

Investigating the Boundary Element Method for the Helmholtz Equation in Fractal Domains

Fachhochschule Aachen, Campus Jülich
Fachbereich 9 Medizintechnik und Technomathematik
Studiengang Angewandte Mathematik und Informatik

Masterarbeit von
Maria Eleousia Widhi Anugerah
Matrikelnummer: 3176832

Jülich, August 2026

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Masterarbeit mit dem Thema

Investigating the Boundary Element Method for the Helmholtz Equation in Fractal
Domains

selbstständig angefertigt und verfasst habe. Es sind keine anderen als die angegebenen
Quellen und Hilfsmittel benutzt worden

Jülich, den 26. August 2026



Maria Eleousia Widhi Anugerah

Diese Arbeit wurde betreut von:

1. Prüfer: Prof. Dr. rer. nat. habil. Andreas Kleefeld (Jülich Supercomputing Centre, Forschungszentrum Jülich and Fachhochschule Aachen)
2. Prüfer: Prof. Dr. rer. nat. Matthias Grajewski (Fachhochschule Aachen)

Contents

Abstract	iii
1 Introduction	1
2 Theoretical Backgrounds	3
2.1 Helmholtz and Laplace Equations	3
2.2 Classifications of the Boundary Value Problem	5
2.3 Acoustic Scattering Problems	7
2.4 Boundary Properties	8
2.5 Potential Operators and Jump Conditions	11
2.6 Boundary Integral Equations (BIEs)	13
2.7 Lagrange Basis Functions	14
2.7.1 Lagrange Basis Functions in One-Dimension	15
2.7.2 Boundary Discretization via Piecewise Polynomial Mapping	16
2.7.3 Basis Function Expansion for the Auxiliary Density	19
2.8 The Collocation Method	23
2.9 The Galerkin Method	28
2.10 Convergence Rate	31
2.10.1 General Convergence Rate of Projection Methods	32
2.10.2 Convergence Rate of the Collocation Method	35
2.10.3 Convergence Rate of the Galerkin Method	37
3 Development of the Galerkin Solver Framework	40
3.1 Software Architecture of the Galerkin Solver Framework	40
3.2 Parametric Boundary Representation	42
3.3 Verification using Point Source Incident Fields as True Solutions	49
3.3.1 Methodology	49
3.3.2 Results and Discussions	54
3.4 Verification of Plane Incident Fields Scattering by Unit Discs	65
3.4.1 Methodology	65
3.4.2 Results and Discussions	68
3.5 Verification of Plane Wave Incident Fields Scattering by Pre-Fractal Koch Snowflakes	79
3.5.1 Methodology	80
3.5.2 Results and Discussions	81
3.6 Parametric Analysis of Segment Refinement and Fractal Iteration on So- lution Accuracy in Koch Snowflake	101
3.6.1 Methodology	101

3.6.2	Results and Discussions	102
4	High-Performance Computing (HPC) Integration	108
4.1	Designing the Parallelization Strategy	108
4.1.1	Matrix Free Architecture: Methodology	109
4.1.2	Matrix Free Architecture: Results and Discussions	110
4.1.3	Precomputed Explicit Matrix Architecture: Methodology	111
4.1.4	Precomputed Explicit Matrix Architecture: Results and Discussions	114
4.2	Performance Evaluation of Numerical Integration Routines	117
4.2.1	Methodology	117
4.2.2	Results and Discussions	120
5	Numerical Simulations of Acoustic Scattering on a Higher-Iteration Fractal Domain	122
5.1	Methodology	122
5.2	Results and Discussions	125
6	Conclusion and Outlooks	129

Abstract

Resolving boundary value problems, such as the Helmholtz acoustic scattering problem, on complex, non-smooth geometries introduces numerical and computational challenges. While conventional numerical methods necessitate a full discretization of the domain, the Boundary Element Method (BEM) discretizes only the boundary of the domain. Popular variants of the BEM include the collocation and the Galerkin methods. The collocation method utilizes pointwise enforcements of the boundary condition (strong formulation). Although this approach yields highly accurate results on smooth domains, its precision degrades on non-smooth domains due to the geometric singularities.

To overcome the limitations, the Galerkin method accounts the entire boundary by utilizing a weak formulation. Consequently, the primary objective of this project is to implement a Galerkin solver framework and resolve the Helmholtz acoustic scattering problem up to a fifth-iteration Koch snowflake geometry.

This thesis details the implementation, verification, and optimization phases executed to realize a robust Galerkin solver framework utilizing a piecewise polynomial approximation. The inherent numerical challenges arising from kernel singularities during integral evaluation are resolved through a reformulation of the boundary integral expressions combined with a strategic boundary parameterization scheme. The associated computational challenges are mitigated by isolating the optimal discretization parameters that balance numerical accuracy with computational costs, deploying a multi-node parallelization strategy, and systematically evaluating various adaptive numerical quadrature routines.

The results demonstrate that pairing optimized parameters with distributed HPC pipelines makes highly complex fractal scattering simulations computationally tractable.

1 Introduction

Many problems in physics and engineering, such as acoustic scattering, fluid dynamics, and electrostatics, require the solution of the Partial Differential Equations (PDEs) [3, 14]. In homogeneous domains composed of a single material, the speed of light is constant. Thus, the coefficients of the governing PDEs remain constant and the PDEs can be reformulated as Boundary Integral Equations (BIEs). Notable examples include the Laplace and Helmholtz equations.

A significant advantage of the Boundary Element Method (BEM) over classical numerical methods—such as finite difference, finite element, or finite volume methods—is that only the boundary of the domain needs to be discretized. This reduction in dimensionality leads to lower computational costs for problems with complex or unbounded domains. In a typical BEM formulation, the BIE contains an expression involving the boundary integral operators on the left-hand side and the given boundary data on the right-hand side.

The focus of this work lies on the indirect method. This method introduces an unknown auxiliary density function defined only on the boundary. Once this function is determined, the solution of the BIE is recovered via a potential representation. The popular options from the indirect method are the collocation, or the Galerkin method [1]. These methods belong to the class of projection methods, but differ in how they enforce the equivalence between the left- and right-hand sides of the BIE.

The goal of this thesis project is twofold: to implement a modular and computationally robust Galerkin framework as an alternative for solving BIEs, and to systematically address the challenges associated with non-smooth domains. Because geometric singularities can severely degrade solution accuracy, this research focuses on the numerical simulation of acoustic scattering within complex pre-fractal boundaries. Specifically, the framework is engineered to scale up to the fifth iteration of the Koch snowflake geometry. This target domain introduces exceptional challenges, coupling the mathematical complexity of thousands of sharp corner singularities with a massive computational workload that requires scalable parallel implementation strategies to remain tractable.

The remainder of this thesis is structured into the following core chapters:

- **Chapter 2 (Theoretical Backgrounds)** details the fundamental mathematical theory of the framework. The section begins with the governing Laplace and Helmholtz equations, boundary value classifications, and their applications to

acoustic scattering problems. It outlines essential boundary properties, potential operators, jump conditions, and the resulting boundary integral equations. The chapter also introduces the discrete numerical approximation space formed by the Lagrange basis functions and their usage for the collocation and Galerkin methods. The chapter concludes by deriving the convergence rate of both methods.

- **Chapter 3 (Development of the Galerkin Solver Framework)** outlines the software architecture of the Galerkin solver framework and describes the parameterization of different boundary geometries. It details a rigorous verification process featuring point source wave scattering on four distinct boundaries, plane wave scattering on unit discs, and plane wave scattering on Koch snowflake boundaries. The chapter concludes with an analysis of the Koch snowflake boundary parameters to identify a strategic configuration that optimizes the trade-off between numerical accuracy and computational cost.
- **Chapter 4 (High Performance Computing (HPC) Integration)** examines the computational demands of high-iteration fractal geometries and the corresponding acceleration strategies. It outlines the implementation and performance evaluation of distributed-memory parallelization strategies tailored for HPC systems. Additionally, the chapter investigates various numerical quadrature routines to minimize execution overhead during the matrix assembly.
- **Chapter 5 (Numerical Simulation of Acoustic Scattering on a Higher-Iteration Fractal Domain)** presents the experimental setups used to solve the acoustic scattering problem up to the fifth-iteration Koch snowflake, providing an analysis of the resulting wave propagation fields.
- **Chapter 6 (Summary and Outlooks)** provides conclusion for the results and future improvements for the project.

2 Theoretical Backgrounds

This chapter establishes the mathematical and theoretical foundations necessary for the numerical treatment of the Helmholtz equation. The materials are organized as follows:

- Chapter 2.1 introduces the Helmholtz and the Laplace equations, which form the governing PDEs of the problems under consideration.
- Chapter 2.2 provides a systematic classification of the boundary value problems that arise when the PDEs are coupled with appropriate boundary conditions.
- Chapter 2.3 addresses the specific context of acoustic scattering, where the boundary value problem is formulated for this application.
- Chapter 2.4 examines the geometric and analytical properties of the boundaries.
- Chapter 2.5 introduces potential operators and their associated jump conditions to represent the solution of the boundary value problem.
- Chapter 2.6 derives and classifies the BIEs that emerge depending on the choice of potential operator and boundary condition.
- Chapter 2.7 presents the Lagrange basis functions employed for numerical approximation.
- Chapter 2.8 explains the numerical algorithm of the collocation method.
- Chapter 2.9 explains the numerical implementation of the Galerkin method.
- Chapter 2.10 derives the convergence rate of the collocation- and Galerkin method.

2.1 Helmholtz and Laplace Equations

This chapter presents the physical applications, the mathematical definition, and the fundamental solution for the Helmholtz equation. Subsequently, it addresses the Laplace equation, which emerges as a special case of the Helmholtz equation.

The Helmholtz equation describes the behavior of an acoustic wave [3]. It is also referred

to as the reduced wave equation [8]. A common application of the Helmholtz equation is found in the acoustic scattering problem, where it models how an incident acoustic wave is scattered by an object [3].

The Helmholtz equation in a spatial domain $\mathcal{D} \subset \mathbb{R}^2$ is given by

$$\Delta u + \kappa^2 u = 0, \quad (1)$$

where u denotes the complex-valued field, Δ is the Laplace operator, and $\kappa \in \mathbb{C}$ with $\text{Im}(\kappa) \geq 0$ is the wave number. The wave number κ is defined in terms of the angular frequency ω and the speed of sound c in the medium as

$$\kappa = \frac{\omega}{c}. \quad (2)$$

In two spatial dimensions, the Laplace Operator Δ is given by

$$\Delta := \partial_{x_1 x_1} + \partial_{x_2 x_2}, \quad (3)$$

and it acts on a sufficiently smooth scalar functions.

The fundamental solution for the Helmholtz equation in two dimensions is described by

$$\Phi_\kappa(\mathbf{x}, \mathbf{y}) = \frac{i}{4} H_0^{(1)}(\kappa \|\mathbf{x} - \mathbf{y}\|), \quad (4)$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2, \mathbf{x} \neq \mathbf{y}$ and $H_0^{(1)}$ is the first-kind Hankel function of order zero. The norm $\|\cdot\|$ denotes the Euclidean norm. This function satisfies the Helmholtz equation in (1) for all $\kappa \in \mathbb{C}$. However, there are many more functions that satisfy the Helmholtz equation for all $\kappa \in \mathbb{C}$.

The normal derivative of the fundamental solution from the Helmholtz equation is given by

$$\partial_{\nu(\mathbf{y})} \Phi_\kappa(\mathbf{x}, \mathbf{y}) = \frac{i\kappa}{4 \|\mathbf{x} - \mathbf{y}\|} H_1^{(1)}(\kappa \|\mathbf{x} - \mathbf{y}\|) \langle \mathbf{x} - \mathbf{y}, \nu(\mathbf{y}) \rangle, \quad (5)$$

where $H_1^{(1)}$ denotes the first-kind Hankel function of order one. For future applications in this work, $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^2$ and $\nu(\mathbf{y})$ denotes the exterior normal at a point $\mathbf{y} \in \mathbb{R}^2$ as described in Equation (22).

The Helmholtz equation reduces to the Laplace equation when $\kappa = 0$:

$$\Delta u = 0. \quad (6)$$

The Laplace equation is a second-order PDE, which plays a fundamental role in various physics fields, such as the theory of gravitation, electrostatics, dielectrics, and magneto-statics [14].

The fundamental solution of the Laplace equation in two dimensions is given by

$$\Phi_0(\mathbf{x}, \mathbf{y}) = -\frac{\log(\|\mathbf{x} - \mathbf{y}\|)}{2\pi}, \quad (7)$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2, \mathbf{x} \neq \mathbf{y}$. This function satisfies the Laplace equation described in (6). However, similar to the case of the Helmholtz equation, there are many more functions that satisfy the Laplace equation.

The normal derivative of this fundamental solution can be described by

$$\partial_{\nu(\mathbf{y})}\Phi_0(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi \|\mathbf{x} - \mathbf{y}\|^2} \langle \mathbf{x} - \mathbf{y}, \nu(\mathbf{y}) \rangle. \quad (8)$$

2.2 Classifications of the Boundary Value Problem

As established in Chapter 2.1, the Helmholtz and Laplace equation have an infinite family of solutions. To restrict the physically relevant solution, appropriate boundary conditions must be specified. These conditions prescribe the structural behavior of the field along the boundary from either the interior or exterior perspective. Embedding these boundary constraints within the governing partial differential equation yields a boundary value problem. This chapter systematically classifies these boundary value problems based on the spatial domain of interest and the underlying type of boundary condition.

Boundary value problems such as the Laplace and Helmholtz equations are commonly classified into exterior and interior problems, depending on the domain of interest $\mathcal{D} \subset \mathbb{R}^2$. An interior domain $\mathcal{D} = D_i$ lies inside a closed boundary, while an exterior domain $\mathcal{D} = D_e := \mathbb{R}^2 \setminus \overline{D_i}$ lies outside a closed boundary.

In this work, the behavior of the boundary conditions is further categorized into the Dirichlet and Neumann problems. Given a boundary Γ and the boundary function $g \in C(\Gamma)$. The Dirichlet problem requires finding an unknown function $u \in C(\overline{\mathcal{D}}) \cap C^2(\mathcal{D})$

that satisfies the PDE with the following prescribed condition on the boundary:

$$u = g \quad \text{on } \Gamma. \quad (9)$$

Meanwhile, the boundary condition for the Neumann problem requires finding a solution $u \in C^1(\overline{\mathcal{D}}) \cap C^2(\mathcal{D})$ that satisfies

$$\partial_\nu u = g \quad \text{on } \Gamma. \quad (10)$$

Therefore, the interior Dirichlet problem for the Helmholtz equation can be written as

$$\begin{aligned} \Delta u + \kappa^2 u &= 0 \quad \text{in } D_i, \\ u &= g \quad \text{on } \Gamma. \end{aligned} \quad (11)$$

If $\kappa = 0$, then the equation becomes the interior Dirichlet problem for the Laplace equation.

The interior Neumann problem for the Helmholtz equation is described as

$$\begin{aligned} \Delta u + \kappa^2 u &= 0 \quad \text{in } D_i, \\ \partial_\nu u &= g \quad \text{on } \Gamma. \end{aligned} \quad (12)$$

If $\kappa = 0$, the problem becomes the interior Neumann problem for the Laplace equation. The interior Neumann problem for the Laplace equation has a unique solution, up to the addition of an arbitrary constant, if the following condition is satisfied, as explained in [12]:

$$\int_{\Gamma} g(\mathbf{x}) ds(\mathbf{x}) = 0. \quad (13)$$

The exterior problems have additional conditions, known as the Sommerfeld radiation condition, to specify the behavior of the solution at infinity. The exterior Dirichlet problem for the Helmholtz equation with the Sommerfeld radiation condition is defined as

$$\begin{aligned} \Delta u + \kappa^2 u &= 0 \quad \text{in } D_e, \\ u &= g \quad \text{on } \Gamma, \\ \lim_{r \rightarrow \infty} \sqrt{r} (\partial_r u - i\kappa u) &= 0 \end{aligned} \quad (14)$$

with $r = \|\mathbf{x}\|$, $\mathbf{x} \in D_e$. Meanwhile, the exterior Dirichlet problem for the Laplace

equation with the corresponding Sommerfeld radiation condition is defined as

$$\begin{aligned}\Delta u &= 0 \quad \text{in } D_e, \\ u &= g \quad \text{on } \Gamma, \\ \lim_{r \rightarrow \infty} \sup_{\|\mathbf{x}\| > r} |u(\mathbf{x})| &< \infty.\end{aligned}\tag{15}$$

The exterior Neumann problem for the Helmholtz equation with the Sommerfeld radiation condition is defined as

$$\begin{aligned}\Delta u + \kappa^2 u &= 0 \quad \text{in } D_e, \\ \partial_\nu u &= g \quad \text{on } \Gamma, \\ \lim_{r \rightarrow \infty} \sqrt{r} (\partial_r u - i\kappa u) &= 0.\end{aligned}\tag{16}$$

The exterior Neumann problem for the Laplace equation with the corresponding Sommerfeld radiation condition can be written as

$$\begin{aligned}\Delta u &= 0 \quad \text{in } D_e, \\ \partial_\nu u &= g \quad \text{on } \Gamma, \\ u(r \cos(\theta), r \sin(\theta)) &= \mathcal{O}\left(\frac{1}{r}\right), \\ \partial_r u(r \cos(\theta), r \sin(\theta)) &= \mathcal{O}\left(\frac{1}{r^2}\right).\end{aligned}\tag{17}$$

2.3 Acoustic Scattering Problems

As outlined briefly in Section 2.1, the Helmholtz equation provides the mathematical model for acoustic wave propagation. This section expands upon that foundation by formalizing the specific boundary value problems that arise in a key physical application: acoustic scattering.

The wave motion at point $\mathbf{x} \in \mathbb{R}^2$ and time $t \in \mathbb{R}$ can be described using the velocity potential $U = U(\mathbf{x}, t)$. The velocity field $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ of the wave can be obtained from

$$\mathbf{v} = \frac{1}{\rho} \nabla U,\tag{18}$$

where $\rho = \rho(\mathbf{x}, t)$ denotes the density [6]. Meanwhile, the pressure $P = P(\mathbf{x}, t)$ of the

wave can be derived from the following equation, as described in [6]:

$$P = -\frac{\partial U}{\partial t}. \quad (19)$$

For time-harmonic acoustic waves of the form $U(\mathbf{x}, t) = \text{Re}\{u(\mathbf{x}) \exp(-i\omega t)\}$ with frequency $\omega > 0$, the space dependent part u satisfies the Helmholtz equation in (1). Therefore, the mathematical description of the time-harmonic acoustic waves scattering by an obstacle leads to boundary-value problems for the Helmholtz equation. For an incident wave $u^{\text{inc}} = u^{\text{inc}}(\mathbf{x})$ and a scattering wave $u^{\text{sca}} = u^{\text{sca}}(\mathbf{x})$, the total acoustic wave $u = u(\mathbf{x})$ is of the form $u = u^{\text{inc}} + u^{\text{sca}}$.

Prescribing the scattered wave $u^{\text{sca}} = -u^{\text{inc}}$ on the boundary of the obstacle (i.e., the Dirichlet problem) means that the total pressure must vanish on the boundary [3]. This leads to the acoustic scattering problem for a sound-soft obstacle

$$\begin{aligned} \Delta u^{\text{sca}} + \kappa^2 u^{\text{sca}} &= 0 \quad \text{in } D_e, \\ u^{\text{sca}} &= -u^{\text{inc}} \quad \text{on } \Gamma, \\ \lim_{r \rightarrow \infty} \sqrt{r} (\partial_r u^{\text{sca}} - i\kappa u^{\text{sca}}) &= 0. \end{aligned} \quad (20)$$

The Sommerfeld radiation condition $\lim_{r \rightarrow \infty} \sqrt{r} (\partial_r u^{\text{sca}} - i\kappa u^{\text{sca}}) = 0$ holds uniformly in all directions $\frac{\mathbf{x}}{\|\mathbf{x}\|}$ with $r = \|\mathbf{x}\|$.

Similarly, prescribing $\partial_\nu u^{\text{sca}} = -\partial_\nu u^{\text{inc}}$ on the boundary with the Neumann boundary condition physically corresponds to prescribing the normal component of the velocity of the acoustic wave [3]. This leads to the acoustic scattering problem for a sound-hard obstacle

$$\begin{aligned} \Delta u^{\text{sca}} + \kappa^2 u^{\text{sca}} &= 0 \quad \text{in } D_e, \\ \partial_\nu u^{\text{sca}} &= -\partial_\nu u^{\text{inc}} \quad \text{on } \Gamma, \\ \lim_{r \rightarrow \infty} \sqrt{r} (\partial_r u^{\text{sca}} - i\kappa u^{\text{sca}}) &= 0. \end{aligned} \quad (21)$$

2.4 Boundary Properties

This chapter discusses the geometric and topological properties of boundaries Γ within the context of boundary integral formulations. It begins with the definition of the exterior unit normal vector, followed by a formal categorization of boundary shapes and an analysis of their respective geometric characteristics. Furthermore, this chapter

details the algorithmic construction of a non-smooth fractal domain, specifically focusing on the classical Koch snowflake.

Consider Γ given in a parametric form $m(t) = (m_1(t), m_2(t))^\top$, $t \in [t_1, t_n]$. The exterior normal $\nu(\mathbf{x})$ at point $\mathbf{x} \in \Gamma$ is given by

$$\nu(\mathbf{x}) = \nu(m(t)) = \nu(t) = \frac{1}{\|m'(t)\|} (m'_2(t), -m'_1(t))^\top. \quad (22)$$

This work categorizes Γ into smooth curves and polygonal chains.

The smooth curves follow the following assumptions:

- The curve Γ is a simple-closed curve, i.e., $m(t_1) = m(t_n)$, and $m(t_i) \neq m(t_j)$ for all $t_i \neq t_j$ in (t_1, t_n) , ensuring no self-intersections.
- The parameterization is twice continuously differentiable, i.e., $m(t) \in C^2([t_1, t_n])$ and regular, i.e., $\|m'(t)\| \neq 0$ for all $t \in [t_1, t_n]$.
- The curve Γ is traversed in counter-clockwise direction, which is consistent with the standard orientation of positively oriented boundaries in $\mathbb{R}^2$.
- The curve Γ is of finite arc length, ensuring that integration over the boundary is well-defined.

Examples of smooth curves are ellipses defined with the following parameterization for $t \in [0, 2\pi]$

$$m(t) = (a \cos(t), b \sin(t))^\top, \quad (23)$$

and kites with the following parameterization for $t \in [0, 2\pi]$

$$m(t) = (\cos(t) + b \cos(2t) + c, -a \sin(t))^\top, \quad (24)$$

where $a, b, c \in \mathbb{R}$ are fixed constants.

Polygonal chains, however, do not satisfy all assumptions and exhibit geometric singularities at their sharp corners. Examples of polygonal chains are rhombus and the fractal Koch snowflakes.

The Koch snowflake is constructed iteratively based on the underlying Koch curve [7]. The initial configuration, or iteration zero, is defined as an equilateral triangle. In each

subsequent step, every linear segment of the boundary is trisected into three subintervals of equal length. Another smaller equilateral triangle is then constructed using the central subinterval as its base, which is subsequently removed to form the next iteration of the snowflake geometry. These steps are illustrated in Figure 1. Figure 2 shows Koch snowflakes from different Koch iterations.

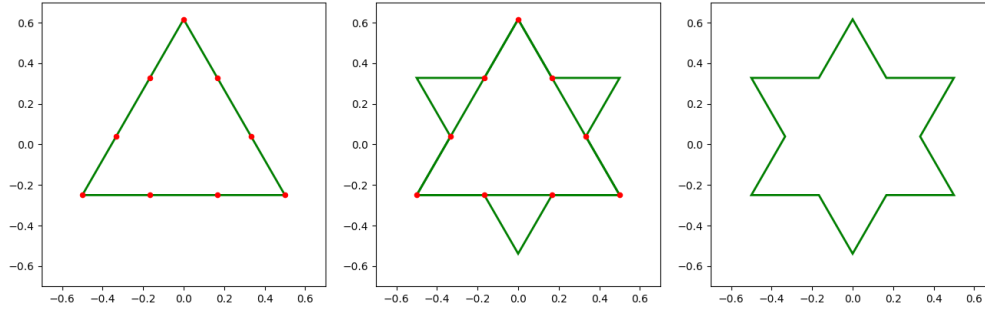


Figure 1: Iterative construction of the Koch snowflake. Left: Initial configuration (iteration 0) consisting of an equilateral triangle, which is trisected into three subintervals. Middle: Outward-pointing equilateral triangle emerges from the middle subinterval in each segment. Right: Removal of the base segment to complete the first iteration. This refinement process is repeated recursively for subsequent generations.

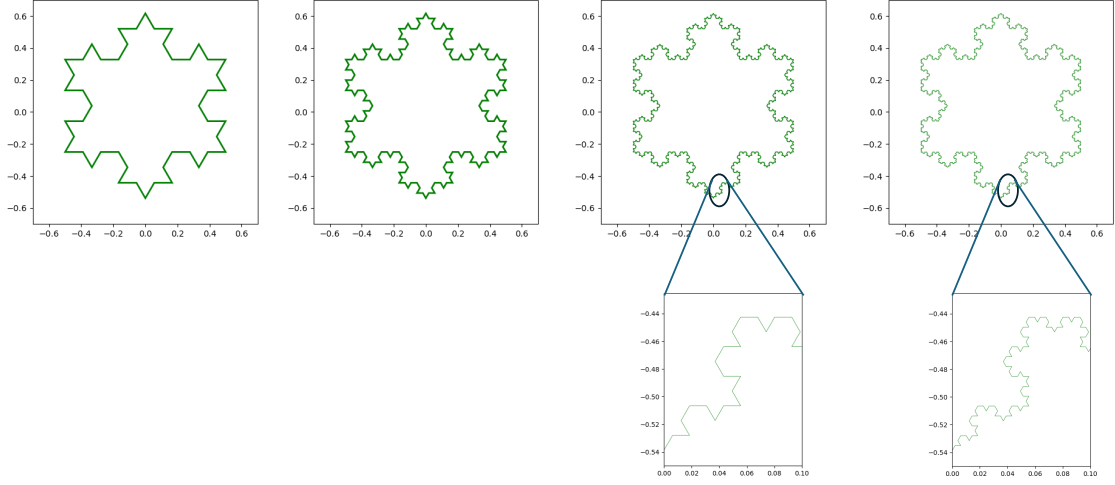


Figure 2: Geometric evolution of the Koch snowflake domain. Top row (left to right): Pre-fractal boundaries generated at iterations $i = 2, 3, 4$, and 5 . Bottom row (left to right): Magnified views of the fourth and fifth iterations, highlighting the microstructural geometric features across the refined sub-elements.

2.5 Potential Operators and Jump Conditions

The initial step in solving a boundary value problem via the BEM consists of selecting a potential operator to represent the domain solution u . This chapter details the formal definitions of various potential operators and characterizes their mathematical behavior when evaluated directly on the boundary, explicitly deriving their associated jump conditions.

The solution u to the Laplace or Helmholtz equation can be represented using either the single-layer potential, the double-layer potential, or a linear combination of both [3]. These representations differ fundamentally in their choice of kernel functions $K(\mathbf{x}, \mathbf{y})$, as formulated below:

$$u(\mathbf{x}) = \int_{\Gamma} \Psi(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) ds(\mathbf{y}), \quad \mathbf{x} \in \mathcal{D}. \quad (25)$$

This work focuses on the indirect method, where an unknown auxiliary density function Ψ is introduced. This method requires solving the BIE for Ψ first, which is subsequently used to evaluate u via the selected potential operator. The definition of the BIE will be

further explained in Section 2.6.

The representation of u with the single-layer potential is defined by

$$u(\mathbf{x}) = \text{SL}_\kappa[\Psi](\mathbf{x}) = \int_{\Gamma} \Psi(\mathbf{y}) \Phi_\kappa(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \mathbb{R}^2 \setminus \Gamma, \quad (26)$$

where $\Phi_\kappa(\mathbf{x}, \mathbf{y})$ denotes the fundamental solution of the Helmholtz equation if $\kappa \neq 0$ or the Laplace equation if $\kappa = 0$.

The representation of u with the double-layer potential is defined by

$$u(\mathbf{x}) = \text{DL}_\kappa[\Psi](\mathbf{x}) = \int_{\Gamma} \Psi(\mathbf{y}) \partial_{\nu(\mathbf{y})} \Phi_\kappa(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \mathbb{R}^2 \setminus \Gamma, \quad (27)$$

where $\partial_{\nu(\mathbf{y})} \Phi_\kappa(\mathbf{x}, \mathbf{y})$ denotes the normal derivative of the fundamental solution of the Helmholtz equation if $\kappa \neq 0$ or the Laplace equation if $\kappa = 0$.

Alternatively, the solution u can be represented as a linear combination of both potentials:

$$u(\mathbf{x}) = \text{DL}_\kappa[\Psi](\mathbf{x}) - i \text{SL}_\kappa[\Psi](\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^2 \setminus \Gamma. \quad (28)$$

The existence and bounded invertibility of the resulting integral operator, obtained after applying the jump relations, can be found in [10, Theorem 2.1.13].

Their behavior across the boundary is characterized by jump conditions [1]. Taking $\mathbf{x} \in \Gamma$, the function becomes $\omega_+(\mathbf{x})$ or $\omega_-(\mathbf{x})$, depending on the limit when $\mathbf{x}$ is approaching from the exterior and interior of the domain, respectively

$$\omega_\pm(\mathbf{x}) = \lim_{h \rightarrow 0^+} \omega(\mathbf{x} \pm h\nu(\mathbf{x})). \quad (29)$$

The function ω is continuous and continuously differentiable as the following function

$$\partial_{\nu(\mathbf{x})} \omega_\pm(\mathbf{x}) = \lim_{h \rightarrow 0^+} \langle \nu(\mathbf{x}), \nabla \omega(\mathbf{x} \pm h\nu(\mathbf{x})) \rangle. \quad (30)$$

Let $\omega(\mathbf{x}) = \text{SL}_\kappa[\Psi](\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^2 \setminus \Gamma$. When $\mathbf{x}$ is approaching the boundary from the

exterior or interior, the function becomes:

$$\begin{aligned}\omega_{\pm}(\mathbf{x}) &= S_{\kappa}[\Psi](\mathbf{x}) \\ &= \int_{\Gamma} \Psi(\mathbf{y}) \Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \Gamma.\end{aligned}\tag{31}$$

The operator S_{κ} denotes the boundary single-layer potential, and it has no jump on the boundary. However, its normal derivative exhibits a jump that introduces a new addition term with a factor $\mp \frac{1}{2}$:

$$\begin{aligned}\partial_{\nu(\mathbf{x})}\omega_{\pm}(\mathbf{x}) &= \mp \frac{1}{2}\Psi(\mathbf{x}) + D_{\kappa}^{\top}[\Psi](\mathbf{x}) \\ &= \mp \frac{1}{2}\Psi(\mathbf{x}) + \int_{\Gamma} \Psi(\mathbf{y}) \partial_{\nu(\mathbf{x})}\Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \Gamma.\end{aligned}\tag{32}$$

For the double layer potential $\omega(\mathbf{x}) = DL_{\kappa}[\Psi](\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^2 \setminus \Gamma$, the function $\omega_{+}(\mathbf{x})$ and $\omega_{-}(\mathbf{x})$ has a jump, when $\mathbf{x}$ approaches the boundary Γ from the exterior and interior of the domain:

$$\begin{aligned}\omega_{\pm}(\mathbf{x}) &= \pm \frac{1}{2}\Psi(\mathbf{x}) + D_{\kappa}[\Psi](\mathbf{x}) \\ &= \pm \frac{1}{2}\Psi(\mathbf{x}) + \int_{\Gamma} \Psi(\mathbf{y}) \partial_{\nu(\mathbf{y})}\Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \Gamma.\end{aligned}\tag{33}$$

The operator D_{κ} denotes the boundary double-layer potential. Meanwhile, its derivative, described by the operator N_{κ} , does not exhibit a jump:

$$\begin{aligned}\partial_{\nu(\mathbf{x})}\omega_{\pm}(\mathbf{x}) &= N_{\kappa}[\Psi](\mathbf{x}) \\ &= \partial_{\nu(\mathbf{x})} \int_{\Gamma} \Psi(\mathbf{y}) \partial_{\nu(\mathbf{y})}\Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}), \quad \mathbf{x} \in \Gamma.\end{aligned}\tag{34}$$

2.6 Boundary Integral Equations (BIEs)

As established in Section 2.5, the solution of a boundary value problem via the BEM begins by expressing u through an appropriate potential operator representation. Subsequently, the prescribed boundary condition (e.g., Dirichlet or Neumann) is imposed by taking the limit as the point $\mathbf{x}$ approaches the boundary. Applying the corresponding jump condition yields the BIE.

This chapter presents the formal definition of various types of BIEs and establishes their application framework based on the underlying boundary value problem. To illustrate

these concepts, a series of characteristic examples is provided, demonstrating how physical boundary conditions determine the choice of the integral formulation.

Depending on the potential operator representation, one distinguishes between BIEs of the first kind and BIEs of the second kind. A general BIE of the second kind has the form

$$\lambda \Psi(\mathbf{x}) + \int_{\Gamma} \Psi(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (35)$$

where $\lambda \in \mathbb{R} \setminus 0$ and in this context, λ is either $-\frac{1}{2}$ or $\frac{1}{2}$. If $\lambda = 0$, the equation reduces to a BIE of the first kind.

The mathematical nature of the resulting BIE depends on the jump relations of the chosen potential operator. If the limit of the operator yields a jump term, a BIE of the second kind is obtained. Otherwise, the formulation results in a BIE of the first kind.

For instance, consider the exterior Dirichlet problem with the boundary condition $u = g$ on Γ . Employing the single-layer potential ansatz $u = \text{SL}_{\kappa}[\Psi]$ yields a BIE of the first kind, as the single-layer potential is continuous across the boundary:

$$\text{S}_{\kappa}[\Psi](\mathbf{x}) = \int_{\Gamma} \Psi(\mathbf{y}) \Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (36)$$

Conversely, for the exterior Neumann problem ($\partial_{\nu} u = g$ on Γ), the same single-layer representation leads to a BIE of the second kind. This is due to the discontinuity in the normal derivative of the single-layer potential, which introduces a jump term:

$$-\frac{1}{2} \Psi(\mathbf{x}) + \text{D}_{\kappa}^{\top}[\Psi](\mathbf{x}) = -\frac{1}{2} \Psi(\mathbf{x}) + \int_{\Gamma} \Psi(\mathbf{y}) \partial_{\nu(\mathbf{x})} \Phi_{\kappa}(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (37)$$

2.7 Lagrange Basis Functions

As established in Section 2.5, the indirect boundary element method introduces an auxiliary density function Ψ . Computing the continuous domain solution u requires a two-step procedure: first, the governing BIE must be resolved for Ψ , which is subsequently utilized to evaluate the primary solution u . In practice, Ψ is discretized using a finite-dimensional basis expansion of the basis functions $\hat{l}_j$.

Furthermore, the potential operators require the evaluation of line integrals over the boundary Γ . These line integrals can be transformed into standard integrals over a

reference scalar interval by parameterizing the boundary curve Γ . To achieve this goal, the geometric representation of Γ can be approximated using a piecewise polynomial mapping with the same kind of basis functions, denoted by $\widehat{\ell}_k^{(t)}$.

Therefore, it can be concluded that this work employs the Lagrange basis functions for the approximation of the auxiliary function Ψ with $\widehat{l}_j$ and the boundary geometry Γ with $\widehat{\ell}_k^{(t)}$. The key advantage of using the Lagrange interpolation lies in its ability to provide an exact interpolation on a regular grid and the fact that it can preserve smoothness and continuity in the estimated function [4]. The main idea of this technique is to construct Ψ or Γ using piecewise polynomial functions based on a set of known data points.

Section 2.7.1 explains the Lagrange basis functions in one-dimension. Then, the approximation of Γ using discretization that includes these basis functions are discussed in Section 2.7.2. The approximation of Ψ with the basis expansion will be explained in Section 2.7.3.

2.7.1 Lagrange Basis Functions in One-Dimension

This chapter details the construction of the Lagrange basis functions over a reference interval for a prescribed number of degrees of freedom.

For one-dimensional interpolation, given $n + 1$ distinct nodes $s_0, \dots, s_n$ in the reference interval $\sigma = [0, 1]$ and the corresponding function values $f(s_0), \dots, f(s_n)$, there exists a unique polynomial $p(s)$ of degree at most n that satisfies the interpolation condition

$$p(s_t) = f(s_t), \quad t = 0, 1, \dots, n. \quad (38)$$

This polynomial can be expressed in the Lagrange form as

$$p(s) = \sum_{t=0}^n f(s_t) l^{(t)}(s), \quad (39)$$

where $l^{(t)}(s)$ denotes the t -th Lagrange basis polynomial satisfying the following cardinal property on the nodes s_j :

$$l^{(t)}(s_j) = \delta_{tj} = \begin{cases} 1, & t = j, \\ 0, & \text{else.} \end{cases} \quad (40)$$

The t -th Lagrange basis polynomials in one-dimension are explicitly defined by

$$\begin{aligned} l^{(t)}(s) &= \frac{(s - s_0) \cdots (s - s_{t-1})(s - s_{t+1}) \cdots (s - s_n)}{(s_t - s_0) \cdots (s_t - s_{t-1})(s_t - s_{t+1}) \cdots (s_t - s_n)} \\ &= \prod_{\substack{i=0 \\ i \neq t}}^n \frac{(s - s_i)}{(s_t - s_i)}. \end{aligned} \quad (41)$$

Depending on the number of interpolation nodes, this formulation yields the following standard approximations over σ with equidistant nodes:

- Constant approximation ($n = 0$, 1 node): Set at $s_0 = 1/2$, the basis functions reduce to

$$l^{(0)}(s) = 1. \quad (42)$$

- Linear approximation ($n = 1$, 2 nodes): Choosing the endpoints $s_0 = 0$ and $s_1 = 1$, the basis functions are given by

$$l^{(0)}(s) = 1 - s, \quad l^{(1)}(s) = s. \quad (43)$$

- Quadratic approximation ($n = 2$, 3 nodes): Utilizing the nodes $s_0 = 0$, $s_1 = 1/2$, and $s_2 = 1$, and defining $u = 1 - s$, the basis functions read

$$l^{(0)}(s) = u(1 - 2s), \quad l^{(1)}(s) = 4su, \quad l^{(2)}(s) = s(2s - 1). \quad (44)$$

2.7.2 Boundary Discretization via Piecewise Polynomial Mapping

This chapter details the derivation of the parametric approximation of the boundary Γ utilizing one-dimensional Lagrange basis functions.

To discretize the boundary Γ , it is subdivided into n_f segments, referred to as faces and denoted by Δ_k for $k = 1, \dots, n_f$. The geometric approximation of the boundary depends on a fixed number of boundary degrees of freedom (DOF), denoted by δ_Γ , which corresponds to $n + 1$ interpolation nodes per face element Δ_k . Thus, the corresponding polynomial degree is n .

In this work, it is fixed that $\delta_\Gamma = 3$, which implies a quadratic approximation (polynomial

degree $n = 2$) where each face is characterized by three distinct nodes: two vertices and one midpoint. Figure 3 illustrates the placement of these interpolation nodes across each boundary face.

Each node $\mathbf{p}_a^{(b)} \in \Gamma$, with face index $a \in \{1, \dots, n_f\}$ and local node index $b \in \{1, \dots, \delta_\Gamma\}$, is associated with a quadratic basis polynomial $\widehat{\ell}_k^{(t)}$, $k \in \{1, \dots, n_f\}$, $t \in \{1, \dots, \delta_\Gamma\}$ satisfying the cardinal property

$$\widehat{\ell}_k^{(t)}(\mathbf{p}_a^{(b)}) = \begin{cases} 1, & \text{if } a = k \text{ and } b = t, \\ 0, & \text{else.} \end{cases} \quad (45)$$

Furthermore, the localized basis function $\widehat{\ell}_k^{(t)}$ vanishes identically outside their supporting face Δ_k . This is the reason why this function is also known as the piecewise polynomial basis function.

For each face Δ_k , there exists a bijective map $m_k : \sigma = [0, 1] \rightarrow \Delta_k$ such that $\mathbf{p} = m_k(s)$ for $\mathbf{p} \in \Delta_k$ and $s \in \sigma$. The map m_k acts as a geometric deformation that takes a scalar value s from the reference interval σ and projects it onto the corresponding boundary face $\Delta_k \subset \Gamma$ as illustrated in Figure 4 and Figure 5. Because this parameterization maps from a one-dimensional scalar interval, the geometric approximation requires only one-dimensional Lagrange basis polynomials, avoiding the complexity of two-dimensional shape functions. Consequently, the mapping m_k can be approximated as

$$m_k(s) \approx \widetilde{m}_k(s) = \sum_{t=1}^{\delta_\Gamma} \mathbf{p}_k^{(t)} \ell_k^{(t)}(s), \quad s \in [0, 1]. \quad (46)$$

The function $\ell_k^{(t)}$ vanishes outside Δ_k , which can be mathematically defined as

$$\ell_k^{(t)} = \begin{cases} l^{(t)}, & \text{on } \Delta_k, \\ 0, & \text{else.} \end{cases} \quad (47)$$

The Jacobian function $\widetilde{J}_k(s)$ is defined as

$$\widetilde{J}_k(s) = \|\partial_s \widetilde{m}_k(s)\| = \sum_{t=1}^{\delta_\Gamma} \mathbf{p}_k^{(t)} \frac{d\ell_k^{(t)}(s)}{ds}. \quad (48)$$

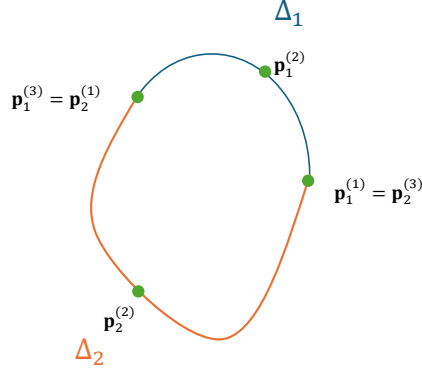


Figure 3: A boundary with two faces ($\delta_\Gamma = 3$). Each face has three nodes: two vertices and one midpoint. Note that a point (e.g. $\mathbf{p}_1^{(1)}$) can be a vertex in Δ_1 as well as in Δ_2 ($\mathbf{p}_2^{(3)}$).

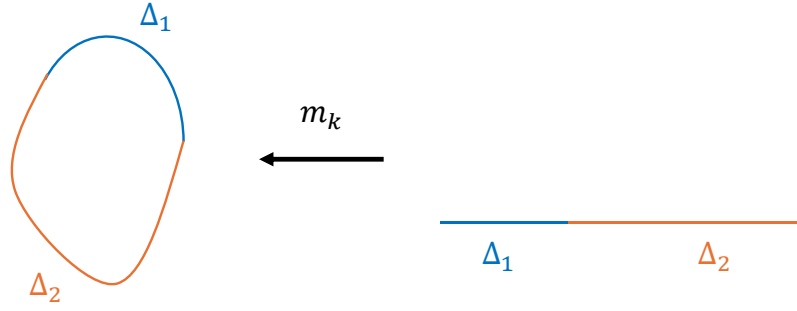


Figure 4: Geometric interpretation of the boundary parameterization. The bijective mapping $m_k, k \in \{1, 2\}$ maps the one-dimensional reference domain $\sigma = [0, 1]$ onto the physical boundary segment Δ_k . Conceptually, this process transforms a straight reference interval into the curved local elements that collectively form the global boundary Γ .

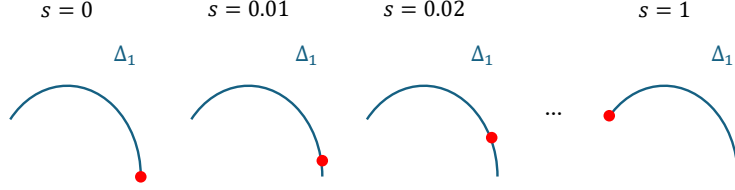


Figure 5: Local boundary parameterization via the mapping m_1 for distinct values of the reference coordinate $s \in \sigma$. Each reference point s is mapped to a specific position on the physical boundary face Δ_1 marked with the red dot. Notably, the parameter boundaries $s = 0$ and $s = 1$ correspond exactly to the initial and terminal vertices of the face, respectively.

2.7.3 Basis Function Expansion for the Auxiliary Density

Computing the domain solution u requires first solving the governing BIE for Ψ , which is subsequently used to evaluate u via the selected potential operator. This chapter details the derivation of the basis function expansion for Ψ by utilizing the one-dimensional Lagrange basis functions.

In practice, Ψ is approximated by a finite linear combination of global basis functions:

$$\Psi(\mathbf{x}) \approx \sum_{j=1}^d c_j \hat{l}_j(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (49)$$

where d denotes the dimension of the discrete approximation space.

To implement this discretization, the boundary Γ is partitioned into n_f faces, denoted by Δ_k for $k = 1, \dots, n_f$. Each face contributes a fixed number of DOF δ , yielding a total approximation space dimension of $d = n_f \delta$. Consequently, the global expansion of Ψ can be refactored into face-wise contributions:

$$\Psi(\mathbf{x}) \approx \sum_{j=1}^d c_j \hat{t}_j(\mathbf{x}) = \sum_{k=1}^{n_f} \sum_{t=1}^{\delta} c_k^{(t)} \hat{l}_k^{(t)}(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (50)$$

The global basis index j maps to the specific face index $k \in \{1, \dots, n_f\}$ via the ceiling function

$$k = k(j) = \left\lceil \frac{j}{\delta} \right\rceil \quad \text{where} \quad \lceil x \rceil = \min\{n \in \mathbb{Z} \mid n \geq x\}. \quad (51)$$

The corresponding local function shape index $t \in \{1, \dots, \delta\}$ is given by

$$t = t(j) = 1 + (j + \delta - 1 \bmod \delta). \quad (52)$$

Furthermore, since any point $\mathbf{x} \in \Delta_k$ can be parameterized via the bijective coordinate mapping $\mathbf{x} = m_k(s)$ for $s \in \sigma = [0, 1]$, the local basis expansion can be evaluated directly on the reference interval σ using the one-dimensional Lagrange basis polynomials $l_k^{(t)}$:

$$\Psi(m_k(s)) \approx \sum_{j=1}^{\delta} c_j l_j(s) = \sum_{k=1}^{n_f} \sum_{t=1}^{\delta} c_k^{(t)} l_k^{(t)}(s), \quad s \in \sigma, \quad (53)$$

where $l_k^{(t)}$ vanishes outside Δ_k as defined in

$$l_k^{(t)} = \begin{cases} l^{(t)}, & \text{on } \Delta_k, \\ 0, & \text{else.} \end{cases} \quad (54)$$

The corresponding Lagrange basis functions $l_j = l_k^{(t)}$ for $\delta \in \{1, 2, 3\}$ in each face are illustrated in Figure 6, 7, 8, respectively.

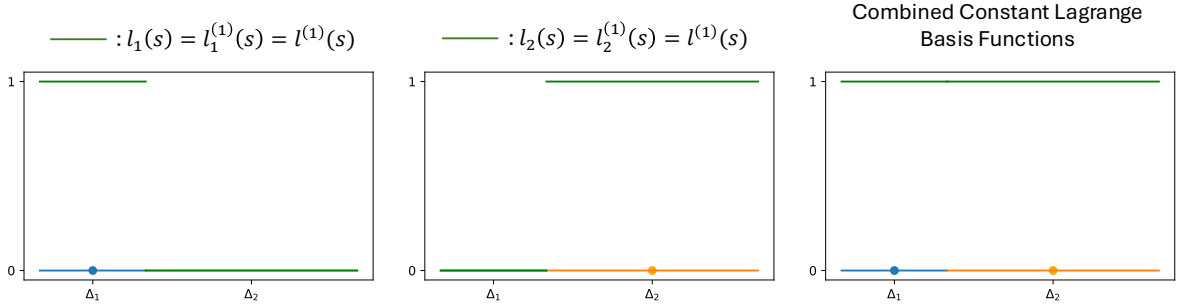


Figure 6: Global Lagrange basis functions with a single DOF ($\delta = 1$) per face across a two-faces mesh (Δ_1, Δ_2) indicated by the green line. The global function $l_1 = l_1^{(1)}$ is supported on Δ_1 , corresponding to the constant reference polynomial $l^{(1)}(s) = 1$, and vanishes identically on Δ_2 . Analogously, $l_2 = l_2^{(1)}$ is supported exclusively on Δ_2 with an identical local profile, vanishing on Δ_1 .

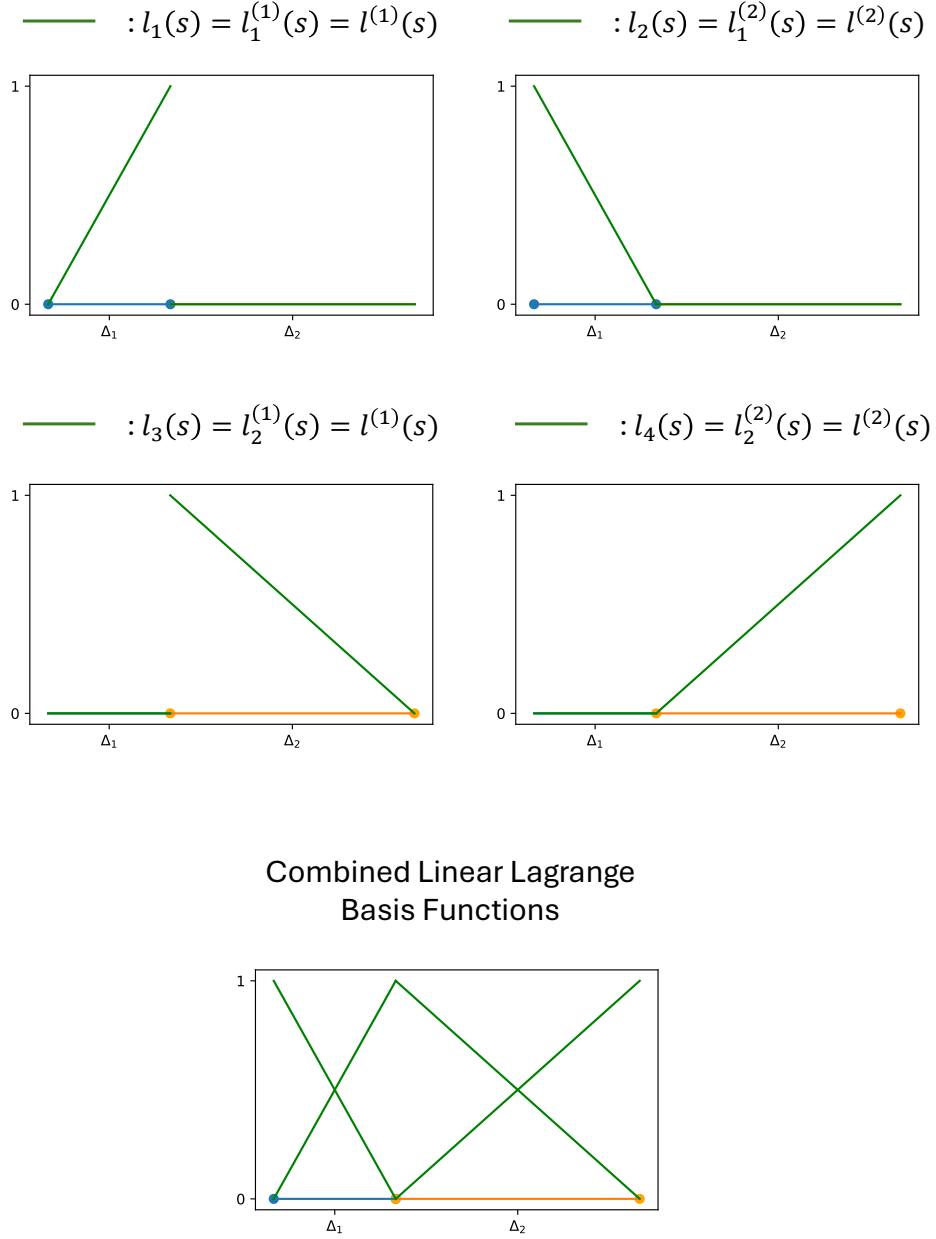


Figure 7: Global Lagrange basis functions with two DOF ($\delta = 2$) per face across a mesh of two faces (Δ_1, Δ_2). The global functions $l_1 = l_1^{(1)}$ and $l_2 = l_1^{(2)}$ are localized to Δ_1 , where they correspond to the reference polynomials $l^{(1)}(s) = 1 - s$ and $l^{(2)}(s) = s$, respectively, and vanish identically on Δ_2 . Analogously, the functions $l_3 = l_2^{(1)}$ and $l_4 = l_2^{(2)}$ are supported on Δ_2 with identical local profiles, vanishing on Δ_1 .

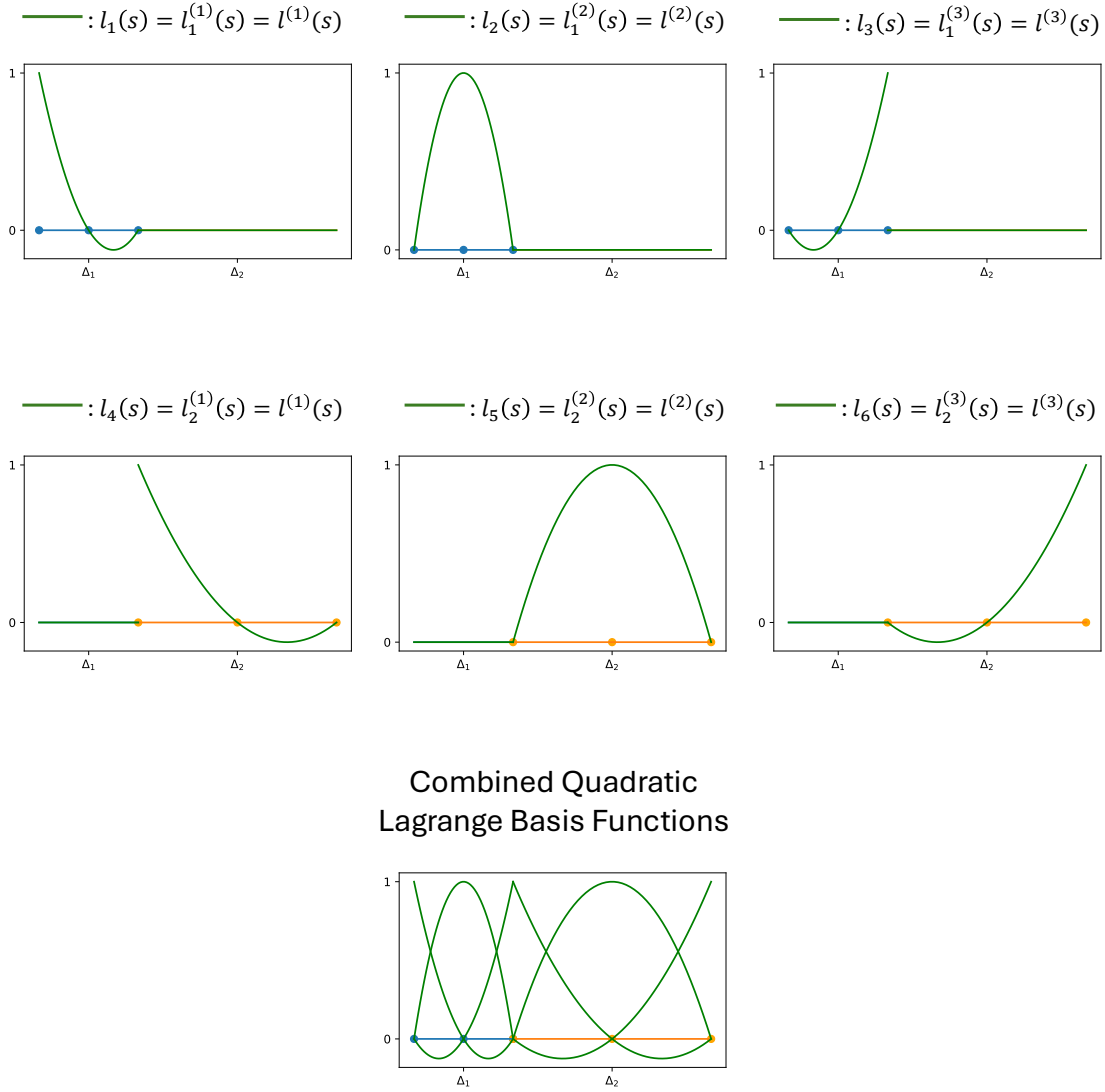


Figure 8: Global Lagrange basis functions with three DOF ($\delta = 3$) per face across a two-element mesh (Δ_1, Δ_2) . The global functions $l_1 = l_1^{(1)}$, $l_2 = l_1^{(2)}$, and $l_3 = l_1^{(3)}$ are localized to Δ_1 , where they correspond to the quadratic reference polynomials $l^{(1)}(s) = u(1 - 2s)$, $l^{(2)}(s) = 4su$, and $l^{(3)}(s) = s(2s - 1)$ (with $u = 1 - s$), respectively. These functions vanish identically on Δ_2 . Analogously, the functions $l_4 = l_2^{(1)}$, $l_5 = l_2^{(2)}$, and $l_6 = l_2^{(3)}$ are supported on Δ_2 with identical local profiles, vanishing on Δ_1 .

2.8 The Collocation Method

The collocation method is a widely used numerical technique for solving BIE. The main idea of this method is based on a strong formulation, where the BIE is enforced to hold exactly at a finite set of prescribed points on the boundary, known as collocation points. This chapter details the numerical approximation of the solution using the collocation method.

To define the collocation points, the boundary Γ is partitioned into n_f segments (referred to as faces), where each face is bounded by two vertices. The number of collocation points per face corresponds to the DOF, denoted by δ . Consequently, a mesh of n_f faces yields a total of $n_f\delta$ collocation points, which matches the dimension d of the discrete approximation space.

Figure 9 illustrates an exemplary distribution of these collocation points. In the collocation framework, the distribution of the collocation points along a given face is not constrained to include the vertices of a face; rather, their spatial positioning can be shifted in between the vertices. However, the specific selection of these nodes significantly influences the evaluation of the boundary integrals, as it directly affects the accuracy of the numerical quadrature and the conditioning of the resulting algebraic system.

Enforcing the second-kind BIE in Equation (35) to hold exactly at each collocation point $\mathbf{v}_i$ yields the discrete system

$$\lambda\Psi(\mathbf{v}_i) + \int_{\Gamma} \Psi(\mathbf{y})K(\mathbf{v}_i, \mathbf{y}) \, ds(\mathbf{y}) = g(\mathbf{v}_i), \quad i = 1, \dots, d. \quad (55)$$

The expansion coefficients c_j from Equation (49) interpolate the density at the collocation nodes, allowing the discrete ansatz to be written as

$$\Psi(\mathbf{y}) \approx \sum_{j=1}^d c_j \hat{l}_j(\mathbf{y}) = \sum_{j=1}^d \Psi(\mathbf{v}_j) \hat{l}_j(\mathbf{y}), \quad \mathbf{y} \in \Gamma. \quad (56)$$

Substituting the approximation in Equation (56) into Equation (55) gives

$$\lambda\Psi(\mathbf{v}_i) + \sum_{j=1}^d \Psi(\mathbf{v}_j) \int_{\Gamma} \hat{l}_j(\mathbf{y})K(\mathbf{v}_i, \mathbf{y}) \, ds(\mathbf{y}) = g(\mathbf{v}_i), \quad i = 1, \dots, d. \quad (57)$$

This relation can be compactly expressed as an algebraic system of linear equation

$$(\lambda \mathbf{I} + \mathbf{A})\mathbf{c} = \mathbf{g}, \quad (58)$$

where $\mathbf{I}$ denotes the identity matrix. The system matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$, the right-hand side vector $\mathbf{g} \in \mathbb{C}^d$, and the unknown coefficient vector $\mathbf{c} \in \mathbb{C}^d$ are defined element-wise by

$$\mathbf{A} = (a_{ij})_{i,j=1}^d, \quad a_{ij} = \int_{\Gamma} \hat{l}_j(\mathbf{y}) K(\mathbf{v}_i, \mathbf{y}) \, ds(\mathbf{y}), \quad (59)$$

$$\mathbf{g} = (g(\mathbf{v}_1), \dots, g(\mathbf{v}_d))^{\top}, \quad (60)$$

$$\mathbf{c} = (\Psi(\mathbf{v}_1), \dots, \Psi(\mathbf{v}_d))^{\top}. \quad (61)$$

To evaluate the boundary line integrals in (59), the localized support of the Lagrange basis functions detailed in Chapter 2.7 is utilized. Because the basis function $\hat{l}_j$ vanishes identically outside its supporting face $\Delta_k, k = k(j)$, the integration domain over the entire boundary Γ naturally restricts to a single face Δ_k . Furthermore, by applying the parametric map $\tilde{m}_k$, the physical line integral over Δ_k is pulled back to a standard one-dimensional integral over the reference domain $\sigma = [0, 1]$. Consequently, the matrix entries a_{ij} can be computed as

$$\begin{aligned} a_{ij} &= \int_{\Gamma} \hat{l}_j(\mathbf{y}) K(\mathbf{v}_i, \mathbf{y}) \, ds(\mathbf{y}) \\ &= \int_{\Delta_k} \hat{l}_j(\mathbf{y}) K(\mathbf{v}_i, \mathbf{y}) \, ds(\mathbf{y}) \\ &\approx \int_{\sigma} l_j(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) \, ds \\ &= \int_{\sigma} l^{(t)}(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) \, ds, \end{aligned} \quad (62)$$

where the face index k and local function shape index t are determined via the structural relations $k(j)$ and $t(j)$ defined in Equations (51) and (52), respectively.

Once the linear system (58) is resolved and the coefficient vector $\mathbf{c}$ is known, the potential solution $u(\mathbf{x})$ at any domain point $\mathbf{x} \in \mathcal{D}$ can be evaluated. Substituting the discrete

basis expansion into the potential representation yields the numerical approximation

$$\begin{aligned}
u(\mathbf{x}) &= \int_{\Gamma} \Psi(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \\
&\approx \int_{\Gamma} \left(\sum_{j=1}^d c_j \widehat{l}_j(\mathbf{y}) \right) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \\
&= \sum_{j=1}^d \left(c_j \int_{\Gamma} \widehat{l}_j(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \right) \\
&= \sum_{j=1}^d \left(c_j \int_{\Delta_k} \widehat{l}_k^{(t)}(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \right) \\
&\approx \sum_{j=1}^d \left(c_j \int_{\sigma} l^{(t)}(s) K(\mathbf{x}, \tilde{m}_k(s)) \tilde{J}_k(s) \, ds \right), \tag{63}
\end{aligned}$$

where the element mappings for k and t follow the same indexing convention as the system matrix by using Equations (51) and (52), respectively.

The collocation method can yield highly accurate results, provided that the right-hand side function $g(\mathbf{v}_i)$ of the pointwise enforcement in Equation (55) is well-defined and continuous at the chosen nodes $\mathbf{v}_i$. However, if the function exhibits discontinuities at $\mathbf{v}_i$, or if it incorporates the exterior unit normal vector along a non-smooth boundary with a geometric singularity (such as when evaluating the normal derivative for a Neumann boundary condition on a polygonal chain), the pointwise enforcement framework can encounter significant numerical difficulties. To mitigate these inherent structural limitations, one can utilize the specialized algorithmic techniques based on [6] detailed in the following paragraphs.

Consider a boundary face Δ_k and its associated local Lagrange basis functions $l_k^{(t)}(s) \equiv l^{(t)}(s)$, where $t \in \{1, \dots, \delta\}$ denotes the local function shape index. Typically, these classical basis functions are constructed using a uniform distribution of nodes across the standard reference interval $\sigma = [0, 1]$, as explained in Section 2.7.1. In the collocation framework, however, the collocation nodes are not restricted to the element vertices. They may be shifted in between the vertices, as illustrated in Figure 9. Specifically, their positions are parameterized within a symmetric sub-interval $[\alpha, 1 - \alpha]$, instead of the standard reference interval $\sigma = [0, 1]$. The parameter $\alpha \in (0, 0.5)$ dictates the inward offset from the vertices.

To ensure that the basis functions preserve their cardinal property shown in (40) at these modified node locations, a linear coordinate scaling is required. By mapping the shifted

interval $[\alpha, 1 - \alpha]$ back onto the standard unit interval $[0, 1]$, the modified basis functions $\tilde{l}_k^{(t)}(s)$ are defined as

$$\tilde{l}_k^{(t)} = \begin{cases} \tilde{l}^{(t)}, & \text{on } \Delta_k, \\ 0, & \text{else,} \end{cases} \quad (64)$$

where

$$\tilde{l}^{(t)}(s) = l^{(t)}\left(\frac{s - \alpha}{1 - 2\alpha}\right). \quad (65)$$

Figure 10 depicts the comparison when the collocation points coincide with the vertices and when they are shifted in between the vertices together with the corresponding modification of the Lagrange basis function. This transformation guarantees that the adjusted basis functions satisfy the standard Kronecker delta property at the chosen collocation positions:

$$\tilde{l}^{(t)}(s_j) = \delta_{tj} = \begin{cases} 1, & \text{if } t = j, \\ 0, & \text{if } t \neq j. \end{cases} \quad (66)$$

In this case, the matrix entries a_{ij} are assembled using the adjusted basis functions $\tilde{l}^{(t)}(s)$:

$$a_{ij} \approx \int_{\sigma} \tilde{l}^{(t)}(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) \, ds, \quad (67)$$

where the indices k and t follow Equations (51) and (52), respectively.

Accordingly, once the discrete density coefficients c_j are determined, the domain solution $u(\mathbf{x})$ is reconstructed by consistency using the same modified basis expansion over the reference interval:

$$u(\mathbf{x}) \approx \sum_{j=1}^d \left(c_j \int_{\sigma} \tilde{l}^{(t)}(s) K(\mathbf{x}, \tilde{m}_k(s)) \tilde{J}_k(s) \, ds \right), \quad (68)$$

where the indices k and t also follow Equations (51) and (52), respectively.

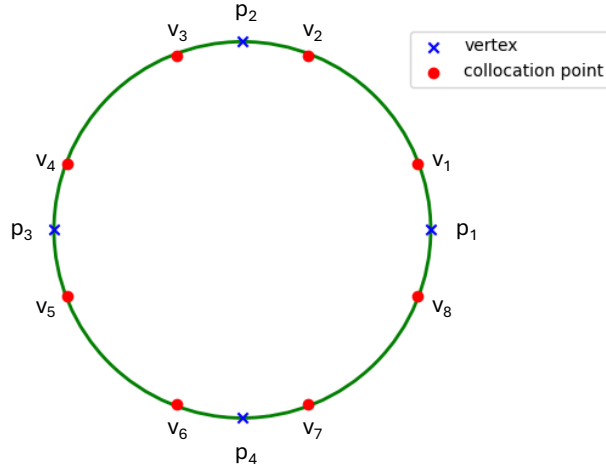


Figure 9: Exemplary distribution of collocation points on a circular boundary partitioned into four faces ($n_f = 4$). The vertices $\mathbf{p}_k$ delineate the face boundaries (e.g., The first face is bounded by $\mathbf{p}_1$ and $\mathbf{p}_2$). Each face contains $\delta = 2$ collocation points, yielding a total approximation space dimension of $d = 8$.

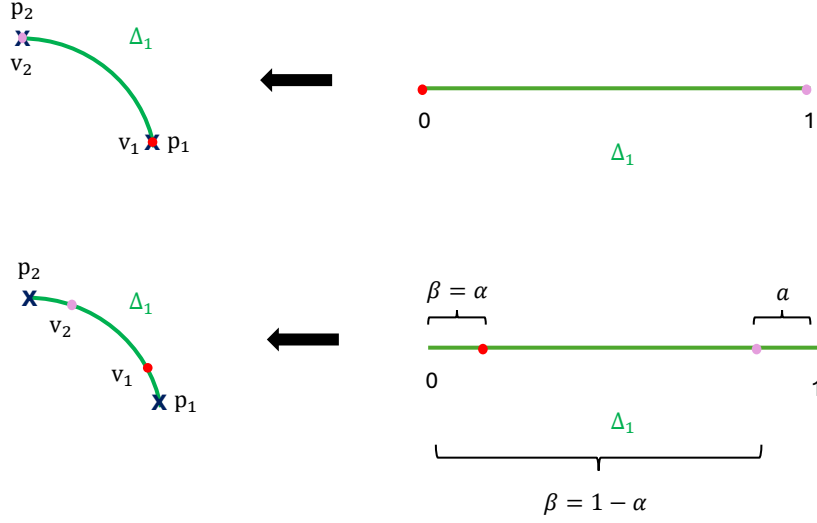


Figure 10: Comparison of collocation point placements on a single face ($\delta = 2$). **Top:** Standard configuration where the first collocation points $\mathbf{v}_1$ (red dot) and the second collocation point $\mathbf{v}_2$ (pink dot) coincide with the initial and terminal vertices ($\mathbf{p}_1$ and $\mathbf{p}_2$) of Δ_1 . The underlying Lagrange basis functions must satisfy the standard conditions $l_1^{(1)}(0) = 1$, $l_1^{(1)}(1) = 0$, $l_1^{(2)}(0) = 0$, and $l_1^{(2)}(1) = 1$, which are fulfilled by the linear polynomials in (43). **Bottom:** Shifted configuration where the collocation points are moved between the vertices, bounded by the sub-interval $[\alpha, 1 - \alpha]$. β denotes the distance of the collocation point from 0. The first collocation points $\mathbf{v}_1$ (red dot) has the distance $\beta = \alpha$, while $\mathbf{v}_2$ (pink dot) has the distance $\beta = 1 - \alpha$. To satisfy the modified interpolation conditions $l_1^{(1)}(\alpha) = 1$, $l_1^{(1)}(1 - \alpha) = 0$, $l_1^{(2)}(\alpha) = 0$, and $l_1^{(2)}(1 - \alpha) = 1$, the basis functions are adjusted via an affine scaling to read $\tilde{l}_1^{(t)}(s) = l_1^{(t)}\left(\frac{s - \alpha}{1 - 2\alpha}\right)$ for $t = 1, 2$.

2.9 The Galerkin Method

This chapter compares the collocation and the Galerkin method. Moreover, it discusses the numerical approximation of the solution using the Galerkin method.

While the collocation method enforces the BIE exclusively at discrete collocation points, the Galerkin method adopts a variational approach based on a weak formulation. Instead of requiring the BIE to hold pointwise, the Galerkin method projects the residual—the

difference between the left- and right-hand sides of the equation—orthogonally onto a finite-dimensional subspace. This weak formulation is constructed by taking the L^2 -inner product of the BIE against a chosen set of basis functions. This integral testing naturally buffers and mitigates the singular behavior induced by non-smooth boundary data. Consequently, the framework can successfully isolate a well-posed weak solution to the boundary value problem even when the continuous field is not sufficiently smooth. Furthermore, rather than restricting its attention to isolated coordinates—which can cause under-sampling of the boundary information—the Galerkin approach integrates the physical boundary conditions over the entire support of each element face.

Similar to the collocation scheme, the Galerkin method approximates Ψ within a finite-dimensional subspace by employing the discrete ansatz in Equation (49). While the expansion coefficients c_j in the collocation method depend on the value of Ψ on the collocation nodes, the coefficients c_j in the Galerkin method depend on the orthogonal projection

$$\Psi(\mathbf{y}) \approx \sum_{j=1}^d c_j \hat{l}_j(\mathbf{y}) = \sum_{j=1}^d (\Psi, \hat{l}_j) \hat{l}_j(\mathbf{y}). \quad (69)$$

In this work, $(\cdot, \cdot)$ denotes the L^2 -scalar product. In this case, it denotes the scalar product in $L^2(\Gamma)$:

$$(f, g) = \int_{\Gamma} f(\mathbf{x}) g(\mathbf{x}) \, ds(\mathbf{x}). \quad (70)$$

Equation (69) is substituted into the second-kind BIE to yield

$$\sum_{j=1}^d c_j \left[\lambda \hat{l}_j(\mathbf{x}) + \int_{\Gamma} \hat{l}_j(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \right] = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (71)$$

The boundary Γ is also partitioned into n_f faces, where each face contributes a fixed number of DOF δ . The DOF corresponds to the number of coefficients c_j per face. The total dimension of the approximation space is therefore $d = n_f \delta$.

Enforcing the residual to be orthogonal to a set of basis functions $\{\hat{l}_i\}_{i=1}^d$ yields

$$\sum_{j=1}^d c_j \left[\lambda (\hat{l}_i, \hat{l}_j) + \int_{\Gamma} \int_{\Gamma} \hat{l}_i(\mathbf{x}) \hat{l}_j(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \, ds(\mathbf{x}) \right] = (\hat{l}_i, g), \quad i = 1, \dots, d. \quad (72)$$

This relation can be expressed as the following algebraic system of linear equation

$$\mathbf{A}\mathbf{c} = \mathbf{g}, \quad (73)$$

where the system matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$, the right-hand side vector $\mathbf{g} \in \mathbb{C}^d$, and the unknown coefficient vector $\mathbf{c} \in \mathbb{C}^d$ are defined element-wise by

$$\mathbf{A} = (a_{ij})_{i,j=1}^d, \quad a_{ij} = \lambda \left(\widehat{l}_i, \widehat{l}_j \right) + \int_{\Gamma} \int_{\Gamma} \widehat{l}_i(\mathbf{x}) \widehat{l}_j(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) ds(\mathbf{y}) ds(\mathbf{x}), \quad (74)$$

$$\mathbf{g} = \left(\left(\widehat{l}_1, g \right), \dots, \left(\widehat{l}_d, g \right) \right)^{\top}, \quad (75)$$

$$\mathbf{c} = (c_1, \dots, c_d)^{\top}. \quad (76)$$

Since each $\widehat{l}_j$ vanishes outside $\Delta_{k(j)}$, the inner products $\left(\widehat{l}_i, \widehat{l}_j \right)$ need only be evaluated if they have the same supporting face $\Delta_{k(i)} = \Delta_{k(j)}$, where $k(i), k(j)$ denote the face index as described in (51). If $\left(\widehat{l}_i, \widehat{l}_j \right)$, $i, j = 1, \dots, d$ are being stored in a matrix $\widehat{L}$, then $\widehat{L}$ is a $d \times d$ block diagonal matrix, where each block matrix $\widehat{L}_h$, $h = 1, \dots, n_f$ is a symmetrical matrix of size $\delta \times \delta$

$$\widehat{L} = \begin{pmatrix} \widehat{L}_1 & 0 & \cdots & 0 \\ 0 & \widehat{L}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \widehat{L}_{n_f} \end{pmatrix} \in \mathbb{R}^{d \times d}, \quad (77)$$

$$\widehat{L}_h = \begin{pmatrix} \left(\widehat{l}_{(h-1)\delta+1}, \widehat{l}_{(h-1)\delta+1} \right) & \cdots & \left(\widehat{l}_{(h-1)\delta+1}, \widehat{l}_{h\delta} \right) \\ \vdots & \ddots & \vdots \\ \left(\widehat{l}_{h\delta}, \widehat{l}_{(h-1)\delta+1} \right) & \cdots & \left(\widehat{l}_{h\delta}, \widehat{l}_{h\delta} \right) \end{pmatrix} \in \mathbb{R}^{\delta \times \delta}. \quad (78)$$

To evaluate the L^2 -scalar product $\left(\widehat{l}_i, \widehat{l}_j \right)$ when two basis functions share the same supporting face ($\Delta_{k(i)} = \Delta_{k(j)}$), the parametric map $\widetilde{m}_{k(i)}$ can be employed. This pulls back the physical line integral over $\Delta_{k(i)}$ to a one-dimensional integral over the reference

domain $\sigma = [0, 1]$:

$$\begin{aligned}
(\widehat{l}_i, \widehat{l}_j) &= \int_{\Gamma} \widehat{l}_i(\mathbf{x}) \widehat{l}_j(\mathbf{x}) \, ds(\mathbf{x}) \\
&= \int_{\Delta_{k(i)}} \widehat{l}_{k(i)}^{(t(i))}(\mathbf{x}) \widehat{l}_{k(i)}^{(t(j))}(\mathbf{x}) \, ds(\mathbf{x}) \\
&= \int_{\sigma} l^{(t(i))}(s) l^{(t(j))}(s) \widetilde{J}_{k(i)}(s) \, ds,
\end{aligned} \tag{79}$$

where the face index $k(i) = k(j)$ and the local shape function indices $t(i), t(j)$ are defined in Equations (51) and (52), respectively.

Similarly, the double boundary integral term contributing to the Galerkin matrix entry a_{ij} in (74) can be simplified. Exploiting the compact support of the basis functions and applying the coordinate transformations $\widetilde{m}_{k(i)}$ and $\widetilde{m}_{k(j)}$, the double line integral is reduced to:

$$\begin{aligned}
&\int_{\Gamma} \int_{\Gamma} \widehat{l}_i(\mathbf{x}) \widehat{l}_j(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \, ds(\mathbf{x}) \\
&= \int_{\Delta_{k(i)}} \int_{\Delta_{k(j)}} \widehat{l}_{k(i)}^{(t(i))}(\mathbf{x}) \widehat{l}_{k(j)}^{(t(j))}(\mathbf{y}) K(\mathbf{x}, \mathbf{y}) \, ds(\mathbf{y}) \, ds(\mathbf{x}) \\
&= \int_{\sigma} \int_{\sigma} l^{(t(i))}(p) l^{(t(j))}(s) K(\widetilde{m}_{k(i)}(p), \widetilde{m}_{k(j)}(s)) \widetilde{J}_{k(i)}(p) \widetilde{J}_{k(j)}(s) \, ds \, dp,
\end{aligned} \tag{80}$$

where the mapping of indices $k(i), k(j)$ and $t(i), t(j)$ follows the identical structural conventions in Equations (51) and (52), respectively.

Once the global linear system (73) is resolved to yield the density coefficient vector $\mathbf{c}$, the potential solution $u(\mathbf{x})$ at any domain point $\mathbf{x} \in \mathcal{D}$ can be evaluated. This post-processing step is performed by substituting the discrete expansion into the chosen potential operators, completely analogous to the collocation approach detailed in (63).

2.10 Convergence Rate

This section analyzes the convergence properties of the collocation and the Galerkin projection methods. The convergence is derived based on the analysis developed in [1]. Section 2.10.1 first presents a general upper bound for the projection approximation error, while Section 2.10.2 evaluates the specific convergence characteristics of the collocation method. Utilizing these theoretical groundworks, Section 2.10.3 derives the convergence rates for the Galerkin method.

While the convergence rate in this section is derived for a piecewise linear approximation ($\delta = 2$), the procedure remains analogous for piecewise polynomials of any fixed degree. For simplicity, the following unified notation is adopted:

$$\Psi(\mathbf{x}) = \Psi(m(t)) = \Psi(t), \quad \mathbf{x} \in \Gamma, \quad t \in [a, b]. \quad (81)$$

2.10.1 General Convergence Rate of Projection Methods

This section establishes the general convergence framework for projection methods based on [1], which serves as the foundation for analyzing the collocation and the Galerkin methods. Let $\Psi_d \in \mathcal{P}_d$ denote the numerical approximation of the auxiliary function Ψ . Ideally, the numerical approximation Ψ_d should be as close as possible to the projection of Ψ onto the d -dimensional space $\mathcal{P}_d$ via the projection operator P_d . The goal of this section is to characterize the relationship between the numerical discretization error $\|\Psi - \Psi_d\|$, and the projection error, $\|\Psi - P_d\Psi\|$.

The convergence analysis is executed on a one-dimensional interval $[a, b]$ that parameterizes the original two-dimensional boundary curve Γ . Specifically, the boundary is subdivided into faces Δ_k , $k = 1, \dots, n_f$, each corresponds to a local mapping $m_k: \sigma_k = [0, 1] \rightarrow \Delta_k$. Connecting these support elements σ_k sequentially yields the global parameterization interval $[a, b]$. On this domain, the density function Ψ is approximated by a piecewise linear expansion ($\delta = 2$) using Lagrange basis polynomials l_i , $i = 1, \dots, d$. The global mesh size h is defined as the maximum support length of the reference elements σ_k where a basis function l_i remains non-vanishing. This configuration is illustrated in Figure 11.

The BIE of the second kind from Equation (35) can be written in the operator form as

$$\lambda\Psi - \mathcal{K}\Psi = g, \quad (82)$$

where the operator $-\mathcal{K}$ is defined as

$$-\mathcal{K}\Psi(\mathbf{x}) = \int_{\Gamma} \Psi(\mathbf{y})K(\mathbf{x}, \mathbf{y})ds(\mathbf{y}). \quad (83)$$

The projection operator for the collocation method is the interpolatory projection operator. For the Galerkin method, the projection operator is the orthogonal projection operator. Both operators are linear operators (cf. Section 2.10.2 for the interpolatory- and Section 2.10.3 for the orthogonal projection operator). Because of the linearity,

applying the projection operator P_d to both sides of Equation (82) yields

$$\begin{aligned} P_d(\lambda\Psi - \mathcal{K}\Psi) &= P_d g \\ \Leftrightarrow \lambda P_d \Psi - P_d \mathcal{K}\Psi &= P_d g. \end{aligned} \quad (84)$$

Applying the interpolatory projection operator leads to the finite linear system of the collocation method defined in Equation (58). Similarly, applying the orthogonal projection operator leads to the finite linear system of the Galerkin method defined in Equation (73). These projection operators will be further detailed in the succeeding sections.

Recall that Ψ_d is a function from the projection space $\mathcal{P}_d$ and the operator P_d projects a function onto $\mathcal{P}_d$. The projection of a function from the projection space $\Psi_d \in \mathcal{P}_d$ is the function itself:

$$P_d \Psi_d = \Psi_d, \quad \Psi_d \in \mathcal{P}_d. \quad (85)$$

Therefore, substituting $\Psi = \Psi_d$ in Equation (84) changes the first term of the equation:

$$\begin{aligned} \lambda P_d \Psi_d - P_d \mathcal{K}\Psi_d &= P_d g \\ \Leftrightarrow \lambda \Psi_d - P_d \mathcal{K}\Psi_d &= P_d g. \end{aligned} \quad (86)$$

To find the relationship between the approximation error $\|\Psi - \Psi_d\|$ and the projection error $\|\Psi - P_d \Psi\|$, Equation (86) is subtracted from Equation (84):

$$\begin{aligned} (\lambda P_d \Psi - P_d \mathcal{K}\Psi) - (\lambda \Psi_d - P_d \mathcal{K}\Psi_d) &= P_d g - P_d g \\ \Leftrightarrow \lambda(P_d \Psi - \Psi_d) - P_d(\mathcal{K}\Psi - \mathcal{K}\Psi_d) &= 0 \\ \Leftrightarrow \lambda(P_d \Psi - \Psi_d) - P_d \mathcal{K}(\Psi - \Psi_d) &= 0 \\ \Leftrightarrow \lambda(P_d \Psi - \Psi_d) &= P_d \mathcal{K}(\Psi - \Psi_d) \\ \Leftrightarrow (P_d \Psi - \Psi_d) &= \frac{1}{\lambda} P_d \mathcal{K}(\Psi - \Psi_d). \end{aligned} \quad (87)$$

The total error $\Psi - \Psi_d$ is subdivided into two parts: the part from Ψ to its projection $P_d \Psi$, and the remaining part from $P_d \Psi$ to the numerical solution Ψ_d . This can be written as

$$\Psi - \Psi_d = (\Psi - P_d \Psi) + (P_d \Psi - \Psi_d). \quad (88)$$

The second term in Equation (88) can be substituted with the relationship from Equation

(87) to obtain

$$\begin{aligned}
\Psi - \Psi_d &= (\Psi - P_d \Psi) + \frac{1}{\lambda} P_d \mathcal{K}(\Psi - \Psi_d) \\
&\Leftrightarrow \left(1 - \frac{1}{\lambda} P_d \mathcal{K}\right) (\Psi - \Psi_d) = (\Psi - P_d \Psi) \\
&\Leftrightarrow (\lambda - P_d \mathcal{K}) (\Psi - \Psi_d) = \lambda (\Psi - P_d \Psi).
\end{aligned} \tag{89}$$

According to [1, Theorem 3.1.1], the operator $(\lambda - P_d \mathcal{K})^{-1}$ exists and is uniformly bounded. Therefore,

$$(\Psi - \Psi_d) = \lambda (\lambda - P_d \mathcal{K})^{-1} (\Psi - P_d \Psi). \tag{90}$$

Taking the norms and Cauchy-Schwarz inequality leads to

$$\begin{aligned}
\|\Psi - \Psi_d\| &= \|\lambda (\lambda - P_d \mathcal{K})^{-1} (\Psi - P_d \Psi)\| \\
&\leq |\lambda| \|(\lambda - P_d \mathcal{K})^{-1}\| \|(\Psi - P_d \Psi)\|.
\end{aligned} \tag{91}$$

Therefore, by letting $\mathbf{M} := \|(\lambda - P_d \mathcal{K})^{-1}\|$, it can be concluded that $\|\Psi - \Psi_d\|$ converges to zero at exactly the same speed as $\|\Psi - P_d \Psi\|$:

$$\|\Psi - \Psi_d\| \leq |\lambda| \mathbf{M} \|(\Psi - P_d \Psi)\|. \tag{92}$$

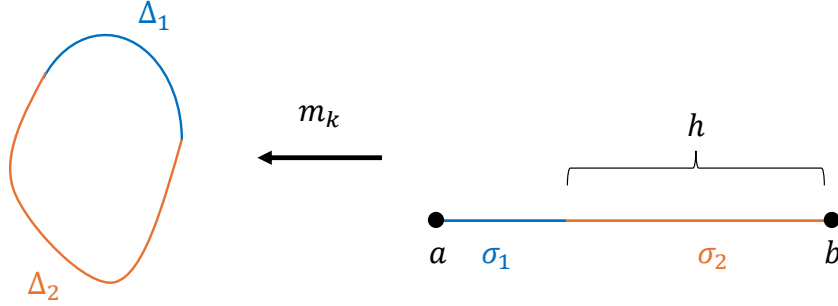


Figure 11: The interval $[a, b]$ for the convergence analysis. It consists of the support elements $\sigma_k, k = 1, \dots, n_f$ that are stitched together. With a certain mapping, $[a, b]$ can be transformed back to obtain the original boundary Γ .

2.10.2 Convergence Rate of the Collocation Method

This section establishes the convergence rate of the collocation method as a function of the mesh size h . The explanation starts with the definition of the projection operator used in the collocation method together with the proof of its linearity. Then, this section derives an upper bound for the error between Ψ and its projection. By linking this result to the general projection framework established in Section 2.10.1, the final convergence rate of $\|\Psi - \Psi_d\|$ with respect to h can be determined.

As briefly mentioned in the preceding section, the projection operator P_d for the collocation method is the interpolatory projection operator I_d :

$$I_d \Psi(t) = \sum_{i=1}^d \Psi(\mathbf{v}_i) l_i(t), \quad (93)$$

where $\mathbf{v}_i$ denotes the collocation nodes. In the collocation method, d corresponds to the number of collocation nodes. This operator is a linear operator because it satisfies the following relationship for any given functions f, g and coefficients a, b :

$$\begin{aligned} I_d (af + bg)(t) &= \sum_{i=1}^d (af + bg)(\mathbf{v}_i) l_i(t) \\ &= \sum_{i=1}^d (af(\mathbf{v}_i) + bg(\mathbf{v}_i)) l_i(t) \\ &= a \sum_{i=1}^d f(\mathbf{v}_i) l_i(t) + b \sum_{i=1}^d g(\mathbf{v}_i) l_i(t) \\ &= a I_d f(t) + b I_d g(t). \end{aligned} \quad (94)$$

Consider a single subinterval inside $[a, b]$ formed by a single support element σ_i , $i \in \{1, \dots, n_f\}$. Assume that each subinterval is bounded by $[t_{i-1}, t_i]$, such that the width of the subinterval h_i can be defined as $h_i = t_i - t_{i-1}$. Therefore the mesh size h can be defined as

$$h = \max_{i \in \{1, \dots, n_f\}} h_i. \quad (95)$$

Furthermore, assume that $\Psi \in C^2([a, b])$. Using Taylor series, the remainder for linear

interpolation at any point $t \in [t_{i-1}, t_i]$ is given by

$$\Psi(t) - (I_d \Psi)(t) = \frac{1}{2}(t - t_{i-1})(t - t_i)\Psi''(\xi_t), \quad (96)$$

where ξ_t is an unknown point inside (t_{i-1}, t_i) that depends on t .

The term $(t - t_{i-1})(t - t_i)$ in Equation (96) is a parabola opening upwards. The maximum absolute value occurs at the midpoint $t = \frac{t_{i-1} + t_i}{2}$, such that

$$\left| \left(\frac{t_{i-1} + t_i}{2} - t_{i-1} \right) \left(\frac{t_{i-1} + t_i}{2} - t_i \right) \right| = \left| \left(\frac{h_i}{2} \right) \left(-\frac{h_i}{2} \right) \right| = \frac{h_i^2}{4}. \quad (97)$$

Moreover, because $\Psi \in C^2([a, b])$, the second derivative term Ψ'' is bounded globally over the interval $[a, b]$ by $\|\Psi''\|_\infty$. Combining this two bounds back into Equation (96) results in

$$\|\Psi - I_d \Psi\|_\infty \leq \frac{1}{2} \frac{h_i^2}{4} \|\Psi''\|_\infty = \frac{h_i^2}{8} \|\Psi''\|_\infty. \quad (98)$$

Since Equation (98) holds for every subinterval, the maximum overall difference between the true solution and its projection is

$$\|\Psi - I_d \Psi\|_\infty \leq \frac{h^2}{8} \|\Psi''\|_\infty. \quad (99)$$

Substituting Equation (99) back into the inequality from Equation (92) results in the following relationship for $\Psi \in C^2([a, b])$:

$$\begin{aligned} \|\Psi - \Psi_d\|_\infty &\leq |\lambda| \mathbf{M} \|\Psi - I_d \Psi\|_\infty \\ &\leq |\lambda| \mathbf{M} \frac{h^2}{8} \|\Psi''\|_\infty. \end{aligned} \quad (100)$$

For $\Psi \in C([a, b])$, the upper bound of the projection error is defined with the modulus of continuity $\omega(\Psi, h)$ as

$$\|\Psi - I_d \Psi\|_\infty \leq \omega(\Psi, h). \quad (101)$$

The convergence rate of the numerical approximation can be derived analogously as

$$\begin{aligned}\|\Psi - \Psi_d\|_\infty &\leq |\lambda| \mathbf{M} \|\Psi - I_d \Psi\|_\infty \\ &\leq |\lambda| \mathbf{M} \omega(\Psi, h).\end{aligned}\tag{102}$$

2.10.3 Convergence Rate of the Galerkin Method

This section derives the convergence rate of the Galerkin method as a function of the mesh size h . The explanation starts with the definition of the orthogonal projection operator used in the Galerkin method together with the proof of its linearity. Then, this section derives the convergence rate by exploiting the property of the orthogonal operator as an L^2 -distance minimizer. This characteristic is coupled with the upper bound for the interpolatory projection in Section 2.10.2 to obtain the convergence rate of the Galerkin method.

The Galerkin method utilizes the orthogonal projection operator O_d that projects $L^2([a, b])$ onto $\mathcal{P}_d$. This operator is defined as

$$O_d \Psi(t) = \sum_{i=1}^d (\Psi, l_i) l_i(t).\tag{103}$$

The proof of the operator's linearity starts with the backward direction: let $af_d + bg_d$ with $f_d = O_d f$ and $g_d = O_d g$ be the candidate such that $O_d(af + bg) = af_d + bg_d$. The candidate has to satisfy the condition for the orthogonal projection. This means that the error of the candidate has to be orthogonal to every basis vector l_i , such that

$$((af + bg) - (af_d + bg_d), l_i) = 0.\tag{104}$$

Because f_d and g_d are also projections, they satisfy the orthogonality condition $(f - f_d, l_i) = 0$ and $(g - g_d, l_i) = 0$. Using the linearity of $(\cdot, \cdot)$, Equation 104 can be written as

$$((af + bg) - (af_d + bg_d), l_i) = a \underbrace{(f - f_d, l_i)}_{=0} + b \underbrace{(g - g_d, l_i)}_{=0} = 0.\tag{105}$$

This immediately confirms the assumption in the backward direction that the candidate satisfies the orthogonality. The forward direction for the proof can be concluded by leveraging the uniqueness of the orthogonal projection operator [1, Section 3.1.2]. Because $O_d(af + bg)$ and the candidate $af_d + bg_d$ satisfy the orthogonality, they must

represent the same projection. Therefore, the linearity of O_d can be concluded:

$$O_d(af + bg) = af_d + bg_d = a O_d f + b O_d g. \quad (106)$$

Based on [1, Section 3.1.2], every projection $P_d \Psi \in \mathcal{P}_d$ minimizes the distance $\|\Psi - \psi\|$ as ψ ranges over the projection space $\mathcal{P}_d$:

$$\|\Psi - P_d \Psi\| = \min_{\psi \in \mathcal{P}_d} \|\Psi - \psi\|. \quad (107)$$

Therefore, the orthogonal projection $O_d \Psi$ is the minimizer for the distance in $\mathcal{P}_d$ with respect to the L^2 -norm:

$$\|\Psi - O_d \Psi\|_{L^2([a,b])} = \min_{\psi \in \mathcal{P}_d} \|\Psi - \psi\|_{L^2([a,b])}. \quad (108)$$

Consequently, if the distance is measured with respect to the L^2 -norm, the distance of $O_d \Psi$ should be smaller or at least the same like $I_d \Psi$ [1, Section 3.3.1]. This leads to the following relationship

$$\begin{aligned} \|\Psi - O_d \Psi\|_{L^2([a,b])} &\leq \|\Psi - I_d \Psi\|_{L^2([a,b])} \\ &= \sqrt{\int_a^b |\Psi(t) - I_d \Psi(t)|^2 dt} \\ &\leq \sqrt{\int_a^b \|\Psi - I_d \Psi\|_\infty^2 dt} \\ &= \sqrt{b-a} \|\Psi - I_d \Psi\|_\infty. \end{aligned} \quad (109)$$

By combining Equation (92), Equation (99), and Equation (109), the following relationship can be established for $\Psi \in C^2[a, b]$:

$$\begin{aligned} \|\Psi - \Psi_d\|_{L^2([a,b])} &\leq |\lambda| \mathbf{M} \|(\Psi - O_d \Psi)\|_{L^2([a,b])} \\ &\leq |\lambda| \mathbf{M} \sqrt{b-a} \|\Psi - I_d \Psi\|_\infty \\ &\leq |\lambda| \mathbf{M} \sqrt{b-a} \frac{h^2}{8} \|\Psi''\|_\infty. \end{aligned} \quad (110)$$

For $\Psi \in C([a, b])$, the upper bound can be established with analogous steps using the upper bound for the interpolatory projection defined in Equation (101). This leads to

the following relationship:

$$\begin{aligned}
\|\Psi - \Psi_d\|_{L^2([a,b])} &\leq |\lambda| \mathbf{M} \|(\Psi - O_d \Psi)\|_{L^2([a,b])} \\
&\leq |\lambda| \mathbf{M} \sqrt{b-a} \|\Psi - I_d \Psi\|_{\infty} \\
&\leq |\lambda| \mathbf{M} \sqrt{b-a} \omega(\Psi, h).
\end{aligned} \tag{111}$$

3 Development of the Galerkin Solver Framework

This chapter presents the progressive development steps required to implement a robust, modular Galerkin solver framework. The materials are organized as follows:

- Section 3.1 describes the software architecture designed to ensure the framework modularity.
- Section 3.2 details the parameterization of four distinct boundaries: ellipse, kite, rhombus, and pre-fractal Koch snowflake. While the system is engineered to accommodate user-defined geometries, it features built-in capabilities to parameterize these four boundaries. These boundary configurations serve as foundational benchmarks to verify the framework.
- Section 3.3 comprises the verification process on the Helmholtz acoustic scattering problems using point source incident fields across all four boundary types.
- Section 3.4 presents the verification process of plane incident fields acoustic scattering by unit discs.
- Section 3.5 details the verification process on the Helmholtz acoustic scattering problems using plane wave incident fields on Koch snowflake boundaries.
- Section 3.6 analyzes the parameter space to identify optimal configurations that balance rigorous numerical accuracy with minimal computational effort. This examination is necessary because the computational overhead to solve boundary value problems on a Koch snowflake geometry scales drastically with advanced fractal iterations.

3.1 Software Architecture of the Galerkin Solver Framework

This chapter outlines the specific boundary value problems supported by the framework and describes its core architectural components.

To implement the Galerkin method described in the preceding chapters, a modular, object-oriented Python library was developed. The architecture is designed specifically to resolve both interior and exterior Helmholtz and Laplace boundary value problems. The framework supports boundary value problems with Dirichlet and Neumann bound-

ary conditions. To ensure adaptability across varying Dirichlet problems, the framework supports a comprehensive suite of integral representations, including single-layer, double-layer, and combined potential operators, as described in Equations (26), (27), and (28), respectively. Meanwhile, the Neumann boundary value problems are supported with the single-layer operator.

The software architecture enforces the separation of concerns software design principle, where the configuration, matrix assembly, and linear algebraic solution are decoupled and managed individually. The pipeline is split into three core functional components: **SolverConfig**, **Executor**, and **GalerkinSolver**.

The initialization of any boundary element simulation begins with the **SolverConfig** component, which encapsulates all physical, geometric, and numerical parameters governing the problem. Through this interface, the user specifies the essential execution parameters:

- The topology of the domain ($\mathcal{D}_i$ or $\mathcal{D}_e$),
- The classification of the boundary constraints (Dirichlet or Neumann),
- The underlying partial differential equation (Laplace or Helmholtz)
- The discrete boundary coordinates outlining Γ ,
- The target physical wave number $\kappa \in \mathbb{C}$ (where $\kappa = 0$ natively triggers the Laplace kernel),
- The local polynomial degree of freedom (δ),
- The explicit mathematical formulation of the boundary data function $g \in C(\Gamma)$,
- The target number of concurrent worker processes.

Upon instantiation, the component executes internal validation routines and automatically maps the problem profile to its corresponding continuous kernel operator $K(\mathbf{x}, \mathbf{y})$ and jump identity coefficient λ . This automated variable binding guarantees runtime consistency across the downstream solver components.

The **Executor** component handles the computationally demanding task of constructing the discrete system matrix **A**. Because boundary element discretizations yield dense, fully populated operator matrices, calculating the double integral for each matrix entry

introduces a significant runtime bottleneck. To alleviate this overhead, the **Executor** is designed as an extensible module capable of supporting multiple parallel execution backends. Crucially, the abstract design of the **Executor** decouples the numerical quadrature routines from the top-level algorithm, allowing alternative computational backends to be integrated seamlessly without altering the core solver logic.

In its baseline implementation, the **Executor** component leverages multiprocessing via the `Joblib` library, utilizing the `Joblib.Parallel` and `Joblib.delayed` abstractions. This framework facilitates dynamic task allocation, where the distribution of the computational workload occurs at runtime based on the instantaneous execution speed of the individual worker processes. This parallelization strategy implements a classic Master-Worker pattern, wherein a centralized master process manages a task queue and dynamically streams the quadrature integral evaluations to free worker processes in the pool upon request.

The orchestration and numerical resolution of the discrete variational system are managed by the **GalerkinSolver**. Operating dynamically on the verified state encapsulated within **SolverConfig**, this component triggers the system matrix assembly via the **Executor** and constructs the corresponding right-hand-side vector $\mathbf{g}$. Once the dense linear system $\mathbf{A}\mathbf{c} = \mathbf{g}$ is fully formulated, the **GalerkinSolver** invokes an iterative linear solver from the SciPy library. Specifically, the Generalized Minimal Residual (GMRES) method from (`scipy.sparse.linalg.gmres`) is employed. This Krylov subspace method utilizes Arnoldi iterations to systematically minimize the residual norm over successive subspaces [13].

3.2 Parametric Boundary Representation

Following the geometric classifications established in Chapter 2.4, this framework provides built-in supports for the two primary classes of boundary topologies: smooth curves and piecewise linear polygonal chains. To evaluate the robustness and convergence behavior of the implemented Galerkin solver, numerical experiments are conducted across domains with four distinct boundary geometries. The standard test suite comprises domains with circle and kite-shaped smooth boundaries, alongside a rhombus and a fractal Koch snowflake serving as representative polygonal chains boundaries. This section describes the implementation of the boundary parametrization and algorithms to generate boundary points for the standard test suite.

The smooth geometries are represented analytically via continuous, periodic parametric mappings. The first configuration utilizes the general ellipse parameterization defined

in Equation (23). Within the library, the corresponding instantiation function accepts three primary input arguments: the total number of boundary points n , the horizontal semi-axis a , and the vertical semi-axis b . For the standard numerical benchmarks defined in the succeeding sections, the circle boundary parameters for the unit discs are fixed to $a = 1$ and $b = 1$. This boundary configuration generates a canonical unit disc centered at the origin, while the boundary resolution parameter n is systematically varied across experiments to evaluate asymptotic convergence. The resulting circle boundary for the unit disc domain is shown in Figure 12 (left).

The second smooth geometry is a non-convex kite-shaped domain generated via the parametric relation given in Equation (24). The underlying software module ingests four parameters: the total number of boundary points n and the shape configuration coefficients a, b , and c . For the standard numerical benchmark defined in the succeeding sections, these geometric parameters are fixed to $a = -1.5$, $b = 0.65$, and $c = -0.65$, while the node count n is varied as an independent experimental variable. The resulting kite boundary is shown in Figure 12 (middle).

The rhombus boundary is parameterized through its total boundary points n and the respective semi-axes in the horizontal and vertical directions, denoted by a and b , respectively. To maintain structural symmetry across the four quadrants, the implementation enforces a strict algebraic constraint on the discretization density, requiring that $n \bmod 8 = 0$. The algorithmic constructor evaluates the side lengths of the boundary segments based on a and b , subsequently distributing the nodes in an equidistant fashion along each independent segment, as explained in Algorithm 1. This procedure ensures a perfectly symmetric rhombus centered at the coordinate origin, as shown in Figure 12 (right). For all experimental simulations in the succeeding sections, both the horizontal and vertical semi-axes are set to unity ($a = b = 1$).

The generation of the fractal Koch snowflake boundary is parameterized by two independent variables: the refinement depth of the fractal iteration (i.e., the Koch iteration), denoted by i , and the number of discrete discretization points allocated per individual segment, denoted by $\tilde{n}$. The conceptual algorithm for Koch snowflake is explained in Algorithm 2.

The construction begins at iteration $i = 0$ with an initiator geometry defined as a canonical equilateral triangle. In the coordinate plane, the vertices of this initial configuration for this framework are explicitly placed at the following positions:

$$\mathbf{p}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{p}_2 = \begin{pmatrix} 0.5 \\ \frac{\sqrt{3}}{2} \end{pmatrix}, \quad \text{and} \quad \mathbf{p}_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \quad (112)$$

Figure 13 shows this initial equilateral triangle with the vertices. This initial state comprises three boundary segments, each possessing an initial segment length L_0 . In this case, $L_0 = 1$. Subsequent fractal iterations ($i > 0$) execute the algorithmic trisection and mid-segment equilateral expansion detailed in Section 2.4. Consequently, the physical length L_i of an individual linear segment decreases monotonically with increasing iteration depth according to the geometric progression:

$$L_i = \frac{L_0}{3^i}. \quad (113)$$

To establish the boundary element mesh, $\tilde{n}$ equidistant nodes are distributed along each individual segment generated at depth i . Figure 14 shows the boundary node distribution with different $\tilde{n}$ in each segment for $i = 1$. The resulting global mesh size parameter h , representing the distance between any two adjacent boundary points within a segment, is uniquely determined by the relation:

$$h = \frac{L_0}{3^i(\tilde{n} - 1)}. \quad (114)$$

It follows from this relationship that if the node density per segment $\tilde{n}$ is held constant while the fractal depth i is increased, the localized mesh size h scales down as $\mathcal{O}(3^{-i})$, packing the boundary points closer together to resolve the geometric features of the fractal boundary, as illustrated in Figure 15.

The number of line segments S_i composing the pre-fractal Koch snowflake boundary at a given iteration $i \geq 0$ scales geometrically according to the relation

$$S_i = 3 \cdot 4^i. \quad (115)$$

Concurrently, the total number of global boundary nodes n generated by the mesh discretization is a function of the sub-element resolution. For elements containing $\tilde{n}$ local nodes per segment, the total number of boundary points n can be derived via the structural relationship

$$n = N(i, \tilde{n}) = S_i \cdot (\tilde{n} - 1). \quad (116)$$

To define a valid line segment, a minimum local resolution of $\tilde{n} = 2$ is required. Note that the boundary is discretized with DOF $\delta_\Gamma = 3$, which requires three nodes per face Δ_k , as previously mentioned in Chapter 2.4. Under this discretization scheme, a single face Δ_k can potentially span two adjacent linear segments simultaneously. This geometric configuration occurs whenever the user defines $\tilde{n}$ as an even integer. In such scenarios,

performing numerical quadrature over the element Δ_k requires integrating directly across a non-smooth vertex, where a sharp corner singularity caused by the undefined normal vector resides, as illustrated in Figure 16. Because the geometric singularity can affect the boundary condition, setting $\tilde{n}$ to an even integer is highly discouraged, as it severely degrades the accuracy of the boundary integral evaluation. The same rule applies for rhombus. However, the constraint $n \bmod 8 = 0$ practically resolves this limitation.

Algorithm 1 Generation of Boundary Points on Rhombus

Input: $n \in \mathbb{N}$ (Total number of boundary points, must be a multiple of 8), $a \in \mathbb{R}$ (Semi-axis in the horizontal direction), $b \in \mathbb{R}$ (Semi-axis in the vertical direction)

Output: $\mathcal{P}$ (Boundary points on rhombus)

- 1: $d \leftarrow n/4$
 - 2: Initialize an empty sequence $\mathcal{P}$ for the boundary points
 - 3: Append d equidistant points on the line segments formed by the point $(a, 0)^\top$ and $(0, b)^\top$ to $\mathcal{P}$
 - 4: Append d equidistant points on the line segments formed by the point $(0, b)^\top$ and $(-a, 0)^\top$ to $\mathcal{P}$
 - 5: Append d equidistant points on the line segments formed by the point $(-a, 0)^\top$ and $(0, -b)^\top$ to $\mathcal{P}$
 - 6: Append d equidistant points on the line segments formed by the point $(0, -b)^\top$ and $(a, 0)^\top$ to $\mathcal{P}$
-

Algorithm 2 Generation of Boundary Points on Koch Snowflake

Input: $i \in \mathbb{N}_0$ (The fractal Koch iteration depth), $\tilde{n} \in \mathbb{N}_{\geq 2}$ (The number of points per boundary segment)

Output: $\mathcal{P}$ (Boundary points on Koch snowflake)

- 1: Initialize a sequence $\mathcal{V}$ with the three vertices in Equation (112) as the base of the equilateral triangle
 - 2: **for** iteration 1 to i **do**
 - 3: Initialize a sequence $\mathcal{V}_{\text{new}}$
 - 4: **for** each segment $(\mathbf{a}, \mathbf{b})$ formed by adjacent vertices in $\mathcal{V}$ **do**
 - 5: Trisect $(\mathbf{a}, \mathbf{b})$, such that it becomes $(\mathbf{a}, \mathbf{p}_L, \mathbf{p}_R, \mathbf{b})$
 - 6: Displace the middle sub-segment $(\mathbf{p}_L, \mathbf{p}_R)$ outward by 60° to form a new, smaller equilateral triangle with apex vertex $\mathbf{p}_{\text{apex}}$
 - 7: Save the segment points as $(\mathbf{a}, \mathbf{p}_L, \mathbf{p}_{\text{apex}}, \mathbf{p}_R)$ in $\mathcal{V}_{\text{new}}$
 - 8: **end for**
 - 9: $\mathcal{V} \leftarrow \mathcal{V}_{\text{new}}$
 - 10: **end for**
 - 11: Linearly interpolate $\tilde{n}$ uniformly spaced points along each segment $(\mathbf{a}, \mathbf{b})$ in $\mathcal{V}$ as a sequence of boundary points $\mathcal{P}$
-

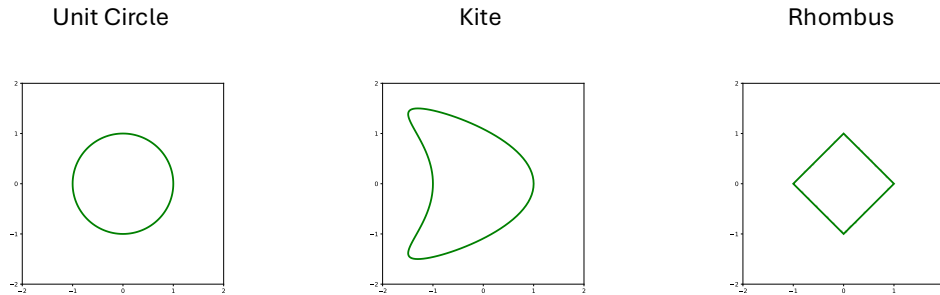


Figure 12: Three out of four boundaries used in the benchmarking of the Galerkin framework: unit circle, kite, and rhombus.

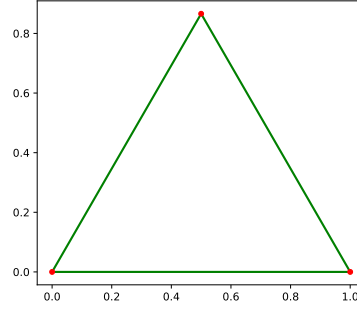


Figure 13: The initial equilateral triangle defining the Koch initiator at iteration $i = 0$. The geometry consists of three linear segments, each possessing a unit length ($L_0 = 1$).

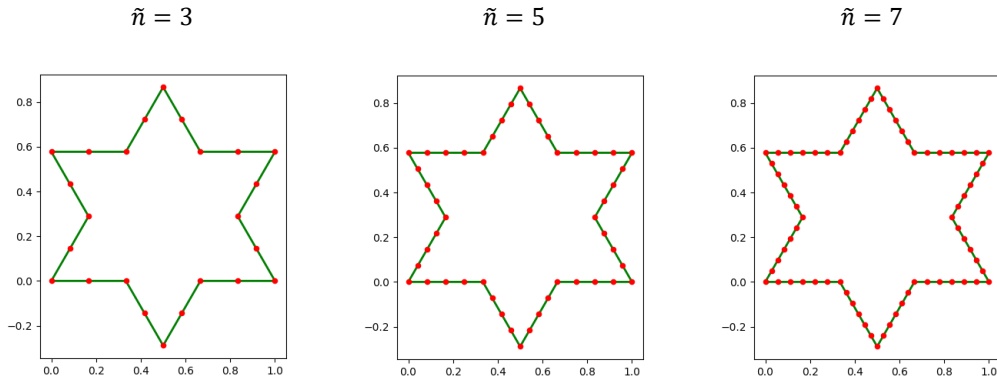


Figure 14: Discrete boundary node distributions on the first-generation Koch snowflake ($i = 1$ and 12 segments). While the underlying geometric domain remains invariant, the nodal resolution is varied across the panels. From left to right, each individual segment is resolved using three, five, and seven equidistant boundary points per segment.

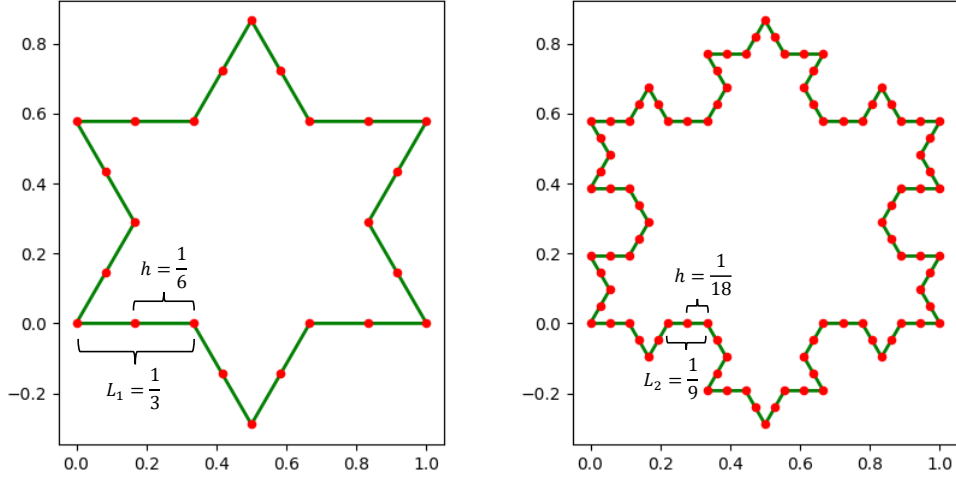


Figure 15: Influence of the geometric fractal refinement i on the local mesh size h for a fixed segment node density ($\tilde{n} = 3$). As the iteration depth i increases, the individual segment lengths contract, compressing the spatial distance between adjacent nodes h , although $\tilde{n}$ stays the same. **Left:** First-generation Koch snowflake with three points per segment ($i = 1, \tilde{n} = 3$) featuring a segment length of $L_1 = \frac{1}{3}$ and a uniform mesh size of $h = \frac{1}{6}$. **Right:** Second-generation Koch snowflake with three points per segment ($i = 2, \tilde{n} = 3$) exhibiting a reduced segment length of $L_2 = \frac{1}{9}$, which consequently scales down the local nodal spacing to $h = \frac{1}{18}$.

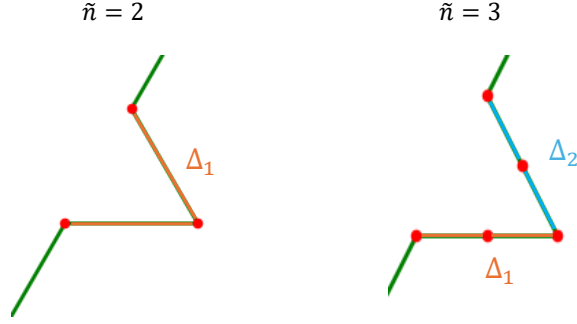


Figure 16: Boundary element discretizations of the Koch snowflake utilizing a local segment node count of $\tilde{n} = 2$ (**left**) and $\tilde{n} = 3$ (**right**). Given a boundary degree of freedom requirement of $\delta_\Gamma = 3$, each discrete face Δ_k must be supported by three nodes consisting of two endpoints and one internal midpoint. **Left:** Setting $\tilde{n}$ to an even integer forces the midpoint to coincide with a geometric corner singularity, requiring numerical quadrature routines to integrate across a non-smooth vertex. **Right:** Restricting $\tilde{n}$ to an odd integer ensures that the local nodes supporting each face Δ_k lie entirely within a single straight segment. Consequently, this alignment isolates the geometric singularities exclusively at element boundaries, avoiding integration across non-smooth corners.

3.3 Verification using Point Source Incident Fields as True Solutions

Following the implementation phase, the framework was rigorously verified by solving the Helmholtz acoustic scattering problems under point source incident fields. Section 3.3.1 outlines the experimental setups and evaluation metrics established for this verification process. The corresponding numerical results are analyzed and discussed in Section 3.3.2.

3.3.1 Methodology

Following the implementation of the Galerkin framework, a verification process was established to validate the numerical accuracy of the solver. The framework was tested

for the Helmholtz problems with a point source incident field utilizing the following physical configurations:

- An exterior Dirichlet problem with a point sourced incident field,
- An exterior Neumann problem with a point sourced incident field,
- An interior Dirichlet problem with a point sourced incident field,
- An interior Neumann problem with a point sourced incident field.

The first validation scenario considered an exterior Dirichlet problem, where the incident field u_e was generated by a point source $\mathbf{x}_0 = (0, 0)^\top$:

$$u_e(\mathbf{p}) = \frac{i}{4} H_0^{(1)}(\kappa \|\mathbf{p} - \mathbf{x}_0\|), \quad \mathbf{p} \in \mathcal{D}_e. \quad (117)$$

The generated point wave incident field on the exterior domain is presented in Figure 17. At the same time, Equation (117) also served as the analytical reference value to validate the numerical solution at an exterior evaluation point $\mathbf{p} = (3, 3)^\top \in \mathcal{D}_e$. The prescribed boundary function $g_e(\mathbf{x})$ for the boundary condition $u = g_e$ on Γ was defined as:

$$g_e(\mathbf{x}) = \frac{i}{4} H_0^{(1)}(\kappa \|\mathbf{x} - \mathbf{x}_0\|), \quad \mathbf{x} \in \Gamma. \quad (118)$$

Because this boundary data corresponds exactly to the fundamental solution of the Helmholtz equation (Equation (4)), the exact continuous field satisfies the governing PDE in $\mathcal{D}_e$ alongside the Sommerfeld radiation condition at infinity. To evaluate the versatility of the framework, this Dirichlet configuration was verified using the single-, double-, and combined-field boundary integral formulations. Throughout the tables in the remainder of this work, the single-, double-, and combined-layer kernels are abbreviated as SL, DL, and DLSL, respectively.

For the exterior Neumann problem under the same physical radiation field, the boundary function was given by the normal derivative of the fundamental solution defined in Equation (118), such that $\partial_\nu u = \partial_\nu g_e$ on Γ . Applying the chain rule to the Hankel function yields the directional derivative:

$$\partial_{\nu(\mathbf{x})} g_e(\mathbf{x}) = -\frac{i\kappa}{4 \|\mathbf{x} - \mathbf{x}_0\|} H_1^{(1)}(\kappa \|\mathbf{x} - \mathbf{x}_0\|) \langle \mathbf{x} - \mathbf{x}_0, \nu(\mathbf{x}) \rangle, \quad \mathbf{x} \in \Gamma. \quad (119)$$

This configuration was tested using the single-layer boundary element representation.

Similar to the Dirichlet case, the accuracy of the computed field at an evaluation point $\mathbf{p} = (3, 3)^\top \in \mathcal{D}_e$ was verified through a pointwise error comparison with the analytical reference value $u_e(\mathbf{p})$ given by Equation (117).

The next validation scenario focuses on the interior Dirichlet problem. The incident field u_i was generated from a pair of symmetric exterior point sources located at $\mathbf{x}_1 = (0, 5)^\top$ and $\mathbf{x}_2 = (0, -5)^\top$:

$$u_i(\mathbf{p}) = \frac{i}{4} \left[H_0^{(1)}(\kappa \|\mathbf{p} - \mathbf{x}_1\|) - H_0^{(1)}(\kappa \|\mathbf{p} - \mathbf{x}_2\|) \right], \quad \mathbf{p} \in \mathcal{D}_i. \quad (120)$$

Figure 18 visualizes the incident field for the interior problems. Similarly, Equation (120) also served as the reference analytical solution for the pointwise validation at an interior evaluation point $\mathbf{p} = (0.3, 0.2)^\top \in \mathcal{D}_i$. The boundary data $u = g_i$ on Γ was given by:

$$g_i(\mathbf{x}) = \frac{i}{4} \left[H_0^{(1)}(\kappa \|\mathbf{x} - \mathbf{x}_1\|) - H_0^{(1)}(\kappa \|\mathbf{x} - \mathbf{x}_2\|) \right], \quad \mathbf{x} \in \Gamma. \quad (121)$$

Since both source locations $\mathbf{x}_1$ and $\mathbf{x}_2$ lie strictly within the exterior domain $\mathcal{D}_e$, the resulting linear combination satisfies the homogeneous Helmholtz equation smoothly throughout the bounded interior domain $\mathcal{D}_i$. This interior Dirichlet framework was benchmarked using the single-, double-, and combined-field boundary integral formulations.

For the corresponding interior Neumann problem, the boundary data $\partial_\nu u = \partial_\nu g_i$ on Γ was defined as the normal trace of the interference field. Taking the directional derivative of Equation (121) with respect to the outward unit normal vector $\nu(\mathbf{x})$ yields:

$$\partial_{\nu(\mathbf{x})} g_i(\mathbf{x}) = -\frac{i\kappa}{4} \left\langle \sum_{m=1}^2 (-1)^{m-1} \frac{H_1^{(1)}(\kappa \|\mathbf{x} - \mathbf{x}_m\|)}{\|\mathbf{x} - \mathbf{x}_m\|} (\mathbf{x} - \mathbf{x}_m), \nu(\mathbf{x}) \right\rangle, \quad \mathbf{x} \in \Gamma. \quad (122)$$

The framework evaluated this Neumann configuration using the single-layer boundary element representation. Similar to the Dirichlet setup, the structural precision of the solver inside the domain was quantified by calculating the absolute pointwise error relative to the reference field $u_i(\mathbf{p})$ from Equation (120) at a specified verification coordinate $\mathbf{p} = (0.3, 0.2)^\top \in \mathcal{D}_i$.

The absolute pointwise error ε_n is defined as:

$$\varepsilon_n(\mathbf{p}) = |u(\mathbf{p}) - u_n(\mathbf{p})|, \quad (123)$$

where $u(\mathbf{p})$ represents the corresponding analytical reference solution of the boundary value problem at an evaluation point $\mathbf{p}$, and $u_n(\mathbf{p})$ denotes the discrete numerical ap-

proximation obtained utilizing a boundary mesh with n nodes.

To quantify the asymptotic grid refinement behavior, the evaluation used the Estimated Order of Convergence (EOC), calculated via the relation:

$$\text{EOC}_{n_1} = \frac{\log(\varepsilon_{n_1}(\mathbf{p})/\varepsilon_{n_2}(\mathbf{p}))}{\log(n_2/n_1)}, \quad (124)$$

where n_1 and n_2 denote two successive boundary node densities satisfying $n_1 < n_2$.

In addition to direct benchmarking against analytical reference solutions, this verification process incorporated the corresponding results from an independent boundary collocation method. Demonstrating that both numerical techniques yield comparable error profiles and convergence rates serves as a cross-method validation to confirm the structural correctness and mathematical consistency of the developed Galerkin solver.

All numerical simulations within this suite were executed using a physical wave number $\kappa = 3$, fixed DOF $\delta = 2$ (linear approximation with two coefficients in each face), and a boundary DOF $\delta_\Gamma = 3$. The comprehensive verification results and accompanying analyses are presented and discussed throughout Section 3.3.2.

The numerical simulations were conducted across four distinct domains: unit disc, kite, rhombus, and pre-fractal Koch snowflake. The circle boundary of the unit disc is parameterized via Equation (23) by setting $a = b = 1$. The boundary of the kite domain is parameterized using Equation (24) with coefficients $a = -1.5$, $b = 0.65$, and $c = 0.65$. The rhombus boundary is generated using Algorithm 1 with semi-axis lengths $a = b = 1$. Finally, the Koch snowflake boundary is constructed using Algorithm 2 and subsequently translated by the offset vector $(-0.5, 0)^\top$.

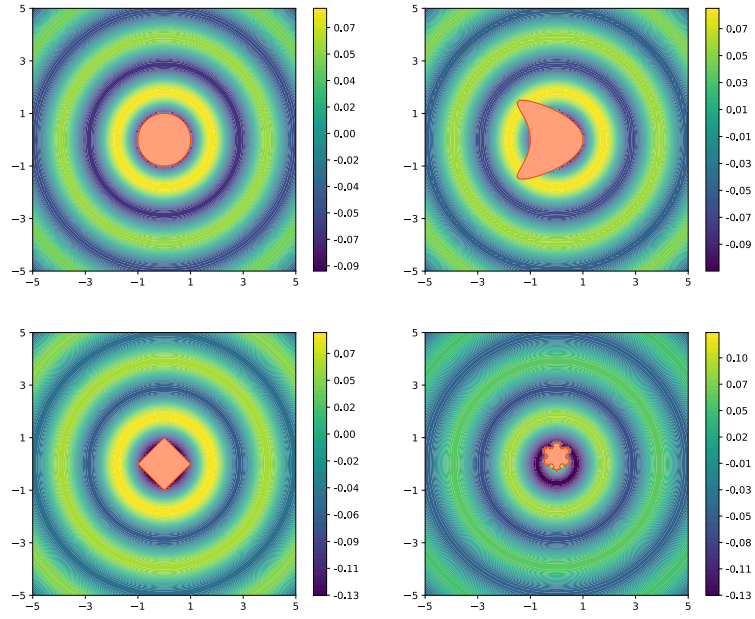


Figure 17: Point wave incident field generated by Equation (118) for the exterior scattering problems by unit disc, kite, rhombus, and second iteration Koch snowflake.

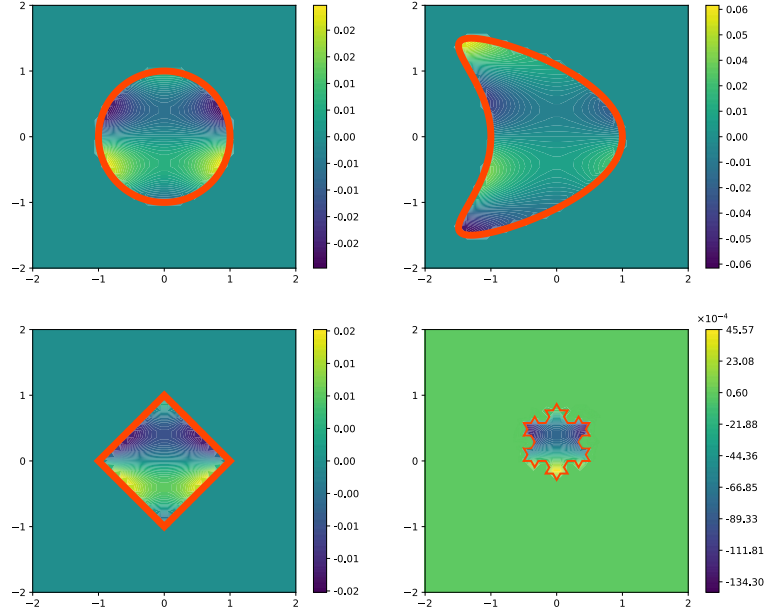


Figure 18: Point wave incident field generated by Equation (121) for the interior scattering problems by unit disc, kite, rhombus, and second iteration Koch snowflake.

3.3.2 Results and Discussions

The primary objective of this section is to present and analyze the verification results for the Galerkin boundary element framework under radiation fields generated by point sources, as outlined in Section 3.3.1.

The results are systematically organized by boundary conditions and domain topologies:

- Tables 1–4 present the absolute error and the EOC from the exterior Dirichlet problem.
- Tables 5–8 detail the absolute error and the EOC from the exterior Neumann problems.
- Tables 9–12 present the absolute error and the EOC from the interior Dirichlet problems.

- Tables 13–16 detail the absolute error and the EOC from the interior Neumann problems.

As a general overview of the results, the empirical data demonstrate that the implemented Galerkin framework shows the expected asymptotic convergence behavior. On coarser meshes with fewer boundary nodes, the discrete spaces yield a predictably higher, yet entirely acceptable, baseline error that mirrors the performance of the established collocation method. Crucially, as the discretization density systematically increases, the absolute error decreases, confirming gains in numerical precision that are comparable with the collocation method.

The MATLAB implementation of the collocation method is based on the framework described by [6], which utilizes two specialized discretization strategies. First, the boundary mesh is generated such that the elements align precisely with the physical vertices of the domain, ensuring that no individual face spans across a sharp corner singularity, matching the case illustrated in Figure 16 (right). Second, the spatial positioning of the collocation evaluation points is optimized, as variations in this parameter directly affect solution accuracy. The scheme leverages a localized optimization where the evaluation points are shifted away from the vertices by setting $\alpha = \frac{1}{2(1-\sqrt{3})}$, a choice derived from the Gauss quadrature rule. This configuration ensures that the collocation nodes lie strictly within the interior of each element face, as illustrated in the bottom panel of Figure 10.

Due to these tailored discretization techniques, the collocation method achieves high accuracy even on domains featuring geometric singularities, particularly across the Dirichlet test configurations. Yet, while the collocation scheme converges steadily throughout the suite, the developed Galerkin framework demonstrates superior accuracy and convergence in the exterior Neumann snowflake (Table 8), interior Neumann rhombus (Table 15), and interior Neumann snowflake (Table 16) test cases.

An explicit evaluation of the computed boundary traces from the collocation method for the Neumann cases confirmed that the normal derivative values remain strictly bounded and did not exhibit localized numerical divergence. This indicates that the observed convergence slowdown of the collocation method on the Koch snowflake is not caused by an arithmetic blow-up at the nodes, but is instead a structural limitation of point-wise discretization schemes on pre-fractal geometries. Because a strong-form collocation scheme samples the boundary data exclusively at discrete points, it inherently undersamples the rapid, localized wave-reflection patterns that occur across the fractal segments. In contrast, the developed Galerkin approach overcomes this sampling bottleneck natively. By projecting the boundary integral equations into a weak variational form, the

framework integrates the continuous physics across the entire element face. This mathematical averaging effectively captures the sub-element field variations and filters out the non-smooth spectral noise introduced by the fractal geometry.

A primary computational challenge shared by both methods is the accurate evaluation of the system matrix $\mathbf{A}$ entries when the underlying kernel K becomes singular. Depending on the boundary integral formulation, the kernel K represents either a fundamental solution (Φ_0 from Equation (7) or Φ_κ from Equation (4)) or its respective normal derivative ($\partial_\nu \Phi_0$ from Equation (8) or $\partial_\nu \Phi_\kappa$ from Equation (5)).

In the collocation framework, the matrix assembly requires evaluating a single integral involving $K(\mathbf{v}_i, \tilde{m}_k(s))$ as defined in Equation (62), which exhibits a pointwise singularity when $\mathbf{v}_i = \tilde{m}_k(s)$. Conversely, the variational Galerkin framework requires a double integration over the product of element domains involving $K(\tilde{m}_{k(i)}(p), \tilde{m}_{k(j)}(s))$ from Equation (80). This kernel possesses a diagonal singularity when evaluating self-overlapping elements, occurring precisely when $k(i) = k(j)$ and $p = s$, which implies $\tilde{m}_{k(i)}(p) = \tilde{m}_{k(j)}(s)$. To maintain the precision of standard numerical quadrature rules, this singularity bottleneck is resolved in both methods by partitioning the integration domains into localized, well-defined sub-intervals.

For the collocation method, let $\beta \in \sigma$ be the parameter to map a certain collocation coordinate, as illustrated in Figure 10. The kernel singularity occurs when $\tilde{m}_k(s) = \mathbf{v}_i$. As an example, consider $\delta = 2$, such that each face has two collocation points and the placement of the collocation point is shifted by the parameter α like in Figure 10 bottom. In this case, the kernel singularity occurs at $\beta = \alpha$ and $\beta = 1 - \alpha$ because $\tilde{m}_k(\alpha) = \mathbf{v}_1$ and $\tilde{m}_k(1 - \alpha) = \mathbf{v}_2$. For a prescribed β , the singular integral is regularized by splitting the domain of integration at the singularity coordinate, such that Equation (62) is modified as follows:

$$\begin{aligned} \int_\sigma l^{(t)}(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) ds &= \int_0^\beta l^{(t)}(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) ds \\ &+ \int_\beta^1 l^{(t)}(s) K(\mathbf{v}_i, \tilde{m}_k(s)) \tilde{J}_k(s) ds. \end{aligned} \quad (125)$$

For the Galerkin formulation, the diagonal singularity occurs along the line $p = s$ within the reference square $\sigma \times \sigma$. To isolate the singularity, the inner double integral is split into two triangular sub-domains where the integrand remains smooth up to the boundary

interface. Therefore, Equation (80) is modified as:

$$\begin{aligned}
& \int_{\sigma} \int_{\sigma} l^{(t(i))}(p) l^{(t(j))}(s) K(\tilde{m}_{k(i)}(p), \tilde{m}_{k(j)}(s)) \tilde{J}_{k(i)}(p) \tilde{J}_{k(j)}(s) \, ds \, dp \\
&= \int_0^1 \int_0^p l^{(t(i))}(p) l^{(t(j))}(s) K(\tilde{m}_{k(i)}(p), \tilde{m}_{k(j)}(s)) \tilde{J}_{k(i)}(p) \tilde{J}_{k(j)}(s) \, ds \, dp \\
&+ \int_0^1 \int_p^1 l^{(t(i))}(p) l^{(t(j))}(s) K(\tilde{m}_{k(i)}(p), \tilde{m}_{k(j)}(s)) \tilde{J}_{k(i)}(p) \tilde{J}_{k(j)}(s) \, ds \, dp. \tag{126}
\end{aligned}$$

Taken together, this validation strategy rigorously demonstrates the structural correctness and precision of the developed Galerkin implementation for the point wave incident fields.

Table 1: **Exterior Dirichlet Problem of a Point Source Incident Field Scattered by Unit Disc**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	3.766 ₋₄	6.631 ₋₄	1.171 ₁	4.007
	16	1.126 ₋₇	4.124 ₋₅	6.601	4.021
	32	1.160 ₋₉	2.541 ₋₆	3.489	4.033
	64	1.033 ₋₁₀	1.552 ₋₇	1.000	4.126
	128	5.166 ₋₁₁	8.889 ₋₉	1.034 ₋₁	5.579
DL	8	2.168 ₋₄	2.387 ₋₄	1.214 ₁	5.103
	16	4.795 ₋₈	6.946 ₋₆	9.140	4.969
	32	8.501 ₋₁₁	2.218 ₋₇	7.875	4.995
	64	3.622 ₋₁₃	6.954 ₋₉	-1.483	4.999
	128	1.013 ₋₁₂	2.174 ₋₁₀	2.081	5.008
DLSL	8	2.168 ₋₄	2.218 ₋₄	1.214 ₁	4.184
	16	4.795 ₋₈	1.220 ₋₅	9.140	4.198
	32	8.501 ₋₁₁	6.649 ₋₇	7.875	4.090
	64	3.622 ₋₁₃	3.904 ₋₈	-1.483	4.141
	128	1.013 ₋₁₂	2.212 ₋₉	2.081	5.571

Table 2: **Exterior Dirichlet Problem of a Point Source Incident Field on a Kite**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	2.630 ₋₂	4.230 ₋₂	4.306	4.752
	16	1.329 ₋₃	1.570 ₋₃	4.668	3.316
	32	5.229 ₋₅	1.576 ₋₄	4.560	3.773
	64	2.217 ₋₆	1.153 ₋₅	4.755	3.047
	128	8.212 ₋₈	1.395 ₋₆	4.944	3.006
DL	8	5.139 ₋₃	7.386 ₋₃	1.585	3.361
	16	1.713 ₋₃	7.190 ₋₄	2.021	2.387 ₋₁
	32	4.219 ₋₄	6.093 ₋₄	4.315	5.569
	64	2.119 ₋₅	1.284 ₋₅	3.971	3.888
	128	1.351 ₋₆	8.673 ₋₇	3.995	3.820
DLSL	8	2.429 ₋₃	1.305 ₋₂	-3.629 ₋₁	1.691
	16	3.124 ₋₃	4.040 ₋₃	3.353	2.924
	32	3.056 ₋₄	5.324 ₋₄	4.671	6.125
	64	1.200 ₋₅	7.631 ₋₆	3.967	3.799
	128	7.670 ₋₇	5.482 ₋₇	3.993	3.621

Table 3: **Exterior Dirichlet Problem of a Point Source Incident Field on a Rhombus**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	5.000 ₋₁	1.792 ₋₁	7.687	7.673
	16	2.425 ₋₃	8.784 ₋₄	4.462	3.517
	32	1.100 ₋₄	7.675 ₋₅	4.581	2.520
	64	4.596 ₋₆	1.338 ₋₅	4.631	2.852
	128	1.854 ₋₇	1.853 ₋₆	4.608	2.948
DL	8	3.112 ₋₃	1.614 ₋₃	4.632	1.731
	16	1.256 ₋₄	4.863 ₋₄	4.854 ₋₁	2.611
	32	8.969 ₋₅	7.960 ₋₅	2.263	2.514
	64	1.869 ₋₅	1.394 ₋₅	2.551	2.611
	128	3.188 ₋₆	2.281 ₋₆	2.629	2.647
DLSL	8	4.139 ₋₃	3.306 ₋₃	3.750	2.922
	16	3.076 ₋₄	4.363 ₋₄	1.823	2.567
	32	8.695 ₋₅	7.363 ₋₅	2.351	2.535
	64	1.705 ₋₅	1.270 ₋₅	2.568	2.621
	128	2.874 ₋₆	2.066 ₋₆	2.633	2.651

Table 4: **Exterior Dirichlet Problem of a Point Source Incident Field on a Koch Snowflake**

Kernel	i	$\tilde{n}$	n	Absolute Error		Estimated Order of Convergence	
				Galerkin	Collocation	Galerkin	Collocation
SL	2	3	96	3.221 ₋₆	1.169 ₋₆	4.700	2.326
	2	5	192	1.239 ₋₇	2.331 ₋₇	3.904	1.607
	2	7	288	2.545 ₋₈	1.215 ₋₇	8.458 ₋₁	1.969
DL	2	3	96	7.633 ₋₅	4.237 ₋₅	2.719	2.638
	2	5	192	1.159 ₋₅	6.809 ₋₆	2.635	2.598
	2	7	288	3.982 ₋₆	2.375 ₋₆	2.626	2.605
DLSL	2	3	96	3.187 ₋₅	1.956 ₋₅	2.839	2.833
	2	5	192	4.453 ₋₆	2.745 ₋₆	2.684	2.694
	2	7	288	1.500 ₋₆	9.210 ₋₇	2.653	2.666

Table 5: **Exterior Neumann Problem of a Point Source Incident Field Scattered by Unit Disc**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	3.880 ₋₄	9.329 ₋₄	1.056 ₁	2.454
	16	2.571 ₋₇	1.702 ₋₄	1.102 ₁	3.004
	32	1.239 ₋₁₀	2.122 ₋₅	4.692	3.014
	64	4.794 ₋₁₂	2.627 ₋₆	2.354	3.010
	128	9.379 ₋₁₃	3.261 ₋₇	-7.703	3.006

Table 6: **Exterior Neumann Problem of a Point Source Incident Field on a Kite**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	1.563 ₋₂	3.202 ₋₂	2.777	4.576
	16	2.280 ₋₃	1.342 ₋₃	2.583	1.684
	32	3.805 ₋₄	4.179 ₋₄	3.978	5.724
	64	2.416 ₋₅	7.905 ₋₆	3.934	4.505
	128	1.580 ₋₆	3.482 ₋₇	3.984	4.423

Table 7: **Exterior Neumann Problem of a Point Source Incident Field on a Rhombus**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	7.378 ₋₂	1.130 ₋₁	6.357	6.814
	16	9.003 ₋₄	1.004 ₋₃	3.530	3.391
	32	7.792 ₋₅	9.569 ₋₅	3.870	3.914
	64	5.330 ₋₆	6.346 ₋₆	3.956	3.977
	128	3.435 ₋₇	4.031 ₋₇	3.985	3.993

Table 8: **Exterior Neumann Problem of a Point Source Incident Field on a Koch Snowflake**

Kernel	i	$\tilde{n}$	n	Absolute Error		Estimated Order of Convergence	
				Galerkin	Collocation	Galerkin	Collocation
SL	2	3	96	5.974 ₋₆	7.333 ₋₄	2.828	9.317 ₋₁
	2	5	192	8.411 ₋₇	3.844 ₋₄	2.810	8.257 ₋₁
	2	7	288	2.692 ₋₇	2.751 ₋₄	2.783	7.985 ₋₁

Table 9: **Interior Dirichlet Problem of a Point Source Incident Field Scattered by Unit Disc**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	1.108 ₋₃	4.194 ₋₃	5.165	6.032
	16	3.088 ₋₅	6.410 ₋₅	4.923	3.248
	32	1.018 ₋₆	6.750 ₋₆	4.765	3.134
	64	3.744 ₋₈	7.689 ₋₇	4.919	3.072
	128	1.238 ₋₉	9.146 ₋₈	4.401	3.019
DL	8	2.770 ₋₃	4.920 ₋₃	3.675	4.105
	16	2.169 ₋₄	2.860 ₋₄	4.328	6.048
	32	1.080 ₋₅	4.321 ₋₆	3.952	3.957
	64	6.977 ₋₇	2.783 ₋₇	3.988	3.987
	128	4.398 ₋₈	1.755 ₋₈	3.997	3.996
DLSL	8	4.088 ₋₃	8.974 ₋₃	3.778	4.647
	16	2.980 ₋₄	3.581 ₋₄	4.145	4.960
	32	1.685 ₋₅	1.150 ₋₅	3.955	3.488
	64	1.087 ₋₆	1.026 ₋₆	3.989	3.314
	128	6.843 ₋₈	1.032 ₋₇	4.002	3.164

Table 10: **Interior Dirichlet Problem of a Point Source Incident Field on a Kite**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	4.707_{-2}	4.731_{-2}	5.608	3.196
	16	9.653_{-4}	5.162_{-3}	4.069	4.023
	32	5.750_{-5}	3.174_{-4}	4.865	3.658
	64	1.973_{-6}	2.515_{-5}	4.781	3.078
	128	7.174_{-8}	2.980_{-6}	4.951	3.021
DL	8	1.846_{-2}	5.133_{-2}	2.308	2.944
	16	3.729_{-3}	6.673_{-3}	4.631	2.083
	32	1.505_{-4}	1.575_{-3}	4.966	6.387
	64	4.814_{-6}	1.882_{-5}	3.903	4.030
	128	3.218_{-7}	1.152_{-6}	3.957	4.055
DLSL	8	2.023_{-2}	5.915_{-2}	1.728	1.814
	16	6.106_{-3}	1.682_{-2}	5.402	2.662
	32	1.444_{-4}	2.658_{-3}	4.350	6.503
	64	7.081_{-6}	2.931_{-5}	4.006	3.329
	128	4.406_{-7}	2.917_{-6}	3.997	3.045

Table 11: **Interior Dirichlet Problem of a Point Source Incident Field on a Rhombus**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	2.518 ₋₃	3.487 ₋₃	6.520	4.459
	16	2.744 ₋₅	1.586 ₋₄	4.053	2.249
	32	1.653 ₋₆	3.336 ₋₅	4.619	2.585
	64	6.727 ₋₈	5.558 ₋₆	4.013	2.606
	128	4.168 ₋₉	9.127 ₋₇	4.232	2.620
DL	8	6.858 ₋₃	7.508 ₋₃	3.490	3.292
	16	6.104 ₋₄	7.668 ₋₄	5.147	5.540
	32	1.722 ₋₅	1.648 ₋₅	3.575	3.482
	64	1.445 ₋₆	1.475 ₋₆	3.943	3.682
	128	9.397 ₋₈	1.149 ₋₇	3.979	3.614
DLSL	8	1.153 ₋₂	1.481 ₋₂	3.835	3.757
	16	8.078 ₋₄	1.095 ₋₃	5.208	5.762
	32	2.186 ₋₅	2.018 ₋₅	3.632	3.312
	64	1.764 ₋₆	2.031 ₋₆	3.939	3.309
	128	1.150 ₋₇	2.049 ₋₇	3.975	3.173

Table 12: **Interior Dirichlet Problem of a Point Source Incident Field on a Koch Snowflake**

Kernel	i	$\tilde{n}$	n	Absolute Error		Estimated Order of Convergence	
				Galerkin	Collocation	Galerkin	Collocation
SL	2	3	96	2.006 ₋₅	5.003 ₋₆	4.300	2.029
	2	5	192	1.018 ₋₆	1.226 ₋₆	6.630	2.588
	2	7	288	6.925 ₋₈	4.292 ₋₇	2.867 ₋₁	2.619
DL	2	3	96	2.858 ₋₅	1.713 ₋₅	2.806	2.420
	2	5	192	4.087 ₋₆	3.201 ₋₆	2.971	2.880
	2	7	288	1.225 ₋₆	9.956 ₋₇	2.897	2.852
DLSL	2	3	96	4.779 ₋₅	4.197 ₋₅	2.839	2.841
	2	5	192	6.680 ₋₆	5.856 ₋₆	2.884	2.864
	2	7	288	2.075 ₋₆	1.833 ₋₆	2.836	2.846

Table 13: **Interior Neumann Problem of a Point Source Incident Field Scattered by Unit Disc**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	4.527 ₋₄	4.275 ₋₃	6.746 ₋₁	3.442
	16	2.836 ₋₄	3.933 ₋₄	4.432	4.510
	32	1.314 ₋₅	1.727 ₋₅	3.959	3.113
	64	8.448 ₋₇	1.996 ₋₆	3.989	3.054
	128	5.319 ₋₈	2.403 ₋₇	3.957	3.026

Table 14: **Interior Neumann Problem of a Point Source Incident Field on a Kite**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	4.628 ₋₂	2.741 ₋₂	2.061	2.328
	16	1.109 ₋₂	5.458 ₋₃	3.927	3.217
	32	7.294 ₋₄	5.869 ₋₄	3.863	4.540
	64	5.011 ₋₅	2.523 ₋₅	3.962	3.246
	128	3.215 ₋₆	2.659 ₋₆	3.991	2.922

Table 15: **Interior Neumann Problem of a Point Source Incident Field on a Rhombus**

Kernel	n	Absolute Error		Estimated Order of Convergence	
		Galerkin	Collocation	Galerkin	Collocation
SL	8	3.705 ₋₃	5.206 ₋₃	3.541	4.268 ₋₁
	16	3.183 ₋₄	3.872 ₋₃	1.244	9.054 ₋₁
	32	1.344 ₋₄	2.067 ₋₃	2.413	7.180 ₋₁
	64	2.523 ₋₅	1.257 ₋₃	2.586	6.875 ₋₁
	128	4.202 ₋₆	7.804 ₋₄	2.637	6.782 ₋₁

Table 16: **Interior Neumann Problem of a Point Source Incident Field on a Koch Snowflake**

Kernel	i	$\tilde{n}$	n	Absolute Error		Estimated Order of Convergence	
				Galerkin	Collocation	Galerkin	Collocation
SL	2	3	96	5.140 ₋₅	2.744 ₋₃	2.779	4.336 ₋₁
	2	5	192	7.491 ₋₆	2.032 ₋₃	2.668	4.591 ₋₁
	2	7	288	2.539 ₋₆	1.687 ₋₃	2.644	4.954 ₋₁

3.4 Verification of Plane Incident Fields Scattering by Unit Discs

Following successful verification using a point source incident field, the validation pipeline continued by solving the Helmholtz acoustic scattering problem on a circular boundary. The experimental setups and evaluation metrics established for this verification process are detailed in Section 3.4.1. The corresponding numerical results are analyzed and discussed in Section 3.4.2.

3.4.1 Methodology

In this verification step, the developed Galerkin solver was evaluated against classical acoustic scattering benchmarks using the plane-wave incident fields to simulate the following canonical Helmholtz acoustic scattering problems:

- Acoustic scattering by a sound-soft circular obstacle governed by a Dirichlet boundary condition,
- Acoustic scattering by a sound-hard circular obstacle governed by a Neumann boundary condition.

The first configuration considered a plane wave incident scattered by a sound-soft unit disc with a homogeneous Dirichlet boundary condition for the total acoustic field $u = 0$, yielding $u^{\text{sca}}(\mathbf{x}) = -u^{\text{inc}}(\mathbf{x})$ on Γ , as explained in Equation (21). The incident plane wave u^{inc} propagating along a chosen unit direction vector $\mathbf{d}$ was expressed as:

$$u^{\text{inc}}(\mathbf{x}) = \exp(i\kappa\langle\mathbf{x}, \mathbf{d}\rangle). \quad (127)$$

While the incident field satisfies the homogeneous Helmholtz equation locally, the induced scattered field u^{sca} satisfies the Sommerfeld radiation condition at infinity. This Dirichlet problem was benchmarked using the single-, double-, and combined-field boundary integral formulations.

Using a Jacobi-Anger series expansion as detailed in [2], the closed-form analytical solution for the scattered field exterior to an origin-centered disc of radius a is given by:

$$u^{\text{sca}}(r, \theta) = - \sum_{m=-\infty}^{\infty} i^m \frac{J_m(\kappa a)}{H_m^{(1)}(\kappa a)} H_m^{(1)}(\kappa r) e^{im(\theta-\theta_0)}, \quad (128)$$

where J_m is the Bessel function of the first kind of order m , $H_m^{(1)}$ is the Hankel function of the first kind of order m , and (r, θ) represent the polar coordinates of the evaluation point. The incident wave angle is defined by $\theta_0 = \arctan(d_y/d_x)$ for $\mathbf{d} = (d_x, d_y)^\top$.

For validation purposes, the geometry was restricted to a unit disc ($a = 1$) and a horizontal incident direction $\mathbf{d} = (1, 0)^\top$, corresponding to $\theta_0 = 0$. For numerical computation with this parameter, the infinite series in (128) was truncated to the following approximation:

$$u^{\text{sca}}(r, \theta) \approx - \sum_{m=-50}^{50} i^m \frac{J_m(\kappa)}{H_m^{(1)}(\kappa)} H_m^{(1)}(\kappa r) e^{im\theta}. \quad (129)$$

Limiting the summation to $-50 \leq m \leq 50$ is numerically justified, as the high-order Bessel functions decay rapidly past this threshold, rendering the omitted terms mathematically negligible.

For a sound-hard unit disc, the normal velocity vanishes on the boundary, leading to an inhomogeneous Neumann condition $\partial_\nu u^{\text{sca}} = -\partial_\nu u^{\text{inc}}$ on Γ . Differentiating the incident wave equation (127) with respect to the outward unit normal vector $\nu(\mathbf{x})$ gives:

$$\partial_{\nu(\mathbf{x})} u^{\text{inc}} = i\kappa \exp(i\kappa \langle \mathbf{x}, \mathbf{d} \rangle) \langle \mathbf{d}, \nu(\mathbf{x}) \rangle. \quad (130)$$

The exact scattering field for this configuration is defined using the derivative of Equation (128):

$$u^{\text{sca}}(r, \theta) = - \sum_{m=-\infty}^{\infty} i^m \frac{J'_m(\kappa a)}{H_m^{(1)'}(\kappa a)} H_m^{(1)}(\kappa r) e^{im(\theta-\theta_0)}, \quad (131)$$

where J'_m and $H_m^{(1)'}$ represent the derivatives of the first kind Bessel and Hankel functions with respect to their arguments. Applying the same canonical verification parameters

($a = 1$, $\mathbf{d} = (1, 0)^\top$), the series was evaluated numerically via the truncated summation:

$$u^{\text{sca}}(r, \theta) \approx - \sum_{m=-50}^{50} i^m \frac{J'_m(\kappa)}{H_m^{(1)'}(\kappa)} H_m^{(1)}(\kappa r) e^{im\theta}. \quad (132)$$

The verification was performed qualitatively and quantitatively. The qualitative inspections were carried out by evaluating the spatial distributions of both the scattered field and its corresponding absolute error field, ensuring that the Galerkin method accurately captures the physical wave structures. Complementing this, quantitative validation was performed by computing the global mean absolute error with respect to the analytical reference solutions. The absolute error $\epsilon_n(\mathbf{p})$ of the real component of the scattered field was evaluated pointwise across the domain via

$$\epsilon_n(\mathbf{p}) = |\text{Re}(u^{\text{sca}}(\mathbf{p}) - u_n^{\text{sca}}(\mathbf{p}))|, \quad (133)$$

where $u^{\text{sca}}(\mathbf{p})$ denotes the continuous reference field at an evaluation coordinate $\mathbf{p} \in \mathcal{D}_e$, and $u_n^{\text{sca}}(\mathbf{p})$ represents the discrete numerical approximation obtained using n boundary nodes. To assess global performance across the entire exterior domain, the mean absolute error $\bar{\epsilon}_n$ was computed over a uniform distribution of N_{eval} evaluation points $\mathbf{p}_i$:

$$\bar{\epsilon}_n = \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} \epsilon_n(\mathbf{p}_i). \quad (134)$$

The reference fields for the benchmark cases were derived from truncated series approximations of the analytical solutions. Specifically, the sound-soft (Dirichlet) disc configuration utilizes Equation (129), whereas the sound-hard (Neumann) disc configuration relies on Equation (132).

To establish a robust baseline, the results were also benchmarked against an independent boundary collocation method. This cross-validation aims to demonstrate that the numerical error of the Galerkin framework tracks consistently within the same order of magnitude as the established collocation scheme, thereby verifying the structural correctness of the implementation.

For the cross-method validation, the reference boundary collocation solver was implemented in accordance with the techniques described by [6]. To shield the strong-form evaluation from nodal geometric singularities, the collocation nodes were shifted internally within each element using the optimal mapping parameter $\alpha = \frac{1}{2(1-\sqrt{3})}$.

The numerical evaluations were carried out on a circular boundary parameterized according to Equation (23) with semi-axes $a = b = 1$. The discrete approximation spaces were configured with a DOF of $\delta = 2$, which prescribes two basis coefficients per element face, while the geometric boundary representation utilized quadratic polynomials according to $\delta_\Gamma = 3$. The scattering simulations were executed at a physical wavenumber of $\kappa = 5$. For the sound-soft obstacle configuration governed by the Dirichlet boundary condition, the framework evaluated three distinct boundary integral operator architectures: the single-, double-, and combined-layer potential formulation. Conversely, the sound-hard obstacle under the Neumann boundary condition was modeled exclusively via the single-layer potential representation. The continuous potential field reconstructions were evaluated on points in the exterior domain inside the bounding region $[-5, 5] \times [-5, 5]$. These inspection paths are detailed and discussed throughout Section 3.4.2.

3.4.2 Results and Discussions

The primary objective of this section is to present and discuss the verification of the Galerkin boundary element framework under plane wave incident fields on unit discs with circle boundaries. To establish a robust baseline, the results were benchmarked against both exact analytical solutions and an independent boundary collocation method.

The spatial topologies of these real-valued reference fields scattered by the unit disc geometry are illustrated in Figure 19.

Following each simulation run, the real component of the complex-valued scattered field computed via the Galerkin framework was qualitatively analyzed alongside both the analytical reference and the boundary collocation fields. The primary objective of this visual inspection is to verify that the developed Galerkin solver accurately replicates the structural characteristics of the continuous wave fields, including spatial phase oscillations, geometric boundary reflections, and physical acoustic shadow zones. As compiled in Tables 17 to 20, the empirical fields generated by the Galerkin formulation match the expected macro-level wave topologies, demonstrating excellent qualitative agreement with both the analytical solutions and the reference collocation baseline.

Tables 21 to 24 display the spatial distribution of the absolute error ϵ_n computed via the Galerkin and boundary collocation methods across various kernels and mesh densities. As monitored by the shrinking scales of the respective color bars, a systematic increase in the number of boundary nodes n results in a predictable, uniform reduction of the absolute error. This steady decay across all evaluated integral kernels quantitatively verified that the implemented Galerkin framework exhibits the expected asymptotic

convergence behavior under systematic grid refinement.

Furthermore, Tables 21 to 24 demonstrate that the absolute error profiles of the implemented Galerkin framework closely mirror those of the independent boundary collocation baseline. This close quantitative agreement across all tested operator representations confirms the structural consistency of the Galerkin framework and independently validates the correctness of the matrix assembly routines.

Tables 21 to 24 also reveal that the absolute error for both methods is elevated in immediate proximity to the boundary compared to evaluation points located further. This behavior does not signify an implementation error within either framework, but is rather a well-documented characteristic of the BEM known as the near-boundary evaluation problem. Let $r = \|\mathbf{p} - \mathbf{x}\|$ denote the Euclidean distance between an evaluation coordinate $\mathbf{p}$ and a point on the boundary $\mathbf{x} \in \Gamma$. As the evaluation point approaches the boundary interface, the distance r tends toward zero, causing the single-layer kernel to exhibit a weakly singular logarithmic dependency ($\log r$), while the normal derivative operators (e.g. the Dirichlet double-layer or the Neumann single-layer operator) scale with a stronger singular inverse dependency ($1/r$). This rapid growth in the integrands poses a significant challenge for standard numerical quadrature rules, naturally resulting in higher localized error profiles near the boundary layer.

Table 21 illustrates the spatial topography of the absolute error field generated by the single-layer potential operator. While the Galerkin formulation yields a highly uniform, low-magnitude error distribution across the interior domain, the collocation benchmark exhibits faint, radially propagating ring-structured ripple artifacts. This structural discrepancy does not stem from an implementation defect within the collocation framework, but rather highlights a fundamental mathematical difference between the strong- and the weak formulation. The strong-form pointwise enforcement of the collocation method restricts residual minimization exclusively to the designated nodal coordinates, leaving unconstrained polynomial interpolation errors to oscillate between nodes. Conversely, the weak formulation of the Galerkin method projects the operator residual onto a continuous test space by integrating the boundary conditions across the entire support of each element face, effectively smoothing out high-frequency discretization noise.

Furthermore, a distinct topographical variation is observable between the error patterns: the Dirichlet single-layer formulation produced concentric, ring-structured ripples, whereas the other operator configurations tended to generate radial, star-like ripples resembling the spokes of a wheel. In the cases of the Dirichlet double-layer, Dirichlet combined-field, and Neumann single-layer formulations, the boundary integral kernel explicitly incorporates a directional derivative mapped by the unit normal vector. Be-

cause the underlying geometry is discretized via a piecewise mesh, the transitions of the unit normal vectors at the element junctions reduce the spatial regularity of the integrand, thereby complicating the localized numerical approximation. The resulting approximation errors are amplified and projected outward along the trajectories of these normal vector components. Conversely, the single-layer kernel depends exclusively on the isotropic Euclidean distance between two points, allowing the discretization errors to disperse more uniformly in all directions across the domain.

Tables 25 and 26 compile the mean absolute errors $\bar{\epsilon}_n$ for the acoustic scattering configurations under examination. Specifically, Table 25 details the error metrics for the sound-soft unit disc across the single-, double-, and combined-field operator formulations, while Table 26 presents the corresponding errors for the sound-hard unit disc utilizing the single-layer representation. The primary objective of these tables is to provide quantitative proof that the Galerkin framework tracks consistently within the same order of error magnitude as the established collocation scheme. Both datasets demonstrate that the developed Galerkin formulation exhibits consistent asymptotic convergence, with error magnitudes that tracks closely alongside the independent boundary collocation baseline. These quantitative agreements successfully validate the structural integrity and precision of the implemented variational framework.

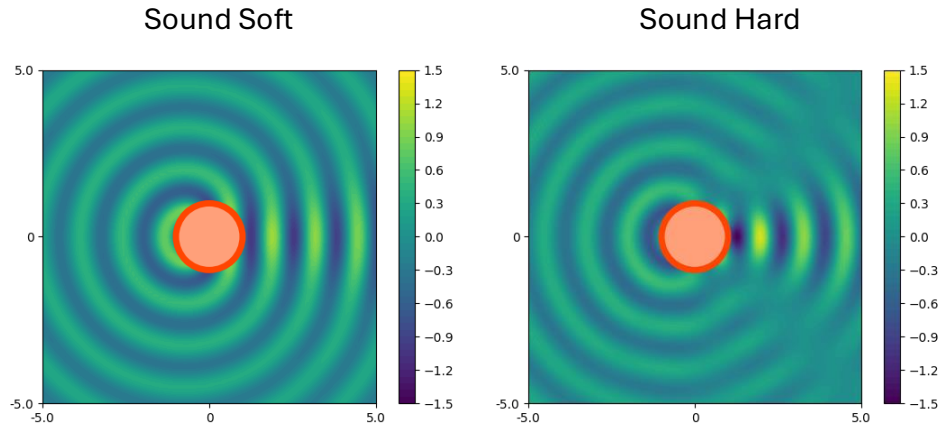


Figure 19: Real components of the reference scattered fields for a sound-soft obstacle (left) and a sound-hard obstacle (right), evaluated via the series approximations in Equation (129) and Equation (132), respectively.

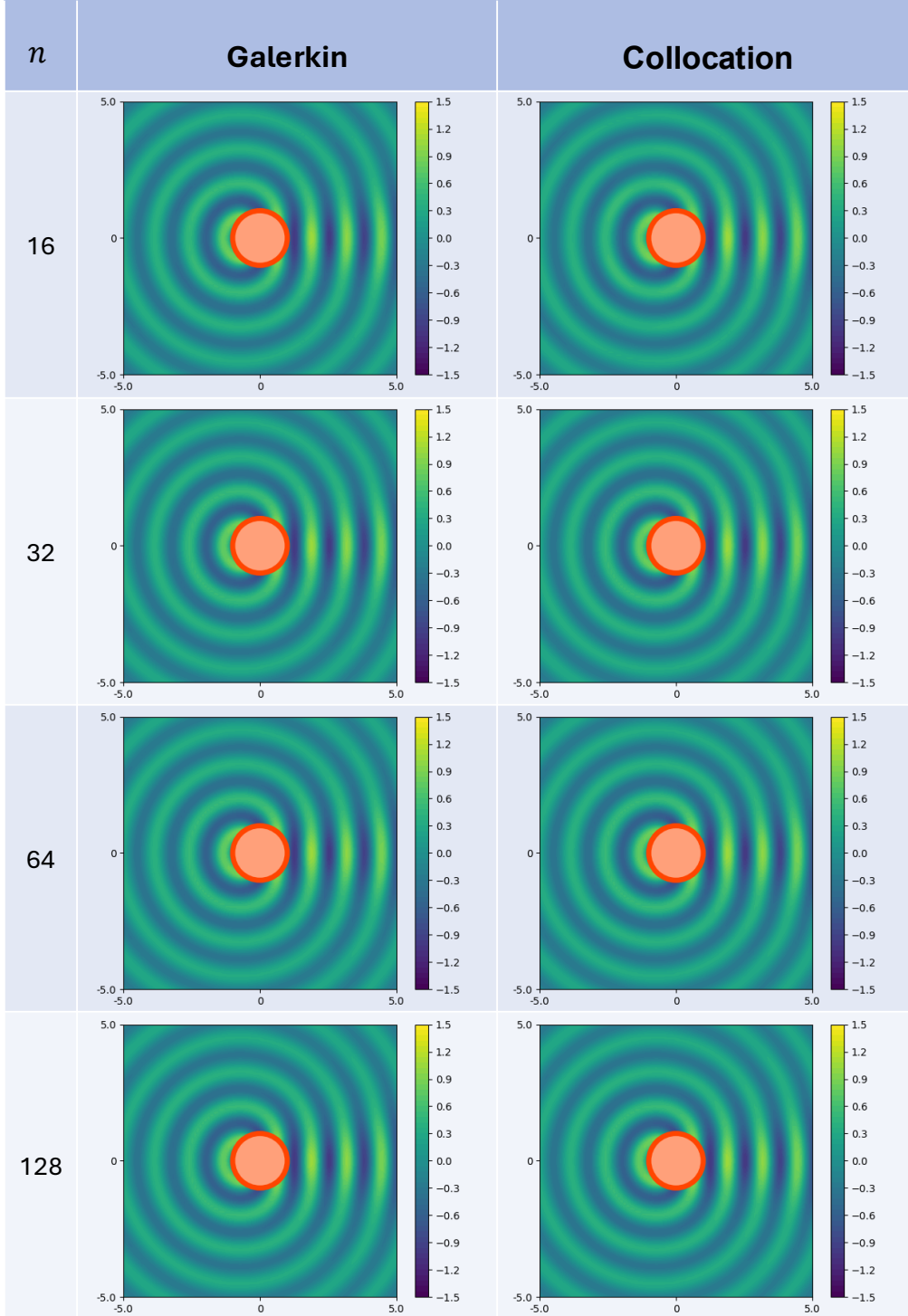


Table 17: **Scattering by a sound-soft unit disc with single-layer kernel.** Real components of the scattered potential field computed using the Galerkin formulation (left) and the collocation baseline (right).

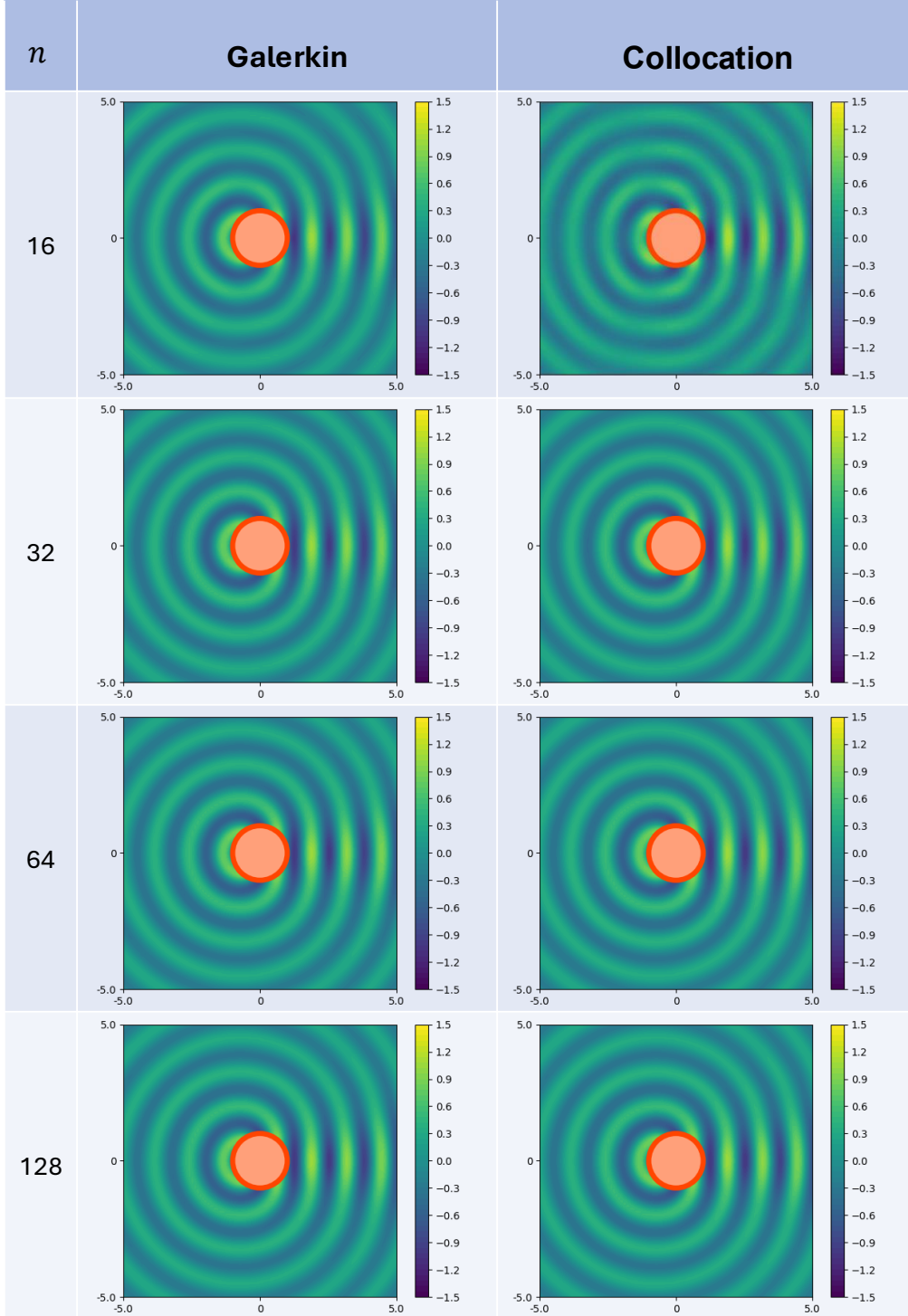


Table 18: **Scattering by a sound-soft unit disc with double-layer kernel.** Real components of the scattered potential field computed using the Galerkin formulation (left) and the collocation baseline (right).

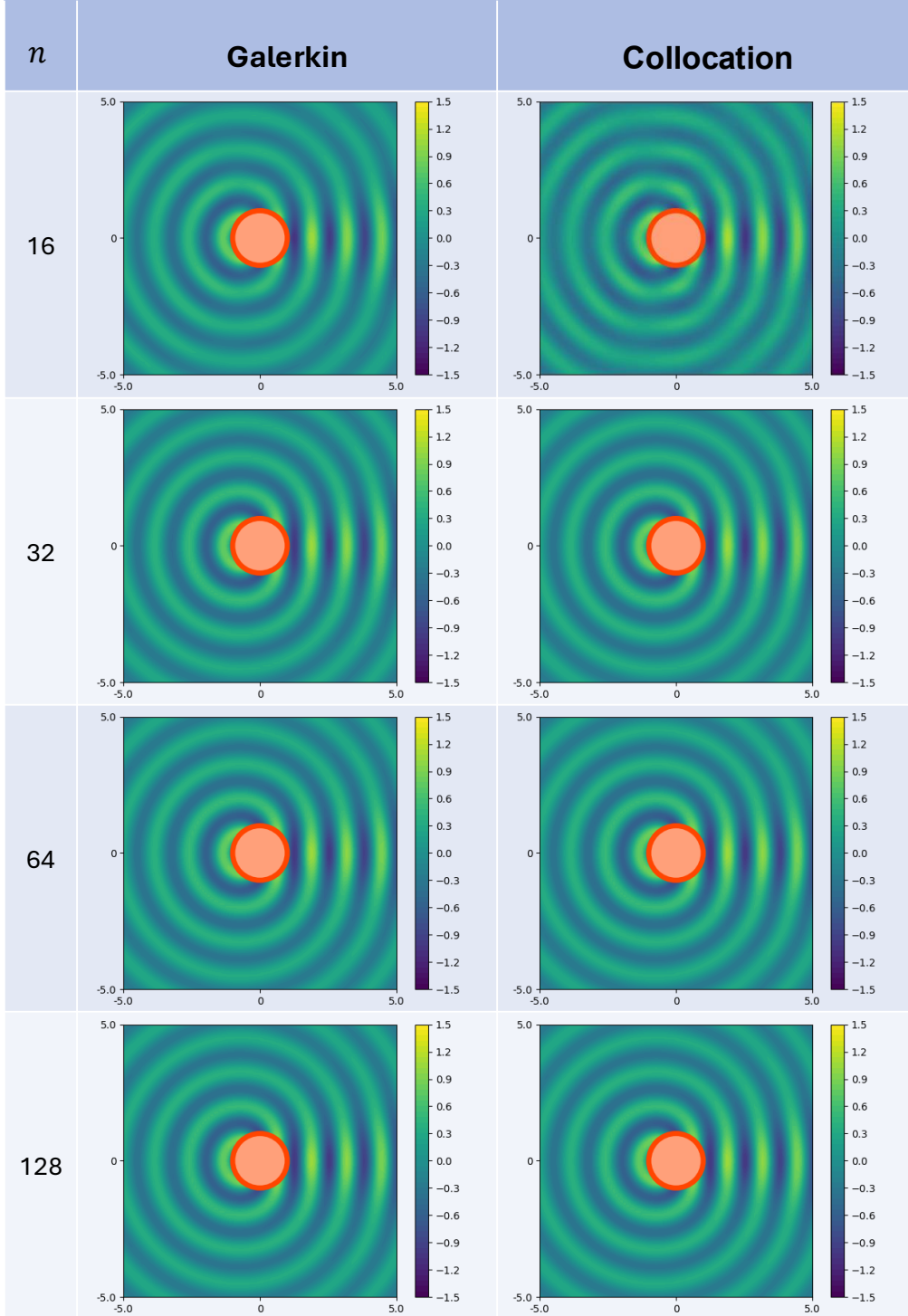


Table 19: **Scattering by a sound-soft unit disc with combined kernel.** Real components of the scattered potential field computed using the Galerkin formulation (left) and the collocation baseline (right).

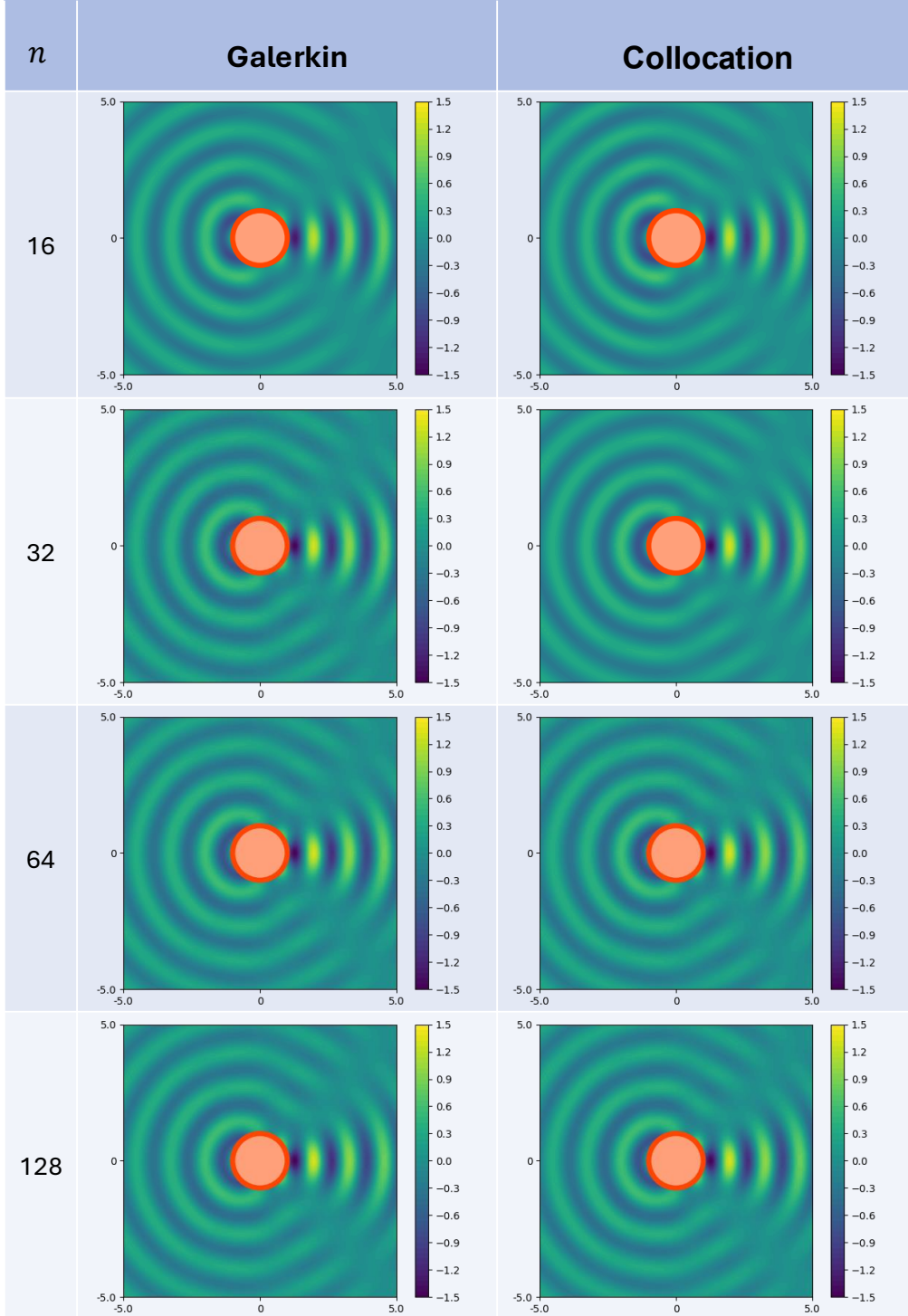


Table 20: **Scattering by a sound-hard unit disc with single-layer kernel.** Real components of the scattered potential field computed using the Galerkin formulation (left) and the collocation baseline (right).

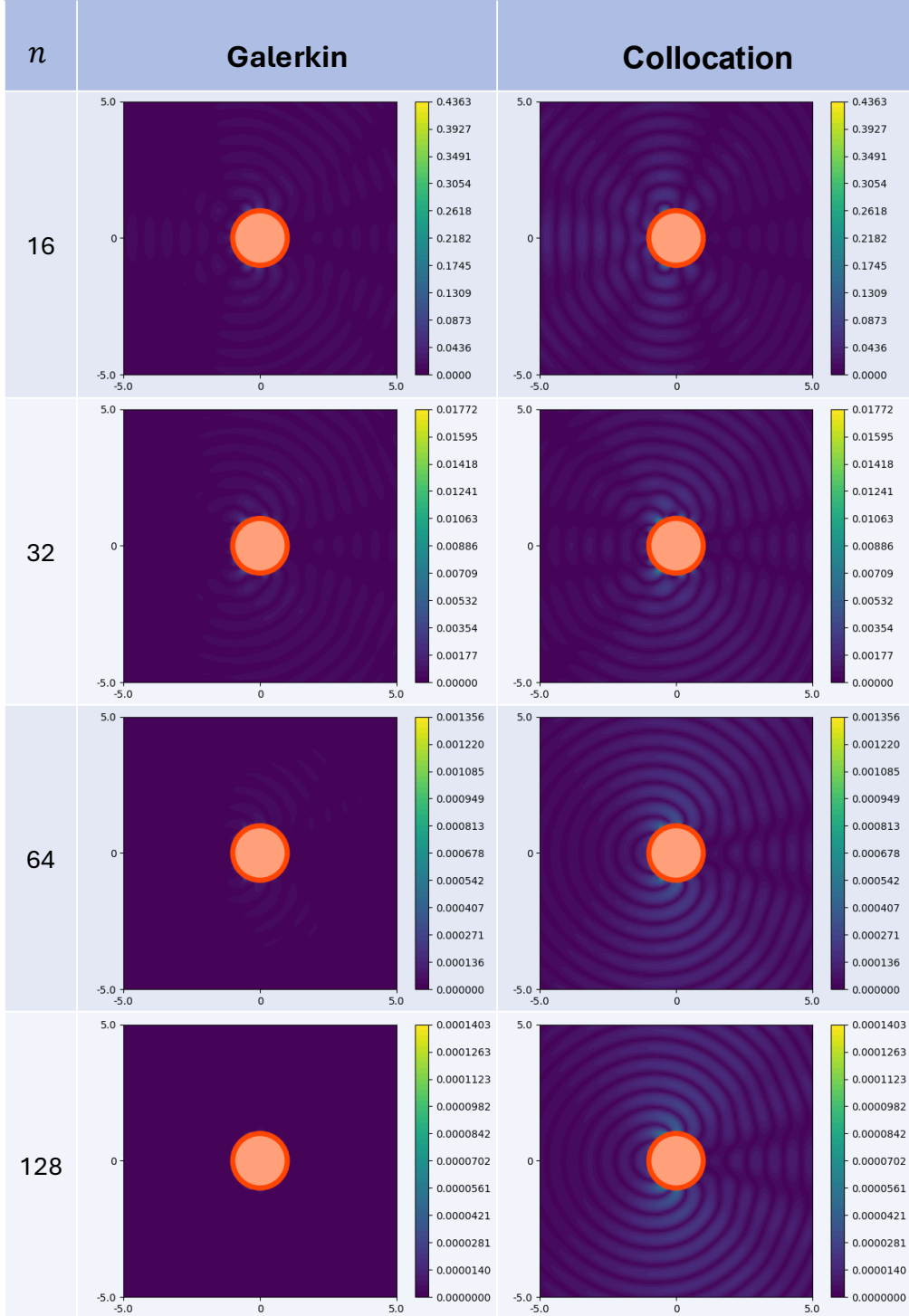


Table 21: **Absolute error of the field scattered by a sound-soft unit disc obtained using the single-layer kernel.** Pointwise absolute error ϵ_n of the reconstructed scattered wave field obtained via the Galerkin method (left) and the boundary collocation method (right).

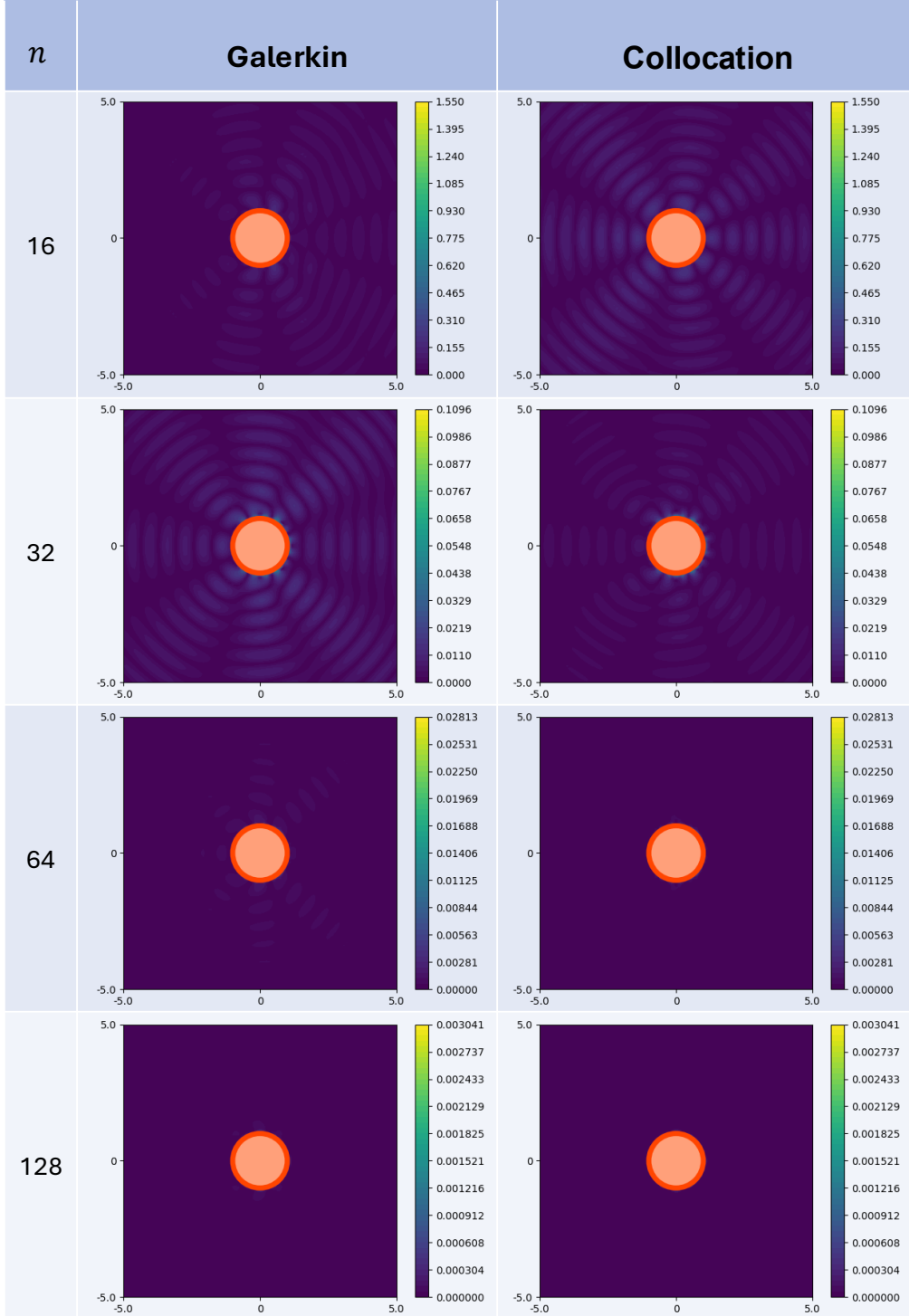


Table 22: **Absolute error of the field scattered by a sound-soft unit disc obtained using the double-layer kernel.** Pointwise absolute error ϵ_n of the reconstructed scattered wave field obtained via the Galerkin method (left) and the boundary collocation method (right).

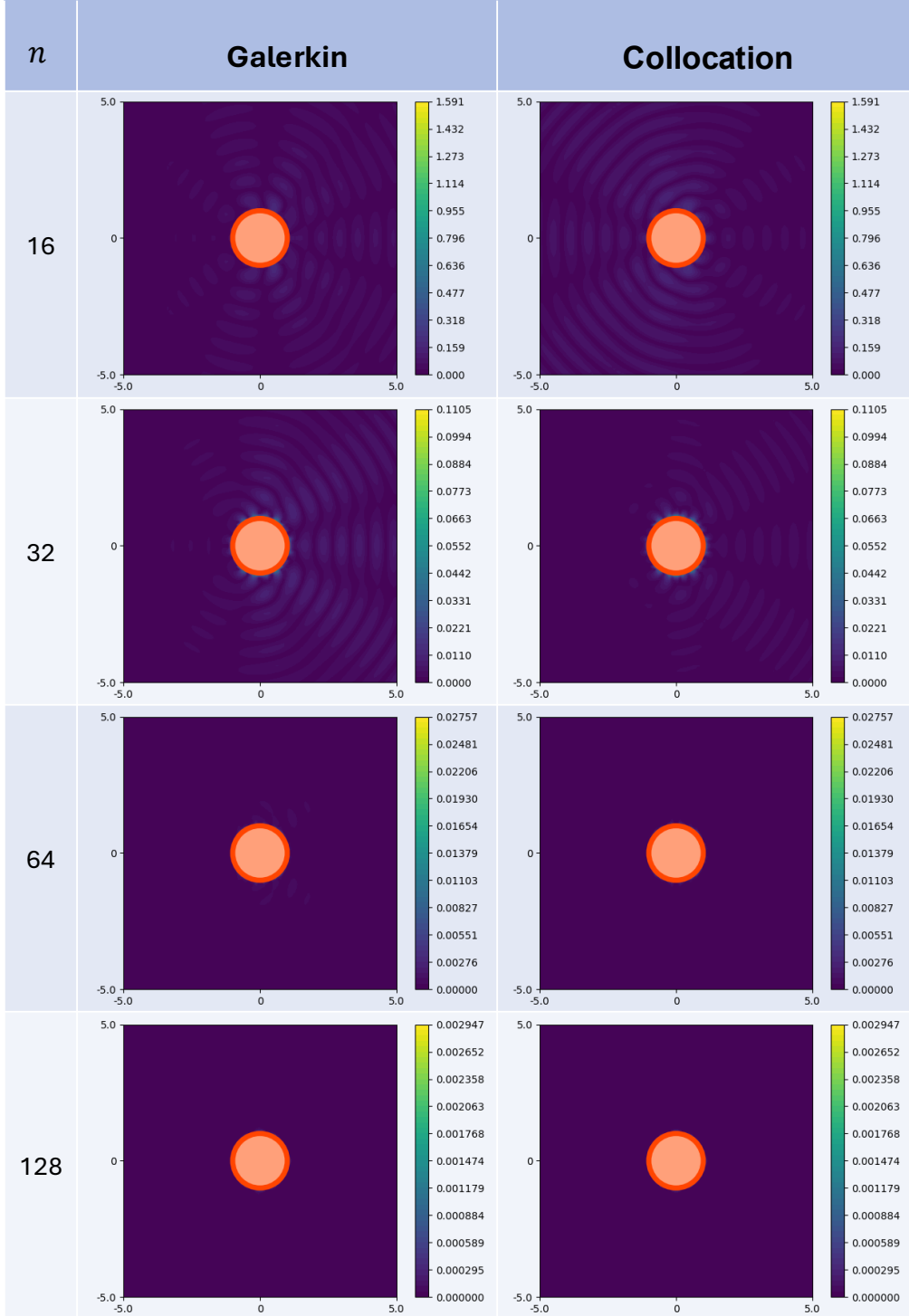


Table 23: **Absolute error of the field scattered by a sound-soft unit disc obtained using the combined kernel.** Pointwise absolute error ϵ_n of the reconstructed scattered wave field obtained via the Galerkin method (left) and the boundary collocation method (right).

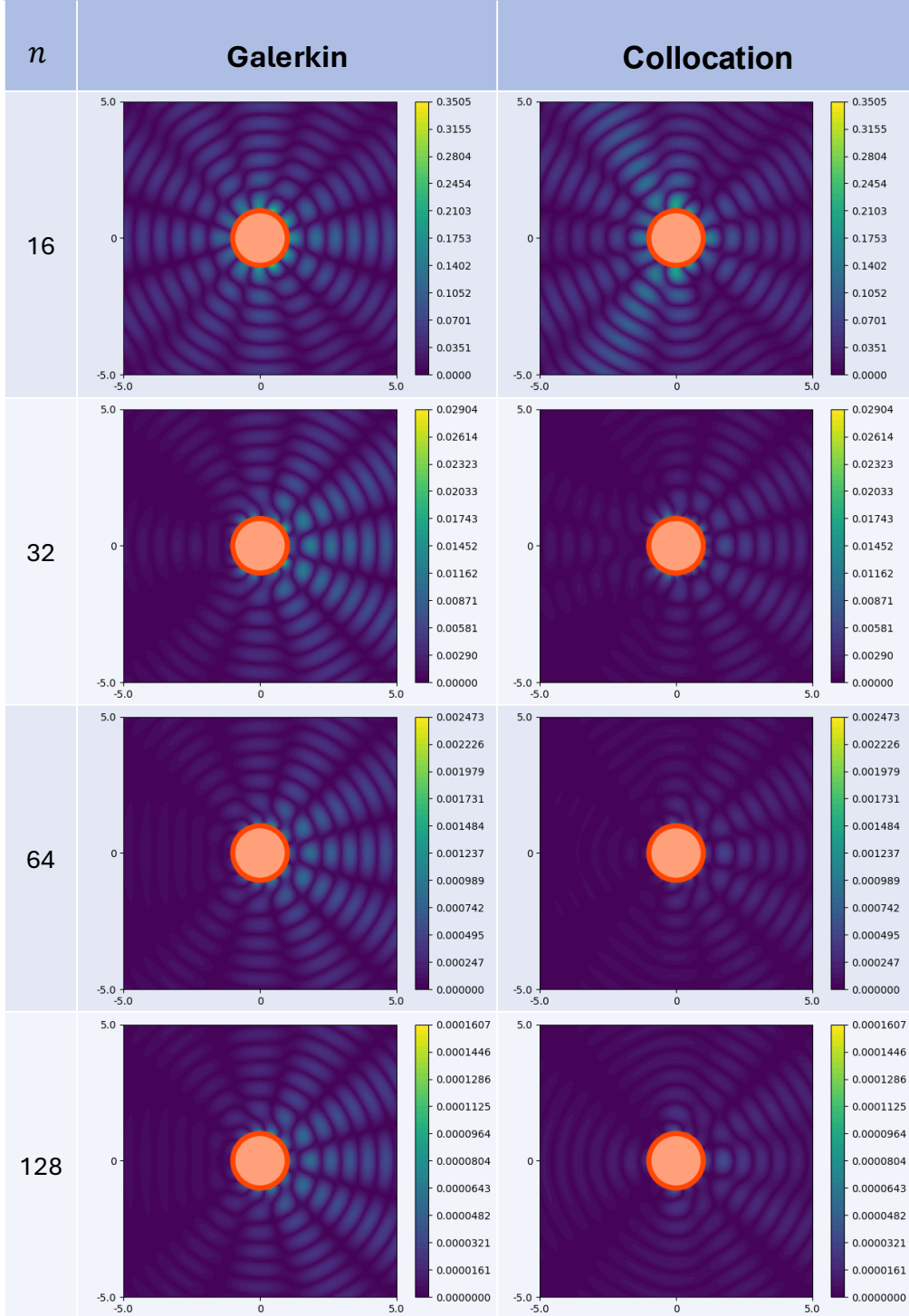


Table 24: **Absolute error of the field scattered by a sound-hard unit disc obtained using single-layer kernel.** Pointwise absolute error ϵ_n of the reconstructed scattered wave field obtained via the Galerkin method (left) and the boundary collocation method (right).

Table 25: **Global Mean Absolute Error of the Acoustic Scattering Problem on a Sound-Soft Unit Disc**

Kernel	n	Mean Absolute Error	
		Collocation	Galerkin
SL	16	6.541 ₋₃	1.343 ₋₂
	32	6.098 ₋₄	2.730 ₋₄
	64	6.418 ₋₅	1.256 ₋₅
	128	5.428 ₋₇	7.571 ₋₆
DL	16	4.859 ₋₂	2.385 ₋₂
	32	3.814 ₋₃	1.987 ₋₃
	64	2.671 ₋₄	1.379 ₋₄
	128	9.617 ₋₆	1.802 ₋₅
DLSL	16	3.773 ₋₂	2.650 ₋₂
	32	2.477 ₋₃	1.471 ₋₃
	64	1.810 ₋₄	1.108 ₋₄
	128	9.111 ₋₆	1.273 ₋₅

Table 26: **Global Mean Absolute Error of the Acoustic Scattering Problem on a Sound-Hard Unit Disc**

Kernel	n	Mean Absolute Error	
		Galerkin	Collocation
SL	16	2.819 ₋₂	2.915 ₋₂
	32	9.558 ₋₄	1.772 ₋₃
	64	6.605 ₋₅	1.273 ₋₄
	128	8.211 ₋₆	5.240 ₋₆

3.5 Verification of Plane Wave Incident Fields Scattering by Pre-Fractal Koch Snowflakes

Following successful verification using a plane wave incident field scattered by unit discs with circle boundaries, the validation pipeline continued by solving the Helmholtz acoustic scattering problem on Koch snowflake boundaries. The experimental setups and evaluation metrics established for this verification process are detailed in Section 3.5.1.

The corresponding numerical results are analyzed and discussed in Section 3.5.2.

3.5.1 Methodology

The developed Galerkin solver was subsequently evaluated under plane-wave incident fields mapped over a highly non-smooth fractal boundary geometry. Specifically, the robustness and performance of the variational framework were investigated using the following configurations of the Koch snowflake domain:

- Acoustic scattering by a sound-soft Koch snowflake governed by a Dirichlet boundary condition,
- Acoustic scattering by a sound-hard Koch snowflake governed by a Neumann boundary condition.

For the sound-soft Koch snowflake, the homogeneous Dirichlet constraint mapped the boundary trace via $u^{\text{sca}} = -u^{\text{inc}}$ using the incident plane wave from Equation (127). The sound-hard Koch snowflake with the Neumann boundary condition $\partial_\nu u^{\text{sca}} = -\partial_\nu u^{\text{inc}}$ on Γ was evaluated via Equation (130).

While closed-form analytical solutions are readily available for canonical geometries like the unit disc, the analytical solutions for highly complex geometries such as the fractal Koch snowflake are not yet established. For manifolds lacking an analytical reference, the numerical verification was rigorously performed via self-convergence analysis. This approach monitors the behavior of the discrete solutions across successive levels of systematic grid refinement to compute the EOC. If the variational routines of the Galerkin framework are implemented correctly, the solution differences between consecutive mesh densities $\tilde{n}$ should diminish predictably, exhibiting a stable Cauchy convergence profile and a well-behaved, asymptotic EOC.

The numerical verification utilized a pre-fractal Koch snowflake boundary at iteration $i = 1$ and evaluated across various mesh refinement levels $\tilde{n}$. The configuration was defined by a wave number of $\kappa = 5$. Under the chosen discretization, each linear boundary element (face) contributed two coefficients ($\delta = 2$) and the local boundary DOF was $\delta_\Gamma = 3$. The generated Koch snowflake geometry based on Algorithm 2 was shifted by $(-0.5, -0.25)^\top$. For the sound-soft boundary configuration governed by the Dirichlet boundary condition, the framework evaluated three distinct boundary integral operator architectures: the single-, double-, and the combined-layer formulation. Conversely, the sound-hard boundary under the Neumann boundary condition was modeled exclusively

via the single-layer potential representation.

The first phase of the validation involved a qualitative analysis via visual inspection of the scattered field and the absolute difference. The pointwise absolute difference is defined by

$$E_{\tilde{n},j} = \left| \operatorname{Re} (u_{\tilde{n}}^{\text{sca}}(\mathbf{p}_j)) - \operatorname{Re} (u_{\tilde{n}+2}^{\text{sca}}(\mathbf{p}_j)) \right|, \quad (135)$$

where $\mathbf{p}_j$ denotes the evaluation point of index j within the exterior region in $[-5, 5] \times [-5, 5]$, and $u_{\tilde{n}}^{\text{sca}}$ represents the scattered field obtained from the Galerkin approximation at refinement level $\tilde{n}$.

For the quantitative validation, the mean absolute difference $\overline{E}_{\tilde{n}}$ between two successive mesh refinements is defined by

$$\overline{E}_{\tilde{n}} = \frac{1}{N_{\text{eval}}} \sum_{j=1}^{N_{\text{eval}}} E_{\tilde{n},j}, \quad (136)$$

where N_{eval} denotes the total number of evaluation points $\mathbf{p}_j$.

The quantitative validation was also performed by analyzing the asymptotic rate of convergence. To determine the asymptotic rate of convergence, the Estimated Order of Convergence ($\text{EOC}_{\tilde{n}}$) was computed based on the ratio of consecutive errors relative to the increased boundary density. The $\text{EOC}_{\tilde{n}}$ is formulated as

$$\text{EOC}_{\tilde{n}} = \frac{\log (\overline{E}_{\tilde{n}} / \overline{E}_{\tilde{n}+2})}{\log (N(i, \tilde{n}+2) / N(i, \tilde{n}))}, \quad (137)$$

where $N(i, \tilde{n})$ is the mapping function that determines the total number of boundary points given the Koch snowflake generation iteration i and the segment refinement density $\tilde{n}$, as defined previously in Equation (116).

The results of this verification are presented and discussed in Section 3.5.2

3.5.2 Results and Discussions

The primary objective of this section is to present and critically evaluate the numerical verification of the developed Galerkin framework using the plane wave incident fields on Koch snowflake domains via the self-convergence analysis.

Visual inspections of the scattered field in Figures 20 through 27 shows that the macroscopic wave fields produced by the coarser meshes exhibit strong visual agreement with those computed on the finer resolution. This close alignment indicates that the implemented Galerkin method maintains structural and physical consistency, reliably capturing core scattering phenomena such as the shadow zones, geometric interference, and directional scattering patterns even at lower spatial resolutions. The absolute difference fields compiled from Figure 28 through Figure 35 indicate that the variation between consecutive wave field solutions decay monotonically under systematic grid refinement. This consistent reduction in the absolute error topology visually demonstrates that the numerical approximations are stabilizing. As the boundary mesh density $\tilde{n}$ increases, the discrete solution traces behave as a Cauchy sequence, confirming the asymptotic consistency of the developed variational framework on these complex domains.

Tables 27 through 30 present the $\overline{E}_{\tilde{n}}$ and the $\text{EOC}_{\tilde{n}}$ values obtained across the various mesh refinement levels. The empirical data strictly adhere to the expected theoretical behavior. Specifically, the mean absolute difference decreased monotonically as the boundary discretization was refined, while the EOC stabilized toward a constant asymptotic value, verifying Cauchy convergence.

This simultaneous reduction in error and stabilization of the convergence rate provides confirmation of the structural and mathematical correctness of the implemented Galerkin framework.

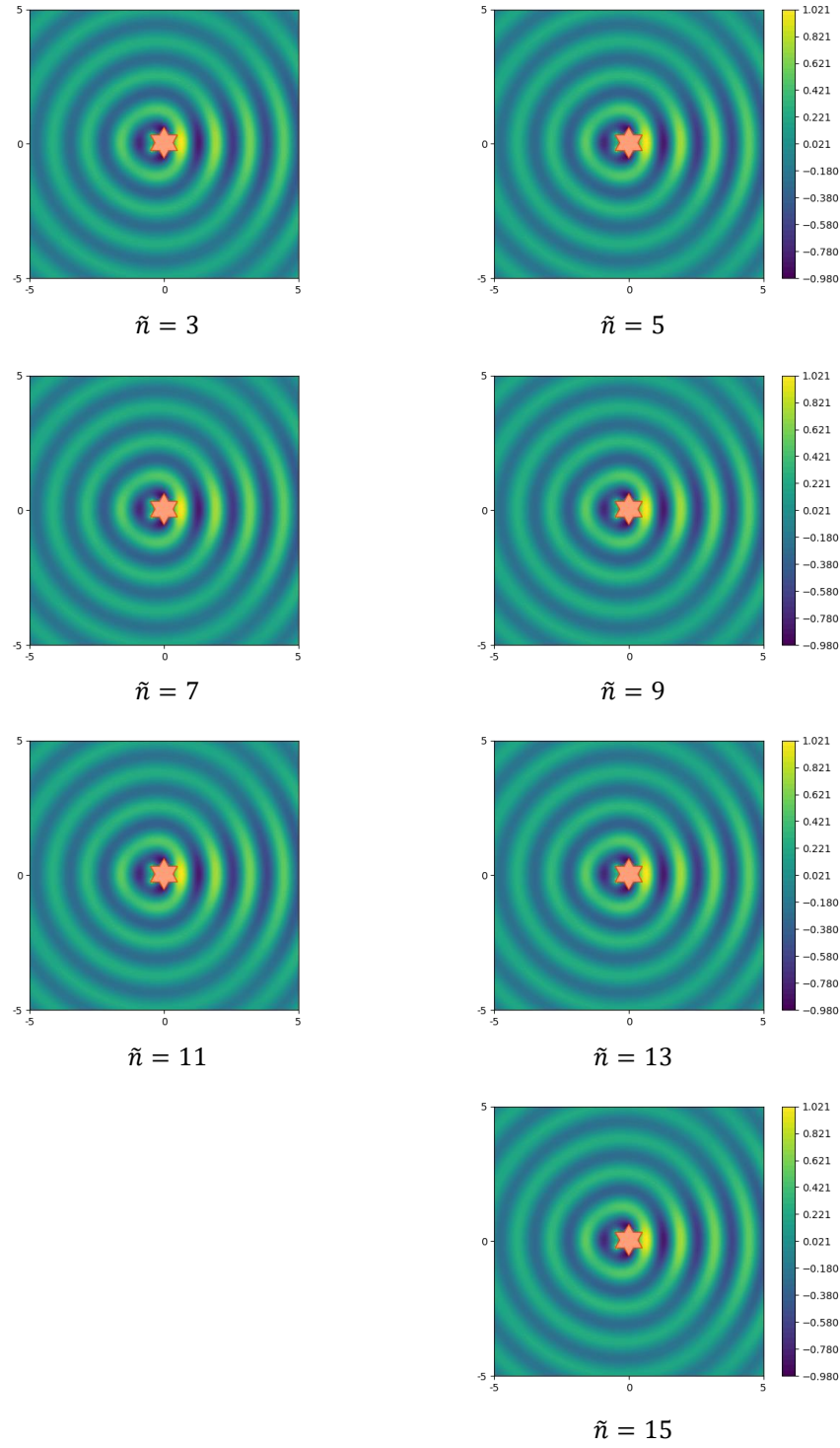


Figure 20: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a single-layer potential kernel.

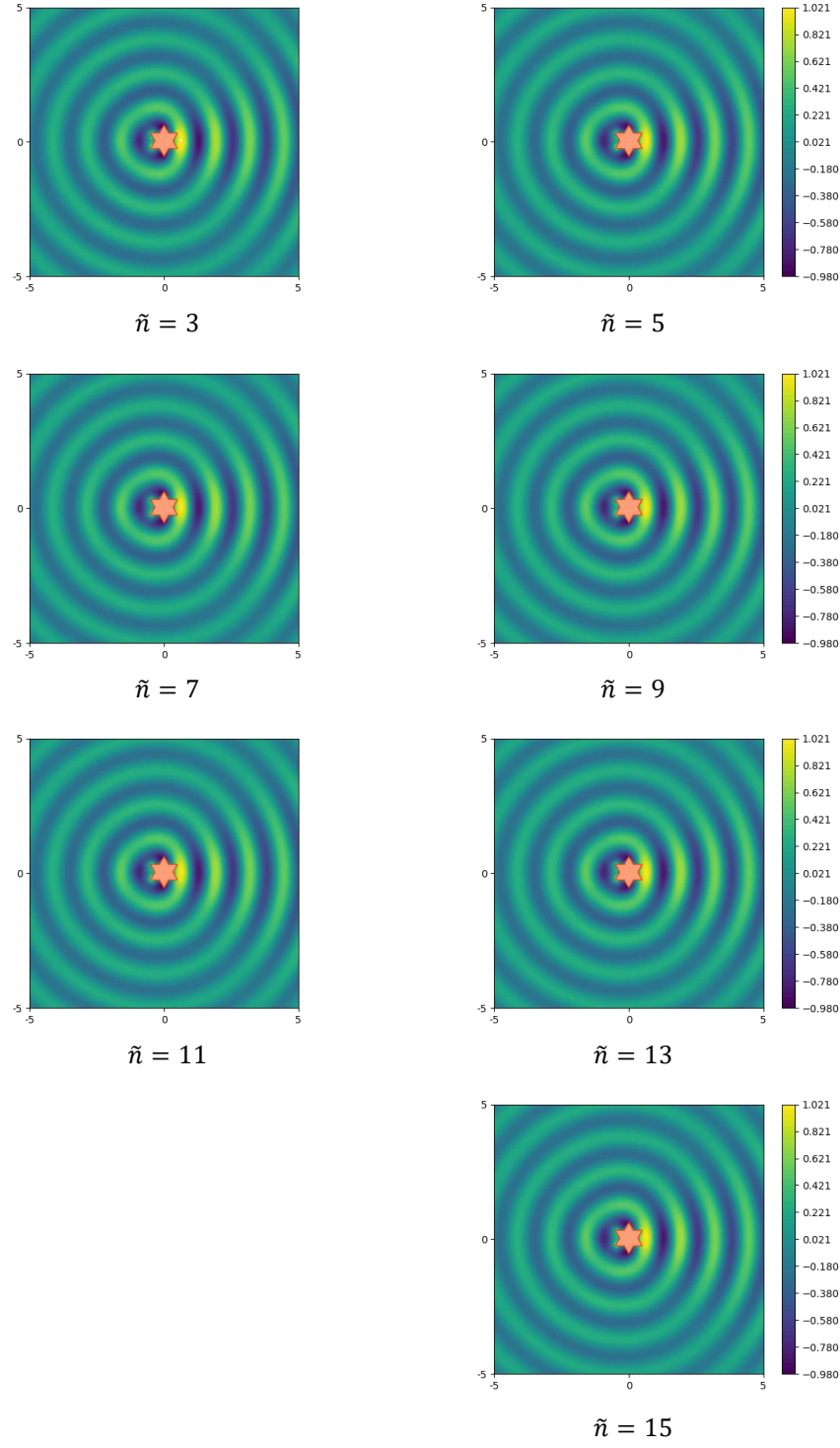


Figure 21: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a double-layer potential kernel.

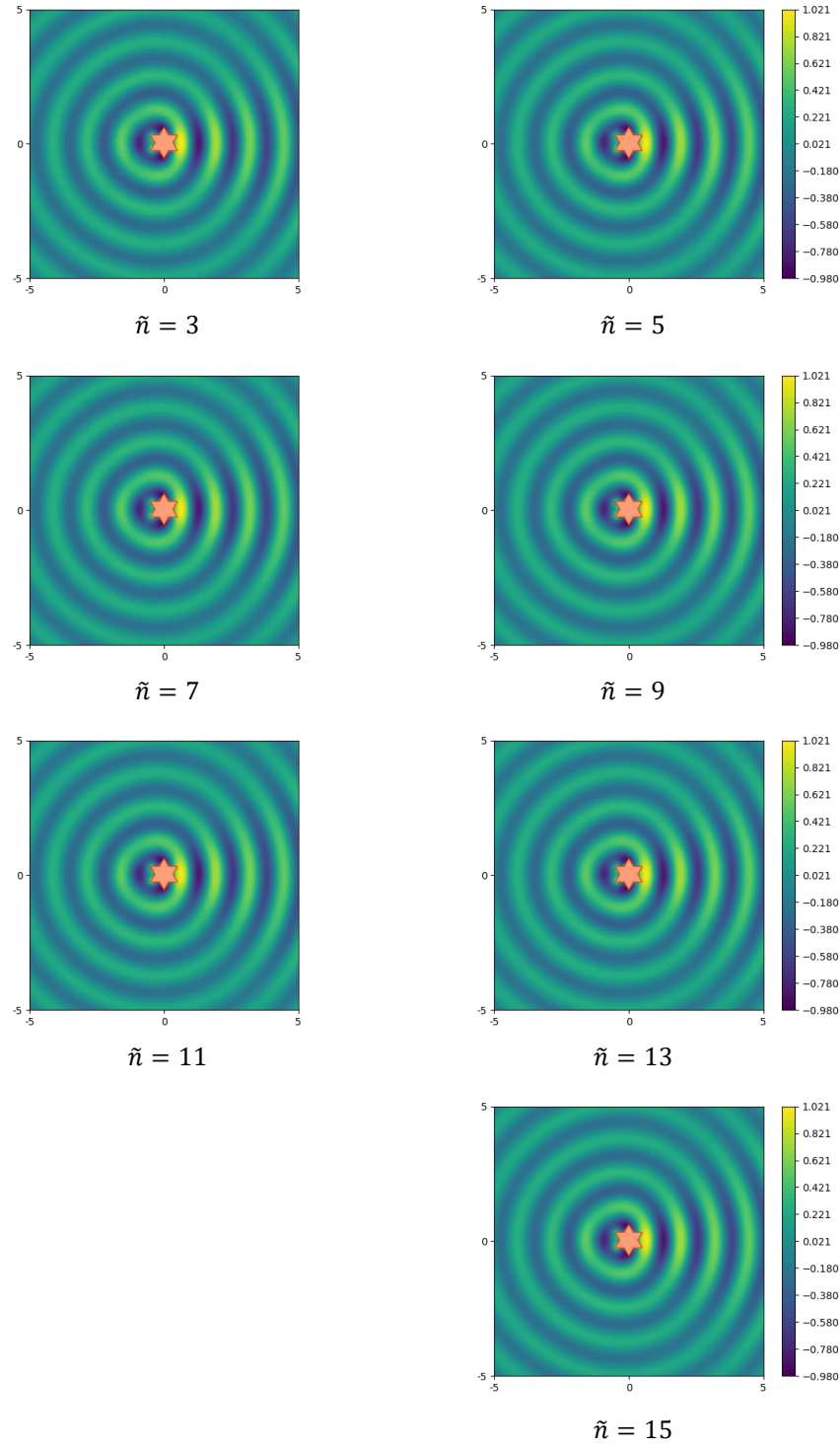


Figure 22: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a combined-layer potential kernel.

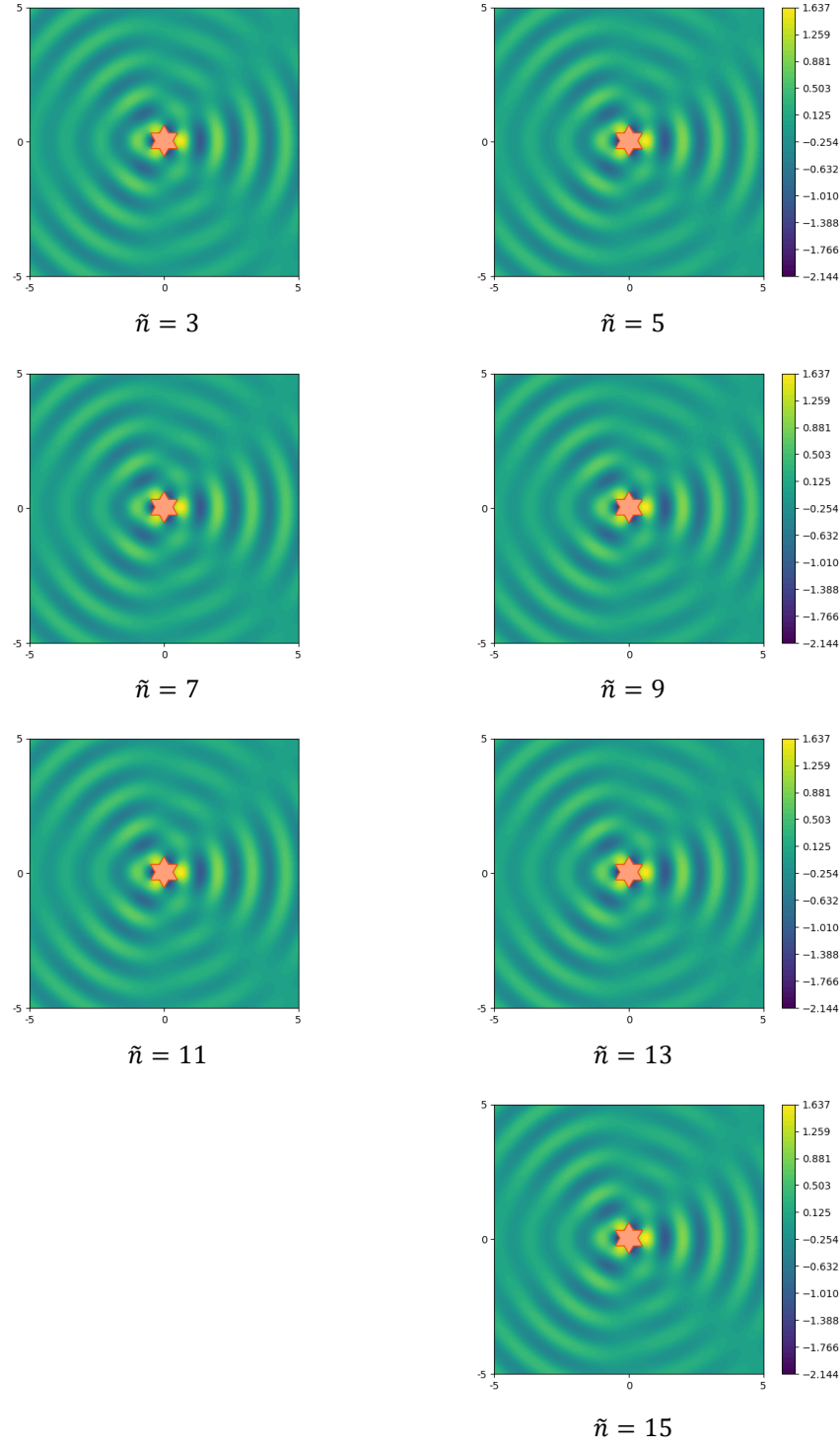


Figure 23: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 1$, Neumann boundary condition), computed using the Galerkin method with a single-layer potential kernel.

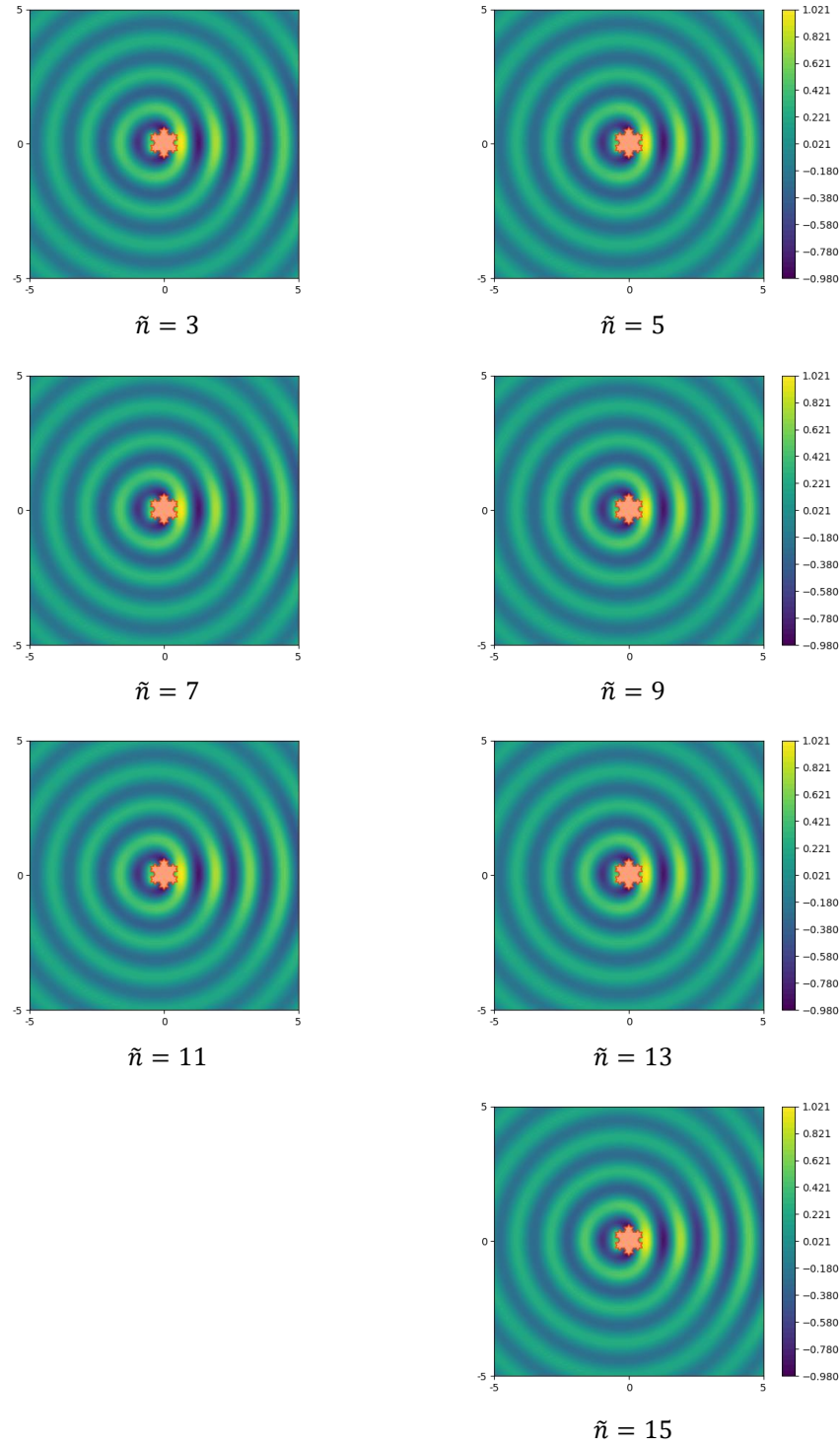


Figure 24: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a single-layer potential kernel.

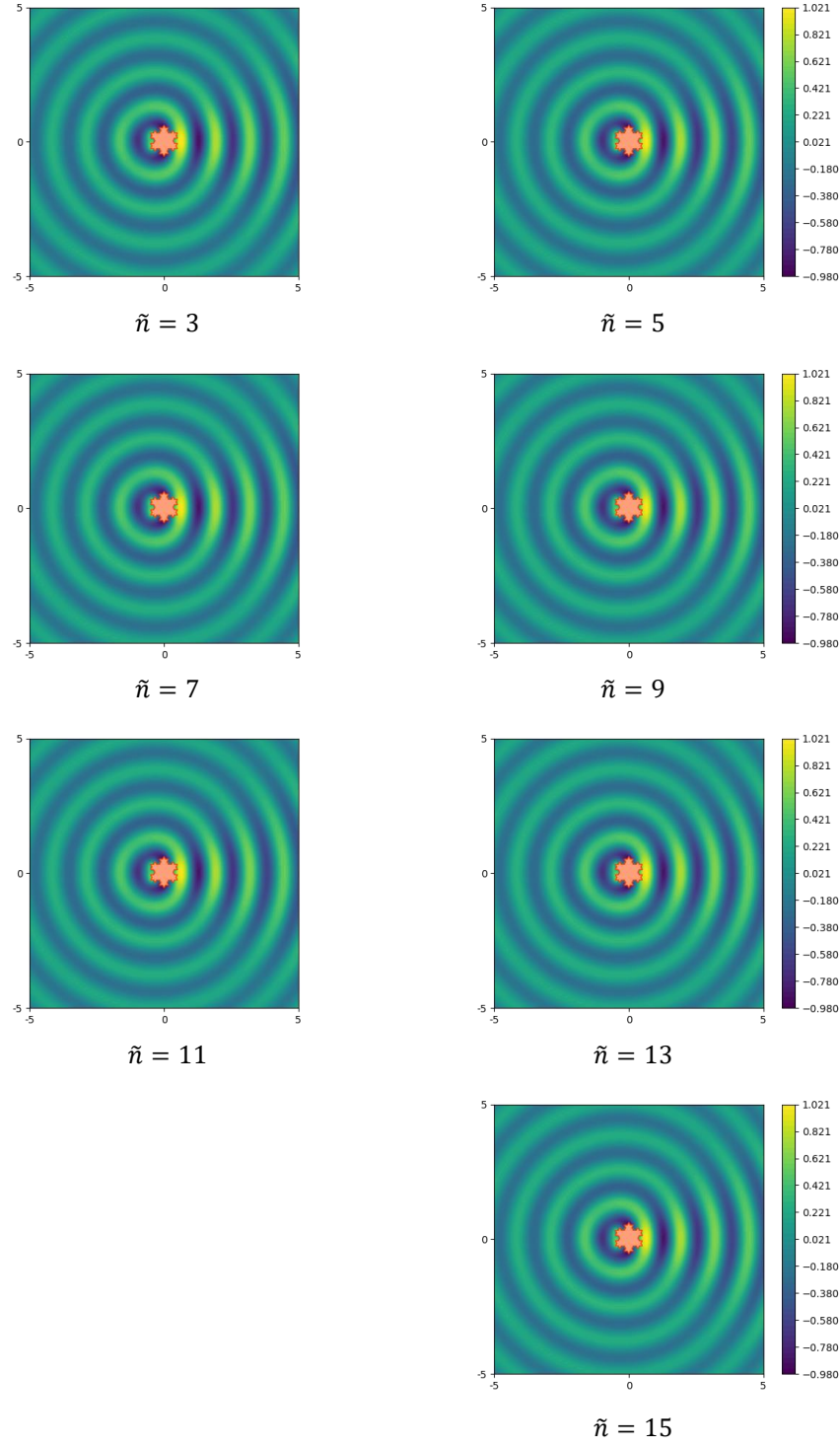


Figure 25: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a double-layer potential kernel.

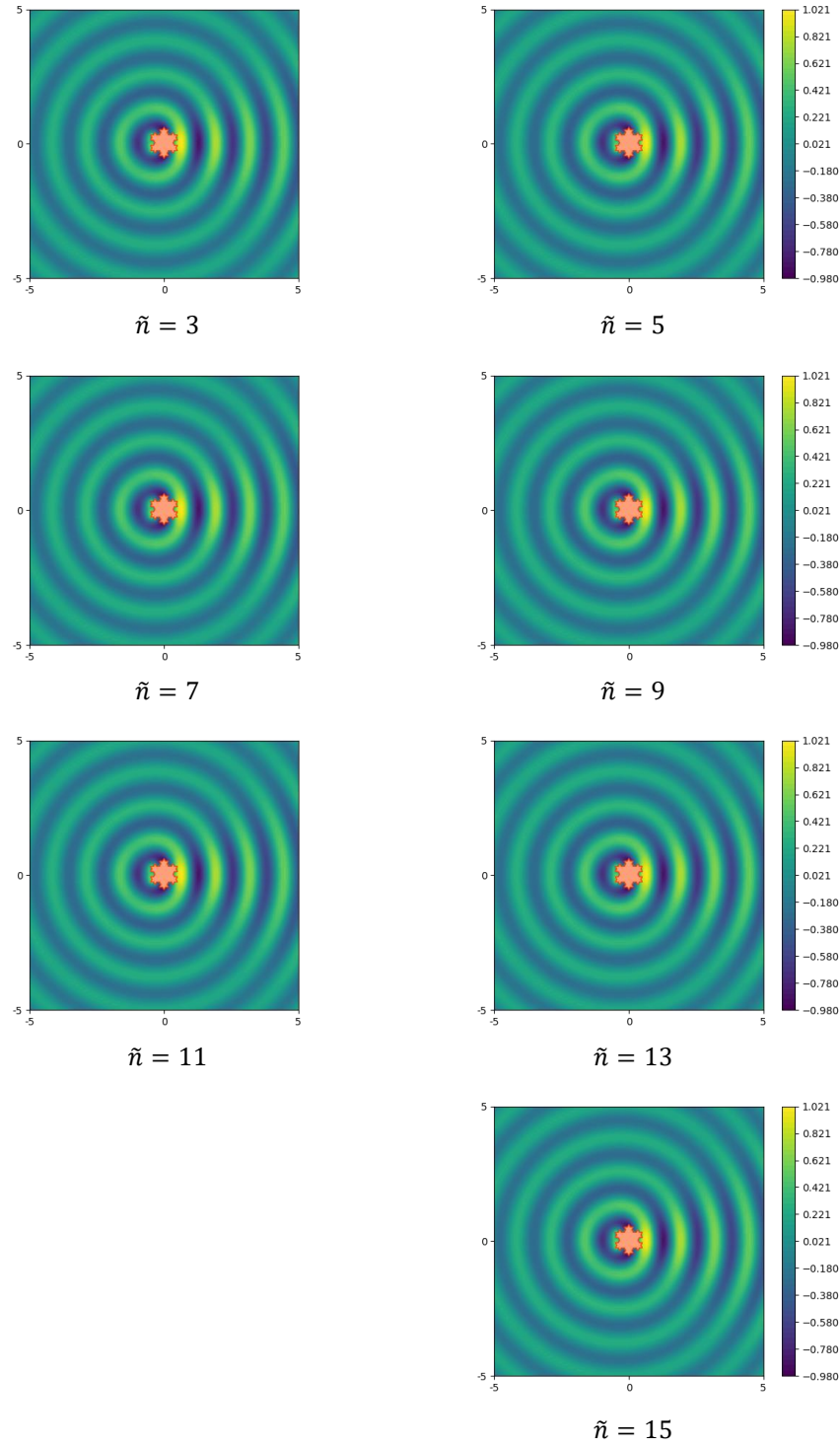


Figure 26: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a combined-layer potential kernel.

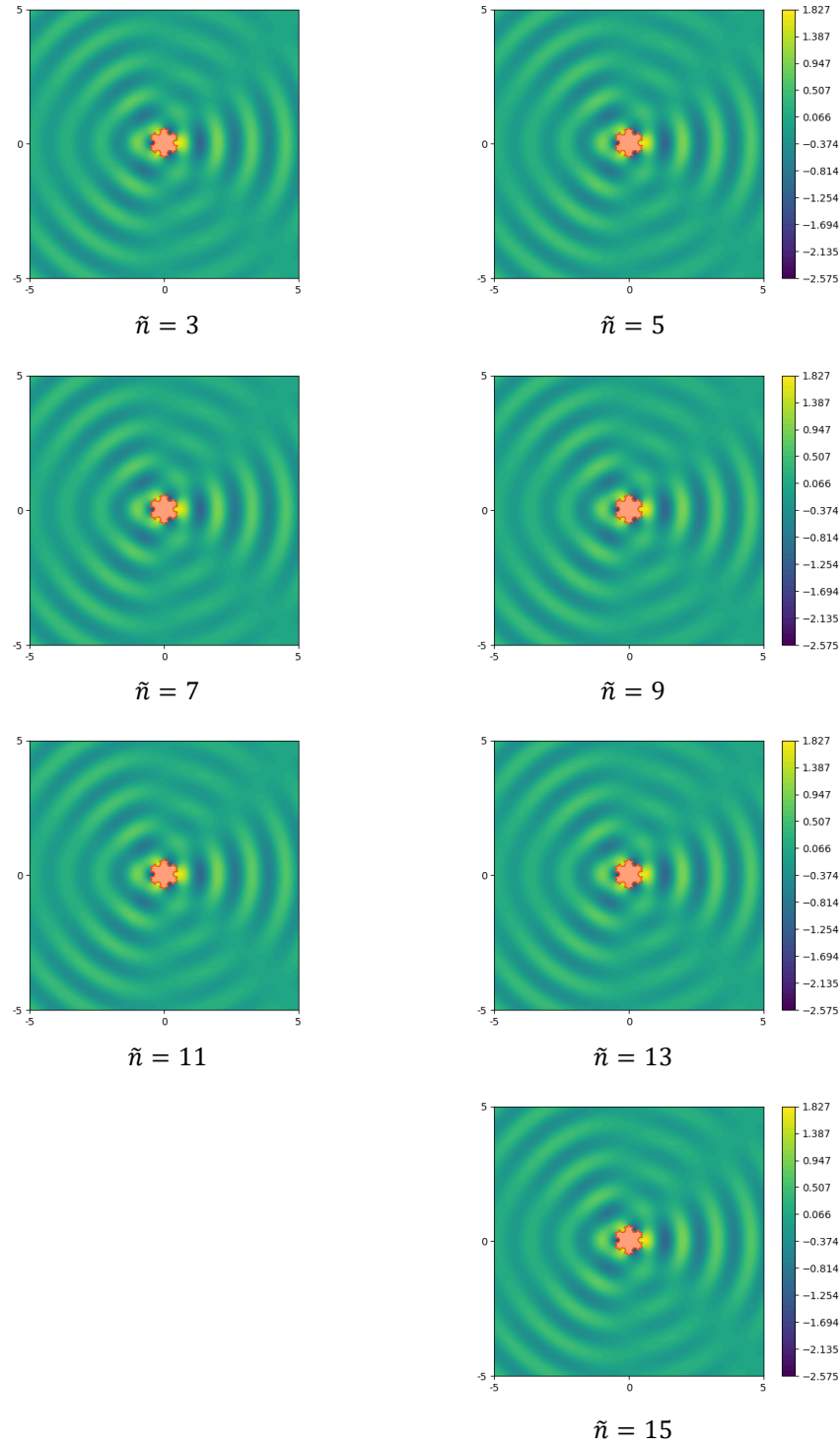


Figure 27: The real part of $u_{\tilde{n}}^{\text{sca}}$ on a Koch snowflake boundary (iteration $i = 2$, Neumann boundary condition), computed using the Galerkin method with a single-layer potential kernel.

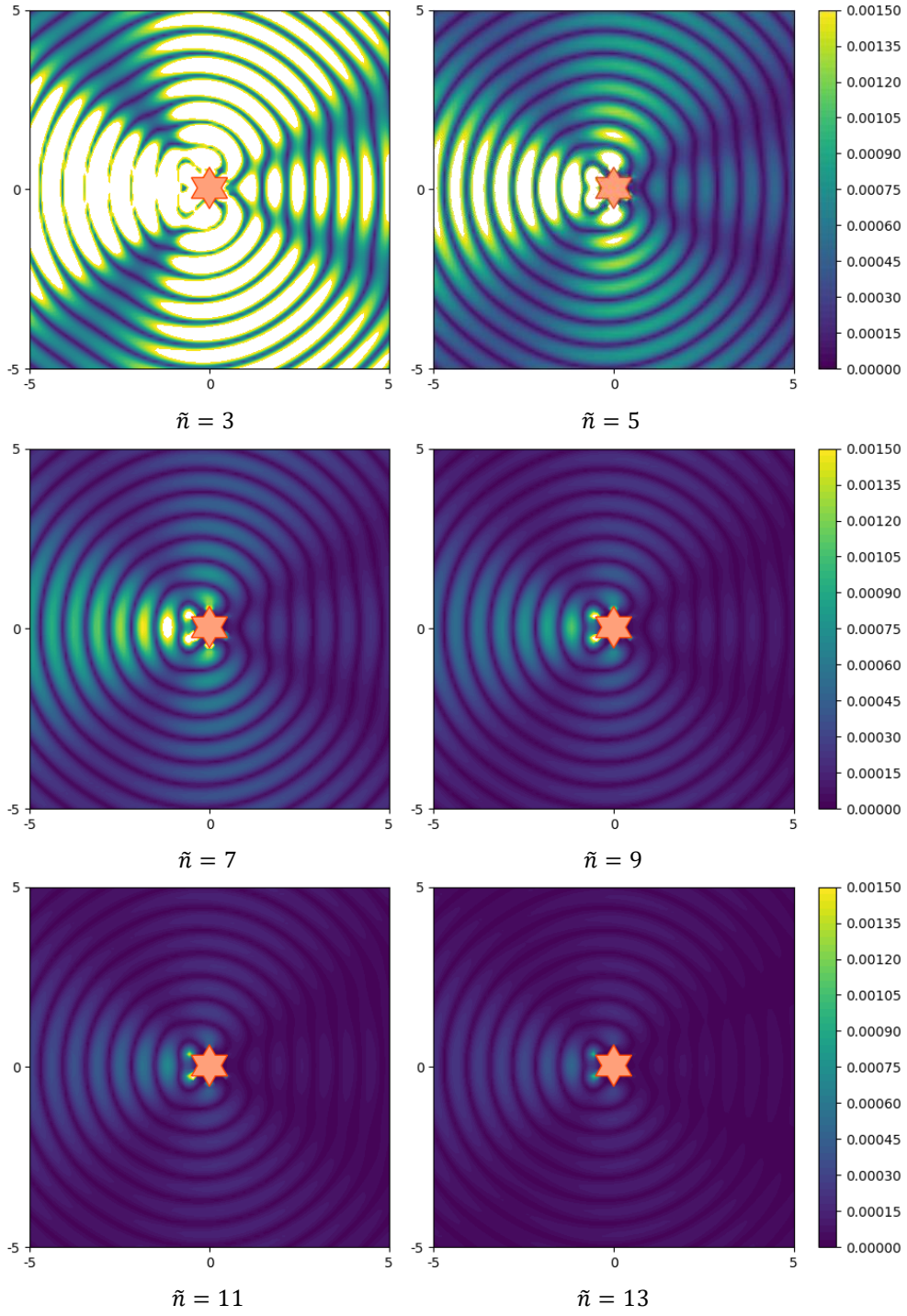


Figure 28: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a single-layer potential kernel.

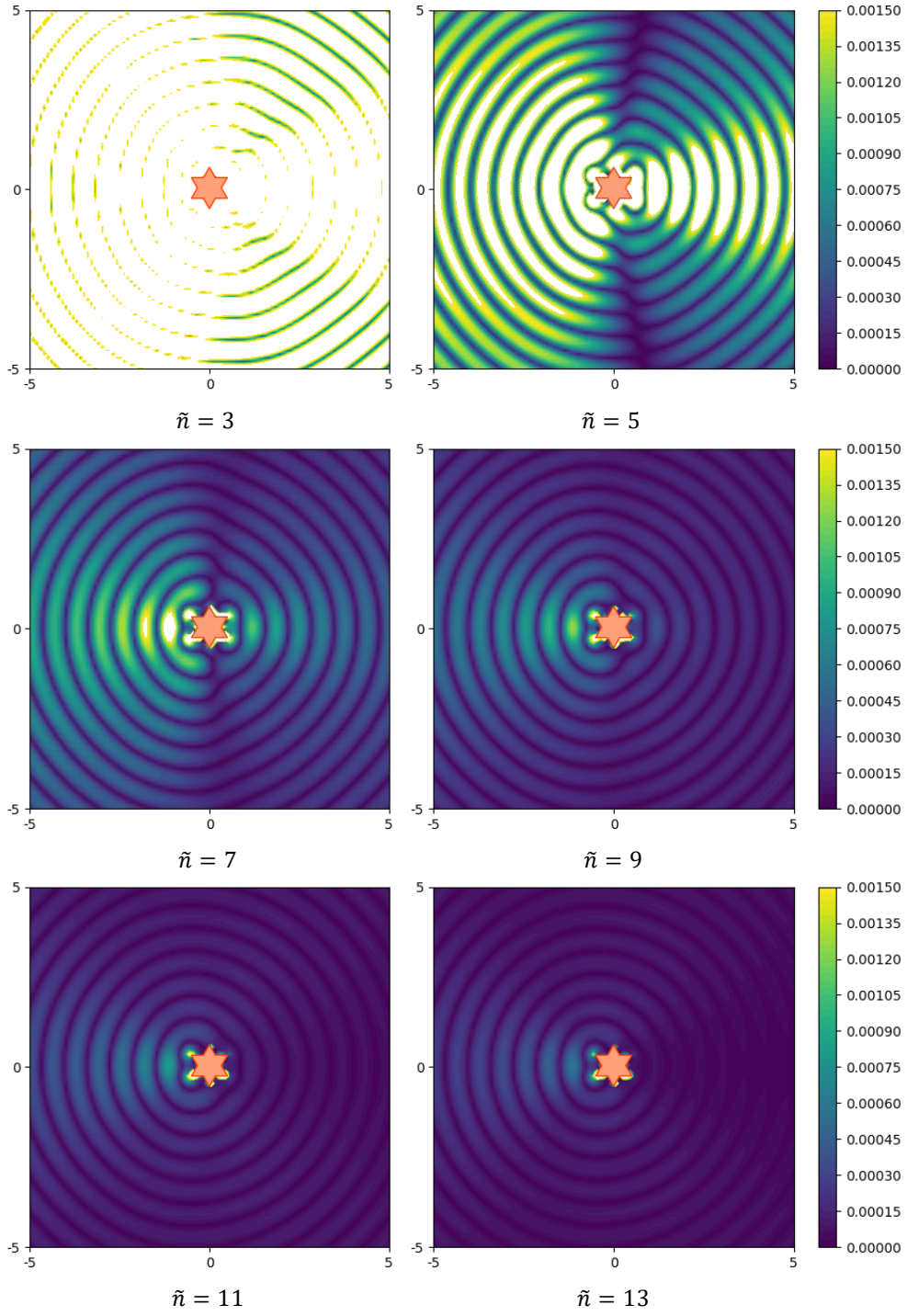


Figure 29: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a double-layer potential kernel.

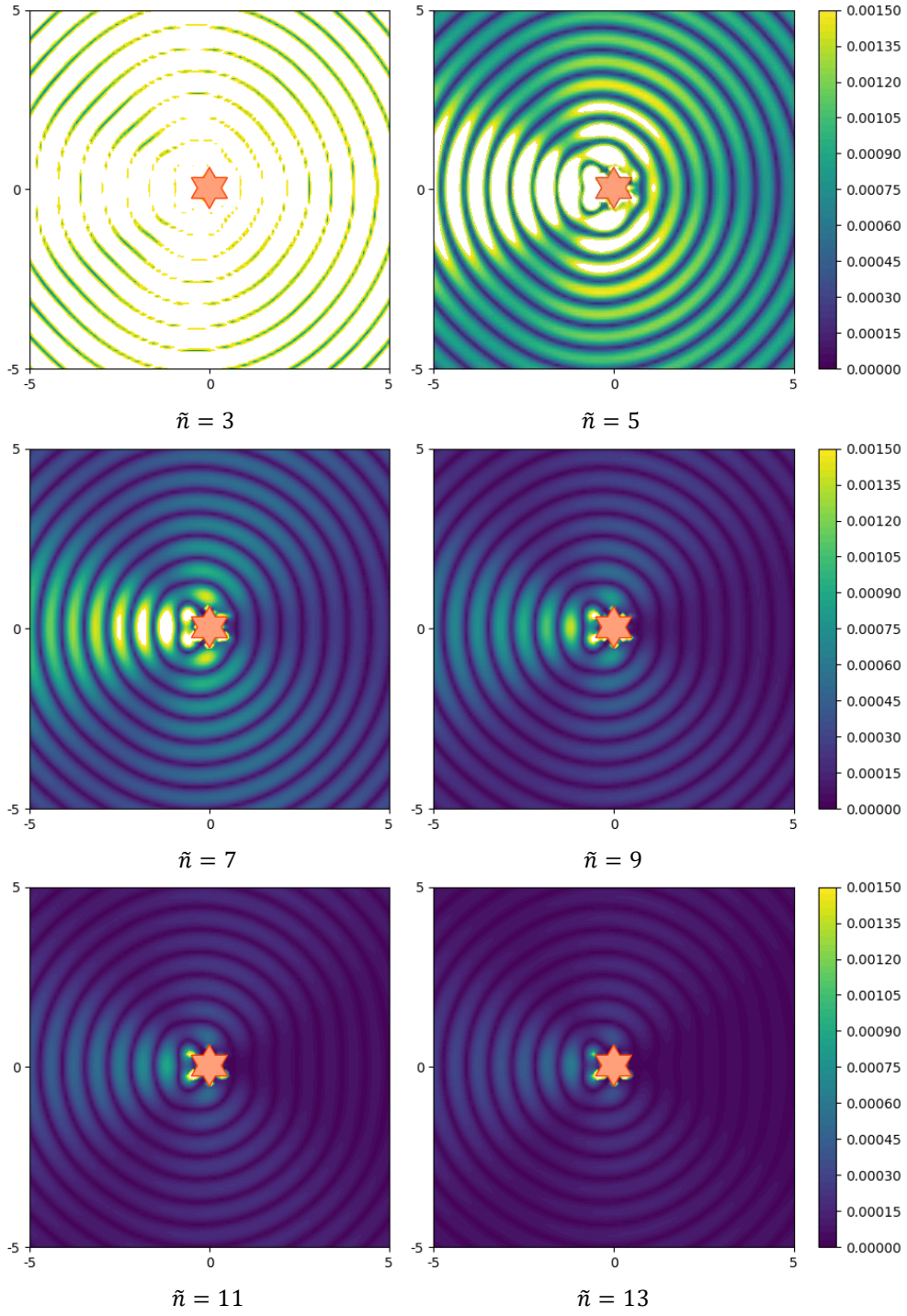


Figure 30: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 1$, Dirichlet boundary condition), computed using the Galerkin method with a combined-layer potential kernel.

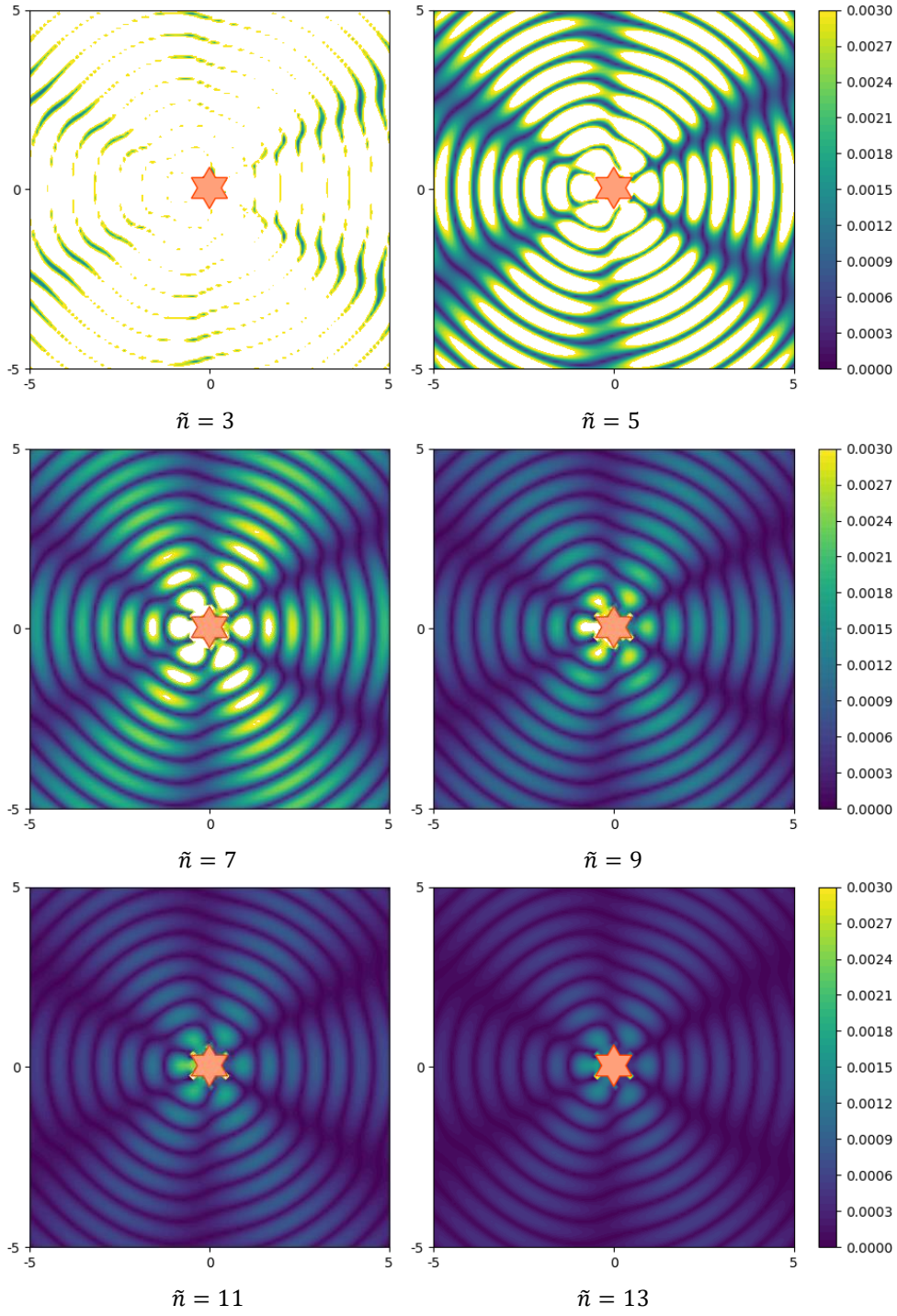


Figure 31: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 1$, Neumann boundary condition), computed using the Galerkin method with a single-layer potential kernel.

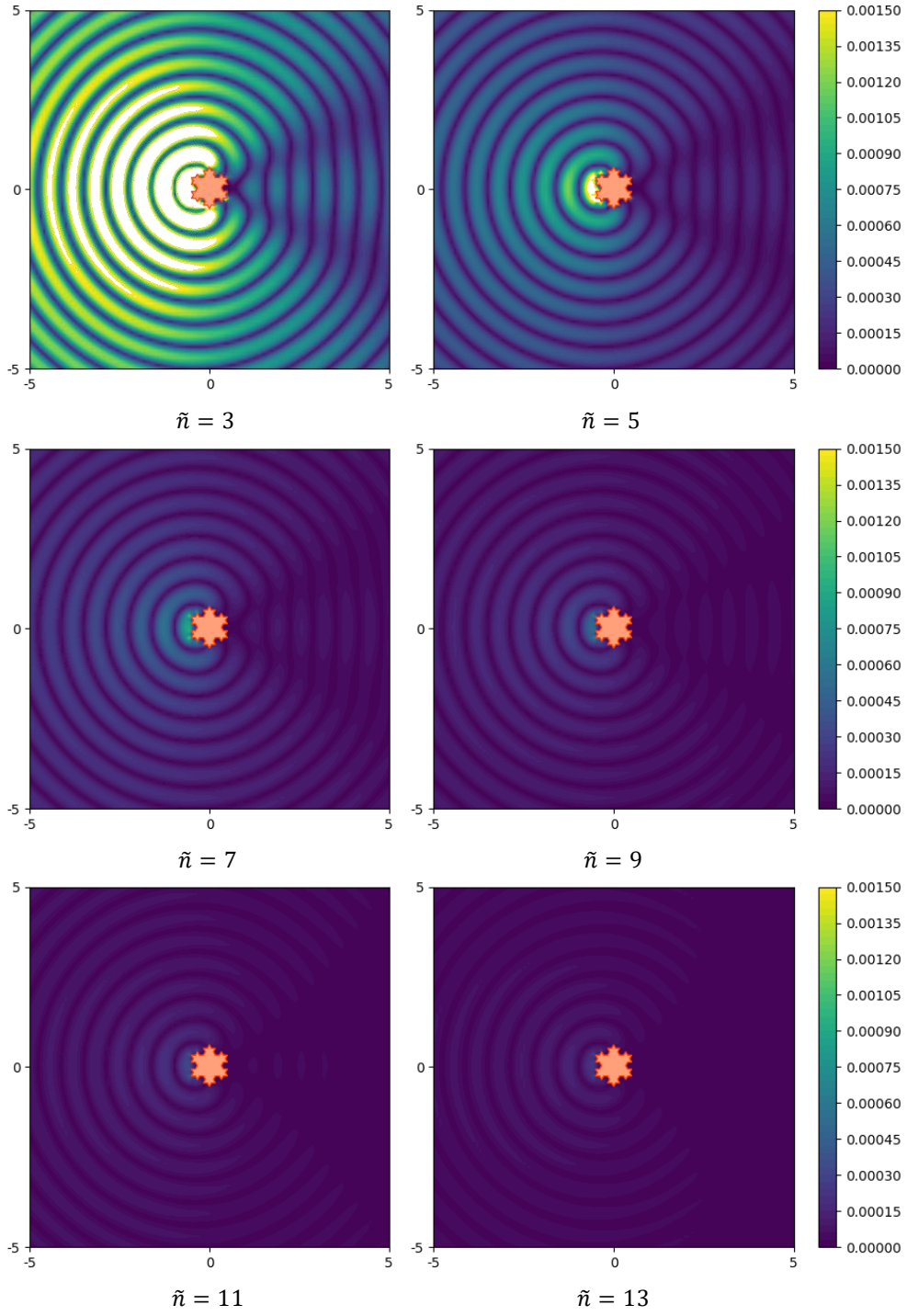


Figure 32: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a single-layer potential kernel.

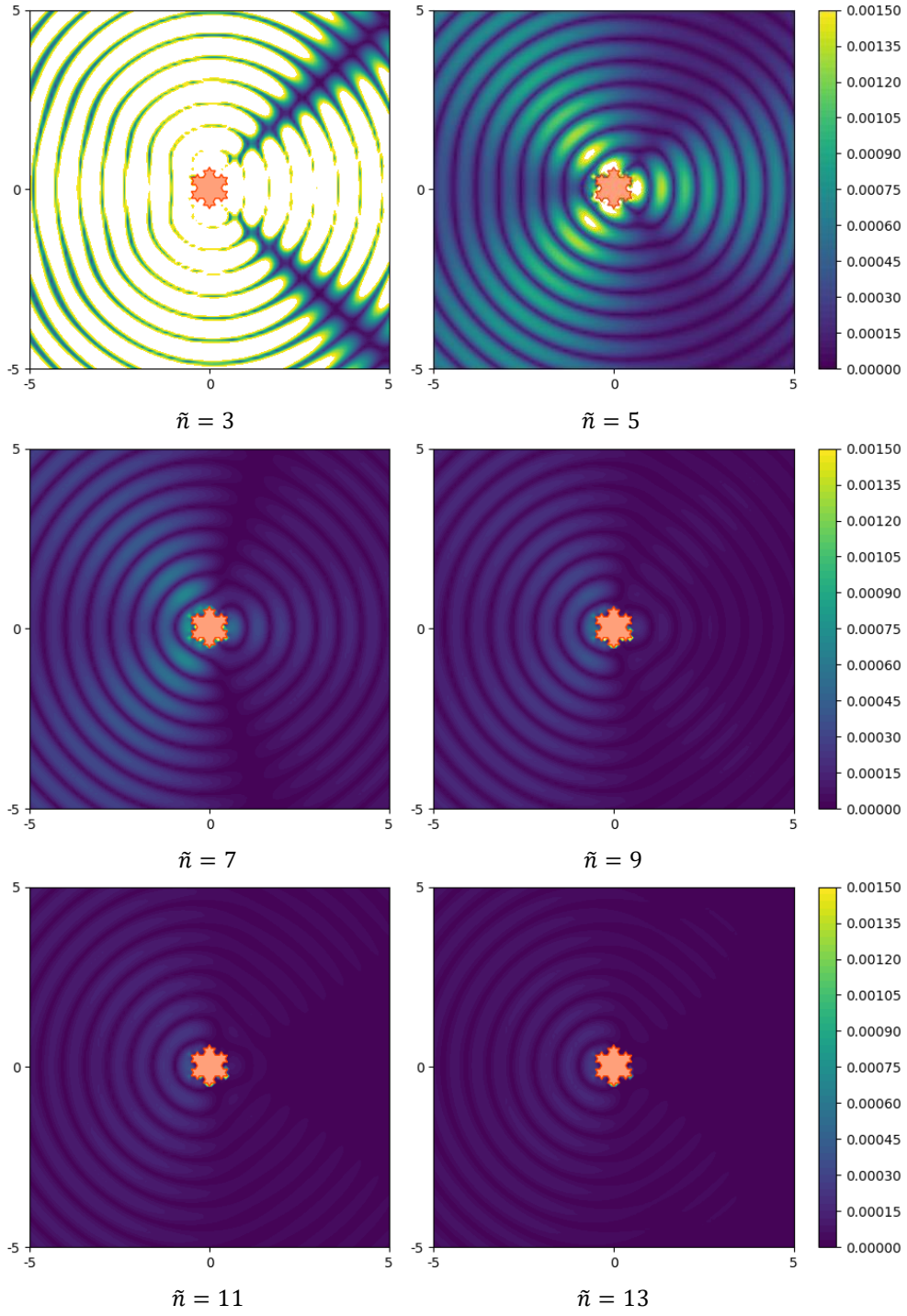


Figure 33: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a double-layer potential kernel.

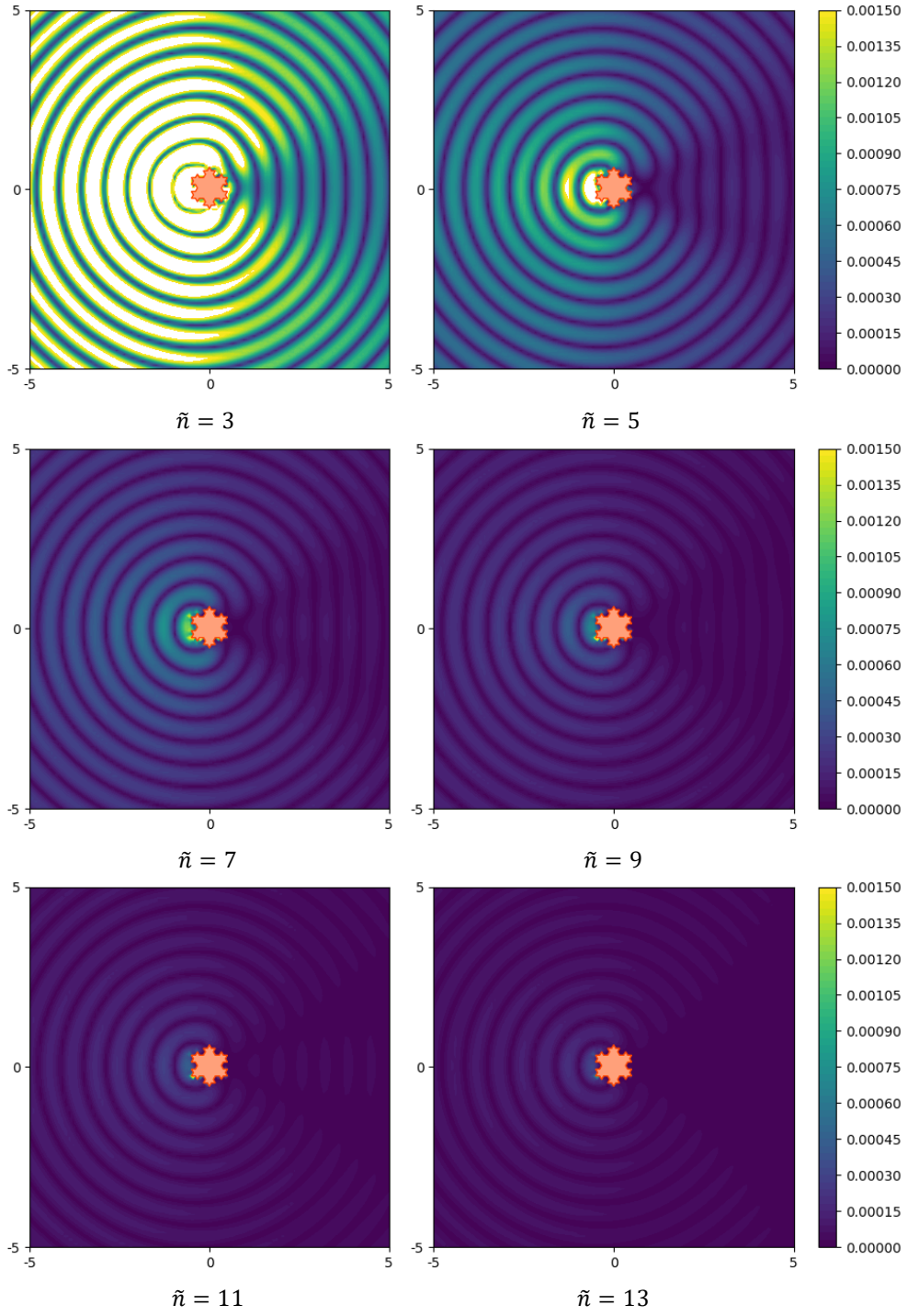


Figure 34: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 2$, Dirichlet boundary condition), computed using the Galerkin method with a combined-layer potential kernel.

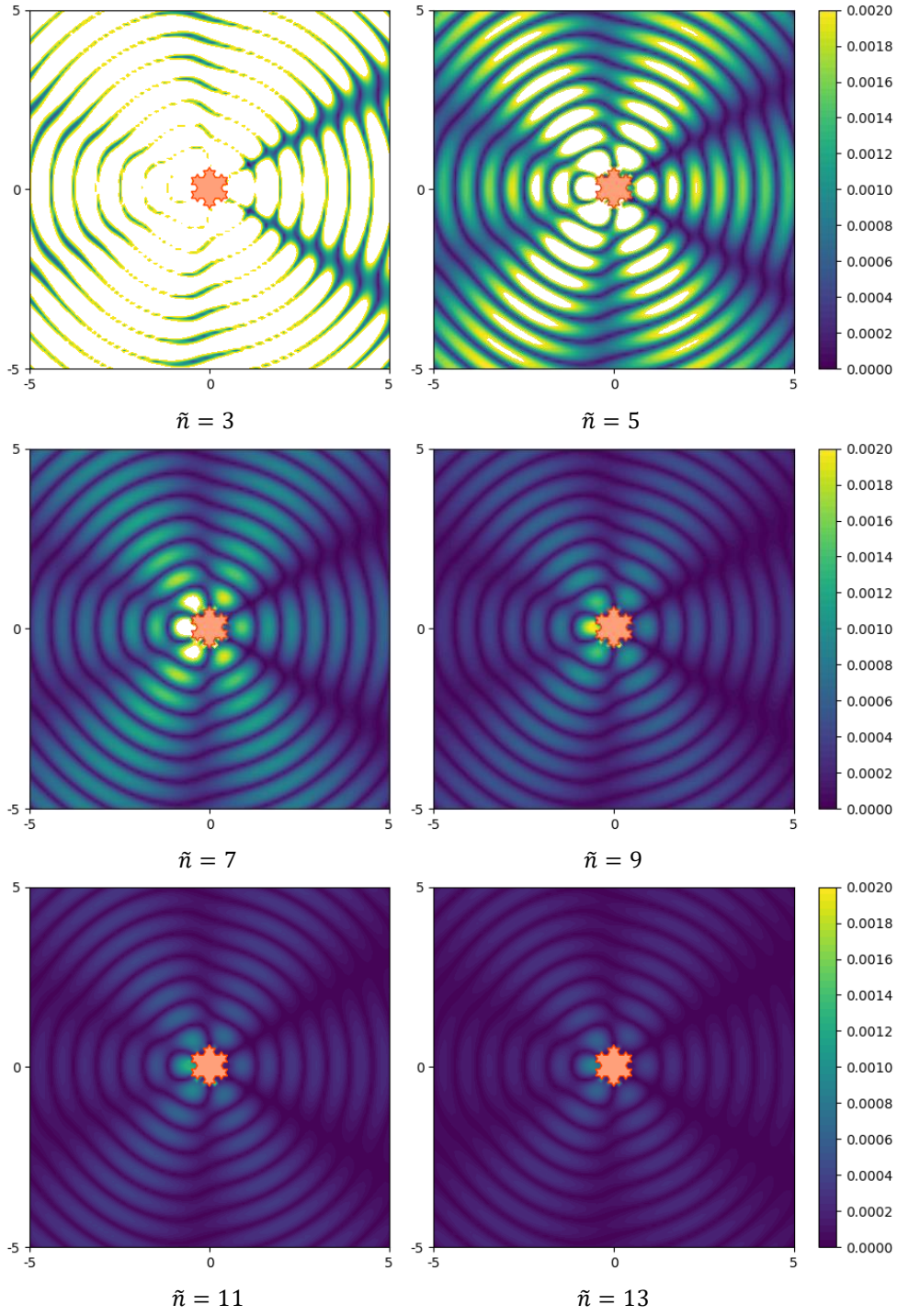


Figure 35: The absolute error field $E_{\tilde{n},j}$ on a Koch snowflake boundary (iteration $i = 2$, Neumann boundary condition), computed using the Galerkin method with a single-layer potential kernel.

Table 27: Mean Absolute Difference and Estimated Order of Convergence from the Single-Layer Potential for a Dirichlet Acoustic Scattering Problem on a Koch Snowflake

i	$\tilde{n}$	n	Mean Absolute Difference	Estimated Order of Convergence
1	3	24	1.593 ₋₃	1.569
	5	48	5.367 ₋₄	1.832
	7	72	2.554 ₋₄	1.952
	9	96	1.456 ₋₄	1.991
	11	120	9.340 ₋₅	2.089
2	3	24	7.514 ₋₄	1.599
	5	48	2.480 ₋₄	1.859
	7	72	1.167 ₋₄	1.951
	9	96	6.657 ₋₅	2.004
	11	120	4.257 ₋₅	2.039

Table 28: Mean Absolute Difference and Estimated Order of Convergence from the Double-Layer Potential for a Dirichlet Acoustic Scattering Problem on a Koch Snowflake

i	$\tilde{n}$	n	Mean Absolute Difference	Estimated Order of Convergence
1	3	24	6.207 ₋₃	2.698
	5	48	9.563 ₋₄	2.436
	7	72	3.562 ₋₄	2.136
	9	96	1.927 ₋₄	2.119
	11	120	1.201 ₋₄	1.973
2	3	24	2.824 ₋₃	2.959
	5	48	3.633 ₋₄	2.346
	7	72	1.403 ₋₄	1.979
	9	96	7.942 ₋₅	1.981
	11	120	5.105 ₋₅	1.973

Table 29: Mean Absolute Difference and Estimated Order of Convergence from the Combined-Layer Potential for a Dirichlet Acoustic Scattering Problem on a Koch Snowflake

i	$\tilde{n}$	n	Mean Absolute Difference	Estimated Order of Convergence
1	3	24	4.444 ₋₃	2.298
	5	48	9.037 ₋₄	2.183
	7	72	3.730 ₋₄	2.084
	9	96	2.048 ₋₄	2.072
	11	120	1.290 ₋₄	2.020
2	3	24	1.131 ₋₃	1.744
	5	48	3.377 ₋₄	1.863
	7	72	1.587 ₋₄	1.939
	9	96	9.084 ₋₅	1.993
	11	120	5.822 ₋₅	2.046

Table 30: Mean Absolute Difference and Estimated Order of Convergence from the Single-Layer Potential for a Neumann Acoustic Scattering Problem on a Koch Snowflake

i	$\tilde{n}$	n	Mean Absolute Difference	Estimated Order of Convergence
1	3	24	1.374 ₋₂	2.251
	5	48	2.886 ₋₃	2.249
	7	72	1.159 ₋₃	2.176
	9	96	6.199 ₋₄	2.150
	11	120	3.837 ₋₄	2.164
2	3	24	4.940 ₋₃	2.216
	5	48	1.064 ₋₃	2.057
	7	72	4.619 ₋₄	2.060
	9	96	2.554 ₋₄	2.061
	11	120	1.613 ₋₄	2.079

3.6 Parametric Analysis of Segment Refinement and Fractal Iteration on Solution Accuracy in Koch Snowflake

This project focuses on solving boundary value problems on Koch snowflake geometries up to the fifth iteration. While the complex fractal boundary introduces severe computational demands, these costs can be minimized by selecting an appropriate segment density $\tilde{n}$ that still preserves high numerical accuracy. Section 3.6.1 describes the core hypothesis concerning the optimal segment density $\tilde{n}$, the criteria used to verify this assumption, and the experimental setups. The empirical results are presented and discussed in Section 3.6.2.

3.6.1 Methodology

As described in Equation (114), the Koch iteration i and the number of points per segment $\tilde{n}$ uniquely determine the distance h between adjacent boundary nodes on Koch snowflake boundary. This characteristic length directly dictates the approximation accuracy of the Galerkin framework, where a smaller distance h corresponds to a higher boundary mesh density and a more precise scattered field approximation. Theoretically, if the fractal generation i increases while $\tilde{n}$ is held constant, the physical compression of the segments naturally shrinks the node spacing h compared to the preceding i as illustrated by an example in Figure 15. Consequently, this geometric contraction implies that an elevated number of points per segment $\tilde{n}$ may not be strictly required to achieve a sufficiently accurate solution at deeper Koch iterations.

Since one of the primary objective of this project is to approximate the scattered field for a Koch snowflake boundary at the fifth generation ($i = 5$), the localized node allocation per segment $\tilde{n}$ must be carefully optimized to maintain computational efficiency. Because the total number of boundary nodes n scales exponentially with successive fractal iterations (cf. Equation (116)), the choice of $\tilde{n}$ directly governs the resulting number of element faces. This geometric expansion scales the dimensionality of the dense Galerkin system matrix, which in turn dictates the number of double integrals required for matrix assembly and consequently determines the overall runtime of the framework.

At a configuration of $i = 5$ and $\tilde{n} = 3$, the characteristic spacing reaches $h = 2.058_{-3}$ across a total of 6,144 global boundary points. Incorporating two coefficients per element face ($\delta = 2$), the Galerkin framework must evaluate more than 37 million double integrals. This baseline allocation already demands substantial computational overhead. Increasing $\tilde{n}$ further would introduce prohibitive resource inflation. Consequently, it is

vital to investigate whether a minimized resolution, such as $\tilde{n} = 3$, is already sufficient to capture the wave physics at the fifth Koch iteration. To resolve this question systematically, preliminary experiments were conducted over lower-order fractal generations spanning $i = 1, 2$, and 3 . Within each iteration, the localized node allocation per segment $\tilde{n}$ was systematically varied. These trials aim to validate the hypothesis that the natural geometric compression of higher iterations allows low localized node counts to preserve numerical precision, thereby establishing an optimal equilibrium between solution accuracy and runtime efficiency.

For the Dirichlet problem with point wave, the exterior boundary data $u = g_e$ was prescribed according to Equation (118) and evaluated utilizing single-, double-, and combined-layer boundary integral operators. Concurrently, the Neumann boundary condition $u = \partial_\nu g_e$ followed Equation (119) and was resolved exclusively via a single-layer potential representation.

The resulting discrete fields were quantitatively verified against the exact continuous solution described in Equation (117) at an evaluation coordinate $\mathbf{p} = (3, 3)^\top$. The Galerkin framework was parameterized with a linear basis configuration ($\delta = 2$), a wave number $\kappa = 5$, and a boundary degree of freedom $\delta_\Gamma = 3$. The Koch snowflake boundary generated by Algorithm 2 was shifted by $(-0.5, -0.25)^\top$. The structural precision was tracked using the pointwise absolute error metric ε_n defined in Equation (123). The underlying hypothesis can be verified if the following two criteria are met:

1. The absolute error ε_n decays monotonically as a function of increasing $\tilde{n}$ (and consequently higher global node counts n) for a fixed Koch iteration i . This behavior validates that the solver's accuracy scales inversely with the discrete spacing h .
2. The extrapolated absolute error ε_n for the parameter $i = 5$ and $\tilde{n} = 3$ remains bounded beneath a sufficient tolerance. This trend confirms that a minimal allocation of $\tilde{n} = 3$ may be sufficient for the target geometry at $i = 5$.

The empirical findings of this structural analysis are presented in Section 3.6.2.

3.6.2 Results and Discussions

The objective of this verification study is to determine whether $\tilde{n} = 3$ can yield high numerical accuracy while simultaneously minimizing computational resource consumption for a boundary value problem on the fifth-iteration Koch snowflake. This hypothesis is verified when the two criteria mentioned in Section 3.6.1 are met.

Table 31 and Table 32 compile the pointwise absolute errors ε_n obtained from these initial numerical trials. The empirical observations closely match theoretical expectations: within any given geometric iteration, the absolute error ε_n decays monotonically under localized mesh refinement. This uniform contraction successfully validates the first criterion, substantiating that the solver’s precision scales inversely with the characteristic node spacing h .

To evaluate the second validation criterion, the quantitative datasets from Table 31 and Table 32 are visualized as semilogarithmic plots in Table 33 and Table 34. Based on the empirical data compiled for $i = 1, 2$, and 3 , the absolute error exhibits an approximately linear decreasing trend on the semilogarithmic scale. Assuming this asymptotic trajectory continues, a predictive error approximation can be performed via trend-line extrapolation using the exponential scaling relation

$$\varepsilon_n = ab^i, \tag{138}$$

where $a, b \in \mathbb{R}$ denote the characteristic regression coefficients. Applying a natural logarithm to both sides maps this nonlinear progression into a decoupled linear equation:

$$\log(\varepsilon_n) = \log(a) + i \log(b). \tag{139}$$

Using a linear extrapolation via the `SciPy` library, the log-linear parameters are extracted from the resolved datasets, and the projected absolute error metrics for the target generation ($i = 5$) are documented in Table 35. These empirical predictions demonstrate that a minimized local resolution of $\tilde{n} = 3$ remains sufficient to preserve high numerical fidelity at deeper fractal generations, thereby confirming the second criterion.

Table 31: **Absolute pointwise Error $\varepsilon_n(\mathbf{p})$ of the Dirichlet Problem Across Various Pre-Fractal Koch Generations and Localized Segment Resolutions**

Kernel	i	$\tilde{n}$	n	h	Absolute Error
SL	1	3	24	1.667_{-1}	4.344_{-4}
		5	48	8.333_{-2}	2.385_{-5}
		7	72	5.556_{-2}	2.786_{-7}
	2	3	96	5.556_{-2}	5.573_{-5}
		5	192	2.778_{-2}	2.180_{-6}
		7	288	1.852_{-2}	1.780_{-7}
	3	3	384	1.852_{-2}	5.929_{-6}
		5	768	9.259_{-3}	2.309_{-7}
		7	1152	6.173_{-3}	1.607_{-8}
DL	1	3	24	1.667_{-1}	5.129_{-5}
		5	48	8.333_{-2}	2.975_{-5}
		7	72	5.556_{-2}	1.194_{-5}
	2	3	96	5.556_{-2}	3.286_{-5}
		5	192	2.778_{-2}	5.043_{-6}
		7	288	1.852_{-2}	1.736_{-6}
	3	3	384	1.852_{-2}	4.579_{-6}
		5	768	9.259_{-3}	6.911_{-7}
		7	1152	6.173_{-3}	2.373_{-7}
DLSL	1	3	24	1.667_{-1}	1.475_{-4}
		5	48	8.333_{-2}	4.825_{-5}
		7	72	5.556_{-2}	1.849_{-5}
	2	3	96	5.556_{-2}	4.343_{-5}
		5	192	2.778_{-2}	6.592_{-6}
		7	288	1.852_{-2}	2.268_{-6}
	3	3	384	1.852_{-2}	6.088_{-6}
		5	768	9.259_{-3}	8.142_{-7}
		7	1152	6.173_{-3}	2.695_{-7}

Table 32: **Absolute pointwise Error $\varepsilon_n(\mathbf{p})$ of the Neumann Problem Across Various Pre-Fractal Koch Generations and Localized Segment Resolutions**

Kernel	i	$\tilde{n}$	n	h	Absolute Error
SL	1	3	24	1.667_{-1}	4.483_{-4}
		5	48	8.333_{-2}	6.937_{-5}
		7	72	5.556_{-2}	2.462_{-5}
	2	3	96	5.556_{-2}	5.789_{-5}
		5	192	2.778_{-2}	1.044_{-5}
		7	288	1.852_{-2}	3.543_{-6}
	3	3	384	1.852_{-2}	6.224_{-6}
		5	768	9.259_{-3}	1.152_{-6}
		7	1152	6.173_{-3}	3.964_{-7}

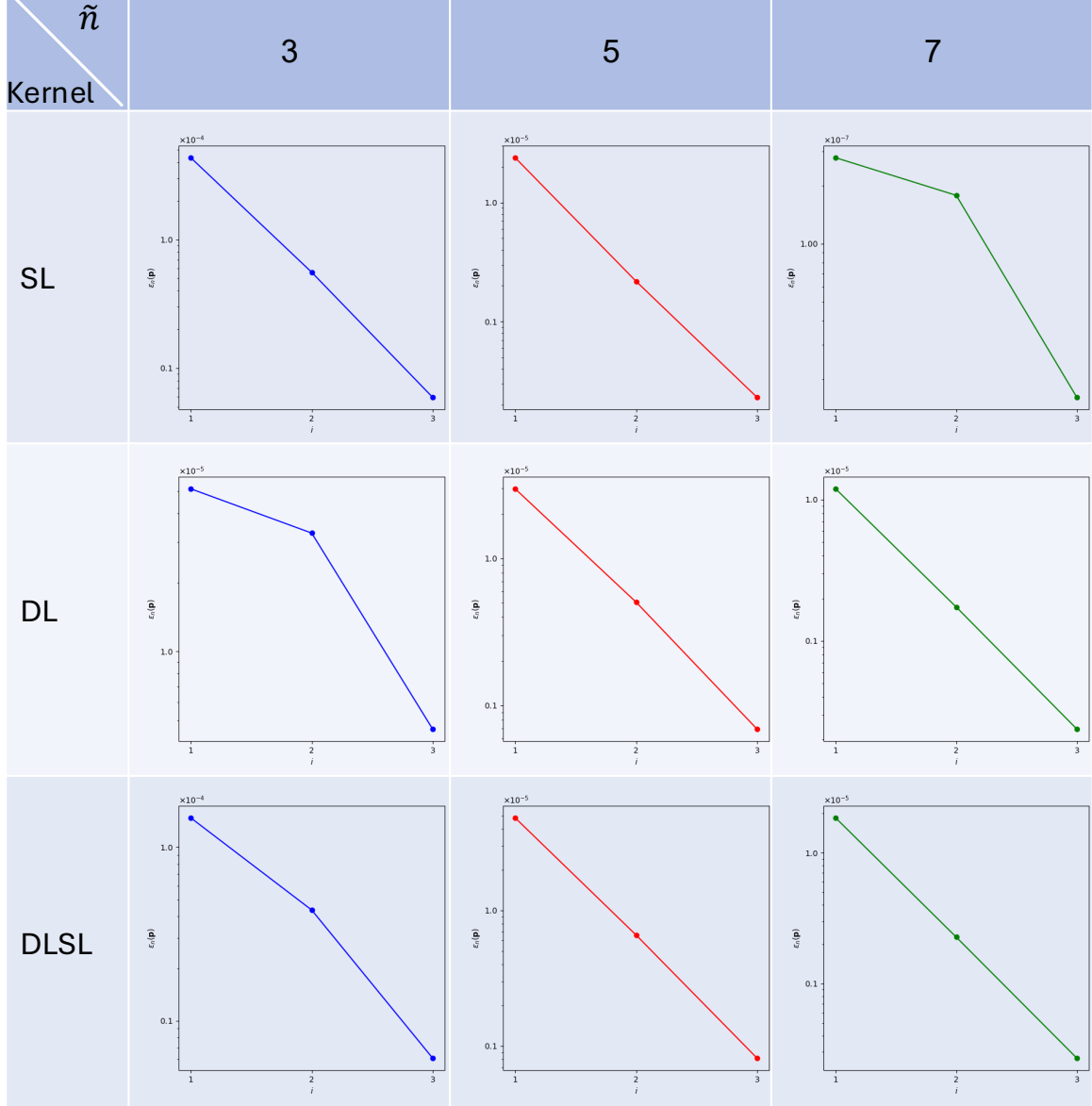


Table 33: Absolute pointwise error $\varepsilon_n(\mathbf{p})$ evaluated over successive pre-fractal Koch generations i under a constant local segment resolution $\tilde{n}$ for the Dirichlet boundary value problem

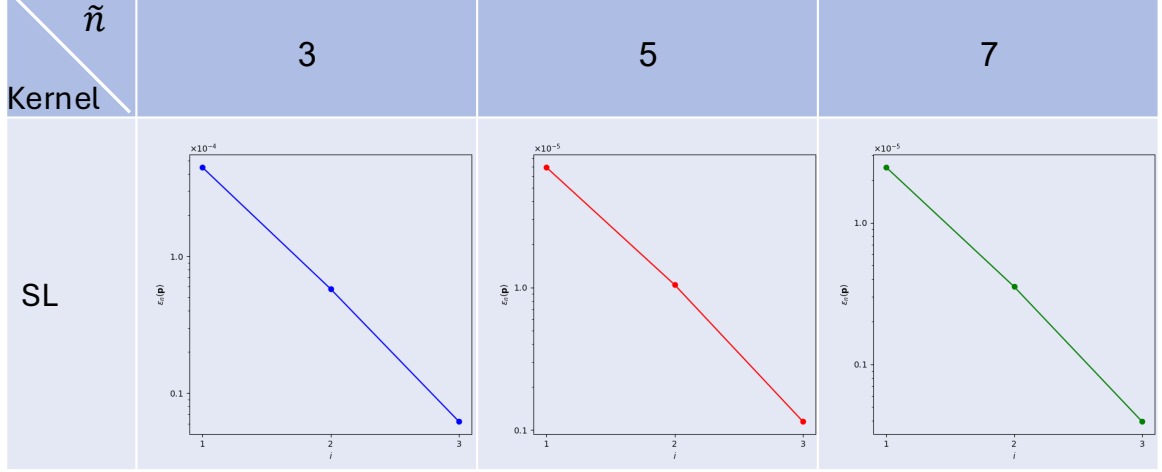


Table 34: Absolute pointwise error $\varepsilon_n(\mathbf{p})$ evaluated over successive pre-fractal Koch generations i under a constant local segment resolution $\tilde{n}$ for the Neumann boundary value problem

Table 35: **Extrapolated Absolute pointwise Error ε_n for the Pre-Fractal Koch Generations $i = 5$ with Segment Density $\tilde{n} = 3$**

Boundary Condition	Kernel	Absolute Error
Dirichlet	SL	6.709 ₋₈
	DL	8.888 ₋₈
	DLSL	1.196 ₋₇
Neumann	SL	7.194 ₋₈

4 High-Performance Computing (HPC) Integration

While the solver framework at the current state successfully resolved lower-iteration geometries, scaling the simulation to a fifth-iteration Koch snowflake boundary introduces an exponential increase in geometric complexity. Under a linear basis approximation framework ($\delta = 2$), this configuration yields a dense spatial discretization of 6,144 boundary nodes, necessitating the evaluation of over 37 million multidimensional double integrals to assemble the system matrix. The resulting dense linear system demands billions of floating-point operations. With respect to this matter, several optimizations were implemented and explained in the following sections:

- Section 4.1 explains the development of advanced parallelization strategies tailored for distributed-memory High-Performance Computing (HPC) architectures. Transitioning the operational pipeline to an HPC system is essential to leverage distributed parallelization strategies, ensuring the massive matrix assembly and solution phases remain computationally tractable.
- Section 4.2 investigates and profiles different numerical integration routines. Optimizing this specific phase is critical, because the numerical evaluation of the double integrals to assemble the system matrix accounts for the vast majority of the total execution runtime.

4.1 Designing the Parallelization Strategy

The baseline implementation of the `Executor` component within the Galerkin framework relies on the dynamic task allocation provided by the `Joblib` library. This model is fundamentally designed for shared-memory, single-node architectures, where the dense system matrix is entirely hosted within local volatile memory without inter-node communication or distributed data structures. However, resolving the acoustic scattering problem on a fifth-generation Koch snowflake boundary introduces a massive, dense system matrix that inflicts a computational overhead during the assembly. To significantly mitigate this runtime bottleneck, the workload must be scaled across multiple compute nodes in an HPC cluster. Consequently, transitioning to a Single Program Multiple Data (SPMD) paradigm becomes essential, as it natively supports distributed-memory environments and coordinates parallel matrix assembly across decoupled hardware nodes.

As its name implies, the SPMD paradigm enables multiple autonomous processors to simultaneously execute identical software instructions while operating on distinct, local-

ized segments of the global dataset. In contrast to the dynamic task allocation executed by `Joblib`, the SPMD model utilizes a static task allocation strategy. Under this concurrency architecture, the structural distribution of the workload is entirely predefined before computational execution begins, establishing a completely deterministic mapping between process identifiers and matrix data structures.

During the development phase, a matrix-free architecture was initially evaluated as an alternative parallelization strategy. Section 4.1.1 details the mechanics of this architecture alongside the experimental configurations established for its performance profiling. The corresponding empirical findings are subsequently analyzed in Section 4.1.2.

In the later development phase, the development strategy was transitioned to a precomputed explicit-matrix architecture. Section 4.1.3 explains the advantages, algorithmic details, and evaluation metrics of this architecture. Section 4.1.4 presents the performance metrics achieved under this design.

4.1.1 Matrix Free Architecture: Methodology

Initially, a matrix-free solver implementation was considered under the SPMD paradigm to maximize the parallel scalability of the framework. As established in Section 2.9, the Galerkin method requires solving the linear system in Equation (73), where the discrete operator matrix $\mathbf{A}$ is inherently dense. Furthermore, adequately resolving a complex pre-fractal geometry requires an elevated global node count, causing $\mathbf{A}$ to expand to a scale that might impose physical memory storage bottlenecks. However, because the framework utilizes an iterative Generalized Minimal Residual (GMRES) algorithm, the solution process exclusively depends on evaluating the matrix-vector product $\mathbf{A}\mathbf{x}$ rather than manipulating the explicit matrix elements directly. Leveraging this operational property, each process is statically assigned a distinct subset of row indices. Instead of allocating memory for the global matrix $\mathbf{A}$, each process computes individual operator coefficients on the fly and immediately multiplies them by the corresponding element of the input vector $\mathbf{x}$. These partial contributions are accumulated directly into local entries of the product vector. Once all assigned rows are processed, the local results from all distributed processes are aggregated via a global reduction step into the final vector, which is dispatched to coordinate the subsequent GMRES iteration.

To support distributed communication, the `Executor` component is refactored to execute the matrix-free evaluation strategy utilizing the `PyTorch` distributed communication backend. To evaluate this matrix-free strategy, a series of performance benchmarks are executed using lower-order pre-fractal generations ($i < 5$), corresponding to relatively

small system dimensions.

Performance evaluations were conducted using pre-fractal Koch snowflake boundaries at generations $i = 2$ and $i = 3$ based on Algorithm 2. The local segment resolution $\tilde{n}$ was varied to alter the global number of boundary points n and the points were shifted by $(-0.5, -0.25)^\top$. The process evaluated the strong scaling capability by varying the number of worker process for each configuration of n . The benchmarks employed an exterior Dirichlet problem from a point-source incident defined in Equation (118). The boundary value problem was resolved via the single-layer boundary representation with a linear basis configuration ($\delta = 2$), wave number $\kappa = 5$, and boundary DOF $\delta_\Gamma = 3$. To ensure reproducibility, all numerical experiments were executed on a dedicated workstation featuring the following hardware and software configuration:

- **CPU Architecture:** x86_64,
- **CPU Model:** Intel(R) Core(TM) i7-14700 (20 Cores / 28 Threads),
- **Operating System:** Rocky Linux 9.8 (Blue Onyx),
- **Python Runtime:** Python 3.11.13,
- **PyTorch Backend:** PyTorch 2.10.0+cu128.

4.1.2 Matrix Free Architecture: Results and Discussions

The purpose of this section is to evaluate the performance of the parallel matrix-free architecture. As established in Section 4.1.1, the term matrix-free architecture denotes an SPMD configuration where the system matrix entries are computed on the fly and immediately multiplied by the operand vector during successive GMRES iterations. By eliminating the requirement to allocate memory for the global system matrix, this strategy was intended to overcome local physical storage limitations and theoretically offers high parallel scalability.

The empirical results of this performance profiling are presented in Table 36. Utilizing the matrix-free strategy for a low-dimensional system operator of merely 96×96 entries, the framework required an execution window of nearly four hours across four parallel worker processes to resolve the boundary value problem. This extreme computational bottleneck confirms that the iterative quadrature evaluation overhead scales unsustainably, even for trivial geometries. Consequently, to achieve acceptable temporal

efficiency, an alternative parallelization paradigm was considered in the following section: the precomputed explicit-matrix strategy.

Table 36: **Total Wall-Clock Runtime of the Matrix-Free Architecture for Solving the Exterior Dirichlet Problem on Pre-Fractal Koch Snowflake Using the Single-Layer Kernel**

i	$\tilde{n}$	n	Number of Processes	Runtime [h]
2	3	96	4	3.97
			8	2.47
			16	1.67
3	2	192	4	10.81
			8	7.08
			16	4.79

4.1.3 Precomputed Explicit Matrix Architecture: Methodology

Based on the empirical data, recalculating the discrete integral operator coefficients on the fly during every independent GMRES iteration introduces an unsustainably high computational overhead. This severe processor bottleneck ultimately diminishes the framework’s scalability potential. Consequently, to eliminate these iterative redundancy costs, the framework’s design was pivoted from a matrix-free configuration to a precomputed explicit-matrix architecture.

The precomputed explicit-matrix architecture adheres to an SPMD paradigm, wherein each distributed process evaluates its assigned matrix coefficients during a dedicated assembly step and preserves them persistently in local memory. Unlike the matrix-free variant, this scheme completely eliminates the need to recalculate operator entries on the fly during successive GMRES iterations. Task allocation across the parallel process topology can be orchestrated via several structural data decomposition strategies, namely column-wise, row-wise, or block-wise partitioning, as illustrated in Figure 36. Assuming a global system matrix of dimensions $d \times d$, a column-wise allocation requires each independent process to compute and store a subset of columns of length d . Conversely, a row-wise distribution assigns a distinct subset of rows of length d to each process. A block-wise configuration subdivides the operator domain into smaller two-dimensional submatrices.

Because the iterative GMRES solver relies heavily on parallel matrix-vector multipli-

cations, where independent inner products are computed across each matrix row and the current solution vector, the row-wise slicing strategy minimizes the inter-node communication footprint. This approach requires only an initial broadcast of the global operand vector $\mathbf{x}$, followed by a collective reduction or gather operation to assemble the finalized product vector. In contrast, column-wise and block-wise decompositions incur substantial network overhead, demanding additional collective communication steps to accumulate partial vector sums across ranks after each local multiplication phase. Consequently, this optimized row-wise distribution was implemented within the **Executor** component in the framework.

The **Executor** component utilizing the precomputed explicit-matrix strategy was implemented with two distinct communication backends: **PyTorch** and **mpi4py**. The **PyTorch** backend was integrated to preserve the architectural path for future hardware acceleration, such as executing heavy integral routines or localized matrix-vector products on Graphics Processing Units (GPUs), even though the current deployment phase remains strictly CPU-based. At this development stage, the distributed matrix partitions and underlying numerical routines rely heavily on the **NumPy** ecosystem. Consequently, leveraging **PyTorch**'s distributed communication primitives introduces an undesirable data-type translation bottleneck, as the local `numpy.ndarray` matrix chunks must be explicitly cast to `torch.Tensor` structures before message transmission and subsequently converted back. To eliminate this unnecessary computational and memory serialization overhead, the **mpi4py** backend was implemented as an alternative option, enabling native communication directly on the underlying memory buffers.

To assess the parallel efficiency of this framework, the performance evaluation was systematically structured around strong and weak scaling metrics. The primary objective of this benchmarking is to demonstrate a significant reduction in total wall-clock runtime compared to the previous matrix-free implementation.

For the strong scaling analysis, the global problem size was held constant while the parallel process count was systematically increased. The empirical speedup S is formally defined as

$$S(p) = \frac{T(1)}{T(p)}, \quad (140)$$

where $T(1)$ and $T(p)$ denote the total execution runtimes utilizing 1 and p processes, respectively. Under ideal scaling conditions, the speedup follows the linear trajectory $S(p) = p$. To quantify the parallel utilization, the strong scaling efficiency E_S is expressed

as

$$E_S(p) = \frac{S(p)}{p} \cdot 100\%. \quad (141)$$

An ideal strong scaling configuration maintains a constant efficiency profile at 100%, indicating perfect computational speedup without parallel overhead.

For the weak scaling analysis, the global problem size was scaled proportionally with the process count to maintain a constant computational workload per process. In this experimental configuration, the localized workload allocation was fixed at 2304 matrix elements per individual process. Operating under a linear basis configuration ($\delta = 2$) for a given Koch generation i , the segment density $\tilde{n}$ was systematically increased alongside the process count p , causing the global matrix dimensions $d \times d = n \times n$ to expand dynamically while keeping the per-process memory footprint constant. The weak scaling efficiency E_W is formally defined as

$$E_W(p) = \frac{T(1)}{T(p)} \cdot 100\%. \quad (142)$$

Under ideal weak scaling conditions, the execution time remains invariant, yielding an efficiency profile of 100% across all processor allocations.

All numerical experiments were executed utilizing the `mpi4py` backend on a dedicated workstation configured with the hardware and software specifications outlined below:

- **CPU Architecture:** x86_64,
- **CPU Model:** Intel(R) Core(TM) i7-14700 (20 Cores / 28 Threads),
- **Operating System:** Rocky Linux 9.8 (Blue Onyx),
- **Python Runtime:** Python 3.11.13,
- **MPI Implementation:** OpenMPI v5.3.0.

The simulations were driven by a point-source incident field enforcing the exterior Dirichlet boundary condition defined in Equation (118). The underlying boundary value problem was resolved via a single-layer potential formulation configured with a linear basis expansion ($\delta = 2$), a wave number of $\kappa = 5$, and a boundary DOF $\delta_\Gamma = 3$. The Koch snowflake boundary was generated based on Algorithm 2 and shifted by $(-0.5, -0.25)^\top$. For each benchmarking trial, two primary metrics were isolated and recorded: the wall-

clock time required to construct the system matrix $\mathbf{A}$ (Matrix Assembly Time) and the duration required to solve the linear system $\mathbf{Ac} = \mathbf{b}$ (Equation Solving Time). The total execution runtime is defined as the direct summation of these two distinct temporal phases.

The scaling properties of the precomputed explicit-matrix strategy implementation are comprehensively verified in Section 4.1.4.

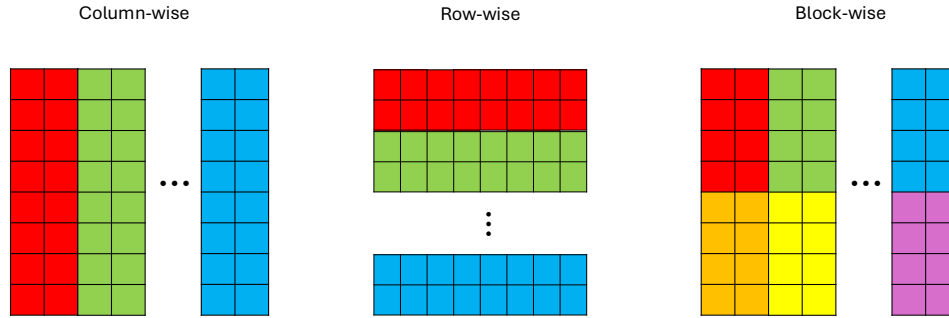


Figure 36: Data decomposition strategies for the task allocation. Each color corresponds to the task associated to a process.

4.1.4 Precomputed Explicit Matrix Architecture: Results and Discussions

This section details the performance evaluation of the precomputed explicit-matrix architecture. This architecture denotes an SPMD configuration where each individual worker process evaluates and caches its assigned system matrix entries prior to the initialization of the GMRES iterations. The matrix is partitioned via a row-wise slicing strategy specifically to minimize inter-node communication overhead.

The quantitative outcomes of the strong scaling experiments are compiled in Table 37. The corresponding parallel speedup and strong scaling efficiency are plotted in Figure 37. The parallel implementation exhibits robust strong scaling characteristics, achieving a speedup of approximately 6 on 8 MPI processes, which corresponds to a parallel efficiency of roughly 75%.

A slight superlinear speedup was observed when transitioning the core allocation from one to two MPI processes. This phenomenon is often attributed to enhanced cache

localized utilization. Conversely, as the process count scales further, the speedup transitions to a sublinear trajectory. Empirical profiling indicated that the Equation Solving Time accounts for less than 0.2% of the comprehensive runtime. Therefore, the primary source of parallel efficiency degradation must be localized within the matrix assembly phase, driven by load imbalances inherent to the adaptive numerical quadrature rules. Because the integration workload varied across geometric elements, faster processes are forced to idle at synchronization barriers until the slowest worker completes its assigned quadrature evaluations.

Another secondary factor likely contributing to the performance degradation is the increased possibility of cache thrashing and memory bus contention. At low process counts, the aggregate working set of the parallel processes, which includes data such as boundary point coordinates, fits entirely within the CPU’s Last-Level Cache (LLC). However, as the process count increases and multiple processes concurrently request identical datasets, the combined working set exceeds the aggregate capacity of the LLC. This capacity limitation causes cache line evictions to accommodate incoming data. Moreover, the limited memory bus capacity for transferring the data requested by the process simultaneously can also prolong core idle time.

The empirical outcomes of the weak scaling evaluation are presented in Table 38, with the resulting weak scaling efficiency plotted in Figure 38. During these scaling experiments, the matrix initialization time exhibited a moderate increase as the process count scales upward. Although the matrix assembly phase requires no active inter-process communication, the localized computational cost per matrix entry does not remain strictly constant. This variation arises because as the segment density $\tilde{n}$ increases, the Euclidean distance $\|\tilde{m}(p) - \tilde{m}(s)\|$ approaches zero for adjacent elements. These interactions necessitate an increased number of adaptive quadrature evaluations compared to well-separated interactions. Moreover, it can be observed that the number of matrix-vector multiplications required by the GMRES solver increased alongside the segment density. This inflation directly impacts the Equation Solving Time, thereby increasing the cumulative execution runtime.

Despite these limitations, distributed-memory parallelization remains essential for solving large-scale Galerkin systems. Since both memory consumption and assembly cost grow quadratically with the number of degrees of freedom, storing and assembling the system matrix on multiple processes enables problem sizes that would exceed the memory capacity of a single compute node. Moreover, the precomputed explicit-matrix architecture yields a substantial and necessary reduction in total execution wall-clock time.

Table 37: **Strong Scaling Performance Profiles for a Fixed System Size of 384×384**

i	$\tilde{n}$	n	Number of Processes	Matrix Assembly Time [h]	Equation Solving Time [s]	Runtime [h]	Speedup	Strong Scaling Efficiency [%]
3	3	384	1	7.92	4.60	7.92	1.00	100.00
			2	3.65	5.00	3.65	2.17	108.40
			4	2.17	6.23	2.17	3.65	91.25
			8	1.32	8.58	1.32	5.98	74.76

Table 38: **Weak Scaling Performance Metrics with a Constant Localized Workload of 2304 Rows/Process**

i	$\tilde{n}$	n	Number of Processes	Matrix Assembly Time [min]	Number of Matrix-Vector Multiplications	Equation Solving Time [s]	Runtime [min]	Weak Scaling Efficiency [%]
2	2	48	1	21.06	32	1.08	21.07	100.00
	3	96	4	14.20	53	2.78	14.25	147.91
	5	192	16	36.52	88	11.82	36.72	57.39

Strong Scaling

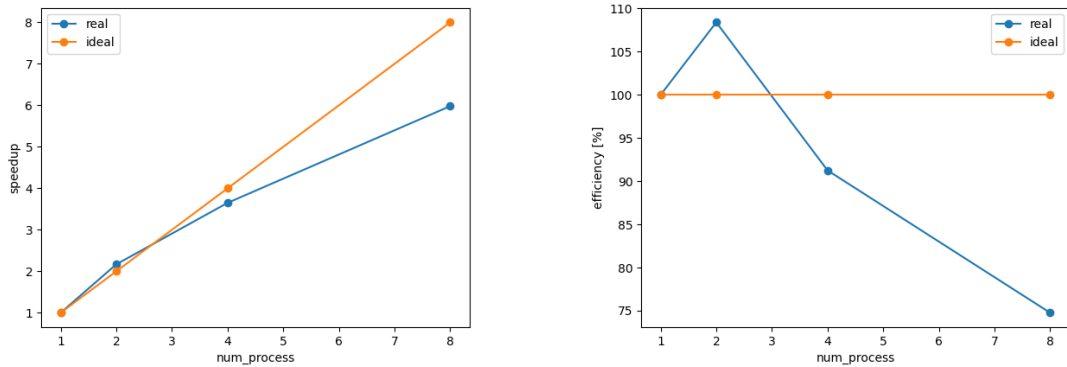


Figure 37: Parallel speedup (left) and strong scaling efficiency (right) as a function of the distributed process count

Weak Scaling

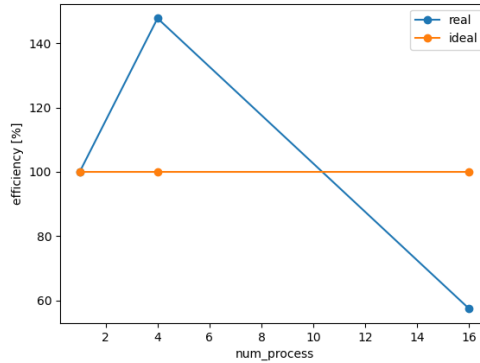


Figure 38: Weak scaling efficiency as a function of the distributed process count

4.2 Performance Evaluation of Numerical Integration Routines

At this stage of development, the matrix assembly phase represented the dominant computational bottleneck of the Galerkin method. This computational cost stems primarily from the double integrals, which consume the vast majority of the execution runtime. To optimize the overall performance of the framework, it is necessary to investigate alternative numerical strategies for evaluating these boundary element integration loops.

Section 4.2.1 outlines the different numerical integration routines and describes the performance metrics. Section 4.2.2 presents the accompanying evaluation results.

4.2.1 Methodology

In the initial implementation of the framework, these double integrals were computed utilizing the `scipy.integrate.dblquad` (short `dblquad`) [16] routine. This function evaluates the integral operators by leveraging the adaptive quadrature algorithms from the Fortran library QUADPACK [9].

An alternative approach for evaluating the double integral in the system matrix's assembly is to utilize the `scipy.integrate.quad_vec` (short `quad_vec`) [17] routine. Unlike `dblquad`, which is designed specifically for the integration of scalar-valued functions,

`quad_vec` is specialized for vector-valued integrands. In the baseline implementation using `dblquad`, the kernel function must be structured to accept scalar parameters and evaluate entries element-by-element for a single matrix position. Conversely, `quad_vec` allows the kernel function to ingest a scalar integration coordinate and process it via an array-based architecture, computing an entire matrix chunk simultaneously. The double integration is subsequently achieved by nesting two successive instances of the `quad_vec` routine.

Another option worth taking into consideration is the `scipy.integrate.cubature` (short `cubature`)[15] integration package. This routine performs adaptive numerical integration for vector-valued functions over multidimensional domains. Unlike `quad_vec`, which requires a nested configuration to resolve multiple dimensions, the `cubature` algorithm handles double integrals natively. This direct approach eliminates the architectural complexity of nested function calls. Furthermore, a primary advantage of this implementation is its potential for hardware acceleration. It offers a possibility to offload the intensive integration process to GPUs when paired with a `PyTorch` backend.

The foundational QUADPACK implementation of the adaptive quadrature routine initiates integration over a global interval $[a, b]$ to generate an initial approximation of both the integral value and its corresponding error bounds. If the localized error estimate exceeds the predefined tolerance, the domain is bisected into two sub-intervals, which are then evaluated independently to yield individual error metrics. In subsequent iterations, the algorithm selectively partitions the specific sub-interval exhibiting the highest error into smaller segments for re-evaluation. This process is executed recursively until the global error falls below the targeted tolerance or a maximum subdivision limit is reached. If the error remains above the threshold when this structural cap is met, `SciPy`'s integration wrapper issues a warning indicating non-convergence.

In contrast, the implementations within `quad_vec` and the `cubature` package modify this classical approach by introducing a vectorized subdivision strategy. Instead of bisecting a single interval per iteration, these algorithms simultaneously subdivide the N intervals containing the largest localized errors. Although this approach seems to increase the number of function evaluations required per quadrature step, it could accelerate the overall error reduction rate. Furthermore, because these concurrent evaluation points can be aggregated into array-based, data-parallel batches, the underlying routines leverage the Single Instruction, Multiple Data (SIMD) units of the processor, ensuring high computational efficiency even during single-core execution.

While it is technically feasible to utilize multiple worker processes to parallelize `quad_vec` and `cubature` routines to distribute the evaluation points of the N worst-error intervals,

this approach is highly undesirable within an SPMD framework. Introducing node-level workers leads to uncontrolled nested parallelism, as SciPy internally spawns worker pools via Python’s native `multiprocessing.pool.Pool` module. These child processes lack the built-in capability to span across separate cluster nodes via network fabrics. This restriction severely limits architectural flexibility, forcing the parent and child processes to compete for resources within a single shared-memory node. Furthermore, spawning these localized child processes incurs significant data serialization (pickling) overhead, as parameters must be copied into isolated process memory spaces. Consequently, to maintain structural compatibility with distributed-memory scaling, only the sequential configurations of `quad_vec` and `cubature` were investigated in this work.

The benchmarking experiments were performed to construct the system matrix of the single-layer potential formulation, configured with a linear basis expansion ($\delta = 2$), a wavenumber of $\kappa = 5$, and a boundary DOF $\delta_\Gamma = 3$. The domain is a unit disc centered at the origin, where the circle boundary for the unit disc is defined using Equation (23) with $a = b = 1$. The performance profile was quantified using two primary metrics: the average matrix assembly wall-time T and the assembly time per matrix entry t , which is defined as

$$t = \frac{T}{n^2}, \quad (143)$$

where n denotes the number of boundary points, making n^2 the total number of matrix elements for the linear basis configuration ($\delta = 2$). The evaluations explored the execution profile across varying boundary densities, specifically $n \in \{16, 32, 64\}$. To ensure statistical reliability, each configuration was executed over three independent trials ($R = 3$), from which the sample mean and standard deviation are computed. The average matrix assembly time $\bar{T}$ is formulated as

$$\bar{T} = \frac{1}{R} \sum_{i=1}^R T_i, \quad (144)$$

where T_i represents the matrix assembly wall-time recorded during the i -th repetition. The corresponding sample standard deviation σ_T is evaluated via

$$\sigma_T = \sqrt{\frac{\sum_{i=1}^R (T_i - \bar{T})^2}{R - 1}}. \quad (145)$$

The average assembly time per entry $\bar{t}$ and its associated standard deviation σ_t were determined analogously by substituting the element-wise metrics t_i into the respective statistical formulations.

To ensure that the integral routines produce numerically equivalent results, the relative Frobenius difference E_F and the relative maximum difference $E_{\max}$ of the assembled matrices were compared. The relative difference E_{rel} is defined as

$$E_F = \frac{\|\mathbf{A}_{\text{dblquad}} - \mathbf{A}\|_F}{\|\mathbf{A}_{\text{dblquad}}\|_F}, \quad (146)$$

where $\mathbf{A}_{\text{dblquad}}$ denotes the system matrix assembled by the `dblquad` routine, and $\mathbf{A}$ denotes the system matrix assembled by the `quad_vec` or the `cubature` routine. In this context, for a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$, the Frobenius norm $\|\cdot\|_F$ is defined as

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}. \quad (147)$$

The relative maximum difference $E_{\max}$ is defined as

$$E_{\max} = \frac{\max\{x \in \mathbb{C} \mid x = |a_{\text{dblquad}_{ij}} - a_{ij}|; i, j = 1, \dots, n\}}{\max\{x \in \mathbb{C} \mid x = |a_{\text{dblquad}_{ij}}|; i, j = 1, \dots, n\}}. \quad (148)$$

The results of these evaluations are presented and discussed in Section 4.2.2.

4.2.2 Results and Discussions

This section analyzes the performance results of three distinct integration methods: `dblquad`, `quad_vec`, and `cubature`.

For the smallest problem size ($n = 16$), the `cubature` routine required a significantly longer execution time than both `dblquad` and `quad_vec`. Benchmarking for `cubature` was therefore halted due to its poor efficiency on small-scale domains.

The empirical results from the remaining experiments utilizing the `dblquad` and `quad_vec` integration routines were compiled in Table 39. These data demonstrate that the baseline `dblquad` routine achieves a faster overall runtime.

Both integration routines were executed with the identical absolute and relative error tolerances of 10^{-8} . Since both routines approximate the same double integrals to comparable requested accuracies, only small differences due to their distinct adaptive refinement strategies are expected. The observed relative Frobenius and maximum differences, ranging from 10^{-11} to 10^{-9} , are therefore consistent with the prescribed quadrature ac-

curacy.

Consequently, since the initial `dblquad` implementation achieved faster assembly time, the baseline integration framework is maintained without further modification.

Table 39: **Comparison of assembly times and numerical precision between `dblquad` and `quad_vec`**

n	Matrix Size	$\bar{T} \pm \sigma_T$ [s]		$\bar{t} \pm \sigma_t$ [ms]		E_F	$E_{\max}$
		<code>dblquad</code>	<code>quad_vec</code>	<code>dblquad</code>	<code>quad_vec</code>		
16	256	202.95 ± 0.50	257.08 ± 1.30	792.77 ± 3.10	1004.22 ± 3.92	1.434_{-11}	1.807_{-11}
32	1024	444.45 ± 0.19	449.45 ± 0.51	434.04 ± 0.42	438.91 ± 0.43	9.152_{-10}	1.410_{-9}
64	4096	1006.40 ± 1.19	1263.64 ± 0.63	245.70 ± 0.06	308.51 ± 0.08	3.791_{-9}	8.249_{-9}

5 Numerical Simulations of Acoustic Scattering on a Higher-Iteration Fractal Domain

After verifying the correctness of the boundary integral implementation and optimizing the matrix assembly and solution process, the complete scattering problem was solved for the fifth iteration Koch snowflake. As mentioned in the previous sections, boundary value problems on higher-iteration pre-fractal domains, such as the fifth-iteration Koch snowflake, introduce significant computational challenges. To discuss this matter, the materials are organized as follows:

- Section 5.1 outlines the numerical difficulties associated with solving boundary value problems on this complex geometry and details the operational pipeline of the Galerkin framework designed to resolve it.
- Section 5.2 presents the numerical results of the Helmholtz acoustic scattering problem from the fifth Koch iteration obtained from the Galerkin framework.

5.1 Methodology

The fifth Koch iteration represents a challenging benchmark because of the intricate fractal geometry. Adequately capturing this boundary structure requires a large number of discretization points, which exponentially increases both the double-integral evaluation workload and the solver runtime. Utilizing the Koch snowflake parameterization described in Algorithm 2, the discrete boundary comprises a total of 6,144 boundary points. Consequently, enforcing a linear basis expansion ($\delta = 2$) necessitates the evaluation of more than 37.7 million double integrals to construct the system matrix. Following the verification of the boundary integral formulation and the optimization of the workflow, the framework was applied to solve the acoustic scattering problem on a fifth-iteration Koch snowflake. The computationally heavy phases of the pipeline in the framework were executed on an HPC cluster using the parallel architecture developed in this work.

Overall, resolving the acoustic scattering problem on the fifth-iteration Koch snowflake with the implemented Galerkin framework comprises the following sequential steps:

- **Generation of the fifth-order Koch snowflake boundary:**
The fifth-order Koch snowflake geometry was generated via Algorithm 2. The boundary was then shifted by $(-0.5, -0.25)^\top$. As established in Section 3.2, it is

crucial to generate the fractal boundary with an odd number of boundary points per segment as a precautionary measure to avoid numerical integration directly over sharp corners, where geometric singularities occur. The evaluation in Section 3.6 suggests that choosing $\tilde{n} = 3$ points per segment is sufficient to achieve high solution accuracy while minimizing computational overhead.

- **Discretization of the boundary condition:**

This phase constructs the right-hand side vector $\mathbf{g}$ of the discrete Galerkin system (Equation (73)). This vector is determined by the incident plane wave and the specific boundary conditions. The solution was approximated via a basis expansion, where each geometric face contributes two coefficients corresponding to the linear basis functions ($\delta = 2$). The underlying boundary value problem was resolved via a boundary DOF $\delta_\Gamma = 3$.

- **Parallel assembly of the boundary integral operator:**

This phase constructs the system matrix $\mathbf{A}$ for the Galerkin linear system using the row-wise allocation scheme of the SPMD precomputed explicit-matrix parallelization strategy established in Section 4.1. This phase was executed on the HPC cluster.

- **Solution of the resulting linear system:**

The assembled Galerkin linear system $\mathbf{A}\mathbf{c} = \mathbf{g}$ is resolved using the GMRES iterative solver implementation from SciPy. This solution phase requires inter-node communication across the HPC node infrastructure, which is managed by the custom `Executor` component via the `mpi4py` backend.

- **Grid evaluation of the boundary integral representation:**

Upon solving for the coefficient vector $\mathbf{c}$, these discrete weights are used to evaluate the approximated scattered field across an exterior Cartesian grid, as formulated in Equation (63). To maximize resource efficiency, this evaluation phase was executed on a local workstation after exporting the calculated solution coefficients from the HPC environment. The acoustic scattering amplitudes were evaluated on the exterior points of the scatterer in the bounded region $[-5, 5] \times [-5, 5]$.

- **Superposition of the total field:**

The final total acoustic field was reconstructed by computing the linear superposition of the incident plane wave and the newly evaluated scattered field components.

Numerical simulation were performed for the acoustic scattering problem on the fifth iteration Koch snowflake using a plane wave incident field with wavenumber $\kappa = 5$. The

real part of the incident wave is illustrated in Figure 39. Numerical simulations for the Dirichlet boundary condition was evaluated using the boundary condition $u^{\text{sca}} = -u^{\text{inc}} = \exp(i\kappa\langle\mathbf{x}, \mathbf{d}\rangle)$ that propagates in the horizontal direction $\mathbf{d} = (1, 0)^\top$. The acoustic scattering problems with the Neumann boundary condition was evaluated using the boundary condition $\partial_\nu u^{\text{sca}} = -\partial_\nu u^{\text{inc}}$ with $\partial_\nu u^{\text{inc}}$, where the plane wave also propagates horizontally. The numerical simulation was further categorized based on the kernel representation:

- Dirichlet acoustic scattering problem resolved with the single-layer kernel,
- Dirichlet acoustic scattering problem resolved with the double-layer kernel,
- Dirichlet acoustic scattering problem resolved with the combined-layer kernel,
- Neumann acoustic scattering problem resolved with the single-layer kernel.

The HPC cluster for the numerical simulation was the JURECA supercomputer at the Jülich Supercomputing Centre [5]. The simulations were performed using the following configurations:

- **Partition name:** dc-cpu (Standard CPU compute nodes on JURECA),
- **Number of reserved nodes:** 20,
- **Number of process per node:** 128,
- **Number of threads per process:** 1,
- **Total number of used CPU cores:** 2560.

The empirical results of the numerical simulation detailing acoustic scattering across the fifth-iteration Koch snowflake boundary are presented and comprehensively analyzed in Section 5.2.

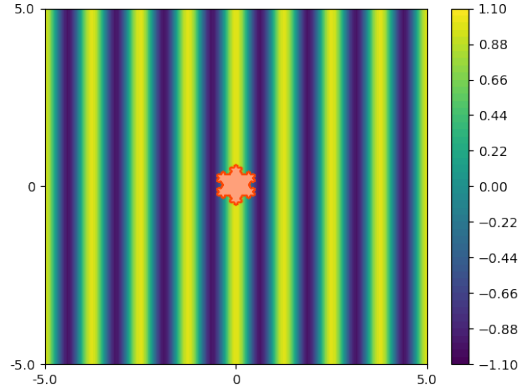


Figure 39: Horizontal plane wave incident field and the fifth-iteration Koch snowflake in the middle

5.2 Results and Discussions

The main objective of this chapter is to present and discuss the numerical simulation results for the acoustic scattering problem on the fifth iteration Koch snowflake.

The numerical simulation of the real part from the scattered field is presented in Figure 40. In all four cases, the scattered field consists of outward propagating wave fronts originating from the obstacle. It can also be observed on the right side of the scatterer that the wave amplitude decreases with increasing distance from the scatterer, which is consistent with the expected behavior of the two-dimensional Helmholtz fundamental solution (Equation (4)).

The scattered fields of the Dirichlet acoustic scattering problem obtained from the single-, double-, and combined-layer formulations are visually identical. They produce a concentric ring-shaped ripple radiating outward. This agreement confirms that the different boundary integral formulations produce equivalent numerical solutions for the same boundary value problem, despite employing different integral operators.

In contrast, the Neumann problem exhibits a substantially different scattering pattern. The reflected energy is redistributed into different directions because the boundary condition prescribes the normal derivative rather than the pressure itself. Instead of nearly concentric waves, the scattered field exhibits a much more directional structure. The oscillation structure has phase change that makes it look like the ring ripple is combined

with a spokes-like-ripple.

The complete execution pipeline with the specific configuration described in Section 5.1 required a total computational runtime of approximately six hours for each simulation. Considering the current JURECA's CPU rate of 2.25 GHz, this processing duration represents an acceptable baseline for resolving dense boundary integral equations on the fifth-iteration fractal geometries.

The total field is presented in Figure 41. The total field was obtained as the superposition of the incident plane wave and the scattered field. This explains the vertical interference fringes seen in the plots combined the faint ripples from the scattered wave.

The presented simulations demonstrate the capability of the developed parallel boundary element implementation to solve large-scale Helmholtz scattering problems on geometrically complex obstacles. The fifth-order Koch snowflake represents a demanding benchmark due to its large number of boundary elements and intricate geometry. Nevertheless, the parallel implementation successfully computed both the scattered and total fields. Furthermore, the close agreement between the three Dirichlet formulations confirms the correctness of the implementation, while the distinct Neumann solution illustrates the expected physical influence of the boundary conditions on the scattering behavior.

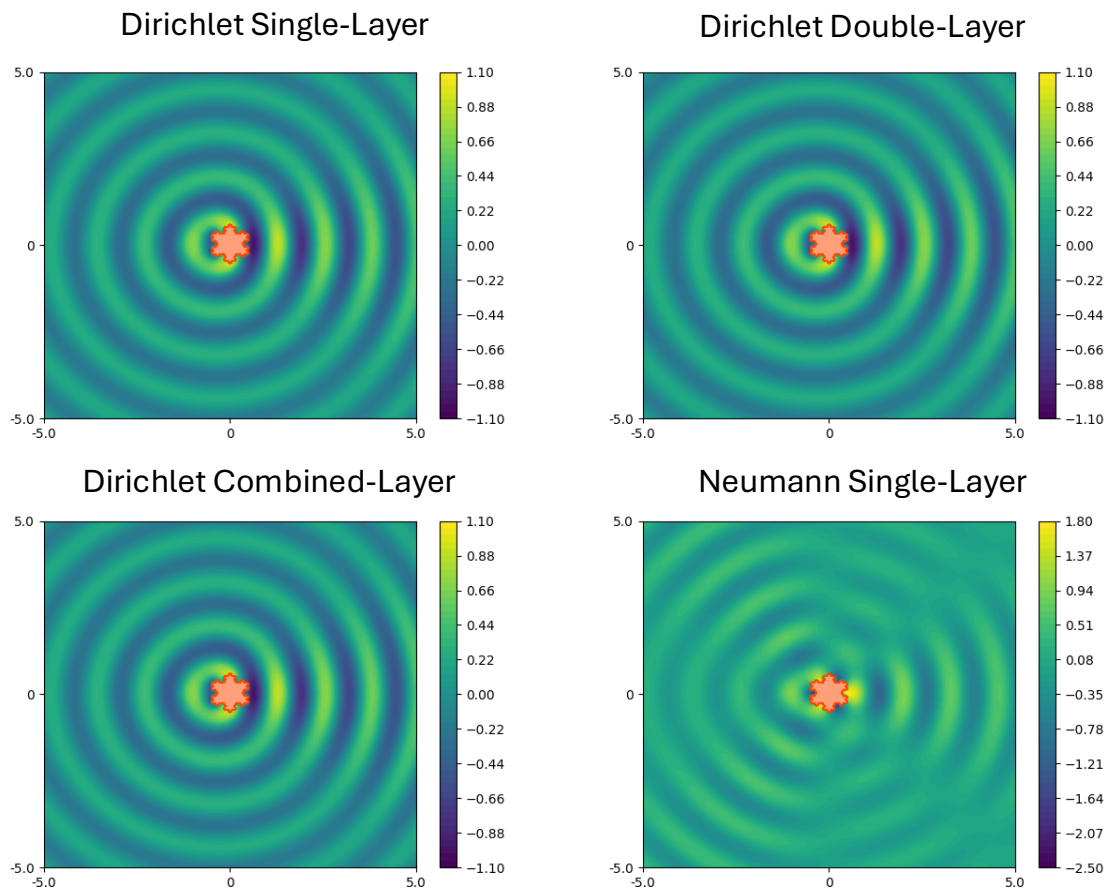


Figure 40: Real part of the scattered field on the fifth-iteration Koch snowflake boundary

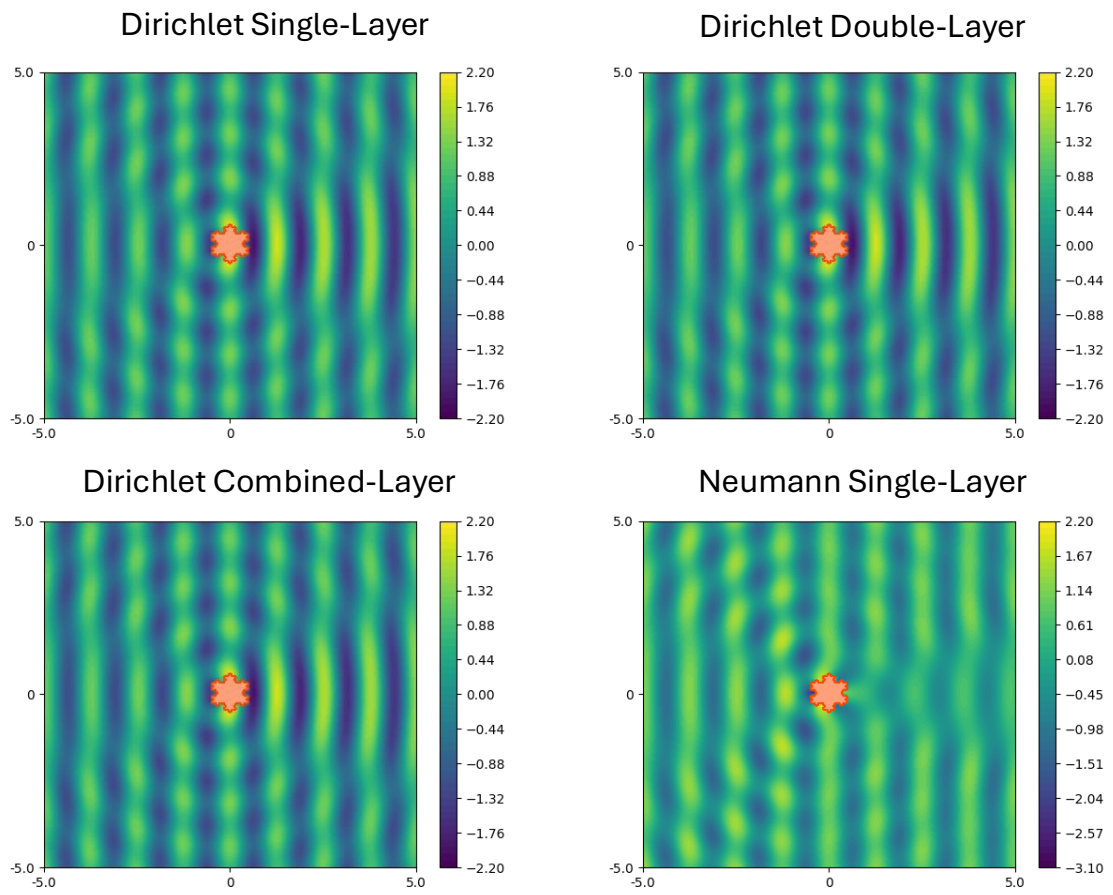


Figure 41: Real part of the total field on the fifth-iteration Koch snowflake boundary

6 Conclusion and Outlooks

This project successfully developed a robust and modular Galerkin solver framework implemented in Python. This work demonstrates that it is possible to systematically address the numerical and computational challenges associated with solving boundary value problems on complex, non-smooth domains by utilizing piecewise polynomial basis functions. The numerical difficulties induced by kernel singularities during the integral evaluation were successfully resolved through a reformulation of the integral expressions combined with a strategic boundary parameterization. Concurrently, computational performance was optimized by isolating the boundary parameter that balance numerical accuracy with computational costs, implementing a multi-node parallelization strategy, and evaluating various adaptive numerical quadrature routines. Following this rigorous verification and optimization pipeline, the framework successfully delivered solutions for the complex pre-fractal Koch snowflake boundary up to the fifth iteration.

A performance investigation was conducted for a matrix-free implementation within the framework. This approach eliminates the requirement to allocate memory for the global system matrix, as the matrix-vector products are computed on the fly. However, this architecture necessitates the reevaluation of the boundary integrals for every matrix entry during each iteration of the GMRES solver. Although it theoretically offers superior scalability, the high computational overhead of the repeated quadrature evaluations makes the approach unsustainable even for minor problem size. Consequently, the framework was transitioned to a precomputed explicit-matrix architecture, where each parallel process computes and stores its designated matrix block. While this method drastically reduces execution wall-clock time, it scales quadratically in terms of memory consumption.

The ideal solution to this bottleneck would be a matrix-free architecture paired with accelerated numerical integration routines that balance computational overhead and numerical precision. Currently, the framework relies on fully adaptive numerical integration. While this approach provides robust and highly accurate results for the general case, its execution during each GMRES cycle is computationally expensive. Future work could investigate specialized Gauss quadrature rules designed specifically for the BEM, such as the techniques proposed by Rathsfeld [11]. Leveraging these specialized rules in combination with GPU acceleration could reduce the overall computational runtime, thereby providing a highly practical and scalable solution for large-scale acoustic scattering problems.

Furthermore, future work should systematically analyze the impact of non-smooth bound-

ary geometries on the error bounds, particularly on how the localized loss of boundary smoothness determines the theoretical convergence rates.

Overall, this project demonstrates that pairing optimized parameters with distributed HPC pipelines makes complex pre-fractal scattering simulations computationally manageable.

References

- [1] K.E. Atkinson. *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009. ISBN: 9780521102834. URL: <https://books.google.de/books?id=hNFPwAACAAJ>.
- [2] D. Colton and R Kress. “The Helmholtz Equation”. In: *Inverse Acoustic and Electromagnetic Scattering Theory*. Cham: Springer International Publishing, 2019, pp. 15–42. ISBN: 978-3-030-30351-8. DOI: [10.1007/978-3-030-30351-8_2](https://doi.org/10.1007/978-3-030-30351-8_2). URL: https://doi.org/10.1007/978-3-030-30351-8_2.
- [3] D. Colton and R. Kress. “Integral Equation Methods in Scattering Theory”. In: Philadelphia, PA: Society for Industrial and Applied Mathematics, 2013. DOI: [10.1137/1.9781611973167](https://doi.org/10.1137/1.9781611973167). eprint: <https://epubs.siam.org/doi/pdf/10.1137/1.9781611973167>. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9781611973167>.
- [4] Z. Feng. “Model of Lagrange Two-dimensional Interpolation Based on Dimensionality Reduction”. In: *IAENG International Journal of Applied Mathematics* 55.7 (2025), pp. 2330–2335. DOI: [10.1000/xyz123](https://doi.org/10.1000/xyz123).
- [5] JURECA administrators. *JURECA user documentation*. Accessed: 2026-07. 2026. URL: <https://apps.fz-juelich.de/jsc/hps/jureca/index.html#>.
- [6] A. Kleefeld. “Direct and inverse acoustic scattering for three-dimensional surfaces”. PhD thesis. Dec. 2009.
- [7] H. von Koch. “Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire”. In: (1904).
- [8] C. Müller. “The Reduced Wave Equation”. In: *Foundations of the Mathematical Theory of Electromagnetic Waves*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1969, pp. 79–129. ISBN: 978-3-662-11773-6. DOI: [10.1007/978-3-662-11773-6_4](https://doi.org/10.1007/978-3-662-11773-6_4). URL: https://doi.org/10.1007/978-3-662-11773-6_4.
- [9] R. Piessens et al. *QUADPACK: A Subroutine Package for Automatic Integration*. Vol. 1. Springer Series in Computational Mathematics. Springer-Verlag, 1983. ISBN: 978-3-540-12553-2. DOI: [10.1007/978-3-642-61773-7](https://doi.org/10.1007/978-3-642-61773-7).
- [10] R. Potthast. *Point Sources and Multipoles in Inverse Scattering Theory*. Vol. 427. Apr. 1999. ISBN: 9780429121494. DOI: [10.1201/9781420035483](https://doi.org/10.1201/9781420035483).

- [11] A. Rathsfeld. “Quadrature methods for 2D and 3D problems”. In: *Journal of Computational and Applied Mathematics* 125.1 (2000). Numerical Analysis 2000. Vol. VI: Ordinary Differential Equations and Integral Equations, pp. 439–460. ISSN: 0377-0427. DOI: [https://doi.org/10.1016/S0377-0427\(00\)00484-2](https://doi.org/10.1016/S0377-0427(00)00484-2). URL: <https://www.sciencedirect.com/science/article/pii/S0377042700004842>.
- [12] S. Rjasanow and O. Steinbach. *Fast solution of boundary integral equations (mathematical and analytical techniques with applications to engineering)*. Springer, 2007.
- [13] Y. Saad and M. H. Schultz. “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869. DOI: [10.1137/0907058](https://doi.org/10.1137/0907058). eprint: <https://doi.org/10.1137/0907058>. URL: <https://doi.org/10.1137/0907058>.
- [14] A. P. S. Selvadurai. “Laplace’s equation”. In: *Partial Differential Equations in Mechanics 1: Fundamentals, Laplace’s Equation, Diffusion Equation, Wave Equation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 151–234. ISBN: 978-3-662-04006-5. DOI: [10.1007/978-3-662-04006-5_5](https://doi.org/10.1007/978-3-662-04006-5_5). URL: https://doi.org/10.1007/978-3-662-04006-5_5.
- [15] The SciPy Community. *scipy.integrate.cubature—SciPy v1.18 Reference Guide*. Accessed: 2026-07. 2026. URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.cubature.html>.
- [16] The SciPy Community. *scipy.integrate.dblquad — SciPy v1.18 Reference Guide*. Accessed: 2026-07-06. 2026. URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.dblquad.html>.
- [17] The SciPy Community. *scipy.integrate.quad_vec—SciPy v1.18 Reference Guide*. Accessed: 2026-07. 2026. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quad_vec.html.