(e_k) \quad \text{for all} \quad k = 1, \dots, N_e\},$$

where $\mathbb{P}_1(e_k)$ denotes the set of piecewise linear polynomials over an edge e, and we also define

$$V_H = \{\, v \in H^1_\Delta(\Omega) \mid v|_\Gamma \in V_H^\Gamma\}. \tag{3.9}$$

Let $(g_s)_{s=1,\dots,N_\mathcal{V}}$ be the nodal basis of V_H^Γ, where $N_\mathcal{V}$ denotes the total number of coarse grid nodes. In this chapter, $(g_s)_{s=1,\dots,N_\mathcal{V}}$ is composed of nodal piecewise linear "hat" functions; however, we recall from Chapter 2 that we can also combine the set of the nodal piece-wise linear "hat" functions with the set of higher order

edge-based basis functions. The function $\phi_s \in V_H$ associated to g_s is computed by weakly imposing on each Ω_j

$$\begin{cases} \Delta\phi_{\mathbf{s}} &= 0 \quad \text{in} \quad \Omega_j, \\ \dfrac{\partial\phi_s}{\partial\mathbf{n}} &= 0 \quad \text{on} \quad \partial\Omega_j \cap \partial\Omega_S, \\ \phi_s &= g_s \quad \text{on} \quad \partial\Omega_j \setminus \partial\Omega_S. \end{cases} \tag{3.10}$$

We refer to the space V_H as the Trefftz space. The basis of the discrete version of the Trefftz space is obtained through the piecewise linear finite element approximation of (3.10) which results in the system of the form

$$\tilde{A}_j \boldsymbol{\phi}_s^j = b_s^j,$$

where $\tilde{A}_j$ is the local stiffness matrix and b_s^j accounts for the Dirichlet boundary data in (3.10).

Let $\overline{R}_j$ denote the boolean restriction matrices which restrict from the degrees of freedom in the discretization of Ω to the degrees of freedom in the nonoverlapping subdomain $\overline{\Omega}_j$, and let $\overline{D}_j$ denote partition of unity matrices corresponding to the discretization of $\overline{\Omega}_j$ such that $\sum_{j=1}^N R_j^T D_j R_j = I$. Furthermore, let $\boldsymbol{\phi}_s$ be a global vector which, when restricted to a subdomain Ω_j, returns $\boldsymbol{\phi}_s^j$. That is,

$$\boldsymbol{\phi}_s = \sum_{j \in \mathcal{N}_s} \overline{R}_j^T \overline{D}_j \boldsymbol{\phi}_s^j,$$

for $s = 1, \ldots, N_{\mathcal{V}}$, where $\mathcal{N}_s = \{\, j \mid \mathbf{x}_s \text{ is contained in } \Omega_j \,\}$. The global vectors $\boldsymbol{\phi}_s$ are the basis vectors associated to the coarse grid nodes $\mathbf{x}_s \in \mathcal{V}$. The discrete Trefftz space is then defined as the span of the basis vectors $\boldsymbol{\phi}_s, s = 1, \ldots, N_{\mathcal{V}}$. With this, the coarse transition matrix R_H is such that the kth row of R_H is given by $\boldsymbol{\phi}_k^T$ for $k = 1, \ldots, N_{\mathcal{V}}$.

3.4 Linear Multi-domain Solution Methods

In this section, we summarize linear DD methods, particularly overlapping Schwarz methods, that will be extended in the next section to nonlinear PDEs. We refer the reader to Chapter 2 for a detailed description. Consider some discretized linear system of the form

$$A\mathbf{u} = \mathbf{f}.$$

Let $\left(\Omega_j'\right)_{j=1,\ldots,N}$ denote the overlapping partitioning of Ω such that $\Omega_j \subset \Omega_j'$. In practice, each Ω_j' is constructed by propagating Ω_j by a few layers of triangles. With this, let R_j denote the boolean restriction matrices which restrict from the degrees of freedom in the discretization of Ω to the degrees of freedom in the overlapping subdomain Ω_j', and let D_j denote partition of unity matrices corresponding to the discretization of Ω_j' such that $\sum_{j=1}^N R_j^T D_j R_j = I$.

With this, the iterative RAS method computes the approximation to the solution $\mathbf{u}^{k+1}$ from $\mathbf{u}^k$ by

$$\mathbf{u}^{k+1} = \mathbf{u}^k + M_{RAS,1}^{-1}(\mathbf{f} - A\mathbf{u}^k), \tag{3.11}$$

where

$$M_{RAS,1}^{-1} = \sum_{j=1}^{N} R_j^T D_j [R_j A R_j^T]^{-1} R_j.$$

Additionally, we can accelerate this fixed-point method (3.11) by solving the preconditioned system

$$M_{RAS,1}^{-1} A\mathbf{u} = M_{RAS,1}^{-1}\mathbf{f}, \tag{3.12}$$

with a Krylov type method such as GMRES.

Generally, the number of iterations in (3.11) and (3.12) grows with the number of subdomains in one direction. For example, this increase is linear for the stationary iteration (3.11). To combat this, a coarse level is needed to aid in global communication between all subdomains. Consider the coarse matrix R_H defined in Section 3.3; with this, the two-level preconditioned Krylov method method is given by

$$M_{RAS,2}^{-1} A\mathbf{u} = M_{RAS,2}^{-1}\mathbf{f}, \tag{3.13}$$

where

$$M_{RAS,2}^{-1} = R_H^T (R_H A R_H^T)^{-1} R_H + \sum_{j=1}^{N} R_j^T D_j [R_j A R_j^T]^{-1} R_j. \tag{3.14}$$

3.5 Nonlinear Multi-domain Solution Methods

We present results for various methods to solve the nonlinear system $F(\mathbf{u}) = 0$, where $F(\mathbf{u})$ arises from some discretization of a nonlinear PDE. For the Diffusive Wave model, the residual function $F(\mathbf{u}) = 0$ is given in (3.5).

Preconditioning the linearized Newton system When a linear system is very large, a direct solve is often too expensive and not recommended. Therefore, it is common to solve the linear system in Algorithm 3.1.1 via a preconditioned Krylov solver. We proposed in [13] and recalled in Section 3.3 a coarse space for the two-level RAS preconditioner applied to the Poisson equation on perforated domains. It turns out that the same coarse space can be successfully employed within the two-level RAS preconditoner (3.14) for the linear system in Newton's method (Algorithm 3.1.1). The numerical evidences regarding the efficiency of such a two-level solver are presented in Section 3.6.

Let $J_k = \nabla F(\mathbf{u}^k)$ denote the Jacobian matrix and $\mathbf{F}_k = F(\mathbf{u}^k)$ the residual at a given iterate. The Newton's method update denoted by $\boldsymbol{\delta}_k$ satisfies the system

$$J_k \boldsymbol{\delta}_k = \mathbf{F}_k.$$

The preconditioned system to solve for $\boldsymbol{\delta}_k$ is as follows:

$$M_{RAS,2}^{-1} J_k \boldsymbol{\delta}_k = M_{RAS,2}^{-1} \mathbf{F}_k,$$

where $M_{RAS,2}$ is defined as in (3.14) with system matrix A replaced by J_k,

$$M_{RAS,2}^{-1} = \sum_{j=1}^{N} R_j^T D_j (R_j J_k R_j^T)^{-1} R_j + R_H^T (R_H J_k R_H^T)^{-1} R_H. \tag{3.15}$$

Nonlinear Restrictive Additive Schwarz Iteration Recall for linear systems $A\mathbf{u} = \mathbf{f}$, the Restricted Additive Schwarz (RAS) method can be used as an iterative method or as a preconditioner for Krylov methods. Likewise, for the nonlinear problem, we introduce a nonlinear RAS (NRAS) fixed point iteration. Multiple methods which are to be introduced will require this nonlinear RAS iteration.

Definition 3.1 (An NRAS iteration)**.** Let R_j be the boolean restriction matrices corresponding to the degrees of freedom of Ω_j'. Let the "subdomain update" function G_j, such that, for all $\mathbf{u}$, $G_j(\mathbf{u})$ is the solution of the local nonlinear system

$$R_j F(R_j^T G_j(\mathbf{u}) + (I - R_j^T R_j)\mathbf{u}) = 0. \tag{3.16}$$

An NRAS iteration is computed by first solving each (3.16) then glue local contributions:

$$\text{NRAS}(\mathbf{u}) = \sum_j R_j^T D_j G_j(\mathbf{u}). \tag{3.17}$$

We can use this method as a fixed point iteration, computing iteratively the solution $\mathbf{u^k}$ starting from an initial guess $\mathbf{u^0}$:

$$\mathbf{u}^{k+1} = \text{NRAS}(\mathbf{u}^k). \tag{3.18}$$

Note that the subdomain updates $G_j(\mathbf{u})$ correspond to solving the local problem on Ω_j' with Dirichlet boundary conditions. These problems (3.16) are solved via Newton's method; as the subproblems are small, each Newton linear system is solved using a direct solver.

Remark 3.2 (NRAS is RAS when applied to a linear problem)**.** We comment here on the connection between the nonlinear RAS fixed point iteration (3.18) and the linear RAS fixed point iteration (3.11). Consider the local solutions computed from (3.16). For linear problems such that $F(\mathbf{u}) = A\mathbf{u} - \mathbf{f}$, we have

$$\begin{aligned} R_j F(R_j^T G_j(\mathbf{u}) + (I - R_j^T R_j)\mathbf{u}) &= 0 \\ R_j(A(R_j^T G_j(\mathbf{u}) + (I - R_j^T R_j)\mathbf{u}) - \mathbf{f}) &= 0 \\ R_j A R_j^T G_j(\mathbf{u}) &= R_j A R_j^T R_j \mathbf{u} + R_j(\mathbf{f} - A\mathbf{u}) \\ G_j(\mathbf{u}) &= R_j \mathbf{u} + (R_j A R_j^T)^{-1} R_j(\mathbf{f} - A\mathbf{u}) \end{aligned} \tag{3.19}$$

such that the fixed point iteration (3.18) would be equivalent to

$$\begin{aligned} \mathbf{u}^{k+1} &= \sum_{j=1}^N R_j^T D_j \left(R_j \mathbf{u}^k + (R_j A R_j^T)^{-1} R_j(\mathbf{f} - A\mathbf{u}^k) \right) \\ &= \mathbf{u}^k + \sum_{j=1}^N R_j^T D_j (R_j A R_j^T)^{-1} R_j(\mathbf{f} - A\mathbf{u}^k) \\ &= \mathbf{u}^k + M_{RAS,1}^{-1}(\mathbf{f} - A\mathbf{u}^k), \end{aligned} \tag{3.20}$$

where we have used the partition of unity property $I = \sum_{j=1}^N R_j^T D_j R_j$. Therefore, the NRAS fixed point iteration (3.18) reduces to the linear RAS fixed point iteration (3.11).

One-level RASPEN Recall that in the linear case, we can accelerate the fixed point RAS iteration by Krylov methods. Analogously, in the nonlinear case, we can accelerate the NRAS fixed point iteration (3.18) by applying Newton's method to the fixed point equation

$$\mathcal{F}(\mathbf{u}) := \mathbf{u} - \text{NRAS}(\mathbf{u}) = 0$$

resulting in the One-level RASPEN method [39] detailed in Algorithm 3.5.1. It was shown in [39] that the Jacobian of $\mathcal{F}(\mathbf{u})$ can be computed directly.

Algorithm 3.5.1 One-level RASPEN to solve $F(\mathbf{u}) = 0$

Require: $\mathbf{u}^0$, residual function F, Jacobian ∇F
  for outer iteration $k = 0, \ldots,$ to convergence **do**
    Compute NRAS update $\widehat{\mathbf{u}}^k = \text{NRAS}(\mathbf{u}^k)$;
    Set $\mathcal{F}(\mathbf{u}^k) = \mathbf{u}^k - \widehat{\mathbf{u}}^k$
    Solve $\mathbf{u}^{k+1} = \mathbf{u}^k - [\nabla\mathcal{F}(\mathbf{u}^k)]^{-1}\mathcal{F}(\mathbf{u}^k)$.
  end for
  In other words, solve $\mathcal{F}(\mathbf{u}) = 0$ via Newton's method.

Remark 3.3 (Connection to ASPIN)**.** The ASPIN method is based on the nonlinear Additive Schwarz iteration, while the RASPEN method is based on the nonlinear Restricted Additive Schwarz iteration. As in the linear case, the "restricted" variant contains the addition of partition of unity matrices. Therefore, the nonlinearly preconditioned system for ASPIN is given by

$$\mathcal{F}(\mathbf{u}^k) = \mathbf{u}^k - \sum_j R_j^T G_j(\mathbf{u}^k),$$

where $G_j(\mathbf{u})$ is computed as in (3.16).

Remark 3.4 (Exact computation of the Jacobian)**.** Unlike in the case of ASPIN, for RASPEN the Jacobian is computed exactly. We describe here the computation of the Jacobian $\nabla\mathcal{F}(\mathbf{u}^k)$ for Algorithm 3.5.1. From (3.17), we have

$$\mathcal{F}(\mathbf{u}^k) = \mathbf{u}^k - \text{NRAS}(\mathbf{u}^k) = \mathbf{u}^k - \sum_j R_j^T D_j G_j(\mathbf{u}^k).$$

Then the Jacobian is given by

$$\nabla\mathcal{F}(\mathbf{u}^k) = \nabla\left(\mathbf{u}^k - \sum_j R_j^T D_j G_j(\mathbf{u}^k)\right) = I - \sum_j R_j^T D_j \nabla G_j(\mathbf{u}^k). \tag{3.21}$$

To evaluate this, we must compute $\nabla G_j(\mathbf{u}^k)$ by the gradient of (3.16):

$$\nabla G_j(\mathbf{u}^k) = R_j - [R_j \nabla F(\mathbf{v}_j^k) R_j^T]^{-1} R_j \nabla F(\mathbf{v}_j^k) \tag{3.22}$$

where we have set $\mathbf{v}_j^k = R_j^T G_j(\mathbf{u}^k) + (I - R_j^T R_j)\mathbf{u}^k$ for convenience. Thus, we have

$$\begin{aligned}
\nabla\mathcal{F}(\mathbf{u}^k) &= I - \sum_j R_j^T D_j \nabla G_j(\mathbf{u}^k), \\
&= I - \sum_j R_j^T D_j (R_j - [R_j \nabla F(\mathbf{v}_j^k) R_j^T]^{-1} R_j \nabla F(\mathbf{v}_j^k)), \\
&= \sum_j R_j^T D_j [R_j \nabla F(\mathbf{v}_j^k) R_j^T]^{-1} R_j \nabla F(\mathbf{v}_j^k),
\end{aligned} \tag{3.23}$$

where we have used the partition of unity property $I = \sum_j R_j^T D_j R_j$. We note that the matrix $[R_j \nabla F(\mathbf{v}_j^k) R_j^T]^{-1}$ can be dense and does not need to be directly assembled. Therefore, we set up $\nabla \mathcal{F}(\mathbf{u}^k)$ as a linear operator, allowing us to compute the matrix-vector product $\nabla \mathcal{F}(\mathbf{u}^k) x^k$ for any vector x^k without explicitly computing the matrix $\nabla \mathcal{F}(\mathbf{u}^k)$.

Two-level RASPEN It is well-known that one-level domain decomposition methods are not scalable with respect to the number of subdomains. Specifically, the GMRES method used at each outer iteration of Algorithm 3.5.1 is not scalable without a coarse level. Therefore, we implement a Two-level RASPEN variant.

We choose a Two-level RASPEN correction provided in Algorithm 11 of [68], which allows the coarse correction to be done algebraically. The coarse correction is chosen to be computed after the local nonlinear solves. The authors of [57] provide numerical experiments on multiple different choices of two-level methods, including adding the coarse correction before, after, and before and after the nonlinear RAS iteration. The Two-level RASPEN method we have chosen is given by Algorithm 3.5.2. This method solves the nonlinearly preconditoned system

$$\mathcal{F}(\mathbf{u}) = \mathbf{u} - \text{NRAS}(\mathbf{u}) + R_H^T c_H(\text{NRAS}(\mathbf{u})), \tag{3.24}$$

where the function $c_H(\mathbf{v})$ for a given $\mathbf{v}$ is defined as the solution to the "coarse" nonlinear equation

$$R_H F(\mathbf{v} - R_H^T c_H(\mathbf{v})) = 0. \tag{3.25}$$

Algorithm 3.5.2 Two-level RASPEN to solve $F(\mathbf{u}) = 0$

Require: $\mathbf{u}^0$, residual function F, Jacobian ∇F
  for outer iteration $k = 0, \ldots,$ to convergence **do**
    Compute NRAS update $\widehat{\mathbf{u}}^k = \text{NRAS}(\mathbf{u}^k)$;
    Solve coarse problem $R_H F(\widehat{\mathbf{u}}^k - R_H^T c_H(\widehat{\mathbf{u}}^k)) = 0$ for $c_H(\widehat{\mathbf{u}}^k)$ via Newton's method;
    Set $\mathcal{F}(\mathbf{u}^k) = \mathbf{u}^k - \widehat{\mathbf{u}}^k + R_H^T c_H(\widehat{\mathbf{u}}^k)$;
    Solve $\mathbf{u}^{k+1} = \mathbf{u}^k - [\nabla \mathcal{F}(\mathbf{u}^k)]^{-1} \mathcal{F}(\mathbf{u}^k)$.
  end for
  In other words, solve $\mathcal{F}(\mathbf{u}) = 0$ via Newton's method.

Two-step method Next, we consider a Two-step method proposed in [26] under the name NKS-RAS; see also [17]. This method alternates between an NRAS iteration and a global Newton step at each outer iteration. This method leads to fast convergence (typically quadratic convergence is expected), but unlike Algorithms 3.5.1 and 3.5.2, can not be framed as a nonlinear preconditioning method. The Two-step method is given in Algorithm 3.5.3. The main feature of the Two-step method is the ease of its implementation. Compared to the standard Newton's method, it simply requires an additional NRAS step; moreover, compared to the One and Two-level RASPEN methods, the Two-step method does not involve any dense linear systems, and therefore, in principle, may be implemented based on a direct linear solver. Here, the linearized system of Algorithm 3.5.3 is solved using the same preconditioned GMRES method as in Algorithm 3.1.1.

Algorithm 3.5.3 Two-step method to solve $F(\mathbf{u}) = 0$

Require: $\mathbf{u}^0$, residual function F, Jacobian ∇F
  for outer iteration $k = 0, \ldots,$ to convergence **do**
    Compute NRAS update $\widehat{\mathbf{u}}^k$ =NRAS($\mathbf{u}^k$);
    Compute Newton update $\mathbf{u}^{k+1} = \widehat{\mathbf{u}}^k - [\nabla F(\widehat{\mathbf{u}}^k)]^{-1} F(\widehat{\mathbf{u}}^k)$;
  end for

Anderson Acceleration of Coarse Two-step Method The Two-step method given in Algorithm 3.5.3 involved a global fine-scale linear solve at each outer iteration. We now propose to replace this step with a "coarse" linear solve, using the Trefftz coarse space described in Section 3.3. For this, we can employ a method similar to the Two-step method (3.5.3) but with a "coarse" Newton step at each iteration, in a method similar to our previous works on linear models. We refer to this as the Coarse Two-Step method, which allows us to use the coarse Trefftz space in the iteration and cheapen the cost of the global Newton update. This Coarse Two-step method is a fixed point iteration which involves a coarse Newton update, which we denote by

$$F_c(\mathbf{u}) = \mathbf{u} - R_H^T \boldsymbol{\delta}_H, \quad \text{where} \quad \boldsymbol{\delta}_H = [R_H \nabla F(\mathbf{u}) R_H^T]^{-1} R_H F(\mathbf{u}). \tag{3.26}$$

With this, the fixed point coarse Two-step method is given as follows: for iteration $k = 0, 1, \ldots,$ to convergence, compute

$$\begin{aligned} \widehat{\mathbf{u}}^k &= \text{NRAS}(\mathbf{u}^k), \\ \mathbf{u}^{k+1} &= F_c(\widehat{\mathbf{u}}^k). \end{aligned} \tag{3.27}$$

The convergence of the fixed point method (3.27) to the numerical solution is generally quite slow when compared to the original Two-step method given by Algorithm 3.5.3, as a full update is not done at each iteration. To combat this, we introduce Anderson Acceleration [105] as a way to accelerate the fixed point method (3.27), resulting in Algorithm 3.5.4. The general Anderson acceleration method is given in Procedure 2. Given some $\mathbf{u}^{k+1} = P(\mathbf{u}^k)$, we solve $\mathcal{F}(\mathbf{u}) = P(\mathbf{u}) - \mathbf{u} = 0$ via Anderson acceleration. While the parameter m_k in Algorithm 2 can vary, we generally choose $m_k = 10$ in our numerical experiments and do not explore this parameter further. For convenience, we will refer to the Anderson Acceleration of the Coarse Two-step method (Algorithm 3.5.4) as the "Anderson method" for the duration of the chapter. We note that Anderson's method generally fails to accelerate quadratically convergent fixed point iterations; this is proven with an error estimate in [47].

Algorithm 3.5.4 Anderson Acceleration Applied to Coarse Two-step method to solve $F(\mathbf{u}) = 0$

Require: $\mathbf{u}^0$, residual function F, Jacobian ∇F
  Denote $F_c(\mathbf{u}) := \mathbf{u} - R_H^T [R_H \nabla F(\mathbf{u}) R_H^T]^{-1} R_H F(\mathbf{u})$;
  Solve $V(\mathbf{u}) := F_c(\text{NRAS}(\mathbf{u})) = 0$ by Anderson acceleration, (Procedure 2).
  The solution to $V(\mathbf{u}) = 0$ is the solution to $F(\mathbf{u}) = 0$.

Procedure 2 Anderson Acceleration to solve some general function of the form $V(\mathbf{u}) = P(\mathbf{u}) - \mathbf{u} = 0$

Require: $\mathbf{u}^0$ and $m \geq 1$
  Set $\mathbf{u}^1 = P(\mathbf{u}^0)$;
  for outer iteration $k = 1, 2, \ldots,$ to convergence **do**
    Set $m_k = \min\{m, k\}$;
    Set $f_k = (\mathbf{V}_{k-m_k}, ..., \mathbf{V}_k)$, where $\mathbf{V}_i = P(\mathbf{u}^i) - \mathbf{u}^i$;
    Determine $\alpha^k = (\alpha_0^k, \ldots, \alpha_{m_k}^k)^T$ that solves

$$\min_{\alpha^k = (\alpha_0^k, ..., \alpha_{m_k}^k)^T} ||f_k \alpha^k||_2 \quad \text{such that} \quad \sum_{i=0}^{m_k} \alpha_i^k = 1;$$

    Update $\mathbf{u}^{k+1}$ using Anderson mixing: $\mathbf{u}^{k+1} = \sum_{i=0}^{m_k} \alpha_i^k P(\mathbf{u}^{k-m_k+i})$;
  end for

3.5.1 Complexities/Costs of Fine-scale Methods

We now provide an overview of the complexity of each method. Recall that all proposed methods besides Newton's method involve an NRAS update. Computing the NRAS update (3.17) requires solving a family of local nonlinear systems associated with the subdomains. The local nonlinear systems are computed by Newton's method, with a linear system solved at each iteration. We denote by $N_{it,loc}$ the required number of local Newton iterations averaged over all subdomains. Due to the small size of the subdomains, the linearized system at each Newton iteration is computed via a direct sparse linear solve. This number might vary according to the method used, and we use an upper index to distinguish different case scenarios. We denote by $C(LU_{loc})$ a typical cost of the local LU decompositions on each subdomain, assumed to be approximately the same for every subdomain. Additionally, we denote by $C(\text{local assembly})$ the cost of assembling the local residual and Jacobian vectors, which will be done for all subdomains at each local iteration. The complexity of the NRAS iteration at each outer iteration can be expressed by:

$$C(\text{NRAS}) = N \cdot N_{it,loc} \left(C(LU_{loc}) + C(\text{local assembly}) \right), \tag{3.28}$$

where N is the number of subdomains. Here, (3.28) assumes that the local computations are not done in parallel. With a parallel implementation, instead of the NRAS cost being summed over all subdomains, the cost would be dominated by the subdomain associated to the maximum number of local linear solves.

With the exception of Anderson's method, all methods involve a global linear system which is solved via GMRES at each iteration. The complexity of the GMRES method *for the Two-step and Newton methods* at each outer iteration is given by the following:

$$N \cdot C(LU_{loc}) + C(LU_{coarse}) + C(\text{assembly}) + N_{\text{GM}} \cdot C(\text{GMRES}), \tag{3.29}$$

where $N \cdot C(LU_{loc}) + C(LU_{coarse})$ denotes the cost of assembling the preconditioner at each outer iteration, $C(\text{assembly})$ denotes the cost of assembling the global residual and Jacobian vectors/matrices $F(\mathbf{u}), \nabla F(\mathbf{u})$, and $C(\text{GMRES})$ denotes the cost of each GMRES iteration, which generally involves a sparse matrix-vector multiplication. We denote by N_{GM} the *average* number of GMRES iterations per outer

iteration. That is, N_{GM} denotes the average number of GMRES iterations until convergence of the preconditioned linear systems occurring at each outer iteration, which will depend on the chosen solution strategy. We use the upper index to N_{GM}^i to illustrate that this number can be different and decrease with the addition of an accelerator like a coarse space.

We note that in the RASPEN variants, the LU decompositions and assemblies for the nonlinearly preconditioned system can be reused from the earlier NRAS iteration, reducing the cost of the GMRES method compared to (3.29). That is, the complexity of the GMRES method *for the One and Two-level RASPEN methods* at each outer iteration is given by the following:

$$N_{\text{GM}} \cdot C(\text{GMRES}),$$

where this cost is equal to (3.29) for the Newton and Two-step methods. We denote by $C(\text{coarse})$ the cost of the sparse linear solve used for the coarse nonlinear problem in the Two-level RASPEN algorithm. Therefore, $N_c \cdot C(\text{coarse})$ denotes the total cost of the coarse nonlinear problems per outer iteration, where N_c denotes the average number of coarse linear solves per outer iteration.

With this, the costs for each method are summarized to be

- Newton: $N \cdot C(LU_{loc}) + C(LU_{coarse}) + C(\text{assembly}) + N_{\text{GM}} \cdot C(\text{GMRES})$;
- Two-Step: $C(\text{NRAS}) + N \cdot C(LU_{loc}) + C(LU_{coarse}) + C(\text{assembly}) + N_{\text{GM}} \cdot C(\text{GMRES})$;
- One-level RASPEN: $C(\text{NRAS}) + N_{\text{GM}} \cdot C(\text{GMRES})$;
- Two-level RASPEN: $C(\text{NRAS}) + N_{\text{GM}} \cdot C(\text{GMRES}) + N_c \cdot C(\text{coarse})$;
- Anderson: $C(\text{NRAS}) + C(\text{assembly}) + C(\text{coarse})$.

Furthermore, we quantify the following costs as follows:

- $C(LU_{loc}) = O(N_{loc}^{3/2})$, where $N_{loc} \approx \frac{N_\Omega}{N}$ denotes the number of degrees of freedom in a subdomain and is assumed to be roughly consistent over all subdomains;
- $C(LU_{coarse}) = O(N_{\mathcal{V}}^{3/2})$;
- $C(\text{local assembly}) = O(N_{loc})$;
- $C(\text{assembly}) = O(N_\Omega)$;
- $C(\text{GMRES}) = O(N_\Omega)$ for each sparse matrix-vector product;
- $C(\text{coarse}) = O(N_{\mathcal{V}}^{3/2})$;

where the LU decomposition of some sparse $m \times m$ matrix is supposed to be $O(m^{3/2})$ [59].

We denote the total number of outer iterations by N_k; this number may vary according to the method used, and we can use an upper index to distinguish between methods. Finally, the total cost over all outer iterations is given by:

$$
\begin{aligned}
C(\text{Newton}) &= N_k^0\Big(N \times O(N_{loc}^{3/2}) + (N_{\text{GM}}^0+1) \times O(N_\Omega) + O(N_{\mathcal{V}}^{3/2})\Big);\\
C(\text{Two-Step}) &= N_k^1\Big(N(N_{it,loc}^1+1)\times O(N_{loc}^{3/2}) + (N_{\text{GM}}^1+N_{it,loc}+1)\times O(N_\Omega) + O(N_{\mathcal{V}}^{3/2})\Big);\\
C(\text{1-RASPEN}) &= N_k^2\Big(N\cdot N_{it,loc}^2 \times O(N_{loc}^{3/2}) + (N_{\text{GM}}^2+N_{it,loc}) \times O(N_\Omega)\Big);\\
C(\text{2-RASPEN}) &= N_k^3\Big(N\cdot N_{it,loc}^3 \times O(N_{loc}^{3/2}) + (N_{\text{GM}}^3+N_{it,loc}) \times O(N_\Omega) + N_c\times O(N_{\mathcal{V}}^{3/2})\Big);\\
C(\text{Anderson}) &= N_k^4\Big(N\times N_{it,loc}^4 \times O(N_{loc}^{3/2}) + O(N_\Omega) + O(N_{\mathcal{V}}^{3/2})\Big);
\end{aligned}
$$

where the $O(N_\Omega)$ term arises as the dominant term in the least-squares problem of Anderson's method and $C(\text{assembly})$ is included in Anderson's method as well. Additionally, we have simplified the cost of the local assembly as $N\cdot N_{it,loc}\cdot O(N_{loc}) \approx N_{it,loc}O(N_\Omega)$ from the definition of N_{loc}.

We remark that for a time-dependent problem such as (3.1), there will be an additional N_T term multiplied into the cost of each method, where N_T denotes the number of time steps. In this case, N_k^i would denote the average number of outer iterations per time step. As shown, we can estimate the complexity of individual components of the methods such as the cost of assembly, the cost of a sparse linear solve, and the cost of a matrix-vector product. Additionally, we report the iteration counts such as N_k^i, N_{GM}^i, $N_{it,loc}^i$, and N_c in Section 3.6.

For the Two-step, Newton, and Two-level RASPEN methods, the GMRES method occurring at each outer iteration will be preconditioned with some two-level DD preconditioner. Therefore, we anticipate that N_{GM}^0, N_{GM}^1, and N_{GM}^3 will be approximately equal, with N_{GM}^2 for the One-level RASPEN method growing as we increase N. We also anticipate that $N_{it,loc}^i$ will be consistent between all methods which involve an NRAS solve (each method besides Newton). Therefore, the cost of the Two-step and Two-level RASPEN methods can be directly compared by comparing the the outer iterations N_k, where $N_{\mathcal{V}} << N_\Omega$ such that the cost of a coarse linear solve will be significantly lower than a fine linear solve. For reference, we will still record the number of coarse linear iterations N_c for the Two-level RASPEN to ensure it is not dominating the cost.

For Newton's method, it is clear that the cost of each iteration is lower than the Two-step and RASPEN methods, as there is no NRAS solve involved. However, the main goal of the proposed methods is to substantially reduce the number of outer iterations to make up for the increased complexity within each outer iteration. That is, we expect N_k to be reduced for the other methods, allowing for a potential gain in performance. With regard to Anderson's method, it appears to have quite a low cost at each iteration, with one NRAS solve at each iteration and the cost of the least-squares problem dominated by one $O(N_\Omega)$ term. However, we do expect a higher value of outer iterations for this method N_k^4 when compared to the Two-step and RASPEN variants. We note that while it is difficult to directly compare an outer iteration of Anderson to the other methods, it is still an interesting alternative that is worth exploring in this thesis.

With this, our main concern in terms of complexity lies in the term $N_k^i N_{\text{GM}}^i O(N_\Omega)$, representing the total number of "inner" GMRES iterations. However, when applicable, we also wish to record the term $N_k^i N_{it,loc}^i$, representing the total number of sparse local linear solves on a given subdomain. In Section 3.6, we record the number of outer iterations N_k^i, the average number of GMRES iterations per outer iteration N_{GM}^i, the cumulative number of GMRES iterations $N_k^i N_{\text{GM}}^i$, as well as the average and cumulative (over all outer iterations) local linear solves per subdomain, namely

$N^i_{it,loc}$ and $N^i_k N^i_{it,loc}$.

3.6 Numerical Results

We now provide numerical experiments for the proposed algorithms (Algorithms 1-5) for multiple test cases. These numerical tests will proceed in order of increasing difficulty; they include a stationary porous medium equation on an L-shaped domain, the similar equation on a small realistic urban domain, and the Diffusive Wave model on a large realistic urban domain. For all numerical experiments, the overlap is set to $\frac{1}{20}\mathcal{H}_j$, where $\mathcal{H}_j = \max(x_{max,j} - x_{min,j}, y_{max,j} - y_{min,j})$ and $x_{min,j}$, $y_{min,j}$, $x_{max,j}$, $y_{max,j}$ denote the minimal and maximal x and y coordinates that are contained in Ω_j. We do not explore the effect of overlap further.

3.6.1 Porous Medium Equation on L-shaped Domain

As a first numerical example, referred to as Example 1, we use an L-shaped domain with a reentering corner. The domain is defined by $D = (-1,1)^2$, $\Omega_S = (0,1)^2$ and $\Omega = D \setminus \overline{\Omega_S}$ such that the model domain has a singularity/corner in the upper right quadrant of the domain. With this model domain, we consider a porous medium equation given by

$$\begin{cases} u(x,y) + \operatorname{div}(\nabla(u(x,y)^4)) &= 0 \quad \text{in} \quad \Omega, \\ u(x,y) &= 1 \quad \text{on} \quad y = 1, \\ u(x,y) &= 0 \quad \text{on} \quad x = 1, \\ \dfrac{\partial u(x,y)^4}{\partial \mathbf{n}} &= 0 \quad \text{on} \quad \Gamma_N, \end{cases} \tag{3.30}$$

where $\Gamma_N = \{(x,y) \in \partial\Omega \,|\, x \neq 1 \text{ and } y \neq 1\}$ denotes the Neumann boundary. Consider a piecewise affine finite element discretization, and let $\mathbf{u}$ denote a vector of interior node values of size N_Ω. We denote the nodal "hat" basis functions $(\eta^\ell)_{\ell=1}^{N_\Omega}$. We define the lumped finite-element mass matrix M and finite element stiffness matrix A such that

$$M_{ii} = \sum_{\ell=1}^{N_\Omega} \int_\Omega \eta_i \eta_\ell \, \mathrm{d}\mathbf{x}, \qquad \text{and} \qquad A_{i\ell} = \int_\Omega \nabla\eta_i \cdot \nabla\eta_\ell \, \mathrm{d}\mathbf{x}.$$

With this, the matrix-vector notation of (3.30) becomes

$$F(\mathbf{u}) = M\mathbf{u} + A\max(\mathbf{u}, 0)^4. \tag{3.31}$$

The finite element solution of this equation is shown in Figure 3.2.

We provide numerical results for this example for Newton's method given by Algorithm 3.1.1, the Two-step method from Algorithm 3.5.3, the One-level RASPEN method from Algorithm 3.5.1, the Two-level RASPEN method from Algorithm 3.5.2, and the Anderson acceleration method from Algorithm 3.5.4. For the Two-step and Newton methods, at each outer iteration, the linear system is solved using GMRES method combined with the linear two-level RAS preconditioner (3.15). We show results for coarse partitionings $N = 9 \times 9, 5 \times 5$, and 3×3; The choice of N being a

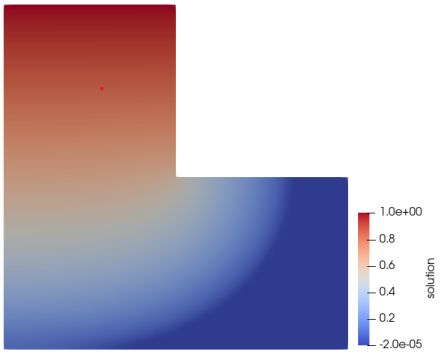


Figure 3.2: Finite element solution of (3.30) for Example 1 on the chosen L-shaped domain.

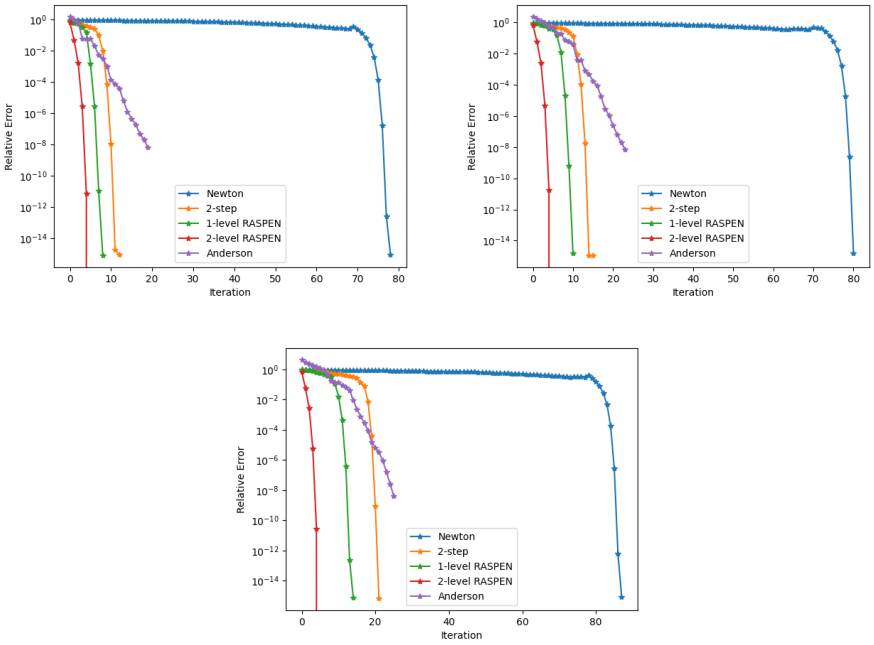


Figure 3.3: Example 1, convergence curves. Top: $N = 3$ (left), $N = 5$ (right). Bottom: $N = 9$.

Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	88	38.1	3353	N/A	N/A
Anderson	31	N/A	N/A	4.35	134.85
2-step	22	35.0	770	3.96	87.12
1-level RASPEN	15	85.3	1280	3.7	55.5
2-level RASPEN	6	19.2(+5.3)	115(+32)	3.94	23.64

Table 3.1: Example 1, $N = 9$, initial guess $\mathbf{u}^0 = 0.05$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	81	33.4	2705	N/A	N/A
Anderson	32	N/A	N/A	4.64	149.0
2-step	16	34.0	544	4.86	77.8
1-level RASPEN	11	53.9	593	5.08	55.9
2-level RASPEN	6	19.2(+4.5)	115(+27)	4.48	26.9

Table 3.2: Example 1, $N = 5$, initial guess $\mathbf{u}^0 = 0.05$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	79	33.7	2662	N/A	N/A
Anderson	25	N/A	N/A	4.36	109.0
2-step	13	38.9	506	6.06	78.8
1-level RASPEN	9	35.4	319	6.96	62.6
2-level RASPEN	6	20.5(+3.8)	123(+23)	5.5	33.0

Table 3.3: Example 1, $N = 3$, initial guess $\mathbf{u}^0 = 0.05$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

square of an odd number ensures the consistency of the mesh sequence in terms of the shape of the elements.

Convergence curves for this example are shown in Figure 3.3. Additionally, results for this example are summarized in Tables 3.1, 3.2, and 3.3. These tables report the total number of outer iterations N_k and, whenever appropriate, also report the average number of GMRES iterations N_{GM}, the total number of GMRES iterations $N_k \cdot N_{\text{GM}}$, the average number of local linear solves on a given subdomain per outer iteration $N_{it,loc}$, and the cumulative number of local linear solves on a given subdomain $N_k \cdot N_{it,loc}$.

In terms of outer iterations, from Figure 3.3, we see that Newton's method results in a very large plateau before the region of quadratic convergence is reached. This difference is stark when compared to the other methods. Additionally, for all number of subdomains N, Two-level RASPEN outperforms the other methods in terms of outer iteration count, with the One-level RASPEN, Two-step, Anderson, and Newton methods following in order. Additionally, we notice that while Anderson does not initially plateau like Newton's method, we do not obtain the eventual steep slope of convergence which the other methods achieve.

We see from Tables 3.1, 3.2, and 3.3 that the Two-level RASPEN method is scalable in terms of both outer iterations and GMRES iterations, in the sense that these values remain approximately the same for all N. In terms of GMRES iterations, we observe that the One-level RASPEN method loses scalability as the number of subdomains increases, more so than the other methods which all involve a coarse component of some type; particularly, as N is doubled in one direction, the GMRES iterations increase and $N_k \cdot N_{\text{GM}}$ about doubles as well. We also observe that New-

ton's method and the Two-step method, which both solve a linearized system with the linear two-level RAS preconditioner (with the Trefftz space as the coarse space), result in similar values of N_{GM} for all N. However, as the number of Two-step outer iterations is lower than the number of Newton outer iterations, the total number of GMRES iterations $N_k \cdot N_{\text{GM}}$ is significantly lower for the Two-step method. The Two-level RASPEN method results in both the smallest value of N_k and the smallest value of N_{GM}; therefore, it clearly results in the lowest number of cumulative GMRES iterations $N_k \cdot N_{\text{GM}}$. While the Two-level RASPEN has the additional $N_k \cdot N_c$ coarse linear solves, these coarse solves are generally less expensive than the global GMRES iterations, and the number of coarse linear solves is not exceedingly large.

In terms of local linear solves per subdomain, we do see a slight dependence of $N_{it,loc}$ on the number of subdomains N for all methods, likely due to the fact that the size of the local problems $\frac{N_\Omega}{N}$ gets larger as N is decreased. However, between methods for a given N, the values of $N_{it,loc}$ appear consistent as expected. Depending on the method, as N_k is increased, clearly $N_k \cdot N_{it,loc}$ will increase accordingly.

3.6.2 Porous Medium Equation on Large Urban Domain

As a second example, referred to as Example 2, we provide numerical results for a Porous Medium equation on a large urban domain of size 160×160 meters such that $D = (-80, 80)^2$ and Ω_S is given by the union of realistic perforations. The buildings are removed from the computational domain, leading to 72 total perforations. The geometry of the buildings have been provided by Métropole Nice Côte d'Azur. After discretization, this model domain contains 35220 triangles and 21317 mesh points. As mentioned, the overlap is set to $\frac{1}{20}\mathcal{H}_j$.

For this example, we ensure that the fine-scale triangulation is identical as the number of subdomains N changes by generating a finer background mesh to be used for various N; specifically, we can a fine-scale triangulation conforming to the coarse partitioning $N = 16 \times 16$ subdomains, then use this background triangulation for $N = 2 \times 2, 4 \times 4$, and 8×8 subdomains.

With this, we consider a porous medium equation given by

$$\begin{cases} u(x,y) + c\,\text{div}(\nabla(u(x,y)^m)) &= 0 \quad \text{in} \quad \Omega, \\ u(x,y) &= c \quad \text{on} \quad x = -80, \\ u(x,y) &= 0 \quad \text{on} \quad x = 80, \\ \dfrac{\partial u(x,y)^m}{\partial \mathbf{n}} &= 0 \quad \text{on} \quad \Gamma_N, \end{cases} \tag{3.32}$$

where $\Gamma_N = \{(x,y) \in \partial\Omega \,|\, x \neq -80 \text{ and } y \neq 80\}$ denotes the Neumann boundary; with this, the boundary of each perforation $(\partial\Omega \cap \partial\Omega_S)$ is included in Γ_N. Here, we take $m = 3$ and $c = 15$ in (3.32). With a similar finite element discretization to Example 1, the matrix-vector notation of (3.32) becomes

$$F(\mathbf{u}) = M\mathbf{u} + c\,A\max(\mathbf{u}, 0)^m,$$

with the finite element solution for this example shown in Figure 3.4.

For this example, we provide numerical results for Newton's method from Algorithm 3.1.1, the Two-step method from Algorithm 3.5.3, the Two-level RASPEN method from Algorithm 3.5.2, and the Anderson acceleration method from Algorithm 3.5.4. As we have established that One-level RASPEN is not a robust method

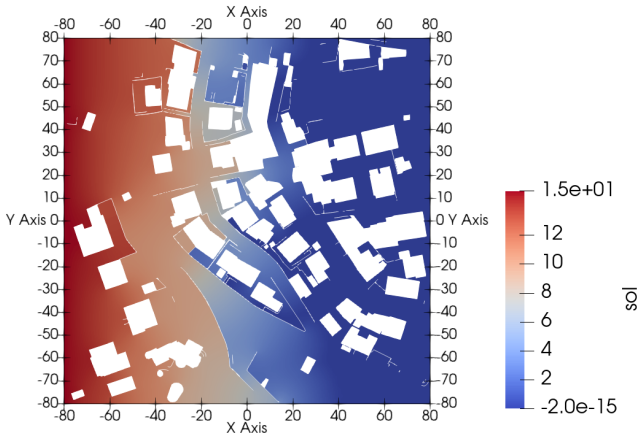


Figure 3.4: Finite element solution of (3.32) for Example 2 on the chosen urban model domain.

for large N, we do not provide results for One-level RASPEN in this example. As in the previous example, for the Two-step and Newton methods, at each outer iteration, the linear system is solved using GMRES method combined with the linear two-level RAS preconditioner (3.15).

The results for this example are summarized in Tables 3.4, 3.5, 3.6, and 3.7 for initial guesses of $\mathbf{u}^0 = 1$ and $\mathbf{u}^0 = 0$ and for subdomain partitioning of $N = 16 \times 16, 8 \times 8, 4 \times 4$, and 2×2, respectively. While the coarse partitionings vary, the background fine-scale triangulation is consistent throughout the numerical experiment. The total number of coarse nodes are given by $N_{\mathcal{V}}$ =784, 332, 149, and 56 for $N = 16 \times 16, 8 \times 8, 4 \times 4$, and 2×2, respectively. These tables report the total number of outer iterations N_k and, whenever appropriate, also report the average number of GMRES iterations N_{GM}, the total number of GMRES iterations $N_k \cdot N_{\text{GM}}$, the average number of local linear solves on a given subdomain per outer iteration $N_{it,loc}$, and the cumulative number of local linear solves on a given subdomain $N_k \cdot N_{it,loc}$.

Corresponding convergence curves for both initial guesses are provided in Figure 3.5. In terms of outer iterations, even more than the previous example, Newton's method has a large plateau in convergence before reaching the region of quadratic convergence. For all number of subdomains N, Two-level RASPEN outperforms the other methods in terms of outer iteration count, with the Two-step, Anderson, and Newton's methods following in increasing order. We particularly see that the outer iterations of the Two-level RASPEN method are extremely robust with respect to N, more so than the other accelerated methods. Additionally, from Tables 3.4, 3.5, 3.6, and 3.7, we see that there is a slight increase in N_k for the Two-step and Anderson methods as N is increased, but the increase in N_k is not proportional to the increase in N. The number of outer iterations N_k stays extremely consistent for the Newton and Two-level RASPEN methods as N is varied. In terms of initial guess, we observe that Newton's method is much more sensitive to the initial guess than the other methods in terms of outer iteration count. In fact, we see that this change in initial guess can result in a 50% increase in iteration count for Newton's method. Other than the outer iterations for Newton's method, there is no significant dependence on the initial guess.

From Tables 3.4, 3.5, 3.6, and 3.7, we see that N_{GM} for the Two-level RASPEN method is significantly smaller than that of the Two-step and Newton methods.

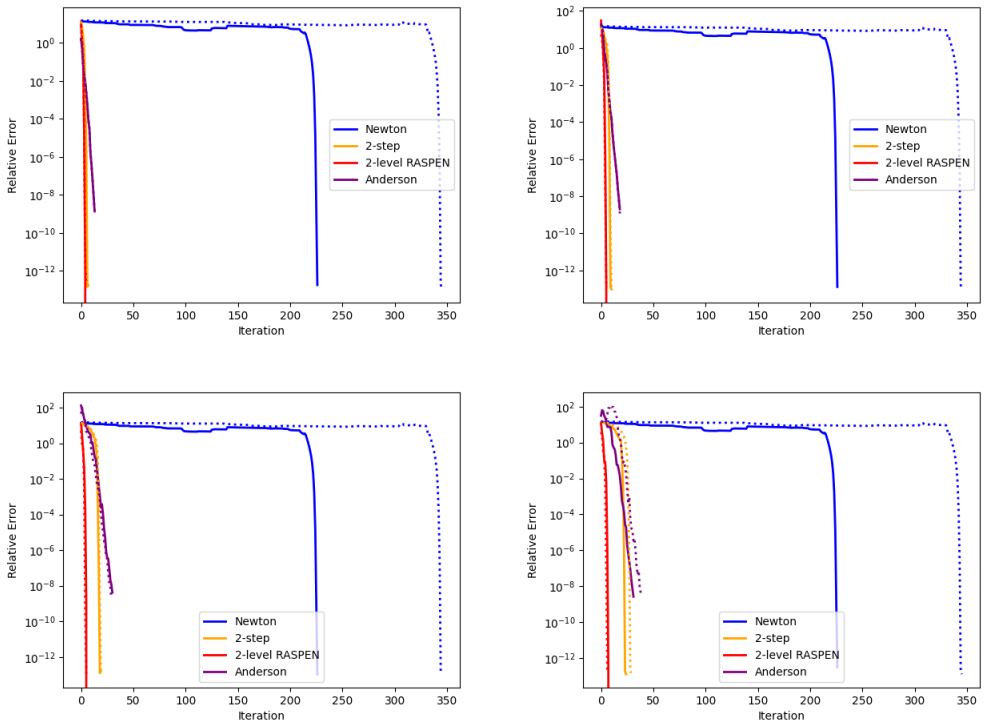


Figure 3.5: Example 2, convergence curves. Solid lines correspond to an initial guess of $\mathbf{u}^0 = 1$, dotted lines correspond to an initial guess of $\mathbf{u}^0 = 0$. Top: $N = 2 \times 2$ (left), $N = 4 \times 4$ (right). Bottom: $N = 8 \times 8$ (left), $N = 16 \times 16$ (right).

Additionally, as the number of outer iterations N_k is the lowest for the Two-level RASPEN method, the total number of GMRES iterations $N_k \cdot N_{\text{GM}}$ is the lowest for the Two-level RASPEN method. As in the previous example, the Two-step method and Newton's method result in around the same number of GMRES iterations per outer iteration N_{GM}, with this value slightly increasing as N is increased; however, this increase is not proportional to the increase in N. As the Two-step method results in fewer outer iterations N_k than Newton's method, the total number of GMRES iterations $N_k \cdot N_{\text{GM}}$ is significantly lower for the Two-step method. This can result in between 10 and 20 times more total GMRES iterations for Newton's method.

In terms of local linear solves per subdomain, our conclusions are similar to those of Example 1. Between methods, the values of $N_{it,loc}$ appear fairly consistent, but depending on N_k, $N_k \cdot N_{it,loc}$ will increase accordingly.

Overall, our conclusions from Example 2 are similar to that of Example 1, with Newton's method particularly struggling in terms of outer iterations and having an additional dependency on the initial guess. However, in terms of solving the linearized system, it appears that the Trefftz coarse space is successful as a component of the linear two-level RAS preconditioner for this nonlinear problem.

3.6.3 Diffusive Wave Model on Large Realistic Urban Domain

In the final numerical experiment, referred to as Example 3, we consider the Diffusive Wave model (3.1) used to model a hypothetical flood in a densely urbanized area

$\mathbf{u}^0 = 1$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	227	45.0	10215	N/A	N/A
Anderson	37	N/A	N/A	4.1	152
2-step	25	41.1	1028	4.14	104
2-RASPEN	9	22.3(+7.4)	201(+67)	3.13	28
$\mathbf{u}^0 = 0$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	346	46.0	15916	N/A	N/A
Anderson	52	N/A	N/A	5.07	264
2-step	30	41.6	1248	2.83	85
2-RASPEN	8	19.2(+9.6)	154(+77)	2.11	17

Table 3.4: Example 2, $N = 16 \times 16$ subdomains, results for initial guesses $\mathbf{u}^0 = 1$ and $\mathbf{u}^0 = 0$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

$\mathbf{u}^0 = 1$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	227	35.2	7990	N/A	N/A
Anderson	35	N/A	N/A	5.03	176
2-step	20	38.4	768	5.42	108
2-RASPEN	7	27.1(+8.0)	190(+56)	5.25	37
$\mathbf{u}^0 = 0$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	346	34.0	11764	N/A	N/A
Anderson	35	N/A	N/A	5.06	177
2-step	20	38.1	762	4.5	90
2-RASPEN	6	19.5(+8.5)	117(+51)	3.85	23

Table 3.5: Example 2, $N = 8 \times 8$ subdomains, results for initial guesses $\mathbf{u}^0 = 1$ and $\mathbf{u}^0 = 0$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

$\mathbf{u}^0 = 1$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	227	26.2	5947	N/A	N/A
Anderson	22	N/A	N/A	7.66	169
2-step	11	30.1	331	9.19	101
2-RASPEN	7	20.4(+5.9)	143(+41)	7.87	55
$\mathbf{u}^0 = 0$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	346	25.4	8788	N/A	N/A
Anderson	22	N/A	N/A	7.96	175
2-step	11	31.9	351	9.66	106
2-RASPEN	7	15.3(+6.3)	107(+44)	6.07	43

Table 3.6: Example 2, $N = 4 \times 4$ subdomains, results for initial guesses $\mathbf{u}^0 = 1$ and $\mathbf{u}^0 = 0$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

$\mathbf{u}^0 = 1$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	227	21.0	4767	N/A	N/A
Anderson	17	N/A	N/A	10.0	170
2-step	8	25.6	205	16.19	130
2-RASPEN	6	15.3(+4.3)	92(+26)	13.33	80
$\mathbf{u}^0 = 0$					
Method	N_k	N_{GM}	$N_k \cdot N_{\text{GM}}$	$N_{it,loc}$	$N_k \cdot N_{it,loc}$
Newton	346	18.3	6332	N/A	N/A
Anderson	17	N/A	N/A	13.93	237
2-step	8	25.6	205	24.53	196
2-RASPEN	6	13.2(+4.3)	79(+26)	26.21	157

Table 3.7: Example 2, $N = 2 \times 2$ subdomains, results for initial guesses $\mathbf{u}^0 = 1$ and $\mathbf{u}^0 = 0$. Values in brackets represent N_c for the N_{GM} column and $N_k \cdot N_c$ for the $N_k \cdot N_{GM}$ column.

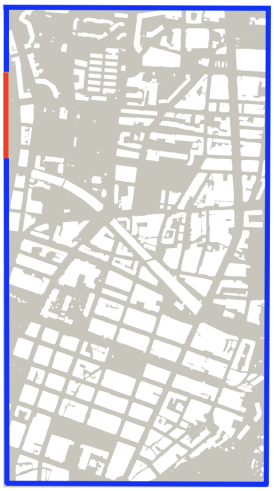


Figure 3.6: Boundary condition (thick blue and red lines) and corresponding model domain for Example 3. Red corresponds to $u = 21.5$ meters, blue corresponds to $u = z_b$. This results in around a 2-meter elevation for the portion of the boundary representing the Paillon river.

of the city of Nice, France. The model domain is depicted in Figure 3.8 and has dimension 850×1500 meters. In this test case scenario, the flood is produced by an overflow of Paillon river in the north-west part of the domain. Once again, the buildings are removed from the computational domain, leading to numerous perforations (447 in total). The bathymetry z_b uses the $1m$ Digital Elevation Model available from [64]. The parameters of the Diffusive Wave model are chosen formula as $\alpha = 1.5$ and $\gamma = 0.5$, based on Chézy's formula, with friction coefficient $c_f = 30$ which corresponding to a rough terrain [107]. We set Dirichlet boundary conditions, with water flow of about 2 meters coming from the Paillon river region and $u = z_b$ elsewhere; this boundary condition is shown in Figure 3.6. The initial condition is taken as equal to z_b and the final simulation time is set to $T_f = 1500$ seconds, which corresponds to a steady state solution.

Similarly to Example 2, we ensure that the fine-scale triangulation is identical as the number of subdomains N changes by generating a finer background mesh to be used for various N. We generate a fine-scale triangulation for $N = 8 \times 16$ subdomains, then use this background triangulation for $N = 1 \times 2, 2 \times 4, 4 \times 8$ and 8×16 subdomain partitionings. Figure 3.7 shows the matching background triangulation for $N = 2 \times 4$ and 8×16 subdomains, respectively.

Consider the NRAS iteration (3.17), which contains the solution of local nonlinear subproblems via Newton's method. In practice, for time-dependent problems, we come across the issue that some local subproblems will not converge for a given time increment $\Delta t_n = t_{n+1} - t_n$ in the residual function (3.5). We have found for our numerical examples that often, it is a small portion of subproblems that will not converge, generally due to stagnation in convergence. Therefore, it may be inefficient to reduce the global time increment for all subproblems and the entire iteration. We implement a local time step reduction strategy which makes it possible to reduce the time increment locally, then use this solution as an initial guess for the original system. Therefore, the *original* local system is still solved on each subdomain. This process is described in Algorithm 3. In this numerical experiment, except for Newton's method, the discussed local adaptive time-stepping method of Algorithm 3 is

Procedure 3 Local subproblem with local time-stepping

Require: $\Delta t, \mathbf{u}^k$, initial guess G_j^0, subdomain index j
Let $G_j(\mathbf{u}^k; G_j^0; \Delta t)$ denote the solution to the local subproblem (3.16) with initial guess G_j^0 and time step Δt used in the residual.
$\Delta t_{worked} = \Delta t$
1: Solve local subproblem with some $\Delta t_{worked} \leq \Delta t$ for $G_j^w = G_j(\mathbf{u}^k; G_j^0; \Delta t_{worked})$:
return $G_j^* = G_j^w$ if $\Delta t_{worked} = \Delta t$
Initialize $\Delta t_{try} = \Delta t_{worked}$
if $\Delta t_{worked} < \Delta t$ **then**
Until convergence:
2: Use G_j^w as initial guess for original Δt, try to solve for $G_j^* = G_j(\mathbf{u}^k; G_j^w; \Delta t)$
if Step 2 fails: Initial guess was insufficient to solve original system **then**
Form better initial guess: Increase $\Delta t_{try} = \frac{\Delta t + \Delta t_{try}}{2}$ and proceed to step 3
else Return G_j^*
end if
3: Obtain new initial guess: Reset and solve for $G_j^w = G_j(\mathbf{u}^k; G_j^w; \Delta t_{try})$
if Step 3 fails: Solve didn't converge for Δt_{try} **then**
Reduce $\Delta t_{try} = \frac{\Delta t_{worked} + \Delta t_{try}}{2}$ and return to step 3
else Solve converged for Δt_{try} and new initial guess G_j^w obtained
Set $\Delta t_{worked} = \Delta t_{try}$ and return to step 2
end if
end if

implemented with a global time increment of $\Delta t = 10$ seconds for each time step for all methods. We find that with this local adaptive time stepping, a global time increment reduction is not necessary for convergence.

For Newton's method, we begin with an initial time increment of $\Delta t_0 = 10$ seconds. If the system at a given time step can not be solved, the global time increment will be reduced by a factor of $\sqrt{2}$ until the reduced system is able to be solved. Then, once the reduced system is solved, we increase the time increment by a factor of $\sqrt{2}$ for the next time step, with a maximum time increment of 10 seconds. This means that $\Delta t_{n+1} = \min(\sqrt{2}\Delta t_n, 10)$. As mentioned, the final time for this experiment is $T_f = 1500$.

Figure 3.8 reports the simulation results for time $t = 10, 250, 750$, and 1500 seconds. In particular, in color, we report the water depth $\mathbf{h} = \mathbf{u} - z_b(\mathbf{x})$ over the "flooded region" where $\mathbf{h} \geq 1$cm. The ground surface elevation $z_b(\mathbf{x})$ is reported in black and white.

The spatial discretization results in 81444 mesh points and 131581 triangles in the model domain. As mentioned, the fine-scale triangulation will be conforming to the polygonal coarse partitioning. Although we record results for various numbers of subdomains N, the fine-scale mesh will be kept consistent throughout the experiment and is conforming to a $N = 8 \times 16$ coarse mesh; therefore, it will be conforming to $N = 1 \times 2$, $N = 2 \times 4$, $N = 4 \times 8$ meshes as well. The coarse and fine partitionings are shown in Figure 3.7 for multiple coarse partitionings. By the end of the simulation, 38216 points and 60846 triangles are "flooded" such that $h \geq 1$cm. The total number of coarse nodes are given by $N_{\mathcal{V}}$ =795, 357, 126, and 34 for $N = 8 \times 16, 4 \times 8, 2 \times 4$, and 1×2 subdomains, respectively. The overlap is once again set to $\frac{1}{20}\mathcal{H}_j$.

Recall that at every time step, the nonlinear system (3.5) has to be solved. For

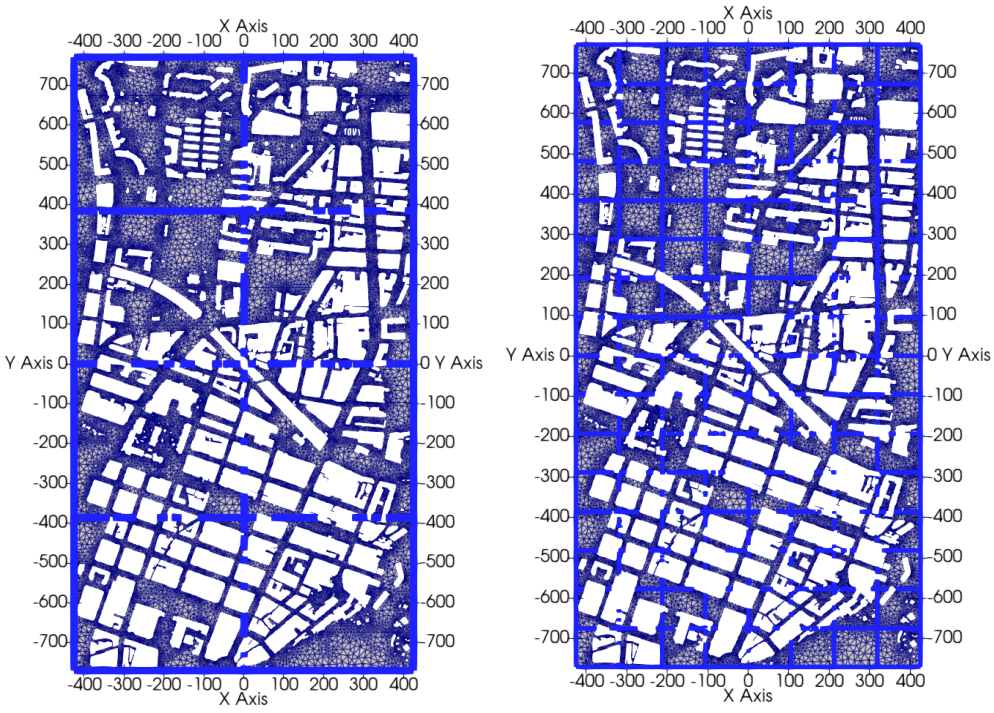


Figure 3.7: Coarse (thick blue lines) and fine (thin blue lines) discretizations for $N = 2 \times 4$ (left) and $N = 8 \times 16$ (right) subdomains for model domain representing the Nice port and Paillon river.

this example, in addition to Newton's method given by Algorithm 3.1.1, we provide numerical results for the One-level RASPEN method from Algorithm 3.5.1, the Two-level RASPEN method from Algorithm 3.5.2, the Two-step method from Algorithm 3.5.3, and the Anderson acceleration method from Algorithm 3.5.4.

Figure 3.9 displays the cumulative number of (successful) outer iterations over the simulation time, obtained for each of the methods for various numbers of subdomains N.

In addition, we record various values in Tables 3.8, 3.9, 3.10 and 3.11. For this time-dependent example, these values report the total number of time steps N_T, the total number of outer iterations over time $N_T \cdot N_k$, and, whenever appropriate, also report the cumulative number of GMRES iterations over time $N_T \cdot N_k \cdot N_{\text{GM}}$ and the total number of local linear solves on a given subdomain over all time steps and outer iterations $N_T \cdot N_k \cdot N_{it,loc}$. For the Two-step and Newton methods, at each outer iteration, the linear system is solved using GMRES method combined with the linear two-level RAS preconditioner (3.15).

Additionally, we report the number of GMRES iterations per outer iteration N_{GM} for each time step in Figure 3.10. As mentioned, the overall (cumulative) number of GMRES iterations is reported in Tables 3.8, 3.9, 3.10, and 3.11 for $N = 8 \times 16, 4 \times 8, 2 \times 4, 1 \times 2$, respectively.

From Figure 3.9, we see that the RASPEN variants (both one and two-level) as well as the Two-step method produce the smallest number of total outer iterations, and are very similar to one another in this regard. Additionally, we see that the Anderson acceleration changes the most as we increase N, with the outer iterations generally increasing. As expected, Newton's method results in a higher number of

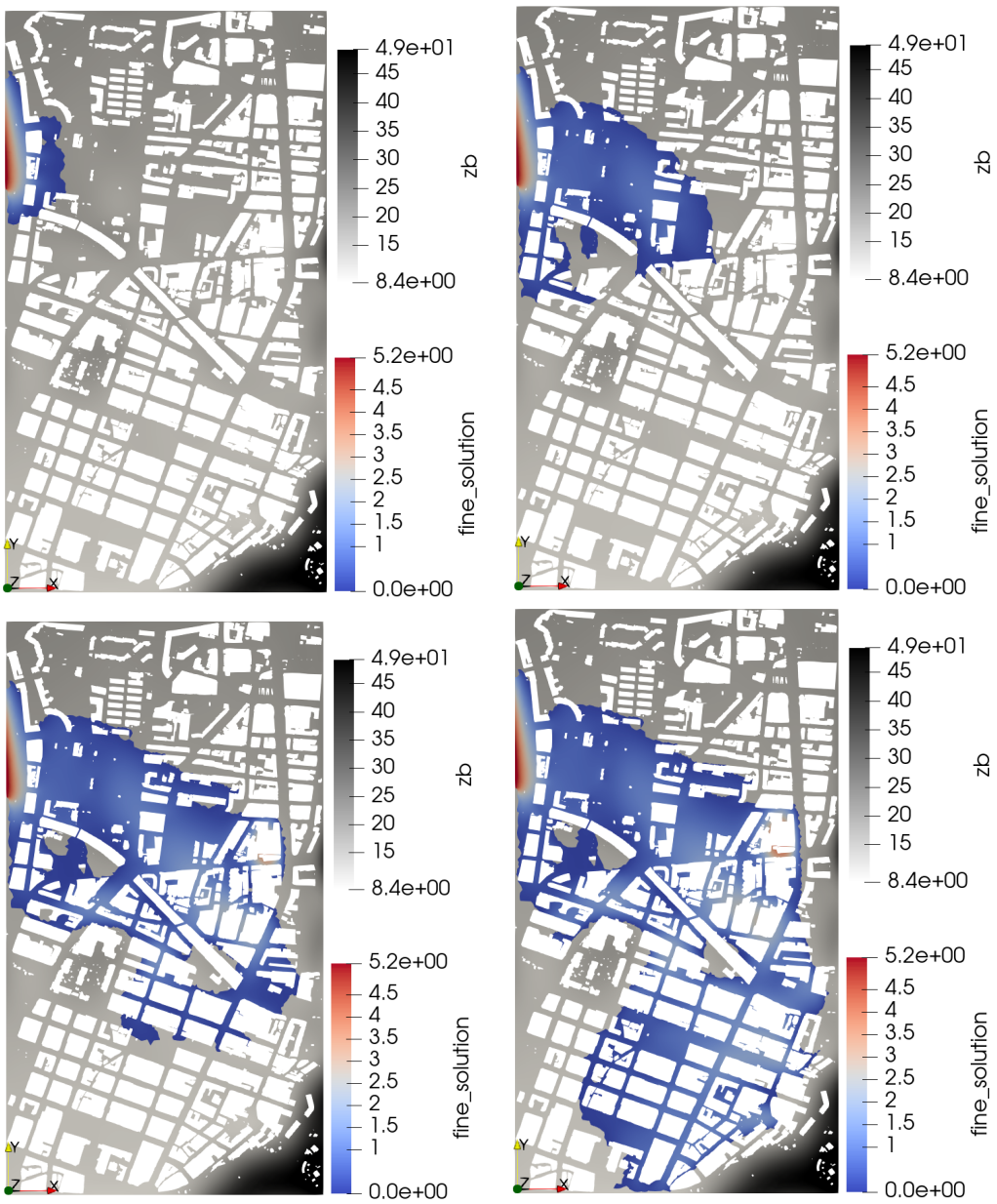


Figure 3.8: Numerical solution of Example 3 at various time steps, where the numerical solution is presented by $\mathbf{h} = \mathbf{u} - z_b(\mathbf{x})$. Top: $t = 10$ (left) seconds, $t = 250$ (right) seconds. Bottom: $t = 750$ (left) seconds, final $t = 1500$ (right) seconds. "Flooded" region (where $\mathbf{h} \geq 1$cm) are shown in color, with underlying bathymetry z_b shown in the background in black and white.

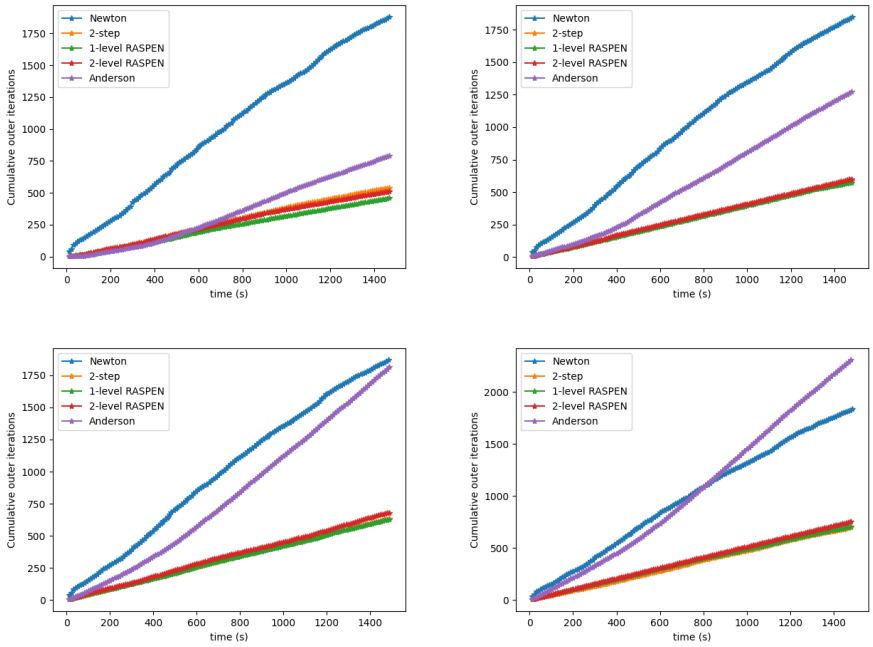


Figure 3.9: Example 3, cumulative outer iterations over time. Top: $N = 1 \times 2$ (left), $N = 2 \times 4$ (right). Bottom: $N = 4 \times 8$ (left), $N = 8 \times 16$ (right).

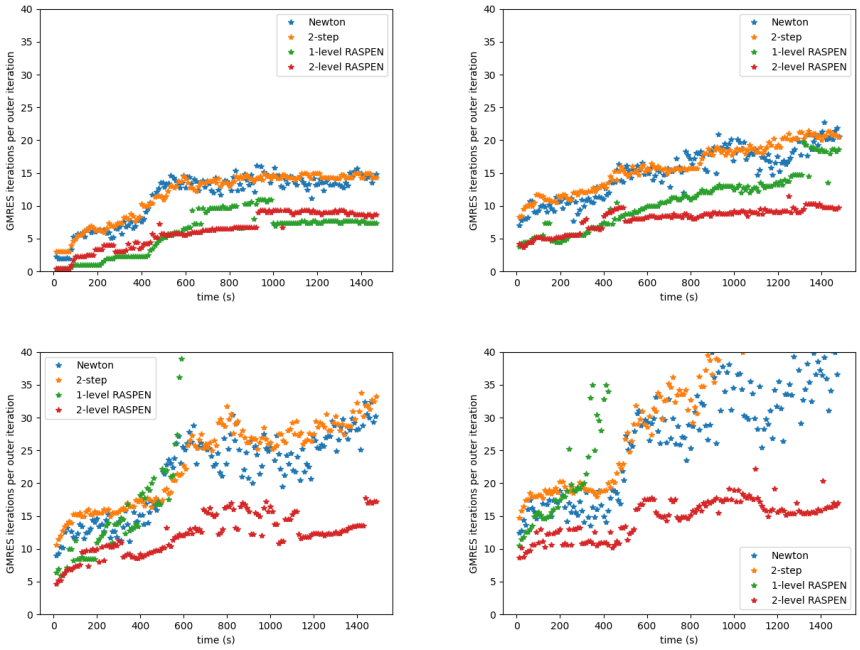


Figure 3.10: Example 3, average number of GMRES iterations per outer iteration N_{GM} at each time step. Top: $N = 1 \times 2$ (left), $N = 2 \times 4$ (right). Bottom: $N = 4 \times 8$ (left), $N = 8 \times 16$ (right).

Method	N_T	$N_T \cdot N_k$	$(N_T \cdot N_k \cdot N_{\text{GM}})$	$N_T \cdot N_k \cdot N_{it,loc}$
Newton	155	1797	44146	N/A
Anderson	150	2524	N/A	6849
2-step	150	687	20029	1557
1-level RASPEN	150	686	100453	1574
2-level RASPEN	150	730	12117(+2907)	2017

Table 3.8: Example 3, $N = 8 \times 16$ subdomains. The value in brackets represents $N_t \cdot N_k \cdot N_c$.

Method	N_T	$N_T \cdot N_k$	$(N_T \cdot N_k \cdot N_{\text{GM}})$	$N_T \cdot N_k \cdot N_{it,loc}$
Newton	160	1807	33432	N/A
Anderson	150	1902	N/A	6967
2-step	150	638	13164	1827
1-level RASPEN	150	624	33117	1824
2-level RASPEN	150	650	8065(+2031)	2334

Table 3.9: Example 3, $N = 4 \times 8$ subdomains. The value in brackets represents $N_t \cdot N_k \cdot N_c$.

Method	N_T	$N_T \cdot N_k$	$(N_T \cdot N_k \cdot N_{\text{GM}})$	$N_T \cdot N_k \cdot N_{it,loc}$
Newton	151	1776	26753	N/A
Anderson	150	1657	N/A	8538
2-step	150	592	9483	2412
1-level RASPEN	150	571	9040	2478
2-level RASPEN	150	595	6328(+1755)	3066

Table 3.10: Example 3, $N = 2 \times 4$ subdomains. The value in brackets represents $N_t \cdot N_k \cdot N_c$.

Method	N_T	$N_T \cdot N_k$	$(N_T \cdot N_k \cdot N_{\text{GM}})$	$N_T \cdot N_k \cdot N_{it,loc}$
Newton	155	1791	21608	N/A
Anderson	150	1225	N/A	7132
2-step	150	597	7794	3831
1-level RASPEN	150	477	3827	3377
2-level RASPEN	150	515	3983(+1114)	3698

Table 3.11: Example 3, $N = 1 \times 2$ subdomains. The value in brackets represents $N_t \cdot N_k \cdot N_c$.

outer iterations when compared to the RASPEN and Two-step methods.

We see from Figure 3.10 that as expected, as the number of subdomains is increased, the One-level RASPEN method is no longer scalable in terms of GMRES iterations; in fact, as N grows, the average GMRES iterations N_{GM} are significantly larger than the other methods such that they are out of the scope of the vertical axis in the figure (green dots in Figure 3.10). Additionally, as the number of subdomains grow, the Two-level RASPEN method clearly becomes the method with the lowest value N_{GM} per outer iteration over time. As in the previous examples, for Newton's method preconditioned with the two-level linear RAS preconditioner (blue dots in Figure 3.10) and the Two-step method preconditioned with the two-level linear RAS preconditioner (orange dots in Figure 3.10), the number of GMRES iterations are not increasing proportionally with N. However, we do see a slight increase as the number of subdomains is increased. We additionally remark that over time, the average number of GMRES iterations per outer iteration N_{GM} in Figure 3.10 increases for each method besides the Two-level RASPEN method; this is generally most likely due to an increased proportion of the model domain being "flooded". That is, the problem becomes increasingly difficult over time. However, this also suggests that the performance of the coarse correction is improved when applied "nonlinearly", as is done in the Two-level RASPEN method.

From Table 3.8, 3.9, 3.10, and 3.11, we see the results from Figures 3.9 and 3.10 summarized. Particularly, Newton's method results in much larger numbers of outer iterations when compared to the nonlinearly preconditioned variants (with the exception of the Anderson acceleration as N grows large). Additionally, the total number of GMRES iterations $N_T \cdot N_k \cdot N_{\text{GM}}$ in the One-level RASPEN method grows significantly as N is increased, more than doubling as N is doubled in one direction. It appears that particularly as the number of subdomains grows, Two-level RASPEN is the most efficient, producing the lowest number of total GMRES iterations with a relatively small number of coarse linear solves. We note again that the coarse linear solves are computationally less expensive than a global fine-scale linear solve. However, we do remark that we do not see the same type of acceleration in terms of outer iterations for Two-level RASPEN like we obtained in the other two numerical examples. Overall, particularly if one is choosing a high value of N, it appears that the Two-level RASPEN method outperforms the others.

We have mentioned that from Figure 3.9, the outer iterations of the Anderson method appear to have noticeably more dependence on N than the outer iterations of the other methods. To view this phenomenon, we provide the average outer iterations per time step N_k *for Anderson's method* in Figure 3.11. We see from Figure 3.11 that the outer iterations do increase noticeably with N. But in fact, the number of outer iterations stays reasonably stable as time increases, and the pattern of outer iterations seems to match that of average GMRES iterations of two-level RASPEN in Figure 3.10. This dependence on N suggests that the coarse space could be improved for this model problem.

3.7 Trefftz Galerkin method

Consider the coarse space provided in Section 3.3. We have shown in Chapter 2 that such a coarse space is capable of providing an accurate approximation of the linear diffusion problems. Furthermore, when combined with overlapping Schwarz methods, we have shown in Section 3.6 that this Trefftz space provides an efficient

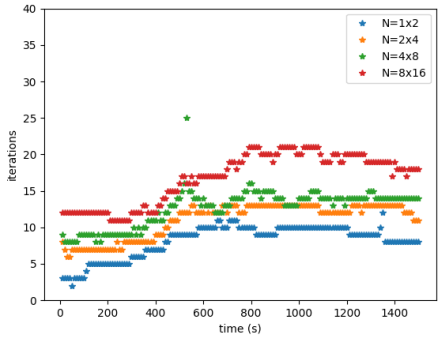


Figure 3.11: Example 3, Anderson outer iterations per time step. Colors correspond to various coarse partitionings.

two-level preconditioner for linear and nonlinear problems. In particular, the numerical experiment reported in Section 3.6.3 shows an excellent performance of the Trefftz-based two-level RASPEN method in the application to the Diffusive Wave equation.

In this section, we discuss the use of the Trefftz space for the approximation of the latter flood model. This method is described and corresponding numerical examples are provided. The main idea is quite simple, and, roughly speaking, consists in writing the Galerkin approximation of (3.2) based on the (conforming) coarse space V_H. In other words, the $\mathbb{P}_1$ finite element space in (3.3) is going to be replaced by the coarse space V_H. In addition, we project the topographical data z_b on the coarse space and perform mass lumping. This allows us to interpret the resulting discretization as a Finite Volume scheme with a local mass conservation property.

Consider the FV- FE discretization of the Diffusive Wave equation introduced in Section 3.2. For the sake of simplicity, let us assume zero Neumann boundary conditions on $\partial\Omega \setminus \partial\Omega_S$. Let $\Phi(\mathbf{u})$ denote the divergence term in the residual $F(\mathbf{u})$, that is

$$\Phi(\mathbf{u})_i = \sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha \left(u_i^{n+1} - u_\ell^{n+1}\right) \quad \text{for } i \in \mathcal{N}.$$

Then, we can express (3.5) as

$$F(\mathbf{u})_i = \frac{m_i}{\Delta t_n}\left(u_i^{n+1} - u_i^n\right) + \Phi(\mathbf{u})_i,$$

while the fine-scale discrete system (with new boundary condition) is expressed as

$$\frac{1}{\Delta t_n}\overline{M}\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) + \Phi(\mathbf{u}^{n+1}) = 0, \tag{3.33}$$

where $\overline{M}$ is the lumped version of the mass matrix M. To obtain the coarse system, we substitute $\mathbf{u}^k = R_H^T \mathbf{u}_H^k$, $k = n, n+1$, into (3.33) and multiply the latter by R_H, yielding

$$\frac{1}{\Delta t_n} R_H \overline{M} R_H^T \left(\mathbf{u}_H^{n+1} - \mathbf{u}_H^n\right) + R_H \Phi(R_H^T \mathbf{u}_H^{n+1}) = 0.$$

Denoting by $\overline{M}_H$ the lumped version of $R_H \overline{M} R_H^T$, that is

$$\left(\overline{M}_H\right)_{i\ell} = \begin{cases} \sum_\ell \left(R_H \overline{M} R_H^T\right)_{i\ell}, & i = \ell, \\ 0, & i \neq \ell, \end{cases}$$

we obtain the following coarse system

$$F_H(\mathbf{u}_H^{n+1}) := \frac{1}{\Delta t_n}\overline{M}_H\left(\mathbf{u}_H^{n+1} - \mathbf{u}_H^n\right) + R_H\Phi(R_H^T\mathbf{u}_H^{n+1}) = 0, \tag{3.34}$$

for each time step t_{n+1}. We refer to the numerical solution of (3.34) via Newton's method as the coarse Trefftz method, where $\mathbf{u}_H^{n+1}$ is the coarse solution at t_{n+1}. We wish to stress that unlike the algorithms mentioned in Section 3.5, this method is a coarse approximation to the fine-scale solution.

The following proposition summarizes some properties of the numerical scheme (3.34). In particular, it is shown to be locally mass conservative and capable of preserving certain stationary solutions.

Proposition 3.7.1. The discretization (3.34) satisfies the following properties:

(a) **Local Mass Conservation**: The scheme is locally mass conservative, in the sense that it can be expressed as

$$\frac{m_{H,i}}{\Delta t_n}\left(u_{H,i}^{n+1} - u_{H,i}^n\right) + \sum_\ell \tau_{H,i\ell}^n\left(u_{H,i}^{n+1} - u_{H,\ell}^{n+1}\right) = 0 \quad \text{for all } i \in \mathcal{N}_\mathcal{V}$$

for some $\tau_{H,i\ell}^n$, with $\tau_{H,i\ell}^n = \tau_{H,\ell i}^n$.

(b) **Preservation of Lakes at Rest**: If $\mathbf{u}_H^0$ is a constant vector, then $\mathbf{u}_H^n = \mathbf{u}_H^0$ satisfies (3.34) for all n; in particular $\Phi(R_H^T\mathbf{u}_H^n) = 0$.

(c) **Preservation of Dry Conditions**: Assume that $z_b \in V_H$, and let $\mathbf{z_{b,H}}$ be such that $(\mathbf{z_{b,H}})_s = z_b(\mathbf{x}_s)$ for $s = 1, \ldots, N_\mathcal{V}$. Then, if $\mathbf{u}_H^0 = \mathbf{z_{b,H}}$, it follows that $\mathbf{u}_H^n = \mathbf{z_{b,H}}$ satisfies (3.34) for all n; in particular, $\Phi(R_H^T\mathbf{u}_H^n) = 0$.

Before proceeding with a proof of Proposition 3.7.1, let us introduce some notations and technical results. For any $m \times k$ matrix A, let $RS(A)$ denote the diagonal "row sum" matrix of the same size defined by

$$RS(A)_{i\ell} = \begin{cases} \displaystyle\sum_\ell A_{i\ell}, & i = \ell, \\ 0, & i \neq \ell. \end{cases}$$

Let $\mathbf{1}_k$ denote a unit vector in $\mathbb{R}^k$. We have the following equivalence,

$$RS(A) = 0 \quad \Leftrightarrow \quad A\mathbf{1}_k = 0 \quad \Leftrightarrow \quad (A\mathbf{u})_i = \sum_\ell A_{i\ell}(u_i - u_l) \text{ for all } \mathbf{u} \in \mathbb{R}^m. \tag{3.35}$$

We further observe that

$$RS(A) = 0 \quad \Rightarrow \quad RS(BA) = 0, \tag{3.36}$$

for any $r \times m$ matrix B. Now, alternatively, let C be a $k \times r$ matrix satisfying the "partition of unity property" $C\mathbf{1}_r = \mathbf{1}_k$, then

$$RS(A) = 0 \quad \Rightarrow \quad RS(AC) = 0. \tag{3.37}$$

Proof of Proposition 3.7.1: In view of (3.35), to prove the statement *(a)* it suffices to show that for any $\mathbf{u}_H^{n+1}$, there exists a symmetric matrix $T_H(\mathbf{u}_H^{n+1})$ satisfying $RS\left(T_H(\mathbf{u}_H^{n+1})\right) = 0$ and such that $R_H\Phi(R_H^T\mathbf{u}_H^{n+1}) = T_H(\mathbf{u}_H^{n+1})\mathbf{u}_H^{n+1}$.

Let us denote by $T(\mathbf{u}^{n+1})$ the matrix with off-diagonal entries given by

$$T(\mathbf{u}^{n+1})_{i\ell} = T(\mathbf{u}^{n+1})_{\ell i} = -\tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha \qquad \text{for} \quad i \neq \ell,$$

and corresponding diagonal entries given by

$$T(\mathbf{u}^{n+1})_{ii} = \sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^n \left(h_{i\ell}^{n+1}\right)^\alpha.$$

We have that $\Phi(\mathbf{u}^{n+1}) = T(\mathbf{u}^{n+1})\mathbf{u}^{n+1}$ and we note that $RS(T(\mathbf{u}^{n+1})) = 0$. Furthermore, we can write

$$R_H \Phi(R_H^T \mathbf{u}_H^{n+1}) = T_H(\mathbf{u}_H^{n+1})\mathbf{u}_H^{n+1}$$

with $T_H(\mathbf{u}_H^{n+1}) = R_H T(R_H^T \mathbf{u}_H^{n+1}) R_H^T$. Clearly, $T_H(\mathbf{u}_H)$ is symmetric for all $\mathbf{u}_H$. Since $R_H^T \mathbf{1}_{N_\mathcal{V}} = \mathbf{1}_{N_\Omega}$, it follows from (3.37) and (3.36) that $RS(T_H(\mathbf{u}_H)) = 0$, which, in view of (3.35), completes the proof of *(a)*.

In view of (3.37), we have $RS(T(R_H^T \mathbf{u}_H^{n+1}) R_H^T) = 0$. Then, the statement *(b)* of the proposition follows from $\Phi(R_H^T \mathbf{u}_H^{n+1}) = T(R_H^T \mathbf{u}_H^{n+1}) R_H^T \mathbf{u}_H^{n+1}$. The statement *(c)* follows from $T(R_H^T \mathbf{z}_{b,H}) = 0$. □

3.7.1 Numerical experiment: Stationary convergence study

To evaluate the precision of the Trefftz Galerkin method, we perform a numerical convergence study for a stationary version of the Diffusive Wave equation posed over a medium size urban domain. The geometries considered in this test case, shown on Figure 3.12, are based on the realistic structural topography data of the city of Nice. The domain is a square with a length of 360 meters such that $D = (-180, 180)^2$ and Ω_S is given by the union of realistic perforations representing buildings and walls. More specifically, we are solving the following nonlinear elliptic problem

$$\begin{cases} \epsilon u(x,y) + \operatorname{div}(u(x,y)^\alpha \nabla u) &= 0 \quad \text{in} \quad \Omega, \\ u(x,y) &= 1 \quad \text{on} \quad x = -180, \\ u(x,y) &= 0 \quad \text{on} \quad x = 180, \\ u(x,y)^\alpha \dfrac{\partial u(x,y)}{\partial \mathbf{n}} &= 0 \quad \text{on} \quad \Gamma_N, \end{cases} \tag{3.38}$$

where $\Gamma_N = \{(x,y) \in \partial\Omega \,|\, x \neq -180 \text{ and } y \neq 180\}$ denotes the Neumann boundary such that each perforation $(\partial\Omega \cap \Omega_S)$ is included in Γ_N.

For both cases ("walls" and "no walls"), Figure 3.12 reports the numerical solution obtained by the fine-scale discretization, and the solution values above zero are reported in color. Because the diffusion operator in (3.38) degenerates at $u = 0$, and given the choice of the boundary conditions, the support of the solution does not extend over the whole domain.

To assess convergence rates of the coarse discretization, we perform the computations using the sequence of meshes with $N = 8^2, 16^2, 32^2, 64^2, 128^2$ and 256^2 regularly spaced coarse cells. The error is evaluated by comparing the solution resulting from the coarse discretization to the one obtained by the fine-scale method. Figure 3.13 exhibits the relative error in L^2 and H^1 norms for both sub-cases. For this nonlinear stationary case, the overall performance of the coarse discretization is

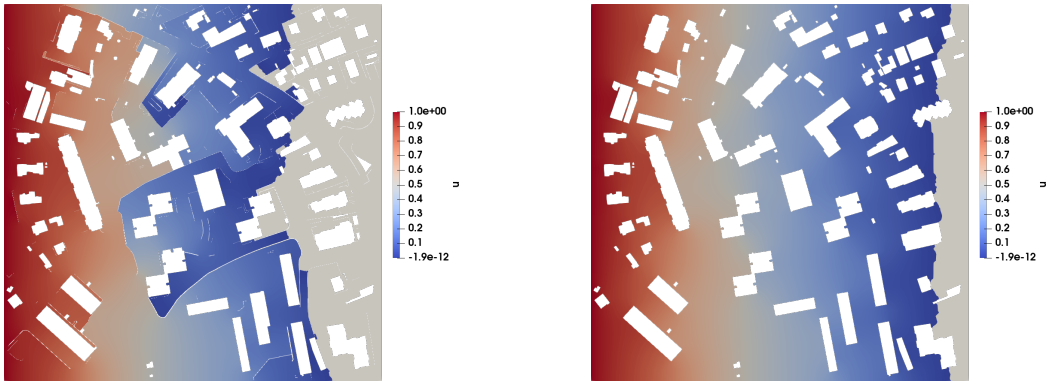


Figure 3.12: Approximate solution for the cases "walls" (left) and "no walls" (right).

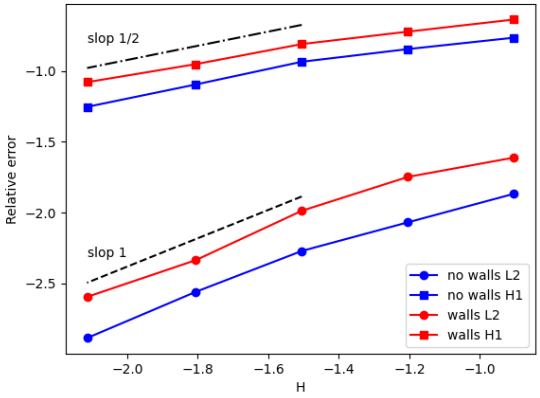


Figure 3.13: Log-log plot of relative L^2 and H^1 error for cases with and without walls as the function of the characteristic coarse cell size $H = \sqrt{N}$.

quite satisfactory, as it manages to provide a reasonable approximation of the solution using only a fraction of fine degrees of freedom. In particular, we observe from Table 3.12 that the relative L^2 error of order 10^{-2} can be achieved with $N = 16^2$ and $N = 32^2$ coarse cells for cases without and with walls, respectively. This is equivalent to using 2.7% (without walls) and 4% (with walls) of the fine degrees of freedom.

3.7.2 Numerical experiment: Large Urban Flood Model

We consider once again the large urban test case presented in Section 3.6.3 with slightly different boundary conditions. As before, the constant water level is imposed at the portion of the boundary located at the north-west of the domain, modeling the overflow of Paillon river. The south (bottom) boundary is subject to the Dirichlet boundary condition $u = z_b$, while on the remainder of the boundary a zero Neumann condition is imposed. We compare the approximate solution obtained by the coarse method presented above and the fine-scale solution based on the scheme introduced in Section 3.2. We consider the coarse discretization based on the regular partitioning of the domain into $N = 16 \times 32, 32 \times 64$ and 64×128 coarse cells. We note that the use of the coarse method allows for a significant reduction of the number of degrees of freedom, ranging in between factor 20 for the finest coarse

	Case "no walls"						Case "walls"					
$\sqrt{N}$	N_Ω	$N_\mathcal{V}$	err L^2	eoc	err H^1	eoc	N_Ω	$N_\mathcal{V}$	err L^2	eoc	err H^1	eoc
8	17790	234	0.0136	-	0.1718	-	4606	434	0.0245	0.0000	0.2308	0.0000
16	21682	581	0.0085	0.6704	0.1427	0.2678	42459	1027	0.0179	0.4538	0.1895	0.2841
32	33438	1609	0.0054	0.6722	0.1161	0.2973	62027	2490	0.0103	0.7950	0.1548	0.2920
64	60247	4903	0.0028	0.9597	0.0802	0.5337	104935	6639	0.0046	1.1590	0.1117	0.4713
128	180400	16896	0.0013	1.0583	0.0558	0.5125	243088	20313	0.0025	0.8411	0.0833	0.4127
256	636711	61601	0.0006	1.1211	0.0372	0.5835	733728	67989	0.0013	0.9678	0.0590	0.4985

Table 3.12: Relative L^2 and H^1 error and experimental order of convergence of the coarse Trefftz method for sub-cases with and without walls.

N	$N_\mathcal{V}$	$N_\mathcal{T}$	N_Ω
16×32	1813	258 390	148 403
32×64	4293	314 979	179 240
64×128	11070	452 504	254 075

Table 3.13: Number of coarse cells $N = N_x \times N_y$, number of coarse grid nodes $N_\mathcal{V}$, number of triangulation elements $N_\mathcal{T}$, and number of triangulation points N_Ω.

mesh, and factor 80 for the coarsest one. The fine-scale triangulation is generated independently for every coarse partitioning. We report in Table 3.13 the resulting number of triangulation elements and points, as well as the number of coarse grid nodes.

The reference fine-scale model relies on the finite element interpolation of the $1m$ topographical data, which we will refer to as z_h. For the coarse Trefftz method, we perform the projection of the fine-scale topography on the coarse space. More precisely, let $z_h = \sum_{i\in\mathcal{N}} z_i \eta(\mathbf{x}_i)$ and $\mathbf{z} = (z_i, \ldots, z_\mathcal{N})^T$, for any symmetric positive definite $N_\Omega \times N_\Omega$ matrix $\mathcal{A}$, we denote by $\|\|_\mathcal{A}$ the norm induced by $\mathcal{A}$, that is $\|\mathbf{u}\|_\mathcal{A} = \left(\mathbf{u}^T \mathcal{A}\mathbf{u}\right)^{1/2}$. We set

$$\mathbf{z}_H = \underset{\boldsymbol{\zeta}\in\mathbb{R}^{N_\mathcal{V}}}{\arg\min} \|R_H^T \boldsymbol{\zeta} - \mathbf{z}\|_\mathcal{A}.$$

In other words, $\mathbf{z}_H = \left(R_H \mathcal{A} R_H^T\right)^{-1} R_H \mathbf{z}$. The projected fine-scale topography used in the coarse discretization is given by $R_H^T \mathbf{z}_H$. The numerical results presented below use the regularized H^1 projection, that is $\mathcal{A} = A + 0.01\overline{M}$, where A denotes the standard finite element stiffness matrix. Alternatively, the L^2 projection can be used. We have observed that methods based on both L^2 and regularized H^1 projections result in similar accuracy. However, for reasons that are not yet clear, the performance of Newton's method is significantly superior for the regularized H^1 projection.

The simulation covers the total period of $T_f = 900s$ with the target time step of $4s$. To solve the nonlinear system (3.34), we use Newton's method which is stopped once the l^∞ norm residual gets below 10^{-5}. If the iteration count exceeds 40, the time step is reduced as detailed in Section 3.6.

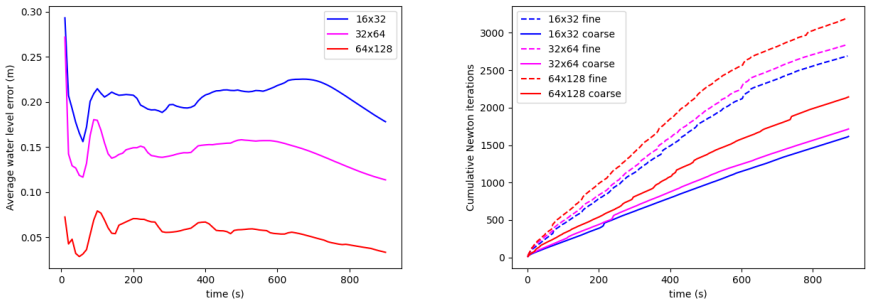


Figure 3.14: Left: Average water level error over time. Right: Cumulative number of Newton's iterations.

Figure 3.15 reports the reference water elevation resulting from the fine-scale model over the "flooded domain"; the flooded domain is defined as where the water depth exceeds $10cm$. The flooded domain resulting from the coarse model is depicted in green. The figure shows the results for the moderately refined coarse grid with $N = 32 \times 64$ coarse cells. We note that the flood extension is fairly well represented by the coarse model. The coarse flood extend is typically a couple of coarse cells further compared to the reference solution.

For all three meshes, the left sub-figure of Figure 3.14 shows the average L^1 error in water surface elevation over the flooded domain. To calculate this L^1 error, for a given mesh and for all $i \in \mathcal{N}$, we denote by $u_i^f(t)$ the continuous piece-wise linear interpolation in time of the nodal values of the reference fine-scale solution. Similarly, for any $i \in \mathcal{N}$, we denote by $u_i^c(t)$ the interpolation in time of the values $\left(R_H^T \mathbf{u}_H^n\right)_i$. Then, for a given $t > 0$ and a positive threshold ϵ the sets of coarse and fine flooded nodes is defined by $\mathcal{N}_{f,\epsilon}(t) = \{i \in \mathcal{N} \,|\, u_i^f(t) - z_i > \epsilon\}$ and $\mathcal{N}_{c,\epsilon}(t) = \{i \in \mathcal{N} \,|\, u_i^c(t) - z_i^c > \epsilon\}$ with $z_i^c = \left(R_H^T \mathbf{z}_H\right)_i$. Then, we define the average L^1 error over the flooded domain as

$$err(t) = \sum_{i \in \mathcal{N}_{f,\epsilon}(t) \cup \mathcal{N}_{c,\epsilon}(t)} m_i |u_i^f(t) - u_i^{ms}(t)| \Big/ \sum_{i \in \mathcal{N}_{f,\epsilon}(t) \cup \mathcal{N}_{c,\epsilon}(t)} m_i.$$

The left sub-figure of Figure 3.14 shows the error $err(t)$ over the simulation period for the coarse meshes with $N = 16 \times 32, 32 \times 64$, and 64×128 coarse cells. We observe that the error remains relatively steady over the simulation period and decreases with the refinement of the coarse mesh. Using the finest coarse partitioning, the error is about $5cm$ in average, which is largely sufficient for practical considerations.

The right sub-figure of Figure 3.14 reports the performance of Newton's method in terms of cumulative Newton iterations over the time interval; performance of the fine-scale method is reported for the sake of comparison. Only successfully time steps are reported. For all three meshes, the results are consistent, with a slight increase in the number of iterations as the mesh is refined. Compared to the fine-scale method, performance of Newton's method in the coarse discretization is significantly improved with the average number of successful Newton iterations per time step of $7, 7.5$ and 9 for $N = 16 \times 32, 32 \times 64$, and 64×128 respectively.

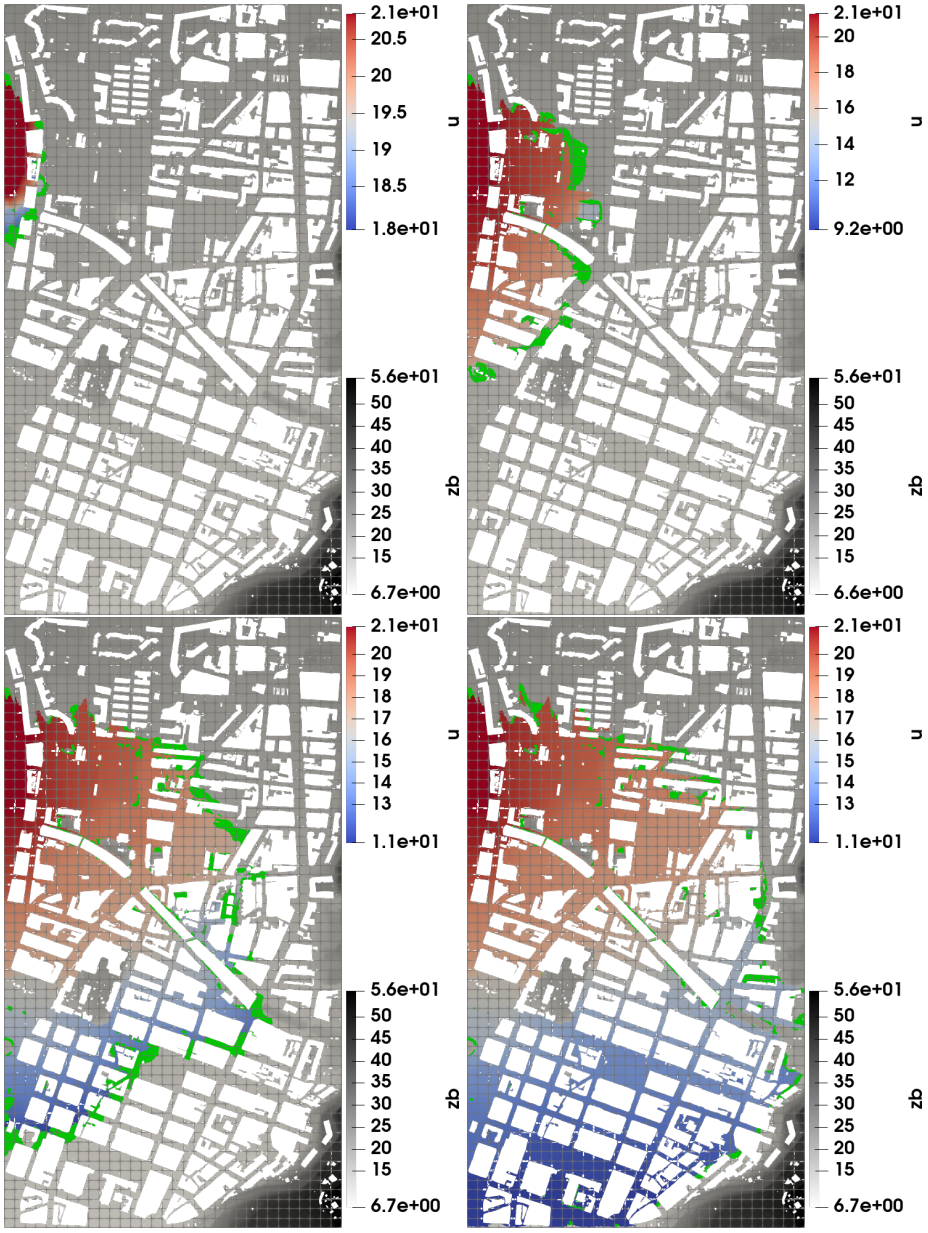


Figure 3.15: Reference water elevation over the flooded domain (in color) and flood extent resulting from the coarse Trefftz method (in green), for $N = 32 \times 64$ and $t = 10, 50, 300$ and 600 s. The underlying topography z_b is shown in black and white.

3.8 Summary and Future Work

This chapter has provided multiple efficient nonlinear domain decomposition methods to solve nonlinear equations on complex, perforated domains. Specifically, we have focused on a Diffusive Wave model on a domain with numerous perforations; these perforations represent buildings and walls/fences in urban areas. The coarse space form [13], designed for linear problems on perforated domains, proves to form efficient two-level methods for our difficult model problem.

From our numerical experiments, we draw the following conclusions. In terms of the number of GMRES iterations, as expected, the One-level RASPEN method is not a robust method as the number of subdomains N grows large. For the linearized systems resulting at each iteration of the Two-step and Newton methods, the Trefftz space performs well as a component of the two-level RAS preconditioner and results in a reasonable number of inner GMRES iterations per outer iteration. However, these iterations slightly increase as N is increased. This effect could be mitigated by updating the basis functions in the Trefftz space as the linearized system is updated. The Two-level RASPEN method further reduces the total number of GMRES iterations compared to the Two-step method, particularly as N grows. However, in terms of total GMRES iterations $N_k \cdot N_{\text{GM}}$, due to the large number of N_k for Newton's method, Newton's method results overall in a much larger total number of GMRES iterations.

In terms of total outer iterations, Newton's method is much more sensitive to the initial guess than the other proposed methods in our experiment, with iteration counts varying by up to 50%. Additionally, Newton's method generally results in extremely large iteration counts when compared to the Two-step and RASPEN variants. The addition of an NRAS solve in the Two-step method greatly accelerates convergence in terms of outer iterations when compared to Newton's method. Specifically, for the stationary examples, the Two-level RASPEN method provides a significant acceleration in terms of outer iterations when compared to the other proposed methods. We do not see the same effect in the time-dependent example, likely due to the time stepping process and choice of the time increment. The Anderson acceleration of a coarse Two-step method is also an interesting alternative to Newton's method, reducing the total iteration count, particularly for the stationary examples. However, for the time-dependent example, we do see a dependence on N in terms of outer iterations for this method. Each iteration of Anderson's method involves a small least-squares problem and is cheaper than a Newton iteration. However, it does not appear to accelerate convergence when compared to methods such as the Two-level RASPEN method.

In terms of implementation, both Newton's method and the Two-step method involve a sparse linear system at each iteration which, in principle, could be solved via a direct solve (provided the system is not too large). Additionally, the Two-step method is quite straightforward to implement when compared to the Two-level RASPEN method, with the only addition from Newton's method being a nonlinear RAS computation at each iteration. The Two-level RASPEN method involves a nonlinear RAS computation, a coarse nonlinear problem, and a global linear system which generally should be solved via a Krylov method. However, once implemented, the Two-level RASPEN method forms an extremely robust and efficient algorithm. Additionally, Anderson acceleration benefits from readily available built-in functions provided by multiple popular programming languages.

Additionally, the use of the Trefftz space for the coarse approximation of the

numerical solution allows us to use a small fraction of the fine-scale degrees of freedom to approximate the solution. For this method, we project the topographical data z_b onto the coarse space V_H and perform mass lumping of the "coarse" mass matrix. Finally, we have found that local time step reduction is an efficient method to eliminate the need for a global time step reduction for all subdomains. Future work involves improved load-balancing of the complexity of the subdomains and parallel implementation to provide detailed runtimes.

Chapter 4

Scientific Machine Learning for Nonlinear Elliptic PDEs with Rough Coefficients

This chapter is mainly taken from a collaboration with Konstantin Brenner and Larissa Miguez da Silva, submitted to the CEMRACS 2023 Proceedings. The corresponding preprint is available on arXiv [16].

Contents

4.1 Introduction

In this Chapter, we focus on heterogeneous model problems of the form

$$\begin{cases} au + \operatorname{div}\left(K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(u, \nabla u)\right) &= 0 \quad \text{in} \quad \Omega, \\ u &= u_D \quad \text{on} \quad \partial\Omega, \end{cases} \tag{4.1}$$

where a is a positive real coefficient and $K_\epsilon(\mathbf{x})$, representing the heterogeneity of the problem, is assumed to satisfy $K_\epsilon(\mathbf{x}) \geq k_{min} > 0$ for all $\mathbf{x} \in \Omega$. Here, Ω may be either an open connected polygonal domain in $\mathbb{R}^2$ or a real interval. In this work, we focus on model problems of the form For the "flux functions" $\mathbf{F}_\epsilon(u, \nabla u)$, we consider either the model

$$\mathbf{F}_\epsilon(u, \nabla u) = -\nabla u^p, \quad p > 1,$$

leading to a porous medium problem, or the $p-$Laplace model

$$\mathbf{F}_\epsilon(u, \nabla u) = -|\nabla u|^p \nabla u, \quad p > 0.$$

The well-posedness of the problem (4.1) is known (see e.g. [103] and [75]).

4.1.1 MsFEM for Linear Problems

We proceed with a description of MsFEM for linear heterogeneous problems before moving to the focus of the chapter, nonlinear problems. It is known that standard numerical methods such as Finite Element Methods often have difficulty approximating multiscale behavior due to the computational cost of resolving all relevant scales. Therefore, multiscale numerical methods have emerged as attractive options for dealing with such problems characterized by significant variations across multiple scales; here, we consider specifically MsFEM [61]. By representing fine-scale features using computationally efficient coarse-scale models, MsFEM can lead to a reduction in both computational time and resources. While we focus on a nonlinear substructured formulation to be described later, we briefly describe the classical MsFEM here as a linear comparison.

With this, consider a linear heterogeneous problem of the form

$$-\text{div}(K_\epsilon \nabla u) = f, \tag{4.2}$$

where K_ϵ is a heterogeneous coefficient matrix. This results in a variational formulation as follows: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla v \cdot K_\epsilon \nabla u \, d\mathbf{x} = \int_\Omega f v \, d\mathbf{x}. \tag{4.3}$$

We remark as well that there are many variants of the MsFEM, but we present the original formulation from [61]. The original space is defined as follows. Consider some (coarse) mesh $\mathcal{T}_H$; here, we take the coarse mesh to be the nonoverlapping rectangular partitioning of Ω denoted by $(\Omega_j)_{j=1}^N$. We will refer to $(\Omega_j)_{j=1}^N$ as the coarse mesh over Ω.

Here, we denote V_H as the space of piecewise affine functions on the coarse mesh. With this, let $(s_i)_{i=1}^{N_H}$ denote the vertices of the coarse mesh, where N_H is the number of coarse nodes, and let $(g_i)_{i=1}^{N_H}$ be the associated nodal basis for V_H made of "hat" functions, satisfying

$$g_i(s_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$$

for all $1 \leq i, j \leq N_H$. Then, the multiscale basis functions ϕ_i are given as follows: for all $1 \leq i \leq N_H$,

$$\begin{cases} -\text{div}\left(K_\epsilon \nabla \phi_i\right) &= 0 \quad \text{in} \quad \Omega_j, \\ \phi_i &= g_i \quad \text{on} \quad \partial\Omega_j, \end{cases}$$

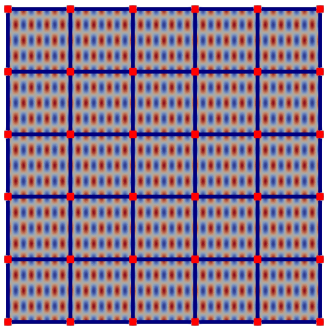


Figure 4.1: Heterogeneous domain $K_\epsilon(\mathbf{x})$ with $N = 5 \times 5$ coarse cells. Nonoverlapping skeleton Γ is denoted by thick dark blue lines.

for all Ω_j. The multiscale space is given by

$$V_{ms} = \text{span}\{\phi_i \mid 1 \leq i \leq N_H\},$$

where $V_{ms} \subset H_0^1(\Omega)$. The Galerkin projection is as follows: Find $u_H \in V_{ms}$ such that

$$\int_\Omega \nabla v_H \cdot K_\epsilon \nabla u_H \, d\mathbf{x} = \int_\Omega f \, v_H \, d\mathbf{x} \quad \text{for all} \quad v_H \in V_{ms}.$$

4.1.2 Nonlinear Substructured and Approximate Substructured Problem

As MsFEM strategies do not naturally extend to nonlinear problems [32, 44], we focus on the computation of local Dirichlet-to-Neumann (DtN) operators. Specifically, with coarse and fine partitionings of the domain, we can compute local DtN operators on each subdomain. These local DtN operators provide a relationship between Dirichlet and Neumann boundary data; these local operators lead to a nonlinear substructuring method such that the fine-scale discrete nonlinear problem can be written in terms of the local DtN maps.

Consider a finite nonoverlapping rectangular partitioning of Ω denoted by $(\Omega_j)_{j=1}^N$. We denote by Γ the coarse skeleton, that is $\Gamma = \bigcup_{j=1}^N \partial\Omega_j$. With this, each edge of the skeleton Γ is denoted by $\Gamma_{ij} = \partial\Omega_j \cap \partial\Omega_i$, while the set of coarse grid nodes is defined by $(\mathfrak{s}_\alpha)_{\alpha=1}^{\mathcal{N}_\Gamma} = \bigcup_{i,j=1}^N \partial\Gamma_{ij}$. The set of coarse grid nodes can be considered as the set of corners/vertices of the subdomains. An example of the coarse partitioning and corresponding coarse grid nodes is shown in Figure 4.1.

The Dirichlet-to-Neumann (DtN) map provides a mathematical relationship between Dirichlet and Neumann boundary data, allowing us to compute one set of boundary data from the other, which can be very useful in solving boundary value problems. Given a nonoverlapping partitioning of domain, these DtN maps can be defined and computed locally on each subdomain. Let us consider a local Dirichlet problem

$$\begin{cases} a u_j + \text{div}\,(K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(u_j, \nabla u_j)) = 0 & \text{in} \quad \Omega_j, \\ u_j = g_j & \text{on} \quad \partial\Omega_j. \end{cases} \tag{4.4}$$

For a given g_j we formally denote by $\mathrm{DtN}_j(g_j)$ the Neumann traces of u_j on $\partial\Omega_j$, that is

$$\mathrm{DtN}_j(g_j) = K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(u_j, \nabla u_j)|_{\partial\Omega_j} \cdot \mathbf{n}_j \qquad \text{on} \quad \partial\Omega_j,$$

where $\mathbf{n}_j$ denotes the unit outward normal on $\partial\Omega_j$.

Our main idea with the nonlinear substructuring method is to replace the original problem (4.1) by the one which only involves the unknown Dirichlet data g defined over Γ. This new unknown function is determined by imposing a matching flux condition along the edges of the skeleton Γ.

For a regular enough function g defined on Γ, we formally introduce the local jump operator, returning the mismatch in fluxes on a given coarse edge

$$[\![]\!]_{ij} : g \mapsto \mathrm{DtN}_j(g|_{\partial\Omega_j})|_{\Gamma_{ij}} + \mathrm{DtN}_i(g|_{\partial\Omega_i})|_{\Gamma_{ij}}.$$

and the global jump operator $[\![]\!]$ is given by $[\![g]\!]|_{\Gamma_{ij}} = [\![g]\!]_{ij}$. Enforcing zero flux jump at every coarse edge Γ_{ij} leads to a global substructured problem that can be written in a weak form as follows: find g satisfying $g|_{\partial\Omega} = u_D$ such that for all regular enough w vanishing on $\partial\Omega$ it holds that

$$\int_\Gamma [\![g]\!] w \, d\gamma(\mathbf{x}) = 0. \tag{4.5}$$

Let us assume for the moment that we are in the possession of a procedure allowing to compute evaluate the DtN operators. Then, the problem (4.5) can be discretized by taking the test and trial function from a finite dimensional space. In particular, we consider the space $V_H(\Gamma)$ made of the continuous functions on Γ that are affine on every coarse edge of Γ, that is

$$V_H(\Gamma) = \{v \in C^0(\Gamma) \quad \text{s.t.} \quad v|_{\Gamma_{ij}} \in \mathbb{P}_1(\Gamma_{ij}) \quad \text{for all} \quad \Gamma_{ij}\}. \tag{4.6}$$

Let $V_{H,0}(\Gamma) = \{v \in V_H(\Gamma) \quad \text{s.t.} \quad v|_{\partial\Omega} = 0\}$ and let $u_{D,H} \in V_H(\Gamma)$ denote some approximation of u_D, obtained for example by interpolation. To form the approximate substructured problem in $V_H(\Gamma)$, one then replaces the problem (4.5) by the following: find g_H satisfying $g_H|_{\partial\Omega} = u_{H,D}$ and

$$\int_\Gamma [\![g_H]\!] w_H \, d\gamma(\mathbf{x}) = 0 \qquad \text{for all} \quad w_H \in V_{H,0}(\Gamma). \tag{4.7}$$

4.1.3 Scientific Machine Learning

With the renewed interest in Machine Learning (ML) within the scientific computing community, numerous techniques have been proposed to apply ML techniques to the solution of PDEs. In [80], the authors employ DeepONet neural networks to learn nonlinear operators. Additionally, the authors of [9] introduce an approach that combines Finite Element interpolation techniques with Neural Networks (NN) to solve a wide range of direct and inverse problems.

Additionally, there exists a large class of NNs which are referred to as Physics-Informed Neural Networks (PINNs) [92]. PINNs are a class of neural networks that integrate physical laws, typically represented by PDEs, directly into the training process by minimizing a functional which represents the residual of the PDE and its initial and boundary conditions. This ensures that the NN's predictions take into consideration both the physical laws and the traditional data-driven loss components. Many pieces of work have been published on PINNs, including [83, 23, 84].

However, problems with complex domains have led to other methodologies based on DD methods and PINNs, including Extended PINNs [65], Conservative PINNs [66], Variational PINNs [70] and Finite Basis Physics-Informed Neural Networks [85]. Additionally, [62] proposed Differential Machine Learning (DML), originally with a financial application. In DML, supervised learning is "extended" such that ML models are trained not only with values/inputs but also on corresponding derivatives.

In this chapter, we aim to use machine learning techniques to learn the previously discussed nonlinear DtN maps. The neural network is trained to replicate the action of the local nonlinear DtN maps on some coarse subset of the trace space. Once the training is completed, the surrogate DtN operators will be used to solve a nonlinear substructuring method via Newton's method. For training the nonlinear maps, we use an approach similar to DML; that is, we incorporate gradient information into the loss function. Additionally, we impose a monotonicity property into the loss function which is based on the known monotonicity of the DtN maps.

4.1.4 Chapter Outline

The remainder of this chapter is laid out as follows. In Section 4.2, we introduce the finite element discretization, discrete substructured formulation, and discrete DtN maps. We also provide a description of the machine learning process which is used to determine the learned DtN operators and assemble the learned substructured formulation. Section 4.3 contains numerical experiments, showing both the accuracy of the learned DtN maps and the resulting learned approximate substructured formulation. Section 4.4 concludes with a summary.

4.2 Discrete Finite Element Substructuring and DtN Maps

4.2.1 Finite Element Discretization

Let T_h denote the triangulation of Ω which is conforming with respect to partitioning $(\Omega_j)_{j=1}^N$. Let $\{s_k\}_{k=1}^{N_{\overline{\Omega}}}$ be the set of vertices (or points) of T_h. We introduce the finite element space defined by

$$V_h = \{v \mid v \in C^0(\overline{\Omega}), \quad \text{s.t.} \quad v|_t \in \mathbb{P}_1, \quad \text{for all} \quad t \in T_h\},$$

and we denote the corresponding nodal "hat" basis functions $(\eta^k)_{k=1}^{N_{\overline{\Omega}}}$ for each point s_k. Let $V_{h,0} = V_h \cap H_0^1$, and let $u_{h,D} \in V_h$ be some approximation of the boundary data u_D. The Galerkin finite element formulation is as follows:

$$\begin{aligned} &\text{Find } u_h \in V_h \text{ satisfying } u_h|_\Omega = u_{h,D} \text{ such that} \\ &\int_\Omega a u_h v_h - (K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(u_h, \nabla u_h)) \cdot \nabla v_h \, d\mathbf{x} = 0 \qquad \text{for all} \quad v_h \in V_{h,0}. \end{aligned} \tag{4.8}$$

Now, consider the following linear reconstruction operator π_h acting from $\mathbb{R}^{N_\Omega}$ to V_h, such that

$$\pi_h(v)(\mathbf{x}) = \sum_{k=1}^{N_\Omega} v^k \eta^k(\mathbf{x}).$$

We note that π_h is bijective, with $\left(\pi^{-1}(\mathbf{v}_h)\right)^k = v_h(s_k)$ for any triangulation point s_k.

We denote by F the "Neumann residual" function corresponding to (4.8); more precisely, F acts from $\mathbb{R}^{N_{\overline{\Omega}}}$ to itself and has its lth component defined by

$$(F(u))^l = \int_\Omega \pi_h(u)\,\eta^l - K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(\pi_h(u), \nabla\pi_h(u)) \cdot \nabla\eta^l \, d\mathbf{x}, \tag{4.9}$$

for $l = 1, \ldots, N_{\overline{\Omega}}$.

Before moving forward, let us introduce some notation related to the decomposition of the mesh and finite element degrees of freedom induced by the partitioning $(\Omega_j)_{j=1}^N$ of Ω. Let $\mathcal{I} = \{1, \ldots, N_{\overline{\Omega}}\}$ denote the index set of the triangulation points. For any arbitrary $\omega \in \overline{\Omega}$ we introduce a subset of indices

$$\mathcal{I}_\omega = \left\{i_\omega^1, \ldots i_\omega^{N_\omega}\right\} \subset \mathcal{I}$$

such that $i \in \mathcal{I}_\omega$ if and only if the point s_i belongs to ω. We note that the number of elements in $\mathcal{I}_\omega$ is denoted by N_ω.

The standard N_ω by $N_{\overline{\Omega}}$ boolean restriction matrix R_ω "from $\overline{\Omega}$ to ω" is defined by

$$(R_\omega)_{kl} = \begin{cases} 1 & \text{if} \quad l = i_\omega^k, \\ 0 & \text{else}, \end{cases}$$

for any $k = 1, \ldots, N_\omega$ and $l = 1, \ldots, N_{\overline{\Omega}}$. Further, for any $A, B \subset \overline{\Omega}$, we denote

$$R_B^A = R_A (R_B)^T.$$

If $A \subset B$, the matrix R_B^A can be interpreted as a restriction matrix "from B to A". Conversely, if $B \subset A$, the same R_B^A may be interpreted as an extension matrix "from B to A". Let us mention a couple of useful properties of this restriction/extension linear operator. First, we note that for any C such that $A, B \subset C \subset \overline{\Omega}$ we have

$$R_C^A R_B^C = R_B^A = R_{A\cap B}^A R_B^{A\cap B}.$$

With this, the discrete problem (4.8) can be expressed as follows:

$$\text{Find } u \in \mathbb{R}^{N_\Omega} \text{ such that } R_{\partial\Omega} u = R_{\partial\Omega}\pi_h^{-1}\left(u_{h,D}\right) \text{ and satisfying } R_\Omega F(u) = 0. \tag{4.10}$$

We note that the integral in (4.8) can be split into a sum of integrals over Ω_j, meaning that the finite element residual can be assembled by gluing together the local contributions. We detail below this local assembly procedure because of its use in the substructured formulation.

For a subdomain Ω_j, let $\{\eta_j^l\}_{l=1}^{N_{\overline{\Omega}_j}}$ denote the set of local finite element basis functions such that $\eta_j^l(\mathbf{x}) = \eta^{i_{\overline{\Omega}_j}^l}(\mathbf{x})|_{\Omega_j}$ for all $l = 1, \ldots, N_{\overline{\Omega}_j}$. The local function reconstruction operator $\pi_{h,j}$ is defined by $\pi_{h_j}(v_j)(\mathbf{x}) = \sum_{l=1}^{N_{\overline{\Omega}_j}} v^l \eta_j^l(\mathbf{x})$. Similar to (4.9) we define the local residual functions F_j. That is for any $u_j \in \mathbb{R}^{N_{\overline{\Omega}_j}}$ and $l = 1, \ldots, N_{\overline{\Omega}_j}$, we set

$$(F_j(u_j))^l = \int_{\Omega_j} \pi_{h,j}(u)\,\eta_j^l - K_\epsilon(\mathbf{x})\mathbf{F}_\epsilon(\pi_{h,j}(u_j), \nabla\pi_{h,j}(u_j)) \cdot \nabla\eta_j^l \, d\mathbf{x}. \tag{4.11}$$

with $\eta_j^l = \eta^{i^l_{\overline{\Omega}_j}}$. For any $u \in \mathbb{R}^{N_{\overline{\Omega}}}$, the global residual $F(u)$ is obtained from the local components by the following expression

$$F(u) = \sum_{j=1}^{N} R_{\overline{\Omega}_j}^T F_j(R_{\overline{\Omega}_j} u). \tag{4.12}$$

.

4.2.2 Discrete DtN maps

In this section, we introduce the finite element version of the Dirichlet-to-Neumann operator, associated with our model PDE and a subdomain Ω_j. This discrete operator, denoted by $\mathrm{DtN}_{h,j}$, is defined as the nonlinear mapping from the set of degrees of freedom $\mathbb{R}^{N_{\partial\Omega_j}}$ associated with the boundary of Ω_j to itself. We also provide the formula for the (Fréchet) derivative of $\mathrm{DtN}_{h,j}$. The construction of $\mathrm{DtN}_{h,j}$ relies on the definition (4.11) of the local "Neumann" residual F_j.

Let us split the set of indices $\mathcal{I}_{\overline{\Omega}_j}$ into two nonoverlapping subsets $\mathcal{I}_{\Omega_j}$ and $\mathcal{I}_{\partial\Omega_j}$ associated with internal degrees of freedom located in Ω_j and boundary degrees of freedom on $\partial\Omega_j$. This gives the representation of the vector $u_j \in \mathbb{R}^{N_{\overline{\Omega}_j}}$ associated to the subdomain $\overline{\Omega}_j$ in the form $u_j = R_{\Omega_j}^{\overline{\Omega}_j} u_{\Omega_j} + R_{\partial\Omega_j}^{\overline{\Omega}_j} u_{\partial\Omega_j}$ with some $u_{\Omega_j} \in \mathbb{R}^{N_{\Omega_j}}$ and $u_{\partial\Omega_j} \in \mathbb{R}^{N_{\partial\Omega_j}}$. In turn, we express the local Neumann residual as following

$$F_j(u_j) = R_{\Omega_j}^{\overline{\Omega}_j} F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) + R_{\partial\Omega_j}^{\overline{\Omega}_j} F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) \tag{4.13}$$

where

$$F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) = R_{\overline{\Omega}_j}^{\Omega_j} F_j\left(R_{\Omega_j}^{\overline{\Omega}_j} u_{\Omega_j} + R_{\partial\Omega_j}^{\overline{\Omega}_j} u_{\partial\Omega_j}\right)$$

and

$$F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) = R_{\overline{\Omega}_j}^{\partial\Omega_j} F_j\left(R_{\Omega_j}^{\overline{\Omega}_j} u_{\Omega_j} + R_{\partial\Omega_j}^{\overline{\Omega}_j} u_{\partial\Omega_j}\right).$$

The discrete counterpart of the local Dirichlet problem (4.4) can be expressed as:

$$\text{Find } u_{\Omega_j} \text{ such that } F_{\Omega_j}(u_{\Omega_j}, g_{\partial\Omega_j}) = 0 \tag{4.14}$$

for a given $g_{\partial\Omega_j} \in \mathbb{R}^{N_{\partial\Omega_j}}$. Assuming that (4.14) admits a unique solution for any $g_{\partial\Omega_j}$, we define the solution operator G_{Ω_j} from $\mathbb{R}^{N_{\partial\Omega_j}}$ to $\mathbb{R}^{N_{\Omega_j}}$ such that

$$F_{\Omega_j}\left(G_{\Omega_j}(g_{\partial\Omega_j}), g_{\partial\Omega_j}\right) = 0 \tag{4.15}$$

for all $g_{\partial\Omega_j}$. With that, the discrete DtN operator is defined by

$$\mathrm{DtN}_{h,j}(g_{\partial\Omega_j}) = F_{\partial\Omega_j}\left(G_{\Omega_j}(g_{\partial\Omega_j}), g_{\partial\Omega_j}\right). \tag{4.16}$$

Under the assumptions of Implicit Function Theorem (see e.g. Proposition 5.2.4. of [90]) the Fréchet derivative of $\mathrm{DtN}_{h,j}$ can be expressed as

$$\mathrm{DtN}'_{h,j}(u_{\partial\Omega_j}) = \partial_{\Omega_j} F_{\partial\Omega_j}(G_{\Omega_j}(u_{\partial\Omega_j}), u_{\partial\Omega_j}) G'_{\Omega_j}(u_{\partial\Omega_j}) + \partial_{\partial\Omega_j} F_{\partial\Omega_j}(G_{\Omega_j}(u_{\partial\Omega_j}), u_{\partial\Omega_j}). \tag{4.17}$$

Recalling that $u_{\Omega_j} = G_{\Omega_j}(u_{\partial\Omega_j})$, we have

$$\mathrm{DtN}'_{h,j}(u_{\partial\Omega_j}) = \partial_{\Omega_j} F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) G'_{\Omega_j}(u_{\partial\Omega_j}) + \partial_{\partial\Omega_j} F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}), \tag{4.18}$$

where, thanks to the identity

$$F_{\Omega_j}(G_{\Omega_j}(u_{\partial\Omega_j}), u_{\partial\Omega_j}) = 0, \tag{4.19}$$

we have

$$G'_{\Omega_j}(u_{\partial\Omega_j}) = -\left(\partial_{\Omega_j} F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j})\right)^{-1} \partial_{\partial\Omega_j}\left(F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j})\right). \tag{4.20}$$

Plugging (4.20) into (4.17) results in

$$\begin{aligned}\mathrm{DtN}'_{h,j}(u_{\partial\Omega_j}) = &-\partial_{\Omega_j} F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) \left(\partial_{\Omega_j} F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j})\right)^{-1} \partial_{\partial\Omega_j} F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}) \\ &+ \partial_{\partial\Omega_j} F_{\partial\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j}),\end{aligned} \tag{4.21}$$

We remark that computing u_{Ω_j} requires solving the local nonlinear problem (4.14) with $u_{\partial\Omega_j}$ as data. This is typically done by some fixed point method. Once the approximate value of u_{Ω_j} is obtained, computing $\mathrm{DtN}'_{h,j}(u_{\partial\Omega_j})$ requires solving the linear system with right-hand side given by $F_{\Omega_j}(u_{\Omega_j}, u_{\partial\Omega_j})$. If u_{Ω_j} is obtained by exact Newton's method, the former can be done at marginal computational cost.

4.2.3 Discrete Substructured Problems

With $\mathrm{DtN}_{h,j}$ introduced above, let us provide the substructured formulation of the discrete problem (4.10). Let $g \in \mathbb{R}^{N_\Gamma}$ be a vector representing the unknown solution values at the skeleton Γ. Denoting $g_{\partial\Omega_j} = R_\Gamma^{\partial\Omega_j} g$ and $u_{\Omega_j} = G_{\Omega_j}\left(g_{\partial\Omega_j}\right)$, we express the unknown discrete solution as

$$u = R_\Gamma^T g + \sum_{j=1}^{N} R_{\Omega_j}^T u_{\Omega_j}. \tag{4.22}$$

Observing that $R_{\overline{\Omega}_j} u = R_{\partial\Omega_j}^{\overline{\Omega}_j} g_{\partial\Omega_j} + R_{\Omega_j}^{\overline{\Omega}_j} u_{\Omega_j}$, in view of (4.13), we deduce that

$$F_j\left(R_{\overline{\Omega}_j} u\right) = R_{\Omega_j}^{\overline{\Omega}_j} F_{\Omega_j}(u_{\Omega_j}, g_{\partial\Omega_j}) + R_{\partial\Omega_j}^{\overline{\Omega}_j} F_{\partial\Omega_j}(u_{\Omega_j}, g_{\partial\Omega_j}). \tag{4.23}$$

The first term in the right-hand side of (4.23) is zero by definition of u_{Ω_j}, which yields

$$F(u) = \sum_{j=1}^{N} R_{\partial\Omega_j}^T \mathrm{DtN}_{h,j}\left(g_{\partial\Omega_j}\right) = \sum_{j=1}^{N} R_{\partial\Omega_j}^T \mathrm{DtN}_{h,j}\left(R_\Gamma^{\partial\Omega_j} g\right),$$

in view of (4.12) and (4.16). Let us denote

$$F_\Gamma(g) = \sum_{j=1}^{N} R_{\partial\Omega_j}^\Gamma \mathrm{DtN}_{h,j}\left(R_\Gamma^{\partial\Omega_j} g\right). \tag{4.24}$$

Substituting $F(u) = R_\Gamma^T F_\Gamma(g)$ in (4.10) leads to the equation $R_\Omega R_\Gamma^T F_\Gamma(g) = 0$. Since the latter is trivially satisfied for all degrees of freedom not lying on Γ, the system has to be reduced to $\Gamma \cap \Omega$, leading to

$$0 = R_\Omega^{\Gamma\cap\Omega} R_\Omega R_\Gamma^T F_\Gamma(g) = R_\Gamma^{\Gamma\cap\Omega} F_\Gamma(g).$$

We have obtained the following substructured problem: find $g \in \mathbb{R}^{N_\Gamma}$ such that $R_\Gamma^{\partial\Omega} g = R_{\partial\Omega}\pi_h^{-1}(u_{h,D})$ and

$$R_\Omega^{\Gamma\cap\Omega} F_\Gamma(g) = 0. \tag{4.25}$$

4.2.4 Discrete Approximate Substructured Problems and Coarse DtN Maps

We now proceed to the discrete version of the approximate substructured problem (4.7). Similar to the definitions introduced in Section 4.2.1, for any $\omega \subset \Gamma$, we denote by $\mathscr{N}_\omega$ the number of *coarse* nodes belonging to ω, and by $\mathscr{R}_\Gamma^\omega$ the restriction matrix from $\mathbb{R}^{\mathscr{N}_\Gamma}$ to $\mathbb{R}^{\mathscr{N}_\omega}$. As before, the transpose of $\mathscr{R}_\Gamma^\omega$ is denoted by $\mathscr{R}_\omega^\Gamma$.

Let $(\phi^\alpha)_{\alpha=1}^{\mathscr{N}_\Gamma}$ be a nodal basis of $V_H(\Gamma)$ made of "hat" functions associated to the set coarse grid nodes $\{\mathfrak{s}_\alpha\}_{\alpha=1}^{\mathscr{N}_\Gamma}$ (see Figure 4.1 for visual). More precisely, the basis function $\phi^\alpha \in V_H(\Gamma)$ is assumed to satisfy

$$\phi^\alpha(\mathfrak{s}_i) = \begin{cases} 1, & \alpha = i, \\ 0, & \alpha \neq i, \end{cases}$$

for all coarse grid nodes $\mathfrak{s}_i$. Given this nodal basis in $V_H(\Gamma)$ we define the function reconstruction operator π_H from $\mathbb{R}^{\mathscr{N}_\Gamma}$ to $V_H(\Gamma)$ by $\pi_H(v)(\mathbf{x}) = \sum_{\alpha=1}^{\mathscr{N}_\Gamma} v^\alpha \phi^\alpha(\mathbf{x})$. We further denote by $\boldsymbol{\phi}_H$ the N_Γ by $\mathscr{N}_\Gamma$ matrix whose columns contain the values of the basis functions ϕ^α at the skeletal degrees of freedom; more specifically, the lth component of a column index α of $\boldsymbol{\phi}_H$ is given by

$$(\boldsymbol{\phi}_H)^{l,\alpha} = \phi^\alpha(s_{i_\Gamma^l}).$$

for $\alpha = 1, \ldots, \mathscr{N}_\Gamma$ and each $i = 1, \ldots, N_\Gamma$ such that $i_\Gamma^l \in \mathcal{I}_\Gamma$.

The local version of $\boldsymbol{\phi}_H$ is defined by

$$\boldsymbol{\phi}_{Hj} = R_\Gamma^{\partial\Omega_j} \boldsymbol{\phi}_H \mathscr{R}_{\partial\Omega_j}^\Gamma, \tag{4.26}$$

where $\boldsymbol{\phi}_{Hj}$ is composed of the $\mathscr{N}_{\partial\Omega_j}$ local basis vectors of size $N_{\partial\Omega_j}$ corresponding to the local coarse grid nodes on $\partial\Omega_j$. We note that $\boldsymbol{\phi}_{Hj}$ satisfies

$$R_\Gamma^{\partial\Omega_j} \boldsymbol{\phi}_H = \boldsymbol{\phi}_{Hj} \mathscr{R}_\Gamma^{\partial\Omega_j}. \tag{4.27}$$

In order to derive the coarse approximation of the problem (4.25), we first express the latter in the following variational form:

$$\begin{aligned} &\text{Find } g \text{ such that } R_\Gamma^{\partial\Omega} g = R_{\partial\Omega}\pi_h^{-1}(u_{h,D}) \text{ and satisfying} \\ &v^T F_\Gamma(g) = 0 \qquad \text{for all} \quad v \in \operatorname{Im}\left(R_{\Gamma\cap\Omega}^\Gamma\right). \end{aligned} \tag{4.28}$$

The Galerkin approximation of (4.28) using the coarse space $V_H(\Gamma)$ is obtained by setting $g \approx \boldsymbol{\phi}_H g_H$ with some appropriate boundary conditions and taking the test elements v in $\mathrm{Im}\left(R^{\Gamma}_{\Gamma\cap\Omega}\right) \cap \mathrm{Im}\left(\boldsymbol{\phi}_{\mathrm{H}}\right) = \mathrm{Im}\left(\boldsymbol{\phi}_{\mathrm{H}}\right) \cap \mathrm{Im}\left(\mathscr{R}^{\Gamma}_{\Gamma\cap\Omega}\right)$. The latter amount to setting $v = \boldsymbol{\phi}_H \mathscr{R}^{\Gamma}_{\Gamma\cap\Omega}\mathbf{v}_H$ with $v_H \in \mathbb{R}^{\mathscr{N}_{\Gamma\cap\Omega}}$. This leads to the system

$$\mathscr{R}^{\Gamma\cap\Omega}_{\Gamma}\boldsymbol{\phi}_H^T F_{\Gamma}\left(\boldsymbol{\phi}_H g_H\right) = 0.$$

In view of (4.24) and (4.27), we have

$$\sum_{j=1}^{N} \boldsymbol{\phi}_H^T R^{\Gamma}_{\partial\Omega_j} \mathrm{DtN}_{h,j}\left(R^{\partial\Omega_j}_{\Gamma}\boldsymbol{\phi}_H^T g_H\right) = \sum_{j=1}^{N} \mathscr{R}^{\Gamma}_{\partial\Omega_j}\boldsymbol{\phi}_{Hj}^T \mathrm{DtN}_{h,j}\left(\boldsymbol{\phi}_{Hj}\mathscr{R}^{\partial\Omega_j}_{\Gamma} g_H\right).$$

With this, the definition of the coarse DtN operator acting from $\mathbb{R}^{\mathscr{N}_{\partial\Omega_j}}$ to itself given by

$$\mathrm{DtN}_{H,j} = \boldsymbol{\phi}_{Hj}^T \circ \mathrm{DtN}_{h,j} \circ \boldsymbol{\phi}_{Hj}, \tag{4.29}$$

such that we denote

$$F_H(g_H) = \sum_{j=1}^{N} \mathscr{R}^{\Gamma}_{\partial\Omega_j}\mathrm{DtN}_{H,j}\left(\mathscr{R}^{\partial\Omega_j}_{\Gamma} g_H\right). \tag{4.30}$$

Let $u_{H,D} \in V_H(\Gamma)$ be some approximation of u_D, the coarse Galerkin approximation of (4.10), or equivalently, the finite element approximation of (4.7) reads as follows:

$$\text{Find } g_H \in \mathbb{R}^{\mathscr{N}_{\Gamma}} \text{ such that } \mathscr{R}^{\partial\Omega}_{\Gamma} g_H = \mathscr{R}^{\partial\Omega}_{\Gamma}\pi_H^{-1}(u_{H,D}) \text{ and } \mathscr{R}^{\Gamma\cap\Omega}_{\Gamma} F_H(g_H) = 0. \tag{4.31}$$

We remark that even though it is considered a coarse approximation, the computation of the discrete DtN_j relies on fine-scale information and fine-scale input. That is, it still takes in as input a vector of size N_Γ which lives on the fine degrees of freedom along the skeleton Γ. This computation can still become quite expensive and the cost is not to be negated, as DtN_j must be computed at each Newton iteration used to solve the discrete substructured problem.

4.2.5 Learning Dirichlet-to-Neumann Maps

In Section 4.2, we discussed the computation of the DtN map and how by nature, the substructured problem results in the DtN map being computed at each Newton iteration. As this computation can get very expensive, we explore a Scientific Machine Learning (SciML) application to our problem. Specifically, we introduce learned local DtN operators which we denote as $\widetilde{\mathrm{DtN}}_{H,j}$. Once the model is generated, we can call the model at each Newton iteration without the need for a separate computation on each subdomain. We remark that if the heterogeneities are periodic in each subdomain, we can use the same NN model.

Let $\widetilde{\mathrm{DtN}}_{H,j}$ denote some approximation of $\mathrm{DtN}_{H,j}$ which we will refer to as a surrogate model; this surrogate model will be introduced in more details below. By replacing the original operator $\mathrm{DtN}_{H,j}$ by $\widetilde{\mathrm{DtN}}_{H,j}$ in (4.30), we define

$$F_{H,learn}(g_H) = \sum_{j=1}^{N} \mathscr{R}^{\Gamma}_{\partial\Omega_j}\widetilde{\mathrm{DtN}}_{H,j}(\mathscr{R}^{\partial\Omega_j}_{\Gamma} g_H).$$

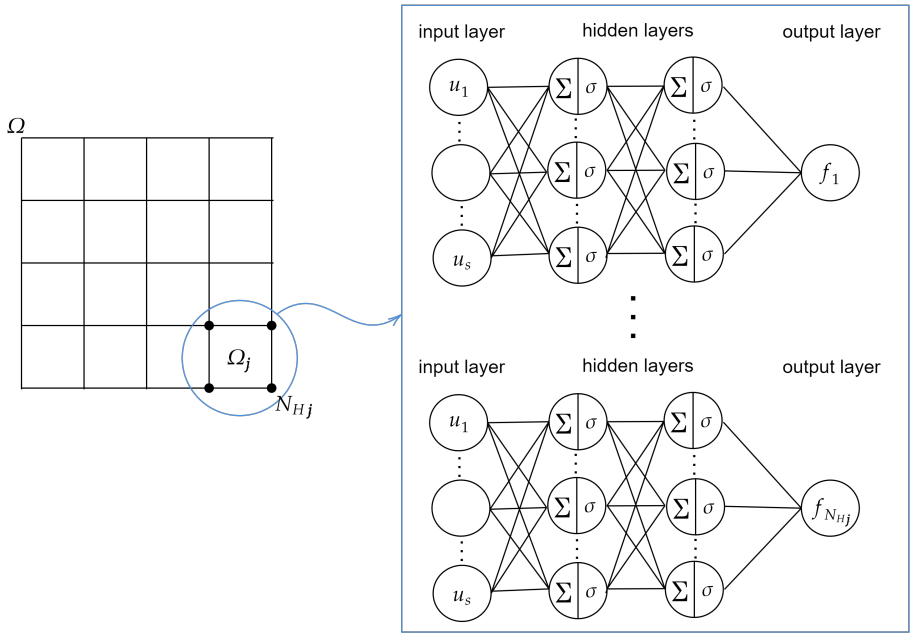


Figure 4.2: Representation of the learning workflow for the operator. Each neural network is depicted as a graph and will be responsible for learning each component of the $\widetilde{\mathrm{DtN}}_{H,j}$ operator. The first column of neurons (from left to right) is the input layer, taking inputs $(u_1, u_2, \cdots, u_s)$, and the last column is the output layer with output $f_{N_{Hj}} \approx \widetilde{\mathrm{DtN}}_{H,j}$. The intermediate columns represent the hidden layers.

and we obtain the problem:

$$\text{Find } \widetilde{g}_H \in \mathbb{R}^{\mathscr{N}_\Gamma} \text{ such that } \mathscr{R}_\Gamma^{\partial\Omega}\widetilde{g}_H = \mathscr{R}_\Gamma^{\partial\Omega}\pi_H^{-1}(u_{H,D}) \text{ and } \mathscr{R}_\Gamma^{\Gamma\cap\Omega}F_{H,learn}(\widetilde{g}_H) = 0. \tag{4.32}$$

We will construct the surrogate mapping $\widetilde{\mathrm{DtN}}_{H,j}$ by learning the individual components of the original map. To learn this operator, we use a fully-connected feed-forward NN. These networks are built as a sequence of layers including an input layer, hidden layers, and an output layer. Each layer performs an affine transformation followed by a nonlinear activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$. For each component $\widetilde{\mathrm{DtN}}^l_{H,j}$, where $l = 1, \ldots, \mathscr{N}_{\partial\Omega_j}$, we approximate it using a neural network expressed as a linear combination of functions:

$$\widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u) = \varphi^k_\theta \circ \ldots \circ \varphi^1_\theta(u),$$

where $\varphi^k_\theta(u) = \sigma(W_k(\theta)u + \mathbf{b}_k(\theta))$. The weights matrix W_k and biases $\mathbf{b}_k$ are determined through training. We illustrate this concept with Figure 4.2.

The solution to the DtN operator is approximated by the following optimization process. Let $U = \{u^s\}_{s=1}^{n_s}$ be a set of sampling vectors sampled in the range $(u_{\min}, u_{\max})$, where n_s denotes the total number of sampling vectors. For each $j = 1, \ldots, N$ and $l = 1, \ldots \mathscr{N}_{\partial\Omega_j}$, find $\theta^* = \arg\min_\theta \mathcal{L}(\theta, U)$, with

$$\mathcal{L}^l_j(\theta, U) = c_0\mathcal{L}^l_{j,0}(\theta, U) + c_1\mathcal{L}^l_{j,1}(\theta, U) + c_{\text{mon}}\mathcal{L}^l_{j,\text{mon}}(\theta) \tag{4.33}$$

where c_0, c_1 and c_{mon} are positive weights, $\mathcal{L}^l_{j,0}(\theta, U)$, $\mathcal{L}^l_{j,1}(\theta, U)$ and $\mathcal{L}^l_{j,\text{mon}}(\theta)$ are defined by (4.34), (4.35) and (4.36) below.

For the loss function (4.33), we have error in sampled values given by

$$\mathcal{L}^l_{j,0}(\theta, U) = \frac{1}{n_s} \sum_s \left(\widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u^s) - \mathrm{DtN}^l_{H,j}(u^s) \right)^2, \tag{4.34}$$

and error with respective derivatives given by

$$\mathcal{L}^l_{j,1}(\theta, U) = \frac{1}{n_s} \sum_{k=1}^{\mathcal{N}_{\partial\Omega_j}} \sum_s \left(\partial_{u_k} \widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u^s) - \partial_{u_k} \mathrm{DtN}^l_{H,j}(u^s) \right)^2. \tag{4.35}$$

In addition ,we introduce the monotonicity loss terms defined either by

$$\mathcal{L}^l_{j,\mathrm{mon}}(\theta) = \int_{(u_{\min}, u_{\max})^{\mathcal{N}_{\partial\Omega_j}}} \left(\left(\partial_{u_l} \widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u) \right)^- \right)^2 + \sum_{k \neq l} \left(\left(\partial_{u_k} \widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u) \right)^+ \right)^2 du, \tag{4.36}$$

or by

$$\mathcal{L}_{j,\mathrm{mon}}(\theta) = \int_{(u_{\min}, u_{\max})^{\mathcal{N}_{\partial\Omega_j}}} \left(\left(\partial_{u_l} \widetilde{\mathrm{DtN}}^l_{H,j}(\theta; u) \right)^- \right)^2 du, \tag{4.37}$$

where $(x)^+ = \max(x, 0)$, $(x)^- = \min(x, 0)$ for any real x, and where the integrals are taken over the training domain, which is an $\mathcal{N}_{\partial\Omega_j}$-dimensional cube. We note that $\mathcal{L}^l_{j,\mathrm{mon}}(\theta)$ does not require any data sampling. The purpose of this terms in the loss function is to enforce certain monotonicity properties that are expected from the original $\mathrm{DtN}_{H,j}$ maps. In particular, for the one-dimensional problem, we expect the Jacobian of $\mathrm{DtN}_{H,j}$ to have positive diagonal elements and non-positive off-diagonal ones. We note that this property is closely related to the discrete maximum principle and is classical for Finite Volume methods. The loss term defined in (4.36) aims to penalize the wrong Jacobian signs. For problems in 2D, the off-diagonal elements of $\mathrm{DtN}'_{H,j}$ do not have specific sign in general. Therefore, for such problems, we will be using (4.37) instead of (4.36). The monotonicity constrained based on (4.36) or (4.36) allows us to improve the robustness of Newton's method for (4.32), which otherwise would be likely to fail.

For training, we generate $\mathrm{DtN}_{H,j}(u^s)$ and $\mathrm{DtN}'_{H,j}(u^s)$ values for each sampling vector $(u^s)_{s=1}^{n_s}$. In terms of choosing sampling points to be used in training, we use a uniform sampling procedure. Given a sampling range (u_{min}, u_{max}), we sample each vector component with m entries equally spaced between u_{min} and u_{max}. Occasionally, these m sampling points can also include additional points, particularly in the "center" of existing training points. This naturally leads to $n_s = m^{\mathcal{N}_{\partial\Omega_j}}$ training vectors for the NN model, where $(u^s) \in \mathbb{R}^{\mathcal{N}_{\partial\Omega_j}}$.

4.3 Numerical experiments

4.3.1 1D degenerate elliptic problem

We first consider the following nonlinear equation

$$au - \left(K_\epsilon(x) \left(u^4 \right)' \right)' = 0 \tag{4.38}$$

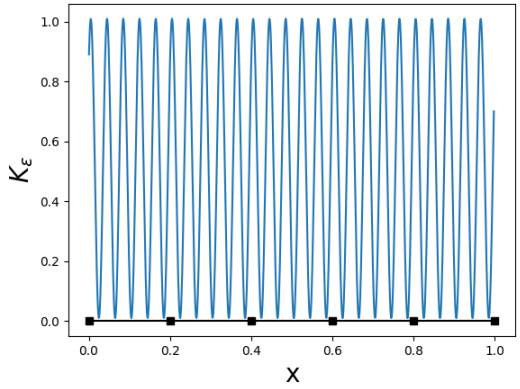


Figure 4.3: Partitioning of the model domain and the plot of $K_\epsilon(x)$.

posed in the domain $(0,1)$ with some non-negative Dirichlet boundary conditions. The diffusion coefficient is given by

$$K_\epsilon(x) = 10^{-2} + \frac{1}{2}\left(1 + \sin\left(10\pi x + \frac{\pi}{4}\right)\right),$$

and $a = 20$. Equation (4.38) can be interpreted as a stationary variant of the porous medium equation [103]. It can be shown that u takes values in the interval $[0, \max(u(0), u(1)]$. We note that, since the diffusion term in (4.38) degenerates at $u = 0$, the solutions of (4.38) may vanish over some portion of the domain. The domain $(0,1)$ is partitioned into five subdomains of equal size. We refer to Figure 4.3 for the illustration of the partitioning and the coefficient K_ϵ. Because the partitioning of the domain is matching the periodicity of $K_\epsilon(x)$, the DtN operators coincide for all the subdomains. As the analytical expression of the subdomain's DtN operator is not available, the latter will be replaced by the approximate one denoted by $\mathrm{DtN}_{h,j}$ and computed by a finite element method using 200 grid points. To be more specific, we use a $\mathbb{P}_1$ finite element method with mass lumping. We note that in the one-dimensional case, the mapping $\mathrm{DtN}_{h,j}$, acting from $\mathbb{R}^2$ to itself coincides with the coarse operator $\mathrm{DtN}_{H,j}$ For the surrogate model $\widetilde{\mathrm{DtN}}_{H,j}$, we use a fully connected neural network with two hidden layers of 64 neurons each. The activation function used in all the numerical experiment is given by $\sigma(u) = \max(u,0)^2$. The model is trained over the domain $[0, u_{max}]^2$ with $u_{max} = 4$. Training points are sampled over a regular grid in $[0, u_{max}]^2$ consisting of $n_s = 2^2, 3^2, 4^2$ and 5^2 points.

The loss function involves both values and derivatives of $\mathrm{DtN}_{H,j}$ at the sampling points with $c_0 = 1$ and $c_1 = 0.1$, as well as the monotonicity component (4.36) with $c_{mon} = 4$. The integral in (4.36) is approximated numerically based on integrand values over a regular 40×40 grid.

Let us begin with a qualitative analysis of the surrogate $\widetilde{\mathrm{DtN}}_{H,j}$ model. Recall that, in 1D, $\mathrm{DtN}_{H,j}$ (coinciding with $\mathrm{DtN}_{h,j}$) maps $\mathbb{R}^2$ to itself, which can be interpreted as mapping left and right Dirichlet data to a pair of left and right fluxes. We report on Figure 4.4 the left flux and its surrogates as a function of the input Dirichlet values. The surface $z = \mathrm{DtN}_{H,j}((u_{\text{left}}, u_{\text{right}}))^1$ shown as solid is the same for all sub-figures. Colored wireframe surfaces show $z = \widetilde{\mathrm{DtN}}_{H,j}((u_{\text{left}}, u_{\text{right}}))^1$ for various number of sampling points (4, 5 and 9 for the first, second and third rows respectively), and different values of coefficients c_α in (4.33) (same for each column). The results obtained using DtN values alone ($c_1 = 0 = c_{mon} = 0$) are shown in the first column, those obtained using both values and derivatives ($c_{mon} = 0$) are reported in

the middle column. Finally, the right column shows the outputs of the model using the additional monotonicity loss term (4.36). For the sake of clearer visualization, the parameter a is set to 1. Unsurprisingly, fitting not only the values, but also the tangential planes has a drastic benefit on the quality of the surrogate model; we see this by comparing the left and middle columns of Figure 4.4. Models with $c_1 > 0$ manage to capture the shape of the response surface reasonably well for very few training points. We also observe that adding the monotonicity loss function does not seem to pollute the interpolation quality; we see this by comparing the middle and right columns of Figure 4.4.

We report on Figure 4.5 the interpolation error of $\mathrm{DtN}_{H,j}$ in relative $L^2((0, u_{max})^2)$ norm as the function of the number of sampling points. The error is evaluated over a regular 20×20 grid in $[0, u_{max}]^2$. Here, in addition to the points sampled over regular $n_s \times n_s$ grid, we include additional points taken at the centers of the sampling grid cells. The decrease of error deteriorates after approximately 25 sampling points. This saturation effect (around error value of $10^{-3.5}$) is even more clear in the second test case (see Figure 4.9 below). The failure in achieving higher accuracy could likely be attributed to our current training algorithm, combined with the fact that the structure of the surrogate model is fixed through the whole experiment.

We further evaluate the performance of the surrogate models for the solution of the substructured problem (4.32), with $u(1) = 0$, and $u(0)$ taking values in $\{1, 2, 3, 4\}$. Once the coarse solution g_H and $\widetilde{g}_H$ are computed by approximately solving (4.31) and (4.32), respectively, we reconstruct the solution inside the subdomains by solving local problems on Ω_j with boundary condition given either by $g_H|_{\partial\Omega_j}$ or $\widetilde{g}_H|_{\partial\Omega_j}$. Figure 4.6 reports the reconstructed solution, using both the original and the surrogate DtN models, for different boundary conditions and varying number of sampling points. Starting from $n_s = 3^2$, the surrogate model produced the solutions which are visually appear to be accurate. The relative $L^2(0, 1)$ error as the function of the number of sampling points is reported by Figure 4.5.

The numerical solution of (4.31) and (4.32) is obtained by Newton's method. In Figure 4.7, we report typical convergence history for Newton's method for the original (4.31) and the surrogate model (4.32) with $u(0) = u_{max}$ and $n_s = 4^2$. Here, the use of the surrogate model leads to convergence of Newton's method with slightly fewer iterations. Most importantly, the inference time of $\widetilde{\mathrm{DtN}}$ is much lower than the time required to evaluate DtN.

4.3.2 1D $p-$Laplace problem

Next, we consider the following equation

$$au - \left(K_\epsilon(x)|u'|^2u'\right)' = 0. \tag{4.39}$$

In this example, all of the numerical parameters are set up as in the previous test case with the exception of the coefficient $a = 5$. Again, we compare the solution obtained using either the original or the surrogate substructured problem ((4.31) or (4.32)). Solution comparison for different boundary conditions and varying sampling point size is reported on Figure 4.8. Starting from $n_s = 3^2$, the solution obtained using the surrogate model appears to be reasonably accurate. Figure 4.9 exhibits convergence of the relative interpolation and solution errors in the $L^2((0, u_{max})^2)$ and $L^2(0, 1)$ norms,respectively. As before, we observe the saturation of the interpolation error as the function of n_s; this phenomenon is even more pronounced than in the

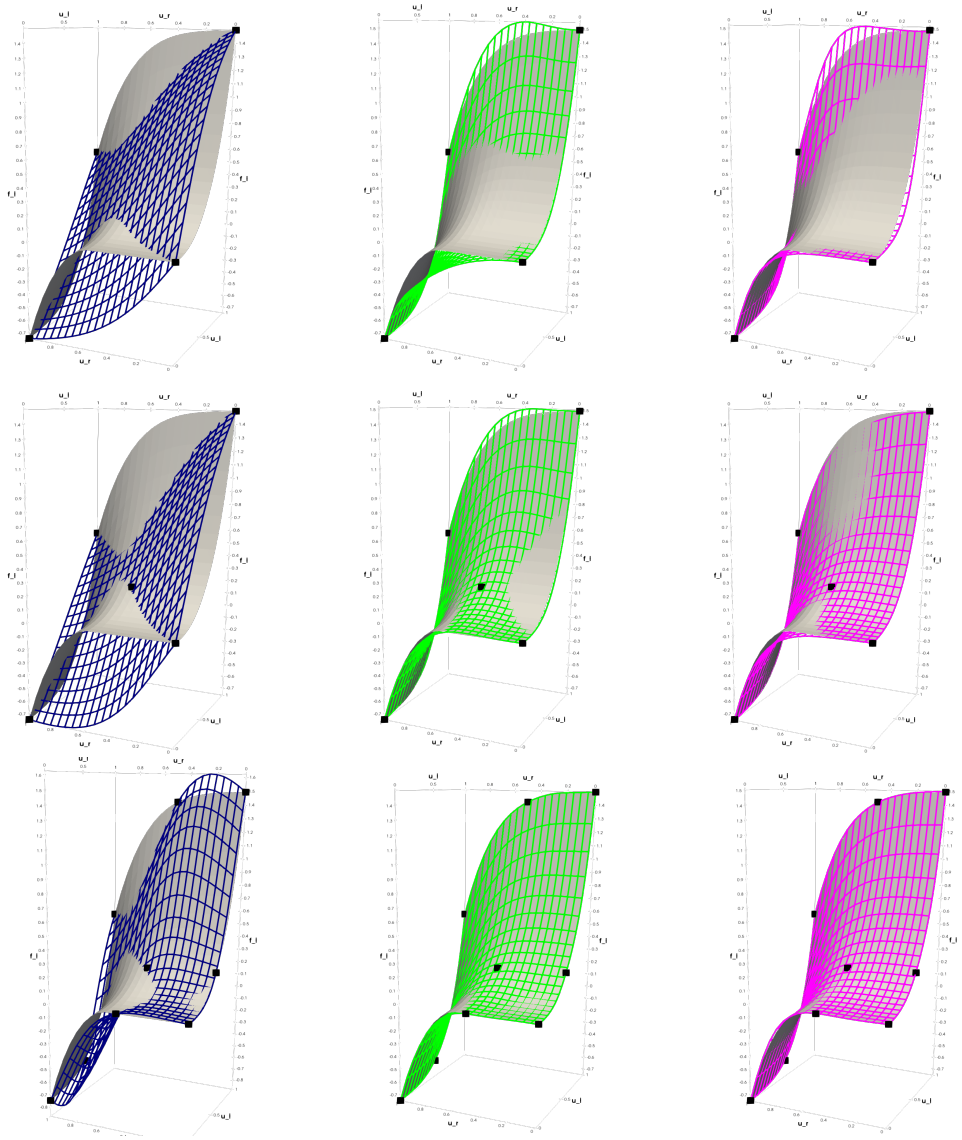


Figure 4.4: Qualitative comparison of the surfaces $z = \mathrm{DtN}_{H,j}((u_{\text{left}}, u_{\text{right}}))^1$ (solid) and $z = \widetilde{\mathrm{DtN}}_{H,j}((u_{\text{left}}, u_{\text{right}}))^1$ (wireframe). Left column (blue): only DtN values included. Middle column (green): DtN and DtN derivatives included. Right column (purple): DtN values, derivatives, and monotonicity assertion included. Top to bottom: $n_s = 2^2$ training points, $n_s = 2^2 + 1$ training points, $n_s = 3^2$ training points.

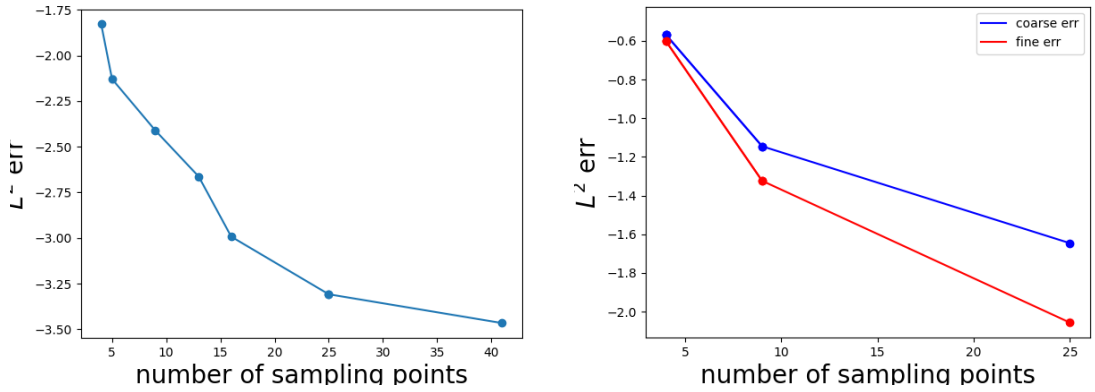


Figure 4.5: Left: Interpolation relative L^2 error. Right: Solution relative L^2 error.

previous test case. Typical convergence of Newton's method is shown by Figure 4.10 for $u(0) = 4$ and $n_s = 4^2$. Although Newton's method requires twice as many iterations for the surrogate model compared to the original one, the former leads to a much faster algorithm due to a much lower inference time.

4.3.3 2D degenerate elliptic problem

We consider a two-dimensional version of (4.38) given by

$$u - \operatorname{div}\left(K_\epsilon(x,y)\nabla u^4\right) = 0 \quad \text{in} \quad \Omega = (0,1)^2, \tag{4.40}$$

combined with the Dirichlet boundary conditions

$$u(x,y) = \max(u_{max}(x+y-1), 0) \qquad \text{on} \qquad \partial\Omega.$$

where $u_{max} = 1.2$. That is, $u(x,y) = 0$ on the lower and left boundaries and $u(x,y) = u_{max}(x+y-1)$ on right and top ones. We set

$$K_\epsilon(x,y) = 10^{-2} + \frac{1}{2}\left(1 + \sin\left(10x + \frac{\pi}{2}\right)\sin\left(5y + \frac{\pi}{2}\right)\right),$$

and we consider a regular partitioning of Ω into 5×5 subdomains (see Figure 4.11 for illustration). Here again, since the partitioning of the domain matches the periodicity of the coefficient K_ϵ, one only needs to build a single surrogate DtN model. More precisely, for every component of $\mathrm{DtN}^l_{H,j}$, we train a specific model $\widetilde{\mathrm{DtN}}^l_{H,j}$ using a fully connected neural network with 2 hidden layers of 20 neurons each and the activation function $\sigma(u) = \max(u,0)^2$.

The reference $\mathrm{DtN}_{H,j}$ operator is computed using mass lumped $\mathbb{P}_1$ finite element method on a fine grid with approximately 660 triangles and 340 nodal degrees of freedom by subdomain. We report on Figure 4.11 the solution using this reference DtN map. The training data is sampled on a regular grid over $[0, u_{max}]^4$. The loss function uses both values and derivatives of $\mathrm{DtN}^l_{H,j}$ at the sampling points with $c_0 = 1$ and $c_1 = 0.1$. The monotonicity of $\widetilde{\mathrm{DtN}}^l_{H,j}$ is enhanced by adding the loss term (4.37) with $c_{mon} = 10$. The integral in (4.37) is approximated by a Monte Carlo method, using 200 random points sampled in $[0, u_{max}]^4$ at every optimization step.

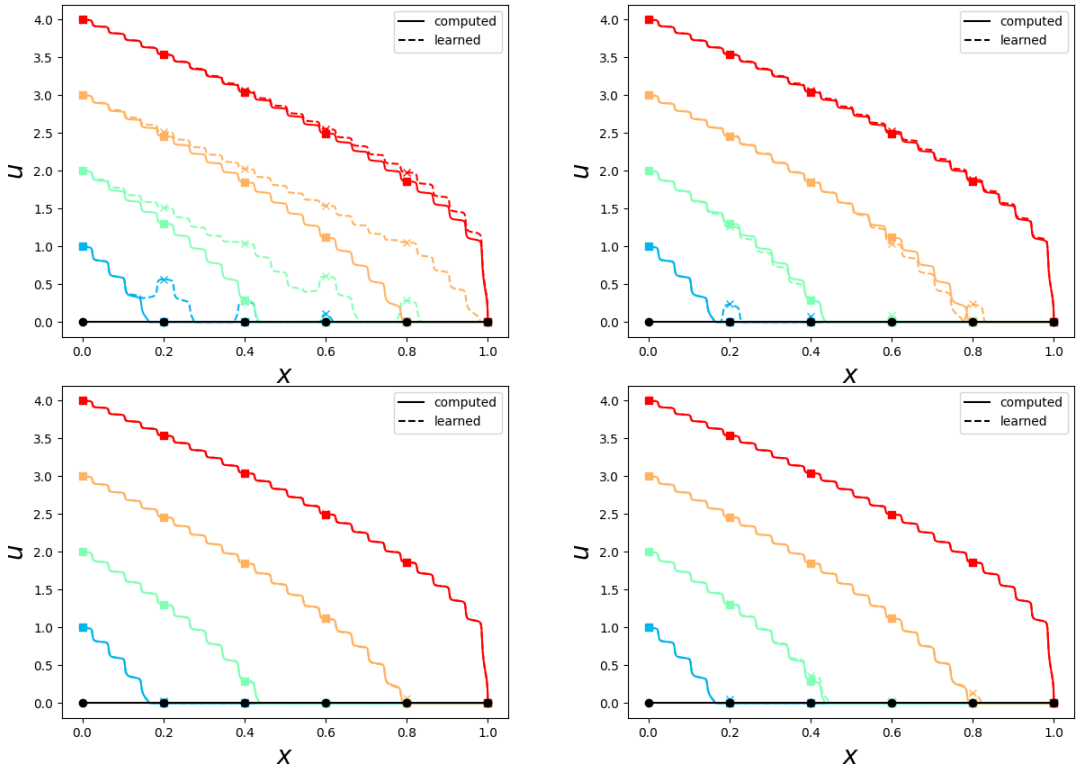


Figure 4.6: Solution profiles of (4.38) for various left boundary conditions using $\mathrm{DtN}_{H,j}$ (solid) and $\widetilde{\mathrm{DtN}}_{H,j}$ (dashed) operators for $n_s = 4, 9, 16$ and 25.

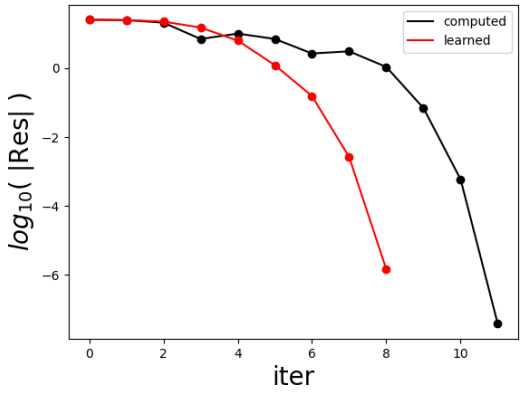


Figure 4.7: Convergence of Newton's method using $\mathrm{DtN}_{H,j}$ (black) and $\widetilde{\mathrm{DtN}}_{H,j}$ (red) operators to solve (4.38).

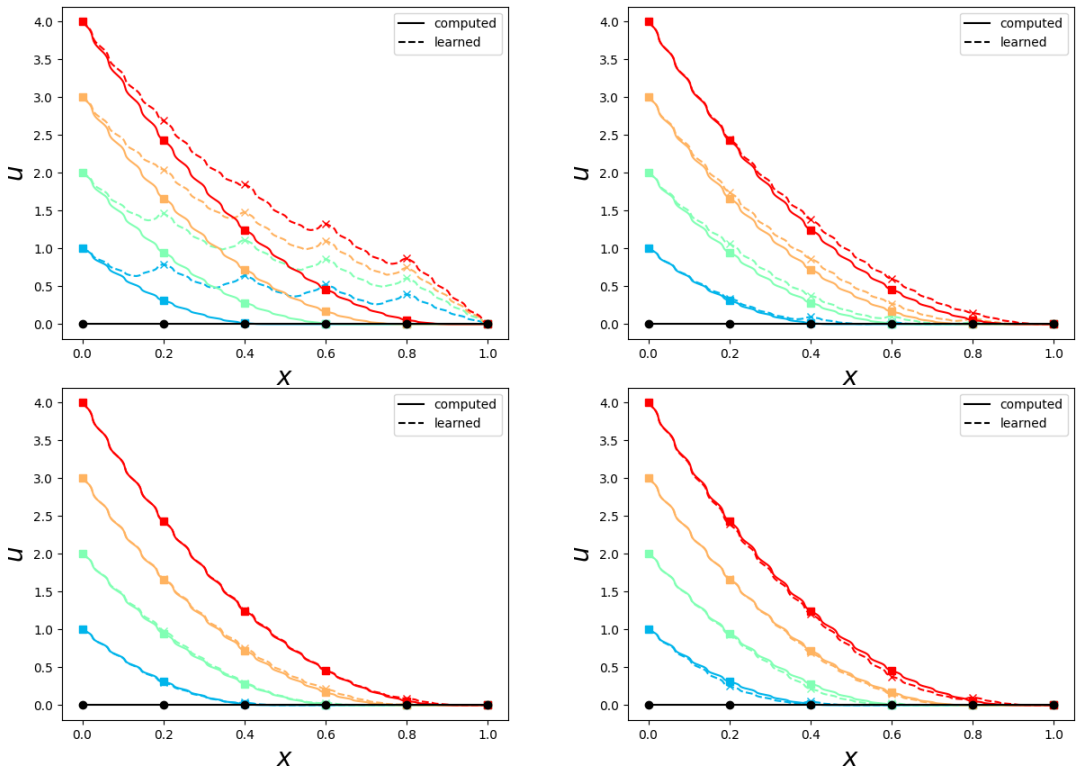


Figure 4.8: Solution profiles of (4.38) for various left boundary conditions using $\mathrm{DtN}_{H,j}$ (solid) and $\widetilde{\mathrm{DtN}}_{H,j}$ (dashed) operators for $n_s = 4, 9, 16$ and 25.

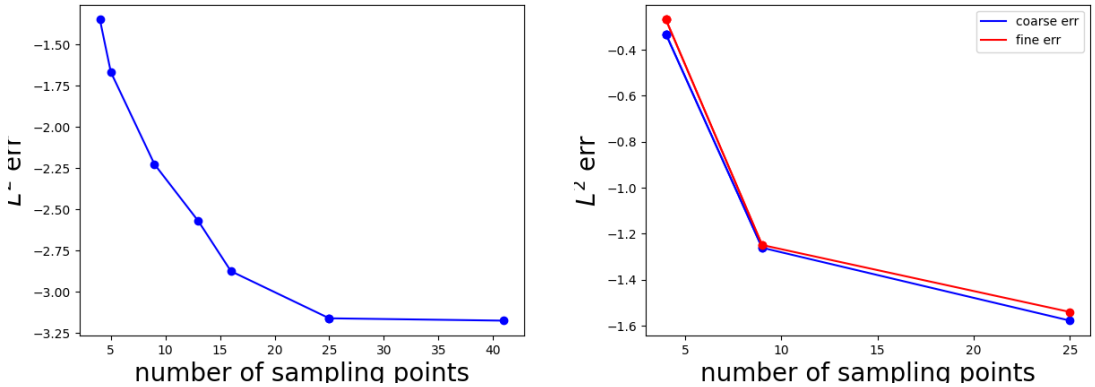


Figure 4.9: Left: Interpolation relative L^2 error. Right: Solution relative L^2 error.

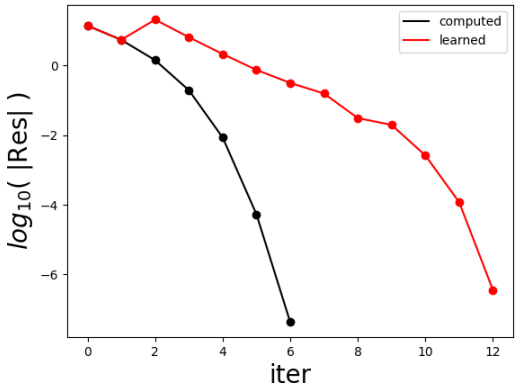


Figure 4.10: Convergence of Newton's method using DtN (black) and $\widetilde{\text{DtN}}$ (red) operators to solve (4.39).

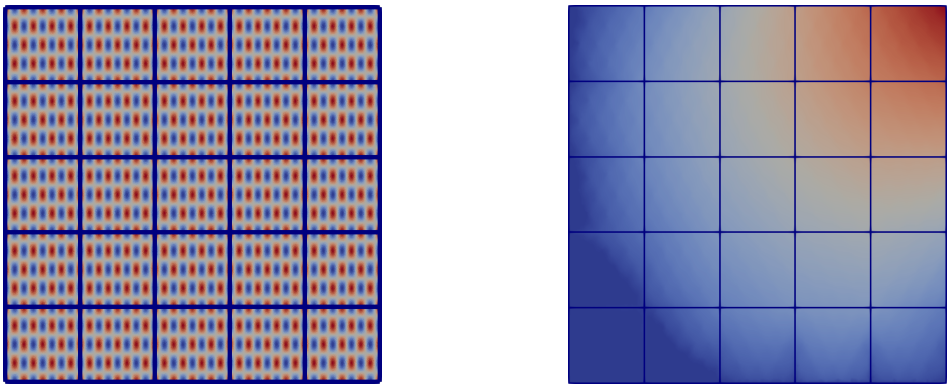


Figure 4.11: Left: Diffusion coefficient $K_\epsilon(x)$. Right: Solution of model problem (4.40) using the original $\text{DtN}_{H,j}$ operator.

n_s	2^4	3^4	4^4	5^4
Rel. error	33%	8%	4%	3%

Table 4.1: Relative $L^2(\Omega)$ error between surrogate model and true models.

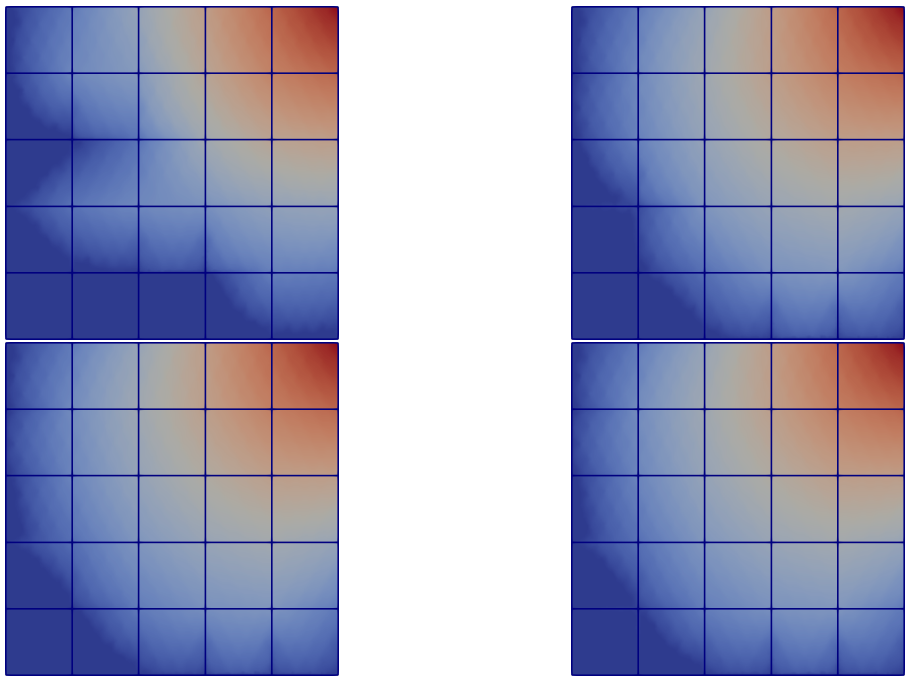


Figure 4.12: Solution of model problem (4.40) using learned $\widetilde{\mathrm{DtN}}$ map for various numbers of sampling points, including $n_s = 2^4$ (top left), $n_s = 3^4$ (top right), $n_s = 4^4$ (bottom left), and $n_s = 5^4$ (bottom right).

The solution obtained using the surrogate models trained with $n_s = 2^4, 3^4, 4^4$ and 5^4 are reported on Figure 4.12, together with solution profiles along the line $x = y$ shown on Figure 4.13 and relative $L^2(\Omega)$ errors shown in Table 4.1. We observe that the error decreases as the number of sampling points grows, with the relative $L^2(\Omega)$ being below 10% starting from $n_s = 3^4$.

Let us recall that the evaluation of the residual function (4.30) based on the original coarse $\mathrm{DtN}_{H,j}$ operator requires the solution of the family of local nonlinear finite element systems, which is computationally demanding. In contrast, the inference time of $\widetilde{\mathrm{DtN}}_{H,j}$ is very low, allowing for fast computation of the solution to the surrogate system (4.32). Although the surrogate solution $\widetilde{g}_H$ may not be sufficiently accurate, it can serve to initialize the nonlinear solver for the original coarse problem (4.31). This idea is illustrated by Figure 4.14. Figure 4.14 reports convergence of the relative $L^2(\Omega)$ error between a very accurate solution to (4.31) and the current Newton's method iterate obtained using either using $\widetilde{\mathrm{DtN}}_{H,j}$ or $\mathrm{DtN}_{H,j}$ with different initialization. The black curve shows convergence of Newton's method for the original problem (4.31), starting from zero initial guess, and is the same for all three sub-figures. Similarly, the red curves depict convergence of Newton's method for the surrogate problem (4.32) for various number $n_s = 2^4, 3^4$ and 4^4, again starting for zero initial guess. The stagnation of the red curve is results the approximation

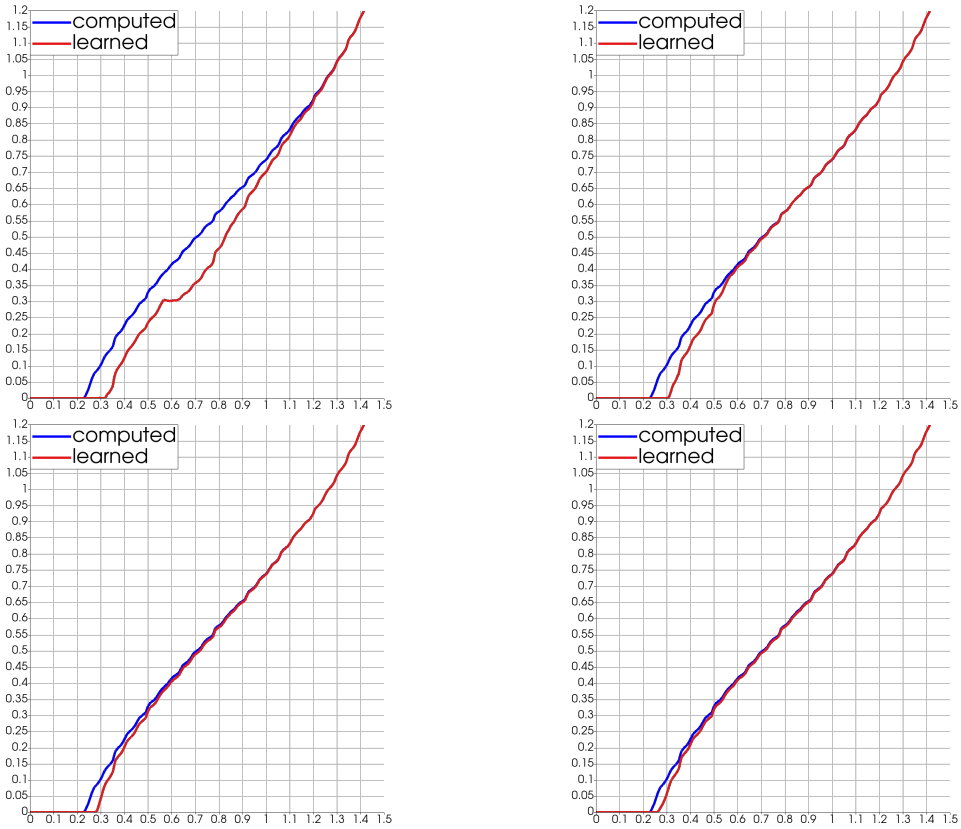


Figure 4.13: Original (blue) and surrogate (red) reconstructed solutions along the line $y = x$ for $n_s = 2^4$ (top left), $n_s = 3^4$ (top right), $n_s = 4^4$ (bottom left) and $n_s = 5^4$ (bottom right).

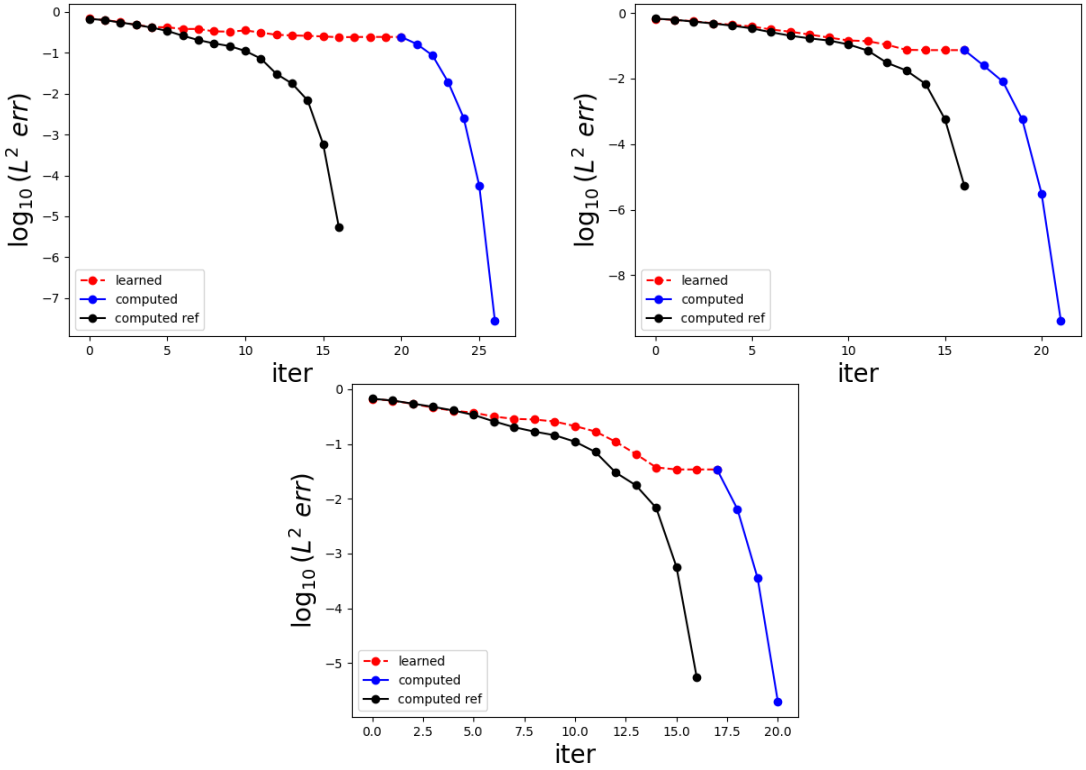


Figure 4.14: L^2 error between the reference solution and the current iterate of Newton's method for (4.32) using zero initial guess (red), and (4.31) with either initial guess set to zero (black) or provided by the approximate solution of (4.32). Top: $n_s = 2^4$ (left), $n_s = 3^4$ (right). Bottom: $n_s = 4^4$.

error associated to $\widetilde{\mathrm{DtN}}_{H,j}$. Finally, the blue curve shows convergence of Newton's method for (4.31) starting from the surrogate solution. For the original coarse problem, convergence up to the tolerance of 10^{-5} is obtained within 17 iterations. In contrast, the Newton's method initialized by the surrogate solution requires up to 4 times fewer iterations in order to achieve the similar accuracy. In particular, there are only 7, 5 or 4 iterations needed for $n_s = 2^4, 3^4$ and 4^4, respectively. Note that even when using a very crude $n_s = 2$ surrogate model, a significant reduction of Newton's step is achieved.

4.4 Conclusions

We have presented the framework that combines the Multi-scale Finite Element (MsFEM) method with techniques from Machine Learning, providing an extension of the former to the case of nonlinear PDEs with highly oscillatory coefficients. Our approach relies on learning Dirichlet-to-Neumann (DtN) maps acting between some low-dimensional (coarse) spaces. The surrogate DtN operators are further combined within a global substructured formulation solved by Newton's method. The optimization process involves fitting values and derivatives of the true coarse DtN, where incorporating derivative information into the loss function plays a pivotal role in accuracy of the learned operator. In addition, we weakly enforce some monotonicity properties of the original model, which improves the performance of Newton's method.

Numerical experiments, performed in 1D and 2D and involving challenging $p-$Laplace and degenerate diffusion equations, have shown promising results. With just a few training points by dimension, the substitution model can approach the solution with an accuracy of a few percent. A further improvement in the accuracy can be achieved by using the "learned" solution as an initial guess for Newton's method applied to the original (not learned) substructured formulation, allowing us to enter directly into the region of quadratic convergence and obtain a more accurate "learned" solution. Although our numerical experiment has been so far limited to the case periodically distributed coefficients, the extension to the non-periodic case is trivial and will be carried in future. Future work also involves incorporating techniques from Principal Component Analysis (PCA) to reduce complexity associated to the derivatives of the DtN maps. Additionally, the inclusion of higher-order derivatives could increase model accuracy. Higher-order optimizers like BFGS, which incorporates curvature/Hessian information into the loss function, may improve the convergence of the loss function.

Chapter 5

Perspectives

This thesis has explored efficient numerical methods to solve linear and nonlinear PDEs on heterogeneous domains, particularly domains containing a very large number of perforations. In our main application concerning urban flood modeling, these perforations represent buildings and walls/fences in urban areas, while the flood is assumed to be governed by the nonlinear parabolic Diffusive Wave equation. The main ingredient used throughout this thesis is the Trefftz coarse space adapted to the geometrical complexity of urban domain, which we introduce in Chapter 2. This coarse space shares many common features with MsFEM and can be used either as a component of a two-level Domain Decomposition method, within a Galerkin method, or in combination with Machine learning techniques. Below, we highlight potential research avenues arising from the current contribution. The discussion is organized following the structure of the manuscript.

Chapter 2 introduces the Trefftz coarse space adapted to urban geometries in the context of a simple linear elliptic equation. The main theoretical contribution concerns the error estimate regarding the H^1-projection over the coarse space. The error analysis does not rely on global regularity of the solution and is performed under some very minimal assumptions regarding the geometry of the domain. However, the right-hand side of the error estimate involves the norm of the (unknown) solution along the coarse edges. This could potentially be improved by modifying the error estimate so that the right-hand side is independent of the solution, and instead dependent on the the source term and boundary data. Additionally, we have observed a higher order of convergence in the L^2 norm compared to the H^1 norm; future work could involve the extension of the error analysis to the L^2 norm based on Aubin-Nitsche trick.

Combined with the Restricted Additive Schwarz (RAS) method, the Trefftz space leads to an efficient and robust iterative solver or preconditioner for linear systems resulting from fine-scale finite element discretizations. This improvement comes at the price of a somewhat larger coarse problem compared to some traditional coarse spaces. The dimension of the coarse space can be reduced by considering zero-order polynomials on the coarse edges that do not intersect any other coarse edge. Compared to the case of coarse functions that are piece-wise affine along the skeleton, this approach can eventually reduce the size of the coarse by at most factor 2.

Finally, we have assumed so far that the urban structures are impervious and thus can be removed from the computational domain. A more realistic representation of

the urban data may involve associating it with some low but positive permeability allowing for some water flow. We expect our methods and analysis to readily extend to such configurations.

Chapter 3 provided multiple efficient nonlinear domain decomposition methods to solve nonlinear equations on complex, perforated domains. Specifically, we have focused on a Diffusive Wave model on a domain with numerous perforations. The Trefftz coarse space introduced in Chapter 2 for linear problems on perforated domains proves to be well-suited as a component of efficient two-level methods for this challenging nonlinear flow model. Particularly, the two-level RASPEN method with the Trefftz space performs extremely well. The performance of the alternative methods are evaluated in terms of the number of nonlinear and linear iterations and the computational complexity is detailed for all considered methods. However, our current implementation does not allow any definitive conclusions to be drawn regarding the rapidity of calculation of the methods under consideration. The main bottleneck is the subdomain loop, which in Python becomes very inefficient as the number of subdomains N increases. This problem can be alleviated either by compiling the code or by performing the calculations in parallel.

For the linearized systems resulting at each iteration of the Two-step and Newton methods, the Trefftz space performs well as a component of the two-level RAS preconditioner and yields a reasonable number of inner GMRES iterations per outer iteration. It results in a fair robustness in terms of GMRES iterations as N is increased, with these iterations generally slightly increasing with N. We believe that the latter effect could be mitigated by updating the coarse basis functions according to the changes in the diffusion coefficient. For example, the basis functions could be updated over time to account for the solution dependent coefficient $k(u, \nabla u)$ in the Diffusive Wave model. We have also proposed a coarse Galerkin method which provides a coarse approximation to the solution of the nonlinear PDE. Future work may involve convergence analysis of this nonlinear method. A parallel implementation would address the cost of the fine-scale assembly in the coarse Galerkin method and is already in progress. Alternatively, the computational cost associated to the fine-scale assembly can be mitigated using hyper-reduction techniques like (Discrete) Empirical Interpolation Method [10], [32], [109].

Chapter 4 explored the combination of machine learning with the efficient numerical solutions of nonlinear PDEs. We have presented the framework that combines the Multi-scale Finite Element (MsFEM) method with techniques from Machine Learning, providing an extension of the former to the case of nonlinear PDEs with highly oscillatory coefficients. Our approach relies on learning Dirichlet-to-Neumann (DtN) maps acting between some low-dimensional (coarse) spaces. Our work into scientific machine learning is introductory, and further exploration is needed to fully understand this topic.

Although numerical experiments have so far been limited to the case of periodically distributed coefficients, the extension to the non-periodic case such as the urban domains discussed in the earlier chapters should be carried out in the future. The main challenge regarding the non-periodic case is that one would need to train as many surrogate models as the number of subdomains. On the other hand, dealing with perforated domains requires higher-dimensional approximate DtN operators, as the number of coarse degrees of freedom by subdomain is not fixed, requiring

larger training sets and surrogate models. However, to some extent, this is merely technical as both the generation of the training data and the training itself are embarrassingly parallel.

Future work also involves incorporating techniques from Principal Component Analysis (PCA) to reduce the complexity associated to the derivatives of the DtN operators. Additionally, the inclusion of higher-order derivatives of the DtN operator could increase model accuracy. The complexity of the resulting loss function can be controlled by only considering higher derivatives of the dominant diagonal part of the operator. Additionally, future work would involve an implementation which allows for higher-order optimizers like BFGS.We remark finally that as an alternative to our proposed method, the DtN operators could be generated with PINNS and this would be an interesting avenue for future work.

Bibliography

[1] J. Aarnes and T. Y. Hou. "Multiscale domain decomposition methods for elliptic problems with high aspect ratios". In: *Acta Math. Appl. Sin. Engl. Ser.* 18.1 (2002), pp. 63–76.

[2] M. Abily, N. Bertrand, O. Delestre, P. Gourbesville, and C.-M. Duluc. "Spatial global sensitivity analysis of high resolution classified topographic data use in 2D urban flood modelling". In: *Environmental Modelling & Software* 77 (2016), pp. 183–195.

[3] R. Alonso, M. Santillana, and C. Dawson. "On the diffusive wave approximation of the shallow water equations". In: *Eur. J. Appl. Math.* 19.5 (2008), pp. 575–606.

[4] L. Andres. "L'apport de la donnée topographique pour la modélisation 3D fine et classifiée d'un territoire". In: *Rev. XYZ* 133.4 (2012), pp. 24–30.

[5] R. Araya, C. Harder, D. Paredes, and F. Valentin. "Multiscale hybrid-mixed method". In: *SIAM J. Numer. Anal.* 51.6 (2013), pp. 3505–3531.

[6] M. Arioli, D. Loghin, and A. J. Wathen. "Stopping criteria for iterations in finite element methods". In: *Numerische Mathematik* 99 (2005), pp. 381–410.

[7] I. Babuska, G. Caloz, and J. E. Osborn. "Special finite element methods for a class of second order elliptic problems with rough coefficients". In: *SIAM Journal on Numerical Analysis* 31.4 (1994), pp. 945–981.

[8] I. Babuska and R. Lipton. "Optimal local approximation spaces for generalized finite element methods with application to multiscale problems". In: *Multiscale Modeling and Simulation* 9.1 (2011), pp. 373–406.

[9] S. Badia, W. Li, and A. F. Martín. "Finite element interpolated neural networks for solving forward and inverse problems". In: *arXiv:2306.06304* (2023).

[10] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. "An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus Mathematique* 339.9 (2004), pp. 667–672.

[11] L. Beir ao da Veiga, A. Chernov, L. Mascotto, and A. Russo. "Basic principles of hp virtual elements on quasiuniform meshes". In: *Mathematical Models and Methods in Applied Sciences* 26.08 (2016), pp. 1567–1598.

[12] M. Boutilier, K. Brenner, and V. Dolean. "A Trefftz-like coarse space for the two-level Schwarz method on perforated domains". In: *International Conference on Domain Decomposition Methods*. Springer. 2022, pp. 77–84.

[13] M. Boutilier, K. Brenner, and V. Dolean. "Robust methods for multiscale coarse approximations of diffusion models in perforated domains". In: *Applied Numerical Mathematics* (2024).

[14] M. Boutilier, K. Brenner, and V. Dolean. "Trefftz approximation space for Poisson equation in perforated domains". In: (2023).

[15] M. Boutilier, K. Brenner, and V. Dolean. "Two-level nonlinear preconditioning methods for flood models posed on perforated domains". In: *arXiv:2406.06189* (2024).

[16] M. Boutilier, K. Brenner, and L. Miguez. "Learning local Dirichlet-to-Neumann maps of nonlinear elliptic PDEs with rough coefficients". In: *arXiv:2405.04433* (2024).

[17] K. Brenner. "On global and monotone convergence of the preconditioned Newton's method for some mildly nonlinear systems". In: *International Conference on Domain Decomposition Methods*. Springer. 2022, pp. 85–92.

[18] K. Brenner, M. Groza, C. Guichard, G. Lebeau, and R. Masson. "Gradient discretization of hybrid dimensional Darcy flows in fractured porous media". In: *Numerische Mathematik* 134 (2016), pp. 569–609.

[19] H. Brezis and P. Mironescu. "Gagliardo–Nirenberg inequalities and non-inequalities: the full story". In: *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*. Vol. 35. 5. Elsevier. 2018, pp. 1355–1376.

[20] F. Brezzi, R. S. Falk, and L. D. Marini. "Basic principles of mixed virtual element methods". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 4.48 (2014), pp. 1227–1240.

[21] C. L. Bris, F. Legoll, and F. Madiot. "Multiscale finite element methods for advection-dominated problems in perforated domains". In: *Multiscale Modeling & Simulation* 17.2 (2019), pp. 773–825.

[22] D. L. Brown and V. Taralova. "A multiscale finite element method for Neumann problems in porous microstructures". In: *Discrete & Continuous Dynamical Systems-S* 9.5 (2016), p. 1299.

[23] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks for heat transfer problems". In: *Journal of Heat Transfer* 143.6 (2021).

[24] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. "Newton-Krylov-Schwarz methods in CFD". In: *Numerical methods for the Navier-Stokes equations: Proceedings of the International Workshop Held at Heidelberg, October 25–28, 1993*. Springer. 1994, pp. 17–30.

[25] X.-C. Cai and D. E. Keyes. "Nonlinearly preconditioned inexact Newton algorithms". In: *SIAM Journal on Scientific Computing* 24.1 (2002), pp. 183–200.

[26] X.-C. Cai and X. Li. "Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity". In: *Siam journal on scientific computing* 33.2 (2011), pp. 746–762.

[27] X.-C. Cai and M. Sarkis. "A restricted additive Schwarz preconditioner for general sparse linear systems". In: *Siam journal on scientific computing* 21.2 (1999), pp. 792–797.

[28] Z. Cai, J. Jones, S. McCormick, and T. Russell. "Control-volume mixed finite element methods". In: *Computational Geosciences* 1 (1997), pp. 289–315.

[29] F. Chaouqui, G. Ciaramella, M. J. Gander, and T. Vanzan. "On the scalability of classical one-level domain-decomposition methods". In: *Vietnam Journal of Mathematics* 46 (2018), pp. 1053–1088.

[30] F. Chaouqui, M. J. Gander, P. M. Kumbhar, and T. Vanzan. "Linear and nonlinear substructured Restricted Additive Schwarz iterations and preconditioning". In: *Numerical Algorithms* 91.1 (2022), pp. 81–107.

[31] F. Chaouqui, M. J. Gander, P. M. Kumbhar, and T. Vanzan. "On the nonlinear Dirichlet–Neumann method and preconditioner for Newton's method". In: *Domain Decomposition Methods in Science and Engineering XXVI*. Springer, 2023, pp. 381–389.

[32] S. Chaturantabut and D. C. Sorensen. "Nonlinear model reduction via discrete empirical interpolation". In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764.

[33] E. T. Chung, Y. Efendiev, G. Li, and M. Vasilyeva. "Generalized multiscale finite element methods for problems in perforated heterogeneous domains". In: *Applicable Analysis* 95.10 (2016), pp. 2254–2279.

[34] E. T. Chung, W. T. Leung, and M. Vasilyeva. "Mixed GMsFEM for second order elliptic problem in perforated domains". In: *Journal of Computational and Applied Mathematics* 304 (2016), pp. 84–99.

[35] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[36] P. Degond, A. Lozinski, B. P. Muljadi, and J. Narski. "Crouzeix-Raviart MsFEM with bubble functions for diffusion and advection-diffusion in perforated media". In: *Communications in Computational Physics* 17.4 (2015), pp. 887–907.

[37] P. Di Giammarco, E. Todini, and P. Lamberti. "A conservative finite elements approach to overland flow: the control volume finite element formulation". In: *Journal of Hydrology* 175.1-4 (1996), pp. 267–291.

[38] C. R. Dohrmann, A. Klawonn, and O. B. Widlund. "Domain decomposition for less regular subdomains: Overlapping Schwarz in two dimensions". In: *SIAM J. Numer. Anal.* 46.4 (2008), pp. 2153–2168.

[39] V. Dolean, M. J. Gander, W. Kheriji, F. Kwok, and R. Masson. "Nonlinear preconditioning: how to use a nonlinear Schwarz method to precondition Newton's method". In: *SIAM Journal on Scientific Computing* 38.6 (2016), A3357–A3380.

[40] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. SIAM, 2015.

[41] V. Dolean, F. Nataf, R. Scheichl, and N. Spillane. "Analysis of a two-level Schwarz method with coarse spaces based on local Dirichlet-to-Neumann maps". In: *Computational Methods in Applied Mathematics* 12.4 (2012), pp. 391–414.

[42] F. Dottori, G. Di Baldassarre, and E. Todini. "Detailed data is welcome, but with a pinch of salt: Accuracy, precision, and uncertainty in flood inundation modeling". In: *Water Resources Research* 49.9 (2013), pp. 6079–6085.

[43] M. Dryja and O. B. Widlund. "Some domain decomposition algorithms for elliptic problems". In: *Iterative methods for large linear systems*. Elsevier, 1990, pp. 273–291.

[44] Y. Efendiev, J. Galvis, and T. Y. Hou. "Generalized multiscale finite element methods (GMsFEM)". In: *Journal of computational physics* 251 (2013), pp. 116–135.

[45] Y. Efendiev and T. Y. Hou. *Multiscale finite element methods: theory and applications*. Vol. 4. Springer Science & Business Media, 2009.

[46] A. Ern, J.-L. Guermond, A. Ern, and J.-L. Guermond. "Finite element interpolation". In: *Theory and Practice of Finite Elements* (2004), pp. 3–80.

[47] C. Evans, S. Pollock, L. G. Rebholz, and M. Xiao. "A proof that Anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically)". In: *SIAM Journal on Numerical Analysis* 58.1 (2020), pp. 788–810.

[48] C. Farhat, J. Mandel, and F. X. Roux. "Optimal convergence properties of the FETI domain decomposition method". In: *Computer methods in applied mechanics and engineering* 115.3-4 (1994), pp. 365–385.

[49] J. Fernández-Pato and P. García-Navarro. "2D zero-inertia model for solution of overland flow problems in flexible meshes". In: *Journal of Hydrologic Engineering* 21.11 (2016), p. 04016038.

[50] J. Galvis and Y. Efendiev. "Domain decomposition preconditioners for multiscale flows in high-contrast media". In: *Multiscale Model. Simul.* 8.4 (2010), pp. 1461–1483.

[51] M. J. Gander. "Optimized Schwarz methods". In: *SIAM Journal on Numerical Analysis* 44.2 (2006), pp. 699–731.

[52] M. J. Gander et al. "Schwarz methods over the course of time". In: *Electron. Trans. Numer. Anal* 31.5 (2008), pp. 228–255.

[53] M. J. Gander, L. Halpern, and K. Santugini-Repiquet. "On optimal coarse spaces for domain decomposition and their approximation". In: *Domain Decomposition Methods in Science and Engineering XXIV 24*. Springer. 2018, pp. 271–280.

[54] M. J. Gander, A. Loneland, and T. Rahman. "Analysis of a new harmonically enriched multiscale coarse space for domain decomposition methods". In: *arXiv:1512.05285* (2015).

[55] I. G. Graham, P. Lechner, and R. Scheichl. "Domain decomposition for multiscale PDEs". In: *Numer. Math.* 106.4 (2007), pp. 589–626.

[56] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach. "An adaptive GDSW coarse space for two-level overlapping Schwarz methods in two dimensions". In: *Domain Decompos. Method. Sci. Eng. XXIV*. 2018, pp. 373–382.

[57] A. Heinlein and M. Lanser. "Additive and hybrid nonlinear two-level Schwarz methods and energy minimizing coarse spaces for unstructured grids". In: *SIAM Journal on Scientific Computing* 42.4 (2020), A2461–A2488.

[58] P. Henning and M. Ohlberger. "The heterogeneous multiscale finite element method for elliptic homogenization problems in perforated domains". In: *Numerische Mathematik* 113 (2009), pp. 601–629.

[59] A. J. Hoffman, M. S. Martin, and D. J. Rose. "Complexity bounds for regular finite difference and finite element grids". In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 364–369.

[60] T. Hou, X.-H. Wu, and Z. Cai. "Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients". In: *Math. Comput.* 68.227 (1999), pp. 913–943.

[61] T. Y. Hou and X.-H. Wu. "A multiscale finite element method for elliptic problems in composite materials and porous media". In: *J. Comput. Phys.* 134.1 (1997), pp. 169–189.

[62] B. Huge and A. Savine. “Differential machine learning”. In: *arXiv:2005.02347* (2020).

[63] F.-N. Hwang and X.-C. Cai. “A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms”. In: *Computer methods in applied mechanics and engineering* 196.8 (2007), pp. 1603–1611.

[64] Institut National de l’Information Géographique et Forestière (IGN). *RGE ALTI® - Digital Terrain Model*. https://geoservices.ign.fr/rgealti. 2024.

[65] A. D. Jagtap and G. E. Karniadakis. “Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations.” In: *AAAI Spring Symposium: MLPS*. 2021.

[66] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. “Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems”. In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028.

[67] P. Jiránek, Z. Strakovs, and M. Vohralík. “A posteriori error estimates including algebraic error and stopping criteria for iterative solvers”. In: *SIAM Journal on Scientific Computing* 32.3 (2010), pp. 1567–1590.

[68] C. Kamthorncharoen. “Two-level Restricted Additive Schwarz Preconditioned Exact Newton with Applications”. PhD thesis. 2022.

[69] G. Karypis and V. Kumar. “METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices”. In: (1997).

[70] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. “Variational physics-informed neural networks for solving partial differential equations”. In: *arXiv:1912.00873* (2019).

[71] A. Klawonn, M. Lanser, and O. Rheinbach. “Nonlinear BDDC methods with approximate solvers”. In: *Electronic Transactions on Numerical Analysis* 49 (2018), pp. 244–273.

[72] A. Klawonn, M. Lanser, O. Rheinbach, and M. Uran. “Nonlinear FETI-DP and BDDC methods: a unified framework and parallel results”. In: *SIAM Journal on Scientific Computing* 39 (2017), pp. C417–C451.

[73] A. Klawonn, M. Lanser, and O. Rheinbach. “Nonlinear FETI-DP and BDDC methods”. In: *SIAM Journal on Scientific Computing* 36.2 (2014), A737–A765.

[74] C. Le Bris, F. Legoll, and A. Lozinski. “An MsFEM type approach for perforated domains”. In: *Multiscale Model. Simul.* 12.3 (2014), pp. 1046–1077.

[75] J. Leray and J. Lions. “Quelques résultats de Visik sur les problèmes elliptiques non linéaires par les méthodes de Minty-Browder”. In: *Séminaire Jean Leray* 1 (1964), pp. 1–19.

[76] P.-L. Lions. “On the Schwarz alternating method. I.” In: *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*. Ed. by R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux. Philadelphia, PA: SIAM, 1988, pp. 1–42.

[77] L. Liu, F.-N. Hwang, L. Luo, X.-C. Cai, and D. E. Keyes. “A nonlinear elimination preconditioned inexact Newton algorithm”. In: *SIAM Journal on Scientific Computing* 44.3 (2022), A1579–A1605.

[78] L. Liu and D. E. Keyes. "Field-split preconditioned inexact Newton algorithms". In: *SIAM Journal on Scientific Computing* 37.3 (2015), A1388–A1409.

[79] L. Liu, D. E. Keyes, and R. Krause. "A note on adaptive nonlinear preconditioning techniques". In: *SIAM Journal on Scientific Computing* 40.2 (2018), A1171–A1186.

[80] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature machine intelligence* 3.3 (2021), pp. 218–229.

[81] C. Ma, R. Scheichl, and T. Dodwell. "Novel design and analysis of generalized finite element methods based on locally optimal spectral approximations". In: *SIAM Journal on Numerical Analysis* 60.1 (2022), pp. 244–273.

[82] A. Målqvist and D. Peterseim. "Localization of elliptic multiscale problems". In: *Mathematics of Computation* 83.290 (2014), pp. 2583–2603.

[83] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. "Physics-informed neural networks for high-speed flows". In: *Computer Methods in Applied Mechanics and Engineering* 360 (2020), p. 112789.

[84] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis. "Physics-informed neural networks for power systems". In: *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE. 2020, pp. 1–5.

[85] B. Moseley, A. Markham, and T. Nissen-Meyer. "Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations". In: *Advances in Computational Mathematics* 49.4 (2023), p. 62.

[86] F. Nataf, F. Rogier, and E. de Sturler. "Optimal interface conditions for domain decomposition methods". PhD thesis. CMAP Ecole Polytechnique, 1994.

[87] F. Nataf, H. Xiang, V. Dolean, and N. Spillane. "A coarse space construction based on local Dirichlet-to-Neumann maps". In: *SIAM J. Sci. Comput.* 33.4 (2011), pp. 1623–1642.

[88] C. Negrello, P. Gosselet, and C. Rey. "Nonlinearly preconditioned FETI solver for substructured formulations of nonlinear problems". In: *Mathematics* 9.24 (2021), p. 3165.

[89] R. A. Nicolaides. "Deflation of conjugate gradients with applications to boundary value problems". In: *SIAM J. Numer. Anal.* 24.2 (1987), pp. 355–365.

[90] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[91] V. M. Ponce. "Diffusion wave modeling of catchment dynamics". In: *Journal of hydraulic engineering* 112.8 (1986), pp. 716–727.

[92] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[93] W. Rudin et al. *Functional analysis*. New York: McGraw-Hill, 1973.

[94] R. Scheichl, P. S. Vassilevski, and L. T. Zikatanov. "Multilevel methods for elliptic problems with highly varying coefficients on nonaligned coarse grids". In: *SIAM J. on Numer. Anal* 50.3 (2012), pp. 1675–1694.

[95] J. E. Schubert and B. F. Sanders. "Building treatments for urban flood inundation models and implications for predictive skill and modeling efficiency". In: *Advances in Water Resources* 41 (2012), pp. 49–64.

[96] H. A. Schwarz. *Ueber einen Grenzübergang durch alternirendes Verfahren*. Zürcher u. Furrer, 1870.

[97] J. R. Shewchuk. "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator". In: *Workshop on applied computational geometry*. Springer. 1996, pp. 203–222.

[98] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. "Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps". In: *Numer. Math.* 126.4 (2014), pp. 741–770.

[99] D. Spiridonov, M. Vasilyeva, and W. T. Leung. "A Generalized Multiscale Finite Element Method (GMsFEM) for perforated domain flows with Robin boundary conditions". In: *Journal of Computational and Applied Mathematics* 357 (2019), pp. 319–328.

[100] A. Toselli and O. Widlund. *Domain decomposition methods-algorithms and theory*. Vol. 34. Springer Science & Business Media, 2004.

[101] A. Turner and N. Chanmeesri. "Shallow flow of water through non-submerged vegetation". In: *Agricultural Water Management* 8.4 (1984), pp. 375–385.

[102] H. A. Van der Vorst. *Iterative Krylov methods for large linear systems*. 13. Cambridge University Press, 2003.

[103] J. L. Vázquez. *The porous medium equation: mathematical theory*. Oxford University Press, 2007.

[104] C. B. Vreugdenhil. *Numerical methods for shallow-water flow*. Vol. 13. Springer Science & Business Media, 2013.

[105] H. F. Walker and P. Ni. "Anderson acceleration for fixed-point iterations". In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735.

[106] E. Weinan and B. Engquist. "The heterogeneous multiscale methods". In: *Communications in Mathematical Sciences* 1.1 (2003), pp. 87–132.

[107] F. M. White, C. Ng, and S. Saimek. *Fluid mechanics*. McGraw-Hill, cop., 2011.

[108] J. Xu and J. Zou. "Some nonoverlapping domain decomposition methods". In: *SIAM review* 40.4 (1998), pp. 857–914.

[109] Y. Yang, M. Ghasemi, E. Gildin, Y. Efendiev, and V. Calo. "Fast multiscale reservoir simulations with POD-DEIM model reduction". In: *SPE Journal* 21.06 (2016), pp. 2141–2154.

[110] D. Yu and S. N. Lane. "Urban fluvial flood modelling using a two-dimensional diffusion-wave treatment, part 1: mesh resolution effects". In: *Hydrological Processes: An International Journal* 20.7 (2006), pp. 1541–1565.

[111] D. Yu and S. N. Lane. "Urban fluvial flood modelling using a two-dimensional diffusion-wave treatment, part 2: development of a sub-grid-scale treatment". In: *Hydrological Processes: An International Journal* 20.7 (2006), pp. 1567–1583.