UNIVERSITY OF CALIFORNIA,
IRVINE

The Smooth Extension Embedding Methods for Free Boundary Problems

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mathematics

by

Dong Yan

Dissertation Committee:
Professor Patrick Guidotti, Chair
Professor Long Chen
Professor Knut Sølna

2024

DEDICATION

To my advisor, who guided me with infinite patience and never gave up on me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Patrick Guidotti, for his patient guidance and professionalism throughout my journey in graduate studies. Patrick's extensive interests across various fields of mathematics have illuminated my path, helping me to develop my research and offering insightful criticism. His accessibility and willingness to devote time to advising me on all matters were invaluable. As an exceptional educator, he transformed me from a passive learner into an active explorer, fostering a profound shift in my approach to learning. Beyond academic matters, Patrick's advice and encouragement have been pivotal at every significant juncture and through the greatest challenges of my graduate life. His wisdom is sure to have a lasting impact on my personal development. Patrick sets an exceptional example for me in his roles as a researcher, educator, and individual embodying positivity, self-discipline, and motivation. To express my utmost respect and enduring gratitude, I would like to quote a Chinese saying here: *Once a teacher, always a father.*

I would also like to thank Professor Long Chen for his comprehensive lecture notes and the engaging articles on his Computational and Applied Math (CAM) blog. These resources have greatly enlightened and guided me throughout my research. I am deeply thankful to Professor Knut Sølna for his tremendous effort in securing financial support during my final year of the PhD program. Without his assistance, completing my studies would have been more challenging. I appreciate Professor Yifeng Yu and Professor Christopher Davis for their support and guidance in my teaching career. Lastly, I want to thank Dr. Daniel Agress for being an exceptional peer mentor, helping me transition into my first PhD research project, and providing invaluable assistance in the early stages of my research. I am grateful for the financial support I received from my advisor and for the Graduate Division Completion Fellowship, both of which helped bring the work on this dissertation to completion.

Special thanks go to the individuals at Harbin Institute of Technology, my alma mater, who have significantly supported me. Thank my advisor, Professor Zhichang Guo, for introducing me to the fascinating world of mathematical research about ten years ago. Thank the team leaders, Professor Jiebao Sun and Professor Boying Wu, for their invaluable guidance in my academic career development. Thank my peers and collaborators—Dr. Ying Wen, Dr. Wenjuan Yao, and Dr. Jingfeng Shao—for their dedication in completing and publishing the projects I was involved in during my undergraduate studies.

I must extend my deepest gratitude to my parents for their invaluable financial and emotional support during my most challenging times. I'm deeply thankful for the unforgettable moments with my friends, from hot pot parties and board game nights to adventurous road trips. Equally precious are the warm online greetings from friends worldwide. Despite the distance, their virtual companionship is just as meaningful and cherished. Special thanks go to Qiao Liang and Jiao Liang, the idol twins, for their encouragement and best wishes. Finally, I thank myself for enduring through numerous obstacles and moments close to giving up throughout these seven years, yet ultimately persevering to the completion of my PhD journey. Ending with Vex King's words dedicated to myself: *You are made of supernova stardust. Your potential to shine is as infinite as the galaxies that surround and support you.*

VITA

Dong Yan

EDUCATION

Doctor of Philosophy in Mathematics **2024**
University of California, Irvine *Irvine, California*

Master of Science in Mathematics **2019**
University of California, Irvine *Irvine, California*

Bachelor of Science in Mathematics **2017**
Harbin Institute of Technology *Harbin, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **2019–2024**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Lecturer **2022–2023**
University of California, Irvine *Irvine, California*

Teaching Assistant **2018–2023**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

The smooth extension embedding method with Chebyshev polynomials **2023**
Numerical Methods for Partial Differential Equations

A Non-Local Diffusion Equation for Noise Removal **2022**
Acta Mathematica Scientia

Hybrid BM3D and PDE filtering for non-parametric single image denoising **2021**
Signal Processing

ABSTRACT OF THE DISSERTATION

The Smooth Extension Embedding Methods for Free Boundary Problems

By

Dong Yan

Doctor of Philosophy in Mathematics

University of California, Irvine, 2024

Professor Patrick Guidotti, Chair

Free boundary problems can be reformulated as shape optimization problems with partial differential equation (PDE) constraints, making them solvable through numerical shape optimization algorithms. The Smooth Extension Embedding Method (SEEM), a novel approach for solving PDEs, leverages straightforward discretizations of complex domains and achieves a high degree of convergence for problems with smooth solutions. This makes SEEM a viable alternative to the finite element methods used in traditional numerical shape optimization algorithms, effectively overcoming key challenges such as mesh degeneration and low-order convergence. To enhance the stability of SEEM while maintaining its high-order accuracy for boundary value problems on non-smooth numerical representations of inherently smooth boundaries, we employ a regularized level set boundary approximation technique. This enhancement broadens the applicability of SEEM to the class of free boundary problems with a shape optimization approach. Through theoretical and practical examples, experiments show that our improved SEEM-based algorithm maintains high-order accuracy in shape gradient approximation and circumvents computational pitfalls like mesh degeneration that can arise during shape evolution. This makes the direct use of a simple shape gradient formula and an explicit gradient descent flow possible. The resulting algorithm employs straightforward domain grids throughout the shape optimization process, resulting in significant computational savings and ease of implementation.

Chapter 1

Introduction

1.1 Free and Moving Boundary Problems

Free and moving boundary problems play a significant role in various engineering and physics disciplines. Their applications span across hydrology, metallurgy, chemical engineering, soil science, molecular biology, and materials science [10], such as melting and solidification processes, tumor growth, and fluid flow around obstacles. Free and moving boundary problems represent a class of complex mathematical and computational challenges encountered across various scientific and engineering disciplines. These problems involve determining the shape and position of boundaries that are not known a priori but instead evolve according to the underlying physical laws of the system.

The numerical simulation of free and moving boundary problems is pivotal for predicting system behavior under varying conditions, in optimization problems, and in the design of new technologies. However, these problems are inherently difficult to solve due to the dynamic nature of the boundaries, which may change shape, move, merge, split, or even disappear over time. Accurately representing and tracking the moving or free boundaries in a numerical

simulation is crucial. Methods such as the boundary element method and front-tracking methods [28] have been applied, but each comes with its own set of challenges, including computational efficiency and the accurate representation of complex boundary geometries.

Ensuring numerical stability and convergence is particularly challenging due to the non-linear nature of the problems and the moving boundaries. Techniques like front-fixing methods and fixed domain methods, which involve transforming the moving boundary to a fixed domain, have been developed to avoid numerical instabilities [5]. However such methods require complex coordinate transformations that can introduce additional computational complexities and sources of errors. Moreover, they may not be suitable for all types of free and moving boundary problems, especially those with highly nonlinear dynamics.

There are no strict definitions that clearly distinguish between free and moving boundary problems. However, Crank [10] suggests a way to differentiate the two terms. The term *free* boundary problem typically refers to situations where the boundary to be determined in the problem is stationary and a steady-state problem exists. In contrast, *moving* boundaries relate to time-dependent problems and the position of the boundary has to be determined as a function of time and space. This categorization becomes particularly useful in the field of shape optimization. Within this context, free and moving boundary problems can be reformulated to minimize a shape functional subject to constraints imposed by partial differential equations (PDEs). Thus, free boundary problems often involve elliptic equations, while moving boundary problems are linked with parabolic equations.

1.2 Shape Optimization Approaches

Shape optimization is a distinct area of research that is intricately linked with free and moving boundary problems. In this thesis, we do not distinguish between shape optimization

problems and free and moving boundary problems. Shape optimization is concerned with determining the optimal shape of a domain to achieve a specific goal, often within various constraints. This process may involve minimizing or maximizing factors such as resistance, energy, or stress, depending on the context. Shape optimization can be directly applied to free and moving boundary problems by considering the free or moving boundary as the target shape for optimization. This approach of focusing on optimizing the shape and position of the boundary can be highly beneficial for design and control objectives. Furthermore, it allows for a seamless integration with physical models, enabling simultaneous optimization of performance while addressing the boundary problem.

These benefits position shape optimization as a vibrant research area with a broad spectrum of applications, ranging from aerodynamic optimal shape design [20, 17, 29], image restoration and segmentation [18], and interface identification in transmission processes [26, 30], to the design of drug-eluting stents [36] and horn-like structures used in devices for acoustic or electromagnetic waves [33]. Recent studies on Stefan problem, a classic example of moving boundary problems which investigates the melting of polar ice caps, have focused on their reformulation as shape optimization problems constrained by parabolic PDEs, further enriched by the inclusion of analytical shape derivatives [8].

To rigorously formulate free and moving boundary problems as shape optimization problems constrained by elliptic or parabolic type PDEs on the underlying domain, and to develop an effective numerical scheme, it necessitates a thorough understanding of differential calculus, spaces of geometries, evolution equations, and other core concepts in analysis when dealing with geometric domains, mirroring the theory of functions of real variables. Delfour and Zolésio have presented a comprehensive overview of mathematical constructions and tools in [11]. The relevant and important concepts and results will be succinctly cited in chapter 4 in order to make this thesis as self-contained as possible.

The numerical investigation of shape optimization problems typically employs a line search

framework, where the initial shape is iteratively evolved based on a velocity field. This field is designed to reduce the shape energy functional, in terms of the minimization problem, until a form of convergence is reached. At this point, the final shape is considered as the numerical approximation of the optimal shape solution. Within this process, two critical questions arise: how to evolve the shapes or define the perturbation of the shape object, and how to determine the descent direction of the shape functional, which involves the concept of shape derivatives and shape gradient. These questions are comprehensively addressed in [32, 11], where methods of domain variation using vector fields are introduced, and in [16], which implies that the shape gradient can be articulated through the solutions of the underlying PDEs. This underscores the importance of PDE-solving methods in numerical shape optimization. In terms of numerical implementation, the finite element method is widely used in this field to solve differential equations.

A prevalent challenge with finite element methods, particularly when addressing free and moving boundary problems where the domains and boundaries are constantly changing, is mesh degeneration. As highlighted in [17], mesh degeneration is one of the bottlenecks of the shape optimization techniques that employ finite element solvers. It is crucial to ensure that the boundary of the transformed domain does not self-intersect during the domain transformation process. This requirement adds complexity to the optimization process, necessitating sophisticated techniques to manage and adapt the mesh dynamically as the shape evolves.

Over the last few decades, considerable literature has emerged to tackle mesh degeneracy in numerical methods for shape optimization problems. A straightforward solution is to re-mesh the computational domain whenever mesh degeneracy occurs [34, 13]. However, re-meshing the entire spatial domain, as opposed to focusing solely on the boundary, the primary subject of shape optimization, entails computations in a high-dimensional space, leading to significant additional computational costs. Alternative techniques aim to preserve

mesh quality throughout the gradient descent process. Methods such as mesh regularization and angle control [4], and geometrically consistent mesh modification [7] have been proposed to adjust and maintain mesh quality.

Moreover, some strategies focus on designing shape gradients to prevent mesh degeneration. For instance, nearly conformal transformations are applied to shape gradients to preserve mesh angles [22], or only restricted perturbation fields induced by normal boundary forces are employed to address mesh degeneracy [14]. Adaptive strategies that guide the line search direction through tangential movements of the boundary nodes [25] are designed to prevent mesh distortion. Additionally, more stable shape evolution schemes, like a semi-implicit Euler discretization for the velocity method with time adaptivity and backtracking line search [12], have been explored to manage error accumulation in explicit time-marching schemes. These varied approaches demonstrate the complexity and the innovative solutions developed to address the challenges of mesh degeneration in shape optimization.

Another prevalent challenge in shape optimization algorithms that use finite element approximation is the low order of convergence. Most shape optimization algorithms employing finite elements rely on linear finite elements. Although this choice benefits from computational simplicity and efficiency, it can lead to a notable decrease in approximation accuracy. Research examining the effect of different shape gradient formulations on convergence order within finite element approximation [19] concludes that shape energies defined in boundary integrals result in a lower order of convergence compared to those defined in volume integrals. This discrepancy arises because finite element solutions possess only limited regularity, making a global integration by parts from volume to boundary expression unfeasible.

The predominant use of linear elements is primarily due to their lower computational demands in terms of both memory and processing power, which is particularly relevant for shape optimization problems that require multiple iterations, making higher degree elements more time-consuming. Furthermore, higher degree elements are known to introduce numerical

instabilities and are more sensitive to mesh quality and distortions, which can deter their use despite their greater accuracy.

In summary, the preference for linear finite elements in shape optimization is a balance between the improved accuracy offered by higher degree elements and the increased computational complexity, implementation challenges, and the potential for instability that they entail. This characteristic of the finite element method hampers the overall convergence order of shape optimization algorithms that rely on it.

Addressing both mesh degeneration and low-order convergence challenges in numerical methods for free and moving boundary problems calls for an alternative approach to boundary value problem solvers. This alternative should ideally minimize the need for extensive mesh generation efforts and be cost-effective in achieving higher orders of convergence. Such a solution would not only streamline the computational process but also enhance the accuracy and efficiency of solving shape optimization problems. The goal is to find and develop techniques that can adapt to changing domain shapes without the need for constant re-meshing, while simultaneously offering improved convergence rates compared to traditional linear finite element methods.

1.3 Smooth Extension Embedding Methods

The *Smooth Extension Embedding Methods* (SEEM) introduced by Agress and Guidotti [3] emerges as a promising PDE-solving technique for formulating an effective and high-order numerical shape optimization approach. SEEM is characterized by its unique and essential features, such as eliminating the need for meshing or re-meshing even when handling complex domains, and achieving a high order of convergence for smooth problems. These attributes make SEEM particularly suitable for overcoming the traditional challenges associated with

mesh degeneration and low-order convergence in shape optimization. Further details on the construction and application of SEEM will be given in Chapter 3.

SEEM addresses problems in complex domains by selecting an optimal smooth extension within a larger, simple hold-all domain, allowing for straightforward discretization. This key advantage removes the necessity to create intricate meshes for domains of varying geometries. The straightforward discretization of the simple hold-all domain also facilitates the discretization of smoothing and differential operators. For instance, fast Fourier transforms can be utilized on a uniform grid for the torus, while Chebyshev transformations are suitable for Chebyshev grids on a box.

The straightforward discretization of the simple hold-all domain can remain constant throughout the shape optimization process, provided that the evolution of the free boundary or the path of the moving boundary remains within the confines of the same hold-all domain. As the boundaries evolve through iterations, only the boundary points need tracking, allowing the domain grids to stay unchanged. This approach significantly lowers the computational effort related to mesh generation commonly found in finite element-based algorithms.

The SEEM approach for solving boundary value problems within a large, simple hold-all domain draws inspiration from the fictitious domain method, although the implementations differ significantly. Essentially, SEEM calculates a discrete extension of the solution to the boundary value problem by selecting a smooth solution from the complete affine family of solutions to the original equations, resulting in an underdetermined problem defined across the entire fictitious domain. Since there is no need to deliberately extend the problem itself to a larger domain, the regularity of the SEEM solution can be easily maintained across the boundary. The actual smoothness of this SEEM extension automatically depends on the smoothness of the analytic solution and the chosen objective functional, allowing for precision tuning of the SEEM solution by choosing different orders of the objective functional. Consequently, if the analytic solution is inherently smooth, the order of SEEM can reach the

limits of what the computing machine can handle.

Beyond the adjustable high-order capability of SEEM when applied to smooth problems, it is noteworthy that shifting to a high-order setting within machine precision limits does not complicate the implementation or extend the processing time. This ease of adjustment is because, for sufficiently smooth problems, the order of SEEM is determined by the order of the smooth norm used in the objective functional. For instance, if the smooth norm is the Sobolev H^p norm, it can be efficiently discretized as a diagonal matrix under fast Fourier transforms. Therefore, transitioning the algorithm from a low order to a high order merely involves modifying the exponent of the diagonal matrix, making the process straightforward and efficient.

An additional advantage of SEEM, which could prove beneficial in future investigations into moving boundary problems, is the utilization of a variant of SEEM employing Chebyshev nodes and polynomials, as outlined in [1]. This variant offers an accurate and efficient space-time scheme suitable for time-dependent problems, such as parabolic PDEs. This becomes particularly noteworthy when applied to moving boundary problems in comparison to traditional time-marching methods that accumulate errors over time. SEEM in a Chebyshev setting empowers the algorithm to maintain accuracy and concurrently reduce computational costs.

The unique attributes of SEEM suggest its potential as a viable alternative to the traditional finite element method for solving PDEs within the shape optimization framework for free and moving boundary problems. Nonetheless, there are limitations that impede its straightforward application to shape optimization techniques.

SEEM demonstrates significant effectiveness for smooth problems, yet the numerical representation of these smooth objects may become non-smooth, particularly in the context of free and moving boundary problems. Here, exact boundaries are unknown and must be

approximated throughout the process, often leading to numerically perturbed or distorted shapes, despite theoretical regularity preservation. The original configuration of SEEM is sensitive to such non-smoothness in numerical boundaries, typically resulting in breakdown after several iterations of shape optimization evolution.

To effectively incorporate SEEM into the shape optimization framework and leverage its unique benefits, enhancing its stability in handling non-smooth numerical representations of underlying smooth shapes is essential. Additionally, a robust boundary representation algorithm is crucial throughout the shape optimization iterations. While there may be other effective boundary value problem solvers suitable for integration into the shape optimization framework for free and moving boundary problems, this thesis focuses on exploring and extending the application of SEEM to various numerical computation facets, setting aside discussions on alternative methods.

1.4 An Overview

Throughout our discussions, we assume that the problems being tested have sufficient regularity, without exhaustively stating the least regularity requirements for each specific scenario. Essentially, our main interest is in problems admitting smooth solutions that are suitable for high-order methods. The methodologies we have developed in this thesis are particularly effective in scenarios with sufficiently smooth solutions. It is important to note that the primary computational tools we use can also address problems with less regularity, where the effectiveness of our methods depends on the regularity of the problem itself. In many real-life examples of free and moving boundary problems, the inherent smoothness is typically guaranteed by the governing physical laws.

The primary contribution of this thesis is the expansion of SEEM to tackle free bound-

ary problems under the framework of shape optimization. Specifically, our contributions manifest in several key areas. We have enhanced the stability of SEEM when addressing boundary value problems (BVPs) on non-smooth numerical representations of inherently smooth boundaries through the use of regularized level set boundary approximation. This enhancement not only facilitates the use of SEEM as a BVP solver but also aids in developing an effective high-order shape optimization approach for free boundary problems. Our experiments reveal that the refined SEEM algorithms maintain high-order accuracy in shape gradient approximation, allowing for the direct application of a straightforward shape gradient formula derived from boundary integral definitions. The shape optimization method based on SEEM successfully avoids computational issues such as mesh degeneration that can occur during shape evolution. Moreover, SEEM utilizes simple domain grids throughout the entire shape optimization process, leading to substantial savings in computational resources. The SEEM-based shape optimization algorithm proves effective in practical problems, such as the obstacle problem, demonstrating high convergence rates and computational efficiency in our experiments.

The structure of the thesis is laid out as follows: Chapter 2 provides a comprehensive overview of boundary approximation techniques, covering both explicit and implicit representations. We then delve into a kernel-based level set method and its regularized variants. Given that the approximated boundary is implicitly defined by the 1-level set of the level set function, we explore different boundary point sampling algorithms tailored to various applications to determine the locations of the boundary points.

In Chapter 3, we introduce a detailed yet comprehensive construction of the original SEEM algorithms. This is followed by an in-depth analysis of the condition numbers and the development of preconditioners for various operators within the algorithm. We propose a regularized preconditioning technique for SEEM, which is inspired by the regularized kernel-based level set boundary approximation algorithm. Our experiments demonstrate that this

improved SEEM exhibits robust stability while preserving accuracy, even when applied to rough discrete boundaries.

In Chapter 4, we introduce basic notations and fundamental results in the field of shape optimization, particularly as they relate to the formulation of free boundary problems. We conclude the chapter with a verification example that demonstrates how SEEM maintains the order of convergence when approximating the shape gradient in the boundary integral formula, whereas the same task using finite element approximation experiences a loss in convergence order.

With the groundwork laid in the previous chapters, Chapter 5 introduces the main algorithm that employs the improved SEEM techniques within a shape optimization approach to solve free boundary problems. This is followed by detailed remarks on each step of the process. To illustrate the effectiveness of SEEM in overcoming the mesh degeneration issue during shape deformation in the shape optimization process, an example of a free boundary problem is provided at the end of this chapter. The algorithm demonstrates a high order of convergence in experiments.

Chapter 6 presents a practical application of our algorithm in solving the obstacle problems. Compared to the existing algorithms based on the universal mesh finite element method, our algorithms exhibit a superior order of convergence, along with additional benefits such as straightforward implementation and reduced computational costs.

This thesis concludes with a summary of the main results and contributions in Chapter 7.

Chapter 2

Boundary Shape Approximation

2.1 Introduction: Boundary Approximation Methods

Shapes, particularly their boundaries, represent the primary unknown variables in free and moving boundary problems. It is crucial to accurately and stably identify and conduct further calculations on these shapes. The iteration of free boundaries and the evolution of moving boundaries necessitate boundary reparametrization. In numerical algorithms, boundaries are expressed as a set of discrete points. A robust boundary approximation algorithm is essential to precisely extract useful geometric information from the given set of points. This ensures that other points on the underlying boundary can be accurately located, furthermore, the differentiation and integration on the boundary can be accurately approximated. This task is challenging in practice as discrete points may not be uniformly distributed along the underlying boundary and could be unordered or affected by computational errors.

Two main approaches exist for representing and approximating shapes: explicit parametrization and implicit level set methods. Explicit parametrization offers advantages such as lower computational costs, given that the parameter domain for the boundary hypersurface $\Gamma \subset \mathbb{R}^d$

lives in the lower dimensional space $\mathbb{R}^{d-1}$. A one-dimensional reduction significantly lowers overall computational costs. Additionally, once the parametric representation of the boundary is established, it is straightforward to generate a new set of boundary points in any desired distribution or to perform boundary integration based on the approximated parametrization.

However, the parametrization approach has notable disadvantages that limit its adaptivity in terms of shape approximation. Finding accurate parameters for each given discrete point is challenging, especially for complex shapes like non-convex ones where polar or spherical coordinates may not be well-defined, leading to multiple boundary points in certain angular directions. Although the arc-length parameter is well-defined for closed boundaries in two dimensions (i.e., closed plane curves), determining the parameters themselves can be difficult. For example, for closed plane curves, one can fix a reference point and follow the curve in one of the two possible directions, using the arc-length from the reference point as the parameter for each discrete point. While this parametrization is well-defined, the mismatch between the approximated arc-length parameter and the exact parameter for each discrete point can significantly impact the overall quality of the boundary shape approximation. The mismatch in the arc-length parameter usually is inevitable due to the fact that the continuous curve is unknown or due to errors in the approximation to the curve. Moreover, generating a coarse approximation to the curve when the given discrete points are not listed in order can also be a challenging task.

Another approach to boundary representation is the implicit level set method, which characterizes an $(d-1)$-dimensional boundary Γ as the zero (or any constant) level set of a function defined inside $\mathbb{R}^d$. Specifically, consider a function $\phi(x) : \mathbb{R}^d \to \mathbb{R}$ where its zero level set $\{x \in \mathbb{R}^d | \phi(x) = 0\}$ delineates the boundary between the interior and exterior of an d-dimensional domain. Although $\phi(x) = 0$ is selected to depict the boundary hypersurface Γ, the choice of the zero level set is not inherently special. The boundary hypersurface Γ

with an $d-1$ dimension, is closed such that the interior and exterior regions are clearly defined.

It has been pointed in [27] that defining boundaries implicitly as the level set of a function offers numerous advantages over explicit parametric methods, where the points comprising the boundary are specifically outlined. In complex scenarios, such as boundaries in three-dimensional space that lack an analytical representation, discretizing explicit parametric representations can be challenging. This process requires selecting a number of points on the two-dimensional surface and noting their connectivity. Connectivity in two dimensional space is determined based on the ordering, while in three dimensions is less straightforward. Moreover, connectivity can change during the evolution of the shapes. However, this issue is seamlessly managed by the implicit level set approach. One of the key benefits of implicit shapes is that there's no need to establish connectivity for discretization. A uniform Cartesian grid in $\mathbb{R}^d$ can be used along with straightforward generalizations of the technology from two spatial dimensions. Possibly the most powerful aspect of implicit surfaces is that it is straightforward to go from two spatial dimensions to three spatial dimensions (or even more).

While implicit representation offers numerous benefits, it also presents challenges depending on the application. In two-dimensional spaces, defining a boundary explicitly involves identifying every point along a curve. A common method for approximating an explicit representation involves discretizing its parameter into a finite set of points, which are not necessarily equally spaced. For each parameter point, the corresponding location on the two-dimensional curve is determined and stored. Increasing the number of points in the discretized parameter space enhances the resolution or detail of the two-dimensional curve. Conversely, the implicit representation used in traditional level set methods identifies the level set function by solving a system of partial differential equations (PDEs). Typically, this level set function is not known analytically and is approximated through numerical

PDE solutions. As a result, the implicit representation must be stored on a discretized bounded subdomain of $\mathbb{R}^2$, highlighting a potential drawback. Instead of merely resolving a one-dimensional parametric interval as in explicit representations, one must resolve a two-dimensional region. More broadly, in $\mathbb{R}^d$, discretizing an explicit representation only requires resolving an $(d-1)$-dimensional set, whereas an implicit representation necessitates resolving an d-dimensional set. This issue can be mitigated, to some extent, by concentrating all points close to the boundary, leaving the rest of the domain less defined. This strategy, known as a local approach, clusters points near the boundary to discretize implicit representations more efficiently.

Moreover, because the level set function ϕ has to be numerically solved from level set equations, which are partial differential equations (PDEs), any subsequent calculations of the normal vector field or curvature on the boundary based on the numerical approximation of the level set function will significantly lose accuracy upon differentiation. This necessitates the use of higher-order schemes in numerical PDE solutions, which can, in a way, increase the computational load. This increase is particularly significant because boundary approximation is a foundational step that will be repeatedly applied throughout the overall process of solving free and moving boundary problems. Therefore, the efficiency of this step is critical to the overall efficiency of solving free and moving boundary problems, highlighting the need for a balance between accuracy and computational speed in these applications.

Consequently, an alternative method for obtaining the level set representation of boundaries through a kernel-based approximation will be introduced in the forthcoming sections 2.2. This approach enables the construction of an analytical level set function for the boundary, requiring only the storage of a few coefficients—typically as many as the number of input discrete boundary points. Furthermore, possessing an explicit formula for the level set function allows for a more accurate approximation of the normal vector field and curvature on the boundary by performing analytical differentiation of the level set function. To derive

such a level set function, it is only necessary to solve a linear system whose size corresponds to the number of boundary sample points. This is in contrast to the traditional level set equation, which must be solved across the entire (or a substantial portion of the) spatial domain. By reducing the computation to a lower dimension, this method significantly conserves computational resources.

2.2 Kernel-Based Level Set Approximation

This section introduces the kernel-based level set approach for approximating boundary shapes, as proposed by Guidotti [15]. For a detailed understanding, readers are encouraged to consult the original work. Below, we offer a concise summary of the procedure and of its key properties.

Given a set of data pairs

$$\mathbb{D} = \{(x^i, y^i) \mid x^i \in \mathbb{R}^d, y^i \in \mathbb{R}^n, i = 0, 1, \ldots, N-1\}.$$

In the context of two-dimensional plane curve or three-dimensional surface approximation, $y^i = 1$ for all i, $x^i \in \mathbb{R}^d$ for $d = 2$ or $d = 3$, indicate the sample locations along the boundary, which may be unevenly distributed or arranged in no particular order.

First consider the non-regularized case, where one aims to find a level set function $\phi(x) : \mathbb{R}^d \to \mathbb{R}$ whose 1-level set exactly interpolates all the sample locations $\{x^i\}_{i=0}^{N-1}$. As proposed in [15], the level set function ϕ is chosen as the unique solution of the following constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\phi\|_{\mathcal{S}}^2, \\ \text{subject to} \quad & \phi(x^i) = y^i, \quad i = 0, 1, \ldots, N-1, \end{aligned}$$

where $||\cdot||_{\mathcal{S}}$ denotes some smooth norm, which is paired with the smooth operator S in the following sense

$$||\phi||_{\mathcal{S}} = ||S^{1/2}\phi||_{L^2}.$$

The corresponding unconstrained optimization problem is given by

$$\phi(x) = \underset{\phi\in\mathcal{D}(\mathbb{R}^d,\mathbb{R})}{\arg\min}\ \max_{\lambda_i\in\mathbb{R}} \left\{ \frac{1}{2}\|\phi\|_{\mathcal{S}}^2 + \sum_{i=0}^{N-1} \lambda_i(\phi(x^i) - y^i) \right\},$$

where the $\mathcal{D}(\mathbb{R}^d, \mathbb{R})$ denotes the corresponding space associated with the smooth norm $\|\cdot\|_{\mathcal{S}}$. The normal equation in the distributional sense is

$$S\phi(x) = \sum_{i=0}^{N-1} \lambda_i \delta_{x^i}(x),$$

where $\delta_{x^i}(x) = \delta(x - x^i)$ is the shifted dirac delta distribution. Apply the inverse operator S^{-1} on both sides, one gets an explicit representation of the non-regularized level set function

$$\phi(x) = \sum_{i=0}^{N-1} \lambda_i S^{-1}\delta_{x^i}(x). \tag{2.1}$$

Denote the kernel function centered at x^i as

$$\phi_i(x) := S^{-1}\delta_{x^i}(x), \tag{2.2}$$

then the coefficients $\Lambda = [\lambda_i]_{i=0}^{N-1}$ can be solved from the linear system

$$M\Lambda = \mathbb{Y}, \tag{2.3}$$

where $\mathbb{Y} = [y^i]_{i=0}^{N-1}$, and M is an $N \times N$ matrix whose ij-th entry is

$$M_{ij} = \phi_i(x^j) = \phi_j(x^i). \tag{2.4}$$

Theoretically, given a smooth norm $\|\cdot\|_{\mathcal{S}}$, formulae (2.1)–(2.4) explicitly construct a function whose 1-level set interpolates the underlying boundary exactly containing all the data points $\{x^i\}_{i=0}^N$. However, in practice the matrix equation (2.3) can be ill-conditioned due to the low quality of the data set.

2.3 Regularized Kernel-Based Level Set Approximation

To mitigate the ill-conditioning of the kernel-based level set algorithm, implementing techniques to control the condition number is crucial for developing a stable boundary approximation algorithm. Achieving stability may involve compromising some interpolation accuracy, which means it is not strictly necessary to enforce the exact validity of $\phi(x^i) = y^i$. Instead, the aim now becomes to find a regularized level set function $\phi_\alpha(x)$ that approximates the data pairs $\mathbb{D}$ to minimize the following functional

$$J_\alpha(\phi) = \frac{\alpha}{2}||\phi||_{\mathcal{S}}^2 + \frac{1}{2}\sum_{i=0}^{N-1} |\phi(x^i) - y^i|^2, \quad \alpha > 0.$$

Here the positive parameter α together with the first term are designed for regularity, and the second term is obviously for data matching. The other notations are the same as the non-regularized case. The corresponding normal equation in the distributional sense is as

follows

$$\alpha S\phi_\alpha(x) = \sum_{i=0}^{N-1}(y^i - \phi_\alpha(x^i))\delta_{x^i}(x).$$

The solution $\phi_\alpha(x)$ of the normal equation above in the boundary shape approximation context represents the regularized level set function whose 1-level-set approximates the smooth boundary. Rewrite the normal equation above, one gets

$$\phi_\alpha(x) = \frac{1}{\alpha}\sum_{i=0}^{N-1}(y^i - \phi_\alpha(x^i))S^{-1}\delta_{x^i}(x). \tag{2.5}$$

Under the notations defined in (2.2) and (2.4) which are independent of the regularization parameter α, the notation $\mathbb{X} = \{x^i\}_{i=0}^{N-1} \subset \mathbb{R}^d$ indicating the set of discrete locations, and $\Phi_{\mathbb{X}} = [\phi_\alpha(x^i)]_{i=0}^{N-1}$ indicating the vector of the regularized level set function ϕ_α evaluated at given locations x^i, the above normal equation can be rewritten as follows

$$(\alpha I + M)\Phi_{\mathbb{X}} = M\mathbb{Y}, \tag{2.6}$$

where I is $N \times N$ identity matrix. Denote the kernel-based representation of the regularized level set function as

$$\phi_\alpha(x) = \sum_{i=0}^{N-1}\lambda_\alpha^i\phi_i(x), \tag{2.7}$$

and the weighted coefficients $\Lambda_\alpha = [\lambda_\alpha^i]_{i=0}^{N-1}$, formula (2.5) yields

$$\lambda_\alpha^i = \frac{1}{\alpha}(y^i - \phi_\alpha(x^i)), \quad i = 0, \ldots, N-1,$$

then in vector form, Λ_α becomes

$$\Lambda_\alpha = \frac{\mathbb{Y} - \Phi_{\mathbb{X}}}{\alpha} = \frac{\mathbb{Y} - (\alpha I + M)^{-1}M\mathbb{Y}}{\alpha},$$

which is equivalent to

$$(\alpha I + M)\Lambda_\alpha = \mathbb{Y}. \tag{2.8}$$

The kernel-based representation (2.7), combined with formulae (2.2), (2.4), and (2.8), fully defines the regularized level set function $\phi_\alpha(x)$. Hence the approximated boundary is

$$\Gamma_\alpha = \{x \in \mathbb{R}^d | \phi_\alpha(x) = 1\}.$$

From (2.6) it is evident that the 1-level set of the regularized ϕ_α function does not exactly interpolate the original dataset $\mathbb{D}$. In fact, the evaluation of the ϕ_α function at the set of given locations $\mathbb{X}$ is $\Phi_{\mathbb{X}} = (\alpha I + M)^{-1} M\mathbb{1}$, which deviates from the unit vector $\mathbb{1}$ when $\alpha > 0$. However, this minor loss in accuracy significantly improves the stability of the boundary approximation algorithm. Furthermore, the approximation error can be minimized by selecting a small value for α.

In [15], the space under consideration is described as follows

$$H^{\frac{d+1}{2}}(\mathbb{R}^d, \mathbb{R}^n) = \{u \in \mathcal{S}' \mid [\xi \mapsto (1 + |\xi|^2)^{\frac{d+1}{2}} \hat{u}(\xi)] \in L^2(\mathbb{R}^d, \mathbb{R}^n)\}.$$

Then the resultant kernel function is

$$\phi_i(x) = S^{-1}\delta_{x^i}(x) = (1 - \Delta)^{-\frac{d+1}{2}}\delta_{x^i}(x) = Ce^{-2\pi|x-x^i|},$$

for some constant C. According to the framework mentioned above, one can replace formula (2.2) by specified kernel functions $\{\phi_i\}_{i=0}^{N-1}$ which are centered at the data locations $\{x^i\}_{i=0}^{N-1}$, then use the remaining formulae to finalize the level set function. In practice, the modified

kernel functions from [15] used in shape optimization are

$$\phi_i(x) = e^{-a|x-x^i|}, \tag{2.9}$$

where a by default is chosen as 1, for specific problems it may be adjusted to scale the radius of numerical support.

2.4 Boundary Points Sampling Methods

From the boundary approximation algorithms discussed in the previous section, we derive a level set function $\phi(x) : \mathbb{R}^d \to \mathbb{R}$, where the boundary of interest is implicitly represented as the 1-level set of the function $\phi(x)$. To perform further calculations at specific points on the curve, it is necessary to sample points from the level set of $\phi(x)$. This step is crucial in various fields, including computer graphics, scientific visualization, and numerical analysis. Although many methods have been proposed in this area, this thesis primarily focuses on introducing two methods that are employed within our research.

2.4.1 Marching Algorithm

The Marching Squares method [24] is a widely utilized algorithm in the field of computer graphics, playing a pivotal role in visualizing and analyzing two-dimensional function evaluations. It primarily generates contour lines by connecting points with the same value and can also sample points along these contours. Its simplicity and efficiency in implementation enables the real-time generation of contours in various applications.

The Marching Squares method segments the input function evaluations into a grid of squares and evaluates the value of the function at each corner of these squares to determine how

the contour intersects the square. Based on these values, the algorithm classifies the square into one of a finite number of cases, each representing a different pattern of intersection. The contour lines are then interpolated within each square according to the identified case, ensuring a continuous and accurate representation of the level set. This methodical approach allows Marching Squares to efficiently and accurately render complex shapes and structures, making it a fundamental tool in the field of computer graphics and data visualization.

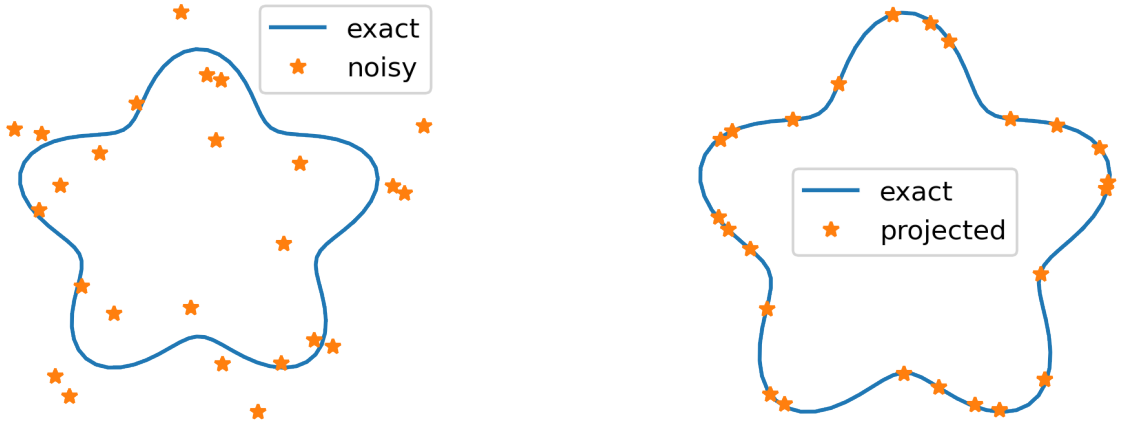
In the algorithm outlined in this thesis, we primarily use the Marching Squares method for sampling boundary points, largely because of its straightforward implementation. In fact, the python function `matplotlib.pyplot.contour` is built based on this algorithm.

2.4.2 Optimization-Based Method

While the Marching Squares method is efficient for implementation, its reliance on linear interpolation can lead to inaccuracies for certain applications. If high precision in boundary point sampling is necessary, one can refine the points obtained from Marching Squares to meet the accuracy requirements.

Recall that sampling points from the level set involves finding $x \in \mathbb{R}^d$ such that $\phi(x) = 1$ for the smooth level set function $\phi(x) : \mathbb{R}^d \to \mathbb{R}$. This constitutes a root-finding problem, and the differentiability of the ϕ function allows for the application of Newton's method, which is quadratically convergent, to efficiently and accurately adjust the sample points. Starting from an initial point $x^0 \in \mathbb{R}^d$, we seek to find the root of the function $f(x) = \phi(x) - 1$ in $\mathbb{R}^d$. Consider the derivative of f at x^k in the following notations

$$A = Df(x^k) = (\nabla\phi(x^k))^\top \in \mathbb{R}^{1\times d},$$



(a) Random Sample (b) Projected Sample

Figure 2.1: The sample points projected back to the exact curve by Newton's method.

and Newton's method provides the following formula for updating the point

$$x^{k+1} = x^k - A^+ f(x^k).$$

Here $A^+ = A^\top (AA^\top)^{-1}$ represents the pseudo-inverse. Notice that the unit normal vector ν can be defined by the level set function ϕ as $\nu = -\frac{\nabla\phi}{|\nabla\phi|}$, where the negative sign indicates that the unit vector points outward, as the level set function defined by the (regularized) kernel-based algorithm has a larger value inside the boundary. The pseudo-inverse matrix can thus be simplified to

$$A^+ = \nabla\phi(|\nabla\phi|^2)^{-1} = -\frac{\nu}{|\nabla\phi|}.$$

Therefore, the sample points can be iteratively updated as follows

$$x^{k+1} = x^k + \left(\frac{\phi(x^k) - 1}{|\nabla\phi(x^k)|}\right)\nu(x^k),$$

until the updating coefficient $|\phi(x^k) - 1|/|\nabla\phi(x^k)|$ falls below a certain small tolerance. This process effectively moves the points in the direction normal to the surface, showcasing Newton's method's capability to direct the points towards the exact boundary.

Figure 2.1 shows how Newton's method can successfully project poorly sampled points, corrupted by Gaussian noise, back onto the precise curve.

Chapter 3

Smooth Extension Embedding Methods

3.1 Introduction: Distinctive Features

The Smooth Extension Embedding Method (SEEM) is a novel efficient and high order boundary value problem (BVP) solving method based on a fictitious domain formulation. It has distinct advantages over several existing BVP solvers.

Comparing to the finite element method, probably the most commonly used BVP solution method in the shape optimization aspect of free and moving boundary problems, the significant difference lies in the discretization of the domain. The finite element method typically uses triangular meshing techniques to adaptively discretize domains with complex geometries. As in free and moving boundary problems the domain changes in each optimization step or evolves in time, there is a need to maintain meshing quality and a computational cost associated to any required remeshing. As the SEEM method uses a simple and fixed grid for a larger hold-all domain for any complex domain and its evolution, there is no (re)meshing

need for SEEM throughout the computational process as long as the hold-all domain contains the full evolution. SEEM captures the geometries of the complex and evolving domains by interpolating their boundary points, which is a lower dimensional task compared to meshing and remeshing of the whole domain. Therefore, a relatively denser sample of points used for boundary characterization becomes affordable and acceptable in an attempt to increase the overall accuracy of the BVP solution. Although SEEM needs to discretize a larger hold-all domain and it seems to waste computational resources in the 'outside' region, its high-order advantage compensates for this. It achieves equivalent accuracy to other low-order methods on a significantly coarser grid when applied to smooth problems.

SEEM has also other advantages when applied in the context of free and moving boundary problems. First, it has been shown in [2] that the order of convergence of SEEM with the pth order Sobolev smoothing kernel is $(p-2)$. Assuming sufficient smoothness, when SEEM is applied to free and moving boundary problems, the BVP solver can theoretically achieve spectral convergence. However, due to the limitations of machine precision, the actual order of convergence cannot be infinite. Despite this, the convergence rate can still be impressively high. In other words, for smooth problems, the high order of convergence of SEEM makes it possible to achieve high accuracy with a limited number of boundary points and a very sparse domain grid. This saves plenty of computational resources, an important consideration in the solution of free and moving boundary problems. In particular more computational effort can be focused on seeking the unknown solution boundary (domain). Second, SEEM can be used with unordered boundary points, though, in its original implementation it still requires the boundary points to be equally distributed along the arc length of the boundary curve. We will see in Section 3.2, SEEM addresses the issue of unordered boundary points by treating it as row permutation of the boundary operator matrix. Therefore, unordered boundary points do not affect the approximation result of SEEM, since row permutation does not impact the linear system solving. In fact, in section 3.3 we will see that the requirement on the boundary point distribution can be relaxed to a much general case. Even if the boundary points form

a non-smooth representation for the smooth shape, SEEM still have the ability to obtain a good numerical BVP solution defined on the underlying domain. This makes SEEM robust enough to serve as the BVP solver for free and moving boundary problems.

A detailed formulation of the SEEM method and its algorithmic implementation will be given in section 3.2. Following that, we will give a more thorough discussion of the two main implementations of SEEM: the iterative and the direct solvers. Subsequently, we introduce a regularized preconditioning technique inspired by the regularized kernel-based boundary approximation algorithm for the iterative approach, as outlined in 3.3. This chapter concludes with experiments demonstrating the efficacy of our proposed regularized preconditioners in reducing the condition number and enabling SEEM to address boundary value problems in domains lacking known analytical representations.

3.2 Original Implementations

For a comprehensive overview of the Smooth Extension Embedding Methods (SEEM) construction, readers are directed to [3, 1, 2]. This section provides a concise presentation of the key concepts to ensure that this thesis be as self-contained as possible. Consider a second order elliptic boundary value problem of the following form

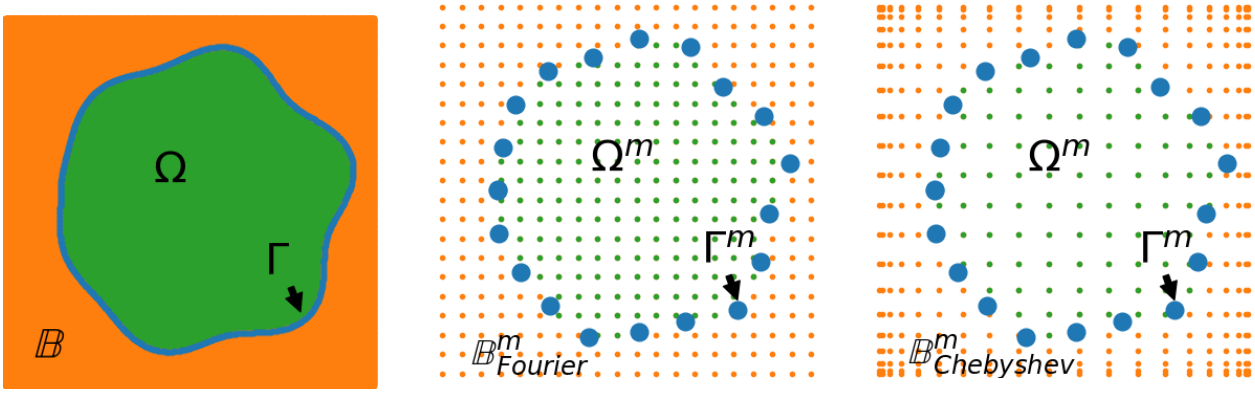
$$\begin{cases} \mathcal{A}u = f & \text{in } \Omega, \\ \mathcal{B}u = g & \text{on } \Gamma = \partial\Omega, \end{cases} \tag{3.1}$$

where $\mathcal{A}$ is a second order elliptic differential operator (such as the Laplace operator $-\Delta$), while $\mathcal{B}$ is a boundary operator (such as, e.g., the trace γ_Γ for Dirichlet problem). The domain Ω has some complex geometries. Traditional methods for boundary value problems when dealing with complex geometries usually require additional implementation and

computational costs. For example, the finite element methods are mesh-based, they have the computational burden of generating a mesh for an arbitrary domain. While the finite difference methods have the simpler formulation of the discretization, but it is not straightforward to implement for complicated domains. Also, traditional spectral methods are only directly available when the domain has a special shape such as in the case of the torus or the rectangular box, where the spectral transformation (such as, e.g., the Fourier transform or the Chebyshev transform) applies.

The SEEM provides a stable and efficient approach to solve boundary value problems on arbitrary domains. It does so by embedding the arbitrary domain Ω into a larger regular domain $\mathbb{B}$ (a box), then it imposes the equations of the boundary value problem (3.1) only in the smaller domain of interest by using numerical discretizations that live on the larger encompassing domain. The solution that the SEEM computes is a smooth extension of the solution within Ω to the large domain $\mathbb{B}$. Many different choices of straightforward discretization $\mathbb{B}^m$ are available for the regular domain $\mathbb{B}$. Figure 3.1 illustrates the domain embedding mentioned above and provides two examples of straightforward discretization. Figure 3.1b (partially) displays a uniform grid that integrates with the Fourier transforms on the torus [3], while Figure 3.1c demonstrates a Chebyshev grid for a rectangular box, suitable for addressing problems involving non-periodicity [1]. Rather than extending the problem artificially across the boundary into the larger domain, like it is done in fictitious domain methods, the SEEM only imposes the differential equations in $\Omega \cup \Gamma$ using the points from the regular grid $\mathbb{B}^m$. The discretization of Ω is simply obtained as $\Omega^m = \mathbb{B}^m \cap \Omega$. The boundary can be discretized by a set points that are not necessarily elements of $\mathbb{B}^m$, such as the points evenly distributed along the arc length of Γ, as shown in Figure 3.1b and 3.1c. The set of chosen boundary points is denoted by Γ^m. In order to impose the boundary conditions on Γ^m, the SEEM uses interpolation operators on the regular grid $\mathbb{B}^m$.

The matrix of the discrete boundary value problem (3.1) denoted as $Cu = b$ can be repre-



(a) (b) (c)

Figure 3.1: Domain embedding and straightforward discretization used in the SEEM.

sented as follows,

$$C = \left[\begin{array}{c} A \in \mathbb{R}^{N_{\Omega^m} \times N_{\mathbb{B}^m}} \\ \hline B \in \mathbb{R}^{N_{\Gamma^m} \times N_{\mathbb{B}^m}} \end{array}\right], \tag{3.2}$$

where N_D indicates the number of points in the corresponding set D (D can be one of $\mathbb{B}^m$, Ω^m, and Γ^m). Each row of the matrix C imposes the interior or the boundary conditions at a single point. Notice that the permutation of the rows in matrix C has no effect on the solvability of system, so that the exact order of boundary points used to generate boundary matrix B makes no difference in the SEEM implementation. Since the domain $\mathbb{B}$ is larger than the domain Ω, the matrix C is a rectangular matrix with fewer rows than columns (supposing, of course, the number of boundary points is fewer than the exterior grid points), therefore the problem (3.1) is typically under-determined. There are many possible extensions of the solution to the larger domain $\mathbb{B}$. However, there exist oscillatory discrete functions satisfying the discrete equations which do not approximate the solution well. Consider a

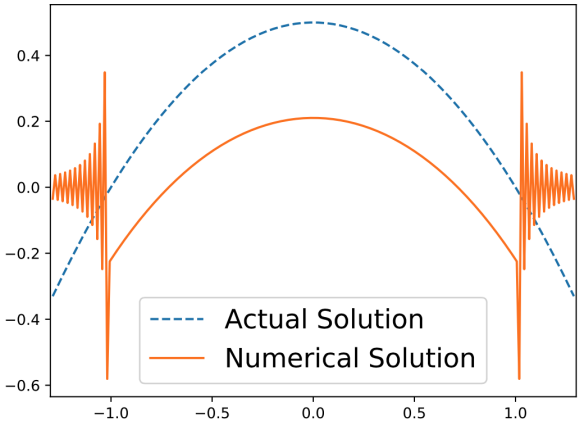


Figure 3.2: A visualization of the oscillations caused by trivial extension with no regularization.

one-dimensional example

$$\begin{cases} -\partial_{xx} u = f, & x \in (-1, 1), \\ u = 0, & x = \pm 1. \end{cases} \tag{3.3}$$

Figure 3.2 displays an oscillatory numerical extension for solving (3.3) without the use of regularization.

The SEEM resolves this issue by reformulating the under-determined problem $Cu = b$ as a constrained optimization problem as follows,

$$\text{argmin}_{\{Cu=b\}} \frac{1}{2}\|u\|_{S_p}^2. \tag{3.4}$$

Rather than extending the PDEs analytically to the complement of the domain, it selects a smooth element of the affine set of solutions. The unconstrained form is as follows,

$$\text{argmin}_u \sup_{\Lambda} \left\{ \frac{1}{2}\|u\|_{S_p}^2 + \Lambda \cdot (Cu - b) \right\},$$

where $\|u\|_{S_p} = \|S_p^{1/2} u\|_{\ell_2}$ can be any smooth norm whose induced operator S_p can be easily

discretized and inverted in the implementation. For instance, when the hold-all discrete box $\mathbb{B}^m$ is the uniform grid of the torus $[-\pi,\pi)^d \subset \mathbb{R}^d$, the smooth norm can be taken as the Sobolev H^p norm which can be efficiently discretized by the fast Fourier transforms (FFT). Specifically, the induced operator is $S_p = (1-\Delta_\pi)^p$, then the regularizer is its inverse operator

$$S_p^{-1} = (1-\Delta_\pi)^{-p}, \tag{3.5}$$

where Δ_π is the Laplacian operator on the torus, and it can be simply discretized as follows

$$S_p^{-1} = (\mathcal{F}^m)^{-1}\mathrm{diag}[1+|k_m|^2]^{-p}\mathcal{F}^m, \tag{3.6}$$

where $\mathcal{F}^m$ and $(\mathcal{F}^m)^{-1}$ are FFT and its inverse, $k_m \in \mathbb{Z}_m^d$ is the frequency vector on the discrete grid $\mathbb{B}^m$. An alternative operator in Chebyshev settings [1] can be obtained in a similar manner,

$$S_p^{-1} = (1-\sum_{i=1}^{d}\mathcal{D}_i^2)^{-p} = (\mathcal{C}^m)^{-1}\mathrm{diag}[1+|k_m|^2]^{-p}\mathcal{C}^m, \tag{3.7}$$

where $\mathcal{D}_i = \sqrt{1-x_i^2}\frac{\partial}{\partial x_i}$. Eventually, the discrete optimization problem (3.4) can be solved from the regularized normal equation with the Schur complement formula

$$u = S_p^{-1}C^\top(CS_p^{-1}C^\top)^{-1}b. \tag{3.8}$$

The primary challenge in numerically implementing SEEM lies in managing the inversion of the Schur complement matrix $CS_p^{-1}C^\top$. As highlighted by the spectral discretization formulas (3.6) and (3.7), the condition number of the problem correlates with m^{2p}, where m represents the one-dimensional grid number in SEEM. Consequently, if $m^{-2p} < \epsilon_{\text{machine}}$, there is a risk of information loss. To prevent this, two distinct strategies are suggested in

[3, 1] for applying formula (3.8): iterative or direct inversion. One method involves using lower-order operators on dense grids, leading to an iterative approach since these dense grids generate large systems more efficiently solved iteratively. Alternatively, employing higher-order operators on coarser grids results in direct solving methods, as the smaller systems from coarser grids can be directly inverted more feasibly. The implementation details and the key features of these two approaches are discussed below.

3.2.1 Iterative Solver

Notice that in the formula (3.8), the main task is to invert the matrix $CS_p^{-1}C^\top$. As the smooth kernel S_p^{-1} is chosen to be symmetric, then the matrix $CS_p^{-1}C^\top$ is symmetric positive definite, thus the conjugate gradient type methods can be applied. However, it has been found that in general the matrix $CS_p^{-1}C^\top$ is ill-conditioned, and this ill-conditioning mainly comes from two aspects, the mismatch in the order between boundary and interior operators, and the high order of the regularizer S_p^{-1}. To tackle the latter issue, one can simply apply relatively lower order regularizer (in practice, SEEM retains its effectiveness for p up to 4 when using iterative solvers), coupled with denser mesh grids to achieve highly accurate results. For the order mismatch issue, a wisely designed preconditioner is used to stably invert the matrix in the preconditioning conjugate gradient (PCG) method. Since the PCG method is the sole iterative technique utilized in SEEM, we refer to the SEEM implementation that uses this iterative solver as the SEEM-PCG method. Recall that the coefficient matrix C consists of domain and boundary parts as defined in (3.2), then matrix $CS_p^{-1}C^\top$ can be regarded as a block matrix

$$CS_p^{-1}C^\top = \begin{bmatrix} AS_p^{-1}A^\top & AS_p^{-1}B^\top \\ BS_p^{-1}A^\top & BS_p^{-1}B^\top \end{bmatrix}.$$

As S_p^{-1} defined in (3.5) represents an operator of order $-2p$, the matrix A represents the discretization of Laplacian operator with order of 2, the boundary matrix B, taking Dirichlet boundary condition as an example, has an order of 0, therefore the overall orders are $4-2p$ and $-2p$ for domain block $AS_p^{-1}A^\top$ and boundary block $BS_p^{-1}B^\top$ respectively. In general, an operator of order $-n$ introduces a polynomial growth of degree n in condition number as the grid size increases. Therefore, the preconditioner ensuring the stability of the iterative solver for large systems should adjust the diagonal blocks in matrix $CS_p^{-1}C^\top$ so that their order reduces to 0. Denote the preconditioner as

$$P = \begin{bmatrix} P_A & 0 \\ 0 & P_B \end{bmatrix},$$

where P_A and P_B are approximate inverses to $AS_p^{-1}A^\top$ and $BS_p^{-1}B^\top$ respectively.

- **Domain operator preconditioning**

  The goal of P_A is to reduce the order of the block $AS_p^{-1}A^\top$ from $4-2p$ to 0 after the preconditioning $(P_A)^{-1/2}\left(AS_p^{-1}A^\top\right)(P_A)^{-1/2}$. For $p=2$, there is no need for preconditioning, i.e. take P_A as the identity matrix $I_{N_{\Omega^m}}$. For $p=3$ and $p=4$, [3] chooses the following preconditioner,

  $$P_A = (1-\Delta_\Omega)^{p-2},$$

  where Δ_Ω is the Laplacian operator on Ω and is discretized by a finite difference scheme.

- **Boundary operator preconditioning**

  When it comes to boundary preconditioning, rather than constructing an operator of order $2p$ for order elimination, the emphasis is placed on directly seeking a rough inverse of the boundary operator. Consider a Dirichlet problem, where the boundary operator consists of evaluations on the boundary, then the rows of B are the discretizations of

the delta distribution supported at the various boundary points. Denote the discrete boundary $\Gamma^m = \{y_i\}_{i=1}^{N_{\Gamma^m}}$ as the set of points on it, then the matrix $BS_p^{-1}B^\top$ is a discretization of the collocation matrix $\mathcal{M}$ whose elements are defined as

$$\mathcal{M}_{ij} = \langle \delta_{y_i}, S_p^{-1}\delta_{y_j} \rangle, \quad i, j = 1, 2, \ldots, N_{\Gamma^m}. \tag{3.9}$$

Thus the preconditioner P_B is obtained by inverting the explicit collocation matrix, i.e.

$$P_B = \mathcal{M}^{-1}.$$

Notice that the dimension of the collocation matrix compared to the dimension of the full boundary value problem is significantly reduced, making direct inversion of the collocation matrix feasible for much denser grids. Additionally, since the inverse collocation matrix is only used as a preconditioner, it does not need to be precisely calculated, then any rough approximation suffices for this purpose.

In the implementation in [3], the fundamental solution $h(y)$ of the smoother S_p defined by

$$h(y) = (S_p^{-1}\delta)(y)$$

is used to build the collocation matrix. The function $h(y)$ is independent of the actual differential and boundary operators arising from the problem, then it can be pre-saved in a lookup table for convenient further use. The explicit collocation matrix is then constructed by interpolating function $h(y)$ at the designated boundary points for specific problems. The elements of the collocation matrix defined in (3.9) after being

applied to the fundamental solution are given by the following formula

$$\mathcal{M}_{ij} = h(y_i - y_j), \quad i, j = 1, 2, \ldots, N_{\Gamma^m}. \tag{3.10}$$

3.2.2 Direct Solver

Another approach to implement the formula (3.8) is to reformulate it as following

$$u = S_p^{-1/2} S_p^{-1/2} C^\top (C S_p^{-1/2} S_p^{-1/2} C^\top)^{-1} b.$$

The pseudo-inverse formula for underdetermined matrix $A^+ = A^\top (A A^\top)^{-1}$ further simplifies the formula as follows

$$u = S_p^{-1/2} (C S_p^{-1/2})^+ b, \tag{3.11}$$

which can be implemented by direct solvers. Notice that the pseudo-inverse in (3.11) significantly decreases the condition number to the square root of that of the full Schur complement matrix in (3.8). This reduction enables the use of the regularizer S_p with a much higher order p when solving boundary value problems. For self-inclusion, two types of direct solvers used in computing the pseudo-inverse of the explicit matrix $CS_p^{-1/2}$ are briefly introduced below. Only for the following discussion, the notation $M = N_{\Omega^m} + N_{\Gamma^m}$ and $N = N_{\mathbb{B}^m}$ is used to describe the dimensions of relevant matrices for simplicity.

- **QR decomposition**

  Notice that $CS_p^{-1/2}$ is an $M \times N$ matrix. Consider the QR decomposition of its transpose

  $$S_p^{-1/2} C^\top = QR,$$

where $Q \in \mathbb{R}^{N\times M}$ is an orthogonal matrix satisfying $Q^\top Q = I_M$, while $R \in \mathbb{R}^{M\times M}$ is an upper triangular matrix. Then the pseudo-inverse of $CS_p^{-1/2}$ can be simplified as follows

$$\begin{aligned}(CS_p^{-1/2})^+ =&(CS_p^{-1/2})^\top \left((CS_p^{-1/2})(CS_p^{-1/2})^\top\right)^{-1}\\ =&QR\left((QR)^\top(QR)\right)^{-1}\\ =&QR(R^\top Q^\top QR)^{-1}\\ =&QR(R^\top R)^{-1}\\ =&Q(R^\top)^{-1}.\end{aligned}$$

Observe that the pseudo-inverse by QR decomposition is simply done by inverting a lower triangular matrix of a smaller order M.

- **Singular value decomposition (SVD)**

  Consider the singular value decomposition of matrix

$$CS_p^{-1/2} = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{M\times M}$ and $V \in \mathbb{R}^{N\times N}$ are orthogonal matrices, $\Sigma \in \mathbb{R}^{M\times N}$ consists of the singular values of $CS_p^{-1/2}$ and is defined as follows

$$\Sigma = \begin{bmatrix} s_1 & & & 0 & \dots & 0\\ & s_2 & & 0 & \dots & 0\\ & & \ddots & \vdots & & \vdots\\ & & & s_M & 0 & \dots & 0 \end{bmatrix}.$$

The pseudo-inverse then is represented as

$$(CS_p^{-1/2})^+ = V\Sigma^+ U^\top,$$

where Σ^+ is the pseudo-inverse of Σ. It is formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix, specifically

$$\Sigma^+ = \begin{bmatrix} \frac{1}{s_1} & & & \\ & \frac{1}{s_2} & & \\ & & \ddots & \\ & & & \frac{1}{s_M} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

3.2.3 Remarks on Different Solvers

In sections 3.2.1 and 3.2.2, the main approaches used in SEEM for solving linear systems are briefly introduced. Both approaches have been proven effective for solving boundary value problems on complex domains. However, as two distinct types of methods, iterative solvers and direct solvers have their own unique features and are typically applied to different types of applications. Furthermore, considering that the main objective of this thesis is to broaden the applicability of SEEM to encompass more general problems like free and moving boundary problems, both solvers mentioned earlier, which were initially tailored for solving boundary value problems, come with limitations that impede their adaptation to broader applications. Detailed comparisons of these solvers will be provided below.

1. **Applicable Scenarios**

   - **The Iterative Solver**

     Iterative solvers are proven to be effective especially for for problems requiring denser grids. Unlike direct solvers, they circumvent the need for explicit matrix

inversion, relying solely on matrix-vector products which can be represented by implicit operators. Consequently, they excel at handling large systems.

The preconditioning technique outlined in the SEEM-PCG solver addresses the disparity in orders between domain and boundary operators, notably reducing the condition number of the system when applied to boundary value problems with analytically known boundaries. Thus, it exhibits exceptional efficiency and stability, particularly on very dense grids and with relatively low-order regularizers. It is worth noting that even though we refer to the regularizers as *relatively low-order*, it can still reach up to $p = 4$. This designation is made in comparison to direct solvers employing pseudo-inverse formulas, where even 12th-order regularizers can be effectively utilized.

It has been demonstrated in section 4.3 of [3] that on an Intel i7-7700HQ CPU, the H^3 solution can be obtained for a 2048^2 grid within in 1 minute, with the PCG method converging in just 45 iterations.

- **The Direct Solver**

  In contrast, direct solvers employing pseudo-inverse formulas facilitate the use of high-order regularizers, making them particularly efficient for sparse grids. With very high-order methods, minimal additional information, such as problem data or computation nodes, is required to achieve equivalent accuracy compared to lower-order methods, where accuracy improvement relies mainly on grid refinement. This significantly conserves computational resources. In SEEM, high-order operators do not entail additional implementation complexity, unlike in the finite element method, where employing high-order elements typically increases both degrees of freedom and computational costs.

  Experiments outlined in [1] demonstrate that direct solvers allow for the utilization of high-order regularizers up to 12th-order. When applied to boundary value problems with sufficiently smooth solutions, the algorithm achieves an L_2 error of

10^{-12} in just 40^2 grids.

2. **Limitations and Drawbacks**

- **The Iterative Solver**

  The effectiveness of the iterative PCG solver heavily relies on the performance of preconditioning. This significantly impacts its efficacy, particularly when dealing with more complex problems such as those involving free and moving boundaries. In such scenarios, where boundary points are unknown and require detection and approximation, achieving well-distributed boundary points precisely located on smooth underlying boundaries proves challenging. Perturbations on boundary points, such as uneven distribution along the arclength or errors and noise accumulated during curve evolution, further exacerbate the condition number of the boundary operator.

  The original design of the preconditioner may no longer effectively reduce the condition number in such cases. A stronger preconditioning strategy is essential to enable the usage of the PCG solver for problems requiring sufficient resolution on boundaries as well as a much denser overall grid.

  Another source of ill-conditioning arises from the density mismatch between the boundary and domain grids. To keep the condition number of the Schur complement matrix within a reasonable range, the density of boundary points needs to remain sparse. However, meeting the boundary resolution requirements in free and moving boundary problems necessitates denser domain grids. This presents a dilemma: allocating too many computational resources to the domain grids becomes prohibitively expensive, especially considering that solving the boundary value problem is only a step in the entire process of addressing free and moving boundary problems.

- **The Direct Solver**

Given the current SEEM implementation, which mandates a fixed ratio of boundary to domain grid density to strike a balance between accuracy and stability, there's a challenge. This fixed density of boundary points may lead to inadequacy when utilizing direct solvers, primarily designed for sparse grids, for nonlinear problems such as free and moving boundary problems. These scenarios impose specific requirements on boundary resolution for identifying unknown boundaries. Hence, for broader applicability, enhancing the utilization of boundary points within the SEEM framework with direct solvers becomes crucial while maintaining the computational efficiency inherent in sparse domain grids.

3.3 Regularized Preconditioned Iterative Solver

As discussed in Section 3.2.1, the iterative solver is particularly sensitive to the condition number of the problem, which is primarily influenced by the domain and boundary components. The original preconditioner P_A effectively reduces both the operator order and the condition number for the domain part. It's also worth noting that the condition number and preconditioning for the domain part are only affected by the mesh size and the level of regularization, independent of the boundary conditions. Therefore, the main focus lies in improving the preconditioner P_B for the boundary operator when expanding the application of the SEEM algorithm.

Experiments have shown that various factors, such as uneven distribution of boundary points, noise in boundary point locations, and discrepancies in densities between boundary and domain points, can substantially increase the condition number of the boundary operator. This, in turn, affects the overall performance of preconditioning in the original SEEM-PCG solver. Many of these factors are commonly encountered in applications involving free and moving boundary problems. Therefore, any modifications made to the boundary precondi-

tioner should be able to effectively manage the complexities associated with such boundary point inputs with the features mentioned above.

Recall from Section 3.2.1 that the original design of the preconditioner for boundary operators (using Dirichlet conditions as an example) involved reformulating the boundary block as a straightforward boundary interpolation problem. Hence, a logical approach for modifying the boundary preconditioner to accommodate more intricate boundary inputs can be inspired by the advanced boundary approximation algorithm discussed in Section 2.3.

In most of the applications of the original SEEM solver, the boundary is usually given by the analytical representation. This ensures that the boundary points used in the SEEM procedure are of very high quality, meaning they are precisely located on the boundary without any errors or noise corruption. Additionally, these points are evenly distributed along the boundary (for example, along the arc length in the case of two-dimensional curves). Consequently, the resulting boundary operators are usually well-conditioned or can be effectively preconditioned. Recall that the boundary preconditioner P_B introduced in section 3.2.1 is the inverse of a matrix $\mathcal{M}$ whose entry is defined by the fundamental solution evaluated at the distance between given boundary points, specifically

$$\mathcal{M}_{ij} = S_p^{-1}\delta(y_i - y_j), \quad i, j = 1, 2, \ldots, N_{\Gamma^m}. \tag{3.12}$$

Recall the kernel-based level set function introduced in section 2.2, built upon the data set $\mathbb{D} = \{(x^i, 1)\}$ that comprises boundary points $\{x^i\}$, employs a kernel decomposition formula given by

$$\phi(x) = \sum_{i=0}^{N-1} \lambda_i S^{-1}\delta_{x^i}(x).$$

In this context, the operator S can essentially represent any smooth operator. Specifically, we designate S as S_p within the SEEM framework. We further update the notation for

boundary points to $y_i \in \Gamma^m$, where $i = 1, 2, \ldots, N_{\Gamma^m}$. Substituting the data set $\mathbb{D}$ into the level set function $\phi(x)$ yields the matrix equation

$$M\Lambda = \mathbb{1},$$

with the coefficient matrix M defined element-wise as follows

$$M_{ij} = S_p^{-1}\delta_{y_i}(y_j) = S_p^{-1}\delta_{y_j}(y_i) = S_p^{-1}\delta(y_i - y_j), \quad i, j = 1, 2, \ldots, N_{\Gamma^m}. \tag{3.13}$$

It is obvious that the coefficient matrix M in level set function is identical to the inverse preconditioner matrix $\mathcal{M}$, i.e.,

$$P_B = M^{-1}.$$

Now suppose that the data set $\mathbb{D}$ consists of boundary points $y_i \in \Gamma^m$, $i = 1, 2, \ldots, N_{\Gamma^m}$, with low quality, then for better stability we use the following regularized level set function to approximate the boundary

$$\phi_\alpha(x) = \sum_{i=0}^{N-1} \lambda_i^\alpha S_p^{-1}\delta_{y_i}(x).$$

The entries in M is still defined the same as in (3.13), however after plugging in the data set $\mathbb{D}$, the corresponding matrix equation becomes

$$(\alpha I + M)\Lambda_\alpha = \mathbb{1}.$$

Therefore, we conclude that the modified boundary preconditioner $P_{B,\alpha}$ is defined as

$$P_{B,\alpha} = (\alpha I + M)^{-1}.$$

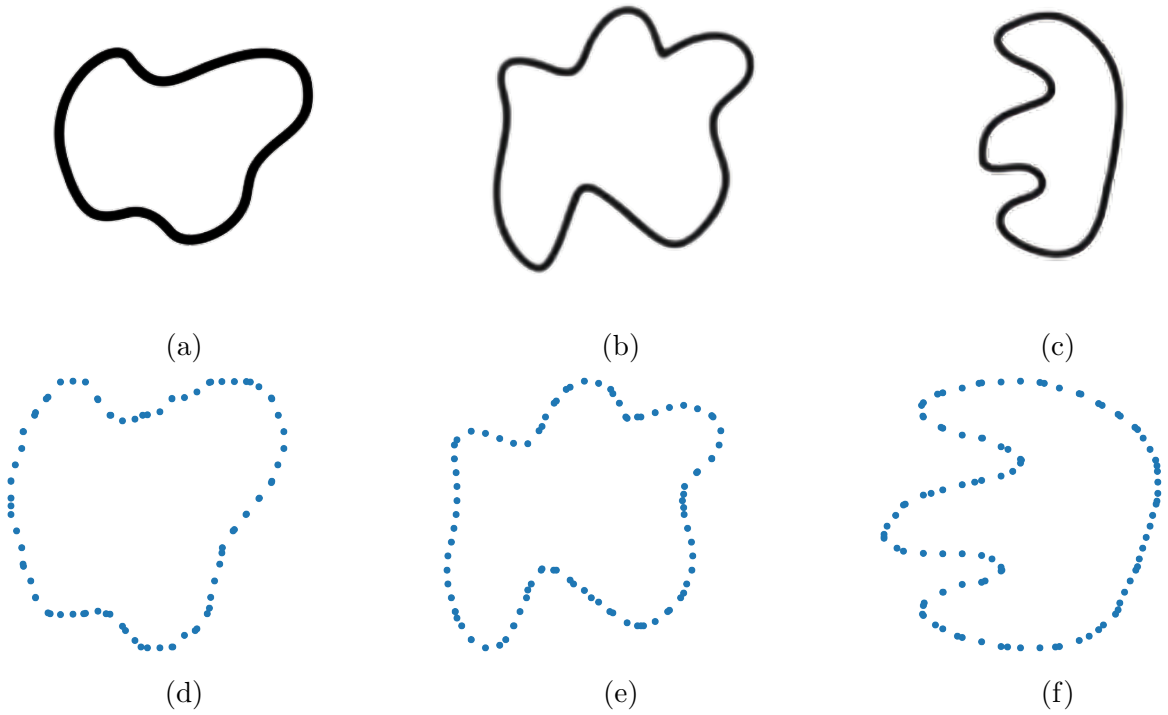


Figure 3.3: Smooth curves and their non-smooth numerical representations for regularized preconditioning test.

Notice that the original preconditioner P_B is identical to $P_{B,0}$.

To evaluate the efficacy of the regularized boundary preconditioner, we selected several smooth sketching curves as boundaries in boundary value problems. The original curves and their corresponding discretizations are illustrated in Figure 3.3. It's noteworthy that, although the original curves presented in the first row appear smooth, their analytical parametrizations are unknown. Therefore, we resorted to using a point sampling technique to obtain their numerical representations, as depicted in the second row. This step introduces elements of non-smoothness into the numerical computation. Indeed, for the discrete representations of all three test curves, the original preconditioners $P_B = P_{B,0}$ are singular, rendering the original SEEM-PCG solver ineffective.

In our investigation of the factors influencing the performance of preconditioning, we tested numerous potential factors. However, some showed no significant correlation with preconditioning performance, including the shape of the test curve, the number of boundary points, the number of SEEM domain grid points, and the regularization level of SEEM itself. These factors, deemed irrelevant, will not be elaborated upon further with detailed data.

Nonetheless, our analysis revealed two factors that significantly affect the performance of the preconditioning: the regularization parameter α in the regularized preconditioner $P_{B,\alpha}$ and the boundary density of SEEM operators. We define this *boundary density* ρ as follows,

$$\rho = \frac{4\pi N}{Lm}. \tag{3.14}$$

Here, N represents the number of boundary points used in SEEM, m is the number of one-dimensional grid points, and L is the perimeter of the underlying closed curve. With this definition, if the curve is a circle with a radius of 2 centered at the origin, then approximately ρ boundary points would be found within each domain grid box. Additionally, according to this definition, the boundary density in the original SEEM settings [3] corresponds to $\rho = 1$.

The effectiveness of the preconditioners $P_{B,\alpha}$ is measured by the *preconditioning rate* η, which is the percentage decrease in the log condition number after preconditioning. Specifically, let $\kappa_0 = \kappa(B)$ and $\kappa_1 = \kappa(P_{B,\alpha}^{1/2} B P_{B,\alpha}^{1/2})$ represent the condition numbers before and after preconditioning, respectively. Then, η is defined as:

$$\eta = \frac{\log(\kappa_0) - \log(\kappa_1)}{\log(\kappa_0)} \times 100\%. \tag{3.15}$$

For the ideal case, the condition number drops to 1 after the preconditioning, then the preconditioning rate is $\eta = 100\%$.

Boundary Density	Condition Number	$P_{10^{-15}}$	$P_{10^{-12}}$	$P_{10^{-9}}$	$P_{10^{-6}}$	$P_{10^{-3}}$
2.515625	5.75×10^{16}	3.69%	54.23%	37.62%	17.48%	-1.93%
1.265625	7.80×10^{16}	69.55%	28.72%	18.03%	17.96%	4.66%
0.828125	4.45×10^{17}	70.86%	32.77%	38.51%	12.24%	-2.62%
0.671875	8.95×10^{16}	71.44%	51.93%	35.98%	7.95%	2.04%
0.484375	2.45×10^{17}	70.42%	53.14%	35.86%	18.82%	-3.37%

Table 3.1: Condition numbers and preconditioning rates for test curve in figure 3.3a with different boundary densities.

Boundary Density	Condition Number	$P_{10^{-15}}$	$P_{10^{-12}}$	$P_{10^{-9}}$	$P_{10^{-6}}$	$P_{10^{-3}}$
2.765625	5.09×10^{16}	33.33%	53.03%	35.04%	16.75%	-3.61%
1.421875	3.73×10^{16}	68.96%	50.83%	34.51%	16.43%	-1.26%
0.921875	1.74×10^{17}	70.17%	52.74%	35.31%	17.71%	3.98%
0.640625	3.13×10^{16}	32.48%	52.43%	32.38%	14.34%	-0.06%
0.546875	2.33×10^{16}	68.87%	51.80%	33.29%	8.48%	-2.32%

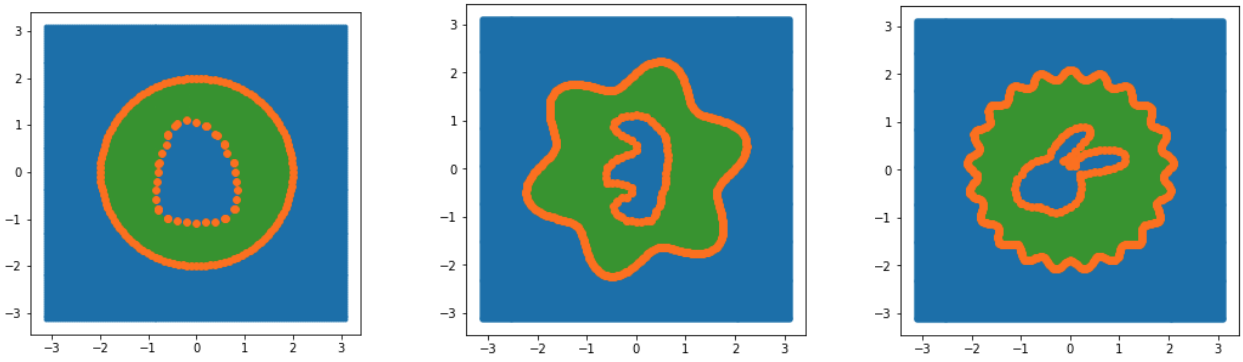
Table 3.2: Condition numbers and preconditioning rates for test curve in figure 3.3b with different boundary densities.

For different preconditioners $P_{B,\alpha}$, each with a unique parameter α, applied to domain-boundary discretizations characterized by varying SEEM boundary densities ρ, we document both the condition number prior to preconditioning and the preconditioning rate η in Tables 3.1–3.3,

The tables 3.1–3.3 demonstrate that with appropriate choices of preconditioning regularizer parameter α and the SEEM boundary density ρ, the condition number of the boundary operator can be significantly lowered. In our regularized PCG algorithm, we default to setting α at 10^{-15} and ρ at 0.5.

Boundary Density	Condition Number	$P_{10^{-15}}$	$P_{10^{-12}}$	$P_{10^{-9}}$	$P_{10^{-6}}$	$P_{10^{-3}}$
2.859375	7.90×10^{16}	38.69%	53.56%	33.99%	18.04%	1.31%
1.390625	3.03×10^{16}	68.79%	50.56%	32.33%	17.75%	-3.18%
0.953125	7.26×10^{16}	69.50%	51.67%	33.86%	17.87%	1.72%
0.703125	2.43×10^{16}	68.61%	50.27%	31.93%	15.42%	0.53%
0.515625	2.81×10^{16}	70.01%	50.41%	28.81%	13.87%	-1.33%

Table 3.3: Condition numbers and preconditioning rates for test curve in figure 3.3c with different boundary densities.



(a) (b) (c)

Figure 3.4: Domain with smooth boundaries for which the analytical representations are unknown.

m	N_{Γ^m}	N_{Σ^m}	L_2 Error	PCG Iteration	CPU Time
256	85	256	2.24×10^{-8}	5700	109.28
128	41	128	3.67×10^{-7}	998	3.01
64	21	64	8.86×10^{-6}	392	0.32
32	13	32	1.05×10^{-4}	226	0.13
16	5	16	4.44×10^{-3}	81	0.02

Table 3.4: Relative L_2 approximation errors, PCG iterations, and CPU times for SEEM-RPCG solving problem defined in Figure 3.4a.

Now, more experiments will be given to show that the SEEM with regularized PCG solvers, which we call it SEEM-RPCG method from now on, can solve BVP with more complex boundaries accurately, efficiently, and stably.

Consider the Dirichlet boundary value problem defined on the domain with analytical representation unknown. Specifically,

$$\begin{cases} -\Delta u = 0, & \text{in } \Omega, \\ u = x^2 - y^2, & \text{on } \partial\Omega = \Gamma \cup \Sigma. \end{cases}$$

The exact solution is $u = x^2 - y^2$. Different domains with inner boundary Γ and outer boundary Σ are shown in Figure 3.4, the domains are labeled with interior (in orange), exterior (in blue), and boundary points (in green), respectively.

m	N_{Γ^m}	N_{Σ^m}	L_2 Error	PCG Iteration	CPU Time
256	91	302	1.95×10^{-8}	9490	194.73
128	47	151	4.16×10^{-7}	2998	9.53
64	23	75	8.17×10^{-6}	1091	0.90
32	9	37	1.73×10^{-4}	325	0.18
16	5	18	2.26×10^{-3}	98	0.03

Table 3.5: Relative L_2 approximation errors, PCG iterations, and CPU times for SEEM-RPCG solving problem defined in Figure 3.4b.

m	N_{Γ^m}	N_{Σ^m}	L_2 Error	PCG Iteration	CPU Time
256	109	316	1.08×10^{-7}	18143	378.92
128	47	158	4.56×10^{-6}	4621	15.02
64	25	79	5.70×10^{-5}	1208	0.99
32	11	39	5.87×10^{-4}	322	0.16
16	5	19	2.36×10^{-3}	91	0.03

Table 3.6: Relative L_2 approximation errors, PCG iterations, and CPU times for SEEM-RPCG solving problem defined in Figure 3.4c.

Tables 3.4–3.6 show the experiment data based on the SEEM-RPCG solver with pre-selected parameters. The regularizer level for SEEM is $l = 4$ for all three test examples. Notations consistent with those in SEEM are used, where m denotes the one-dimensional grid number within SEEM. Furthermore, N_{Γ^m} and N_{Σ^m} represent the number of boundary points sampled from Γ and Σ, respectively. The computations were performed using the standard Python packages on an Apple M2 chip. The stopping criteria for the PCG iterations is 10^{-8}.

Chapter 4

Shape Optimization

According to the definition of free and moving boundary problems given by Crank [10] mentioned in section 1.1, these problems can be formulated as shape optimization problems involving constraints from elliptic and parabolic PDEs, respectively. Thus, this thesis primarily focuses on shape optimization problems constrained by PDEs. The general formulation of the shape optimization problems under consideration is outlined below:

$$\text{minimize } J(\Omega) := j(u(\Omega), \Omega), \quad \text{subject to } E(u(\Omega), \Omega) = 0.$$

Here $u(\Omega)$ represents the solution to the constrained PDE over the domain Ω. The primary variable in the shape optimization problem is the domain Ω, which is to be optimized.

To identify the optimal shape Ω^* that minimizes a given shape energy, it is essential to employ an energy minimization algorithm. These algorithms typically operate iteratively, beginning with an initial shape Ω^0 and progressively deforming it through a series of iterations using a velocity field $\vec{V}$ until a minimal energy state is reached. A key step in solving the minimization problem involves computing the gradient descent velocities $\vec{V}$, which necessitates understanding how slight alterations in the shapes Ω affect the energy functional

value, or in other words, understanding the shape derivatives. We will introduce basic notations and essential findings in this domain, drawing from [11], including a detailed explanation of shape perturbations within any given vector field $\vec{V}$ in section 4.1, followed by a comprehensive definition of shape derivatives and the formula commonly associated with PDE constraint shape optimization in section 4.2. With the shape derivative formula for any given vector field, it becomes possible to determine the descent direction by rendering the shape derivative negative. Consequently, we can straightforwardly achieve a gradient descent iteration, which will be further elaborated in section 4.3. We will conclude this chapter with a verification example illustrating the effectiveness of SEEM in approximating the shape gradient, as discussed in section 4.4. Remarkably, SEEM preserves the convergence order when estimating the shape gradient in the boundary integral formula, whereas the same task using finite element approximation experiences a loss in convergence order.

This thesis prioritizes the development of high order numerical methods within the shape optimization framework to address free boundary problems. Our focus is specifically on problems that, as established in the literature, exhibit sufficient smoothness, where the shape functional is differentiable, and a unique minimal shape can be determined in the context of shape optimization. Consequently, we do not delve into the minimal differentiability requirements essential for the existence of shape derivatives. It's important to note that determining these requirements is a non-trivial task, varying significantly based on the application and necessitating a tailored approach for each case. For readers interested in exploring the existence and uniqueness of minimal shapes in shape optimization further, we recommend consulting sources such as [9, 6] for comprehensive insights.

4.1 Shape Perturbations

This section elaborates on two methods of shape perturbation within the context of shape optimization: the velocity method and the perturbation of identity method. Both methods involve transforming a domain or boundary via specified vector fields, but they differ in their approaches and mathematical formulations. The content in this section primarily references Chapter 4 and Chapter 9 in [11]. For further details, we direct the reader to these chapters.

The velocity method is based on a velocity field that pushes points within a domain or on a boundary in a specified direction over time, resulting in a new shape. This method is defined through a transformation that follows an ordinary differential equation (ODE), which specifies how points move according to the velocity field. Specifically, we introduce this method with the following definition.

Definition 4.1 (Velocity method)**.** *Suppose $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$ is a smooth vector field. Let $x \in \mathbb{R}^d$, consider the following ODE*

$$\begin{cases} \partial_t T(t,x) = \vec{V}(T(t,x)), \quad t > 0 \\ T(0,x) = x \end{cases}$$

which defines a transformation $T(t,x) : [0,\infty) \times \mathbb{R}^d \to \mathbb{R}^d$. For any fixed t, define the set

$$\Omega_{t,\vec{V}} = \{T(t,x) | x \in \Omega \subset \mathbb{R}^d\},$$

$$\Gamma_{t,\vec{V}} = \{T(t,x) | x \in \Gamma \subset \mathbb{R}^d\},$$

to be the perturbation of Ω (or $\Gamma = \partial\Omega$) with respect to the velocity $\vec{V}$ and time step t.

Given the focus on developing high-order methods for problems that are sufficiently smooth, we assume that both shapes and velocity fields exhibit high levels of regularity. It's important

to highlight that to achieve a perturbed shape of a certain regularity, specific conditions for the regularity of $\vec{V}$ and an appropriate value of t are essential.

A more straightforward method for transforming shapes is known as the *perturbation of identity*. This approach typically offers a more direct and uncomplicated formulation compared to other methods.

Definition 4.2 (Perturbation of identity method)**.** *Given a (smooth) vector field* $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$*, consider a family of transformations* $\{T_t\}_{t\in[0,\tau]}$ *such that*

$$T_t = \mathcal{I} + t\vec{V},$$

where $\tau > 0$*, and* $\mathcal{I}$ *is the identity operator on* $\mathbb{R}^d$*. The family* $\{T_t\}$ *yields a family of perturbed shapes*

$$\Omega_{t,\vec{V}} = T_t(\Omega), \quad \Gamma_{t,\vec{V}} = T_t(\Gamma),$$

which are the perturbations of Ω *(or* $\Gamma = \partial\Omega$*) with respect to the velocity* $\vec{V}$ *and time step* t*.*

In view of Banach's fixed point theorem, if the domain Ω is open bounded with $C^{1,1}$-boundary Γ, and the vector field $\vec{V} \in C^{1,1}(\mathbb{R}^d;\mathbb{R}^d)$, then there exists a value $\tau > 0$ such that T_t is invertible for all $t \in [0,\tau]$.

According to [19], from a differential geometry perspective, the perturbation of identity method outlined in Definition 4.2 is considered less versatile than the velocity method described in Definition 4.1. Despite this, both methods ultimately yield the identical formula for calculating the shape gradient, which merely takes into account first order perturbations of the shape functional.

In fact, when we approximate the shape gradient of the shape functional at some specific

domain Ω (or boundary Γ), only the normal component of the vector field $\vec{V}$ on the boundary Γ matters for the shape perturbation, as the tangential velocities only shift the points along the boundary but do not change the shape. Since for any domain with $C^{1,1}$-boundary Γ, there is an open neighborhood $\mathcal{U}(\Gamma)$ of Γ, such that any point z inside $\mathcal{U}(\Gamma)$ can be represented as

$$z = x_\Gamma(z) + s_\Gamma(z)\nu_\Gamma(x_\Gamma(z)).$$

Here $x_\Gamma(z) \in \Gamma$ is the projection of point z onto the boundary Γ, $s_\Gamma(z)$ is the normal displacement of z from Γ, and ν_Γ is the unit normal vector on Γ. Then one can define the perturbed shapes starting from any (sufficiently smooth) normal velocity V that is only defined on the boundary Γ, and extend it into the entire space $\mathbb{R}^d$ by

$$\vec{V}(z) = \begin{cases} \psi_0(s_\Gamma)V(x_\Gamma)\nu_\Gamma(x_\Gamma), & z \in \mathcal{U}(\Gamma), \\ 0, & z \in \mathbb{R}^d/\mathcal{U}(\Gamma). \end{cases}$$

Here ψ_0 is the smooth cut-off function centered at 0 with the properties that $\psi_0(x) = 1$ near 0 and $\psi_0(x) = 0$ away from 0. With this notation, within a small tubular neighborhood of the boundary where $\psi_0(x) \equiv 1$, the vector field $\vec{V}$ is the constant normal extension of the normal velocity V. Hence, the perturbed shapes obtained by both Definition 4.1 and Definition 4.2 are equivalent. Notice that the shape derivatives computed for the shape optimization algorithms are used for evolving the shape locally. Only the information near to the current shape make senses during the entire shape optimization process. Given the nonlinear nature of the shape manifold, shape derivatives must be recalculated in the new tangential space after each iteration to accurately reflect changes and inform subsequent steps.

4.2 Shape Derivatives

The results presented in this section are intended to make the thesis self-contained. For comprehensive proofs and additional details, readers are directed to [32]. This section employs the concept of shape derivatives to analyze how a specific velocity field $\vec{V}$ influences energy changes. Understanding the impact of any given velocity filed $\vec{V}$ on energy allows for the selection of an optimal velocity from the space of admissible velocities that minimizes the energy for a specific shape Ω (or $\Gamma = \partial\Omega$).

Definition 4.3 (Shape derivative of energy functional)**.** *The* shape derivative *of an energy functional $J(\Omega)$ at Ω (or $J(\Gamma)$ at $\Gamma = \partial\Omega$) with respect to velocity field $\vec{V}$ is*

$$dJ(\Omega; \vec{V}) = \lim_{t \to 0} \frac{J(\Omega_{t,\vec{V}}) - J(\Omega)}{t},$$

or

$$dJ(\Gamma; \vec{V}) = \lim_{t \to 0} \frac{J(\Gamma_{t,\vec{V}}) - J(\Gamma)}{t},$$

where $\Omega_{t,\vec{V}}$ and $\Gamma_{t,\vec{V}}$ are defined in Definition 4.1 or 4.2.

It goes without saying that it is desirable that the limits in above definition exist for all possible perturbation directions $\vec{V}$. It is therefore natural to define a shape functional J to be *shape differentiable* at Ω (or Γ) if the mapping

$$dJ(\Omega; \cdot) : C^1(\mathbb{R}^d; \mathbb{R}^d) \to \mathbb{R}, \quad \vec{V} \mapsto dJ(\Omega; \vec{V})$$

is linear and bounded on $C^1(\mathbb{R}^d; \mathbb{R}^d)$. Similarly for the boundary functional $dJ(\Gamma)$, see Chapter 9, Definition 2.2 in [11].

This thesis primarily concentrates on shape optimization problems constrained by PDE, where the energy functional usually is a nonlocal function of the PDE solution $u = u(x, \Omega)$

(or $u(x,\Gamma)$), viewed as a function dependent not only on the spatial variable x, but also on the shape Ω (or Γ) where its associated PDE is defined. To effectively compute the shape derivatives of such energy functionals, it's crucial to consider the derivative of functions with respect to the shapes Ω and Γ. This involves revisiting the concepts of material derivative and shape derivative as discussed in [32].

Definition 4.4 (Material derivative of shape-dependent functions)**.** *For a shape-dependent function $\phi = \phi(x,\Omega)$ (or $\phi(x,\Gamma)$), the* material derivatives $\dot{\phi}(\Omega,\vec{V})$ *at Ω and $\dot{\phi}(\Gamma,\vec{V})$ at Γ in direction $\vec{V}$ are defined as follows*

$$\dot{\phi}(\Omega;\vec{V}) = \lim_{t\to 0}\frac{\phi(T(t,\cdot),\Omega_{t,\vec{V}}) - \phi(\cdot,\Omega)}{t}, \quad \dot{\phi}(\Gamma;\vec{V}) = \lim_{t\to 0}\frac{\phi(T(t,\cdot),\Gamma_{t,\vec{V}}) - \phi(\cdot,\Gamma)}{t},$$

where the notations $T(t,\cdot)$, $\Omega_{t,\vec{V}}$, and $\Gamma_{t,\vec{V}}$ are from Definition 4.1.

Definition 4.5 (Shape derivative of shape-dependent functions)**.** *For a domain-dependent function $\phi = \phi(x,\Omega)$, the shape derivative $\phi'(\Omega,\vec{V})$ at Ω in direction $\vec{V}$ is defined as follows*

$$\phi'(\Omega;\vec{V}) = \dot{\phi}(\Omega;\vec{V}) - \nabla\phi\cdot\vec{V}.$$

While for a boundary-dependent function $\phi = \phi(x,\Gamma)$, the shape derivative $\phi'(\Gamma,\vec{V})$ at Γ in direction $\vec{V}$ is defined as follows

$$\phi'(\Gamma;\vec{V}) = \dot{\phi}(\Gamma;\vec{V}) - \nabla_\Gamma\phi\cdot\vec{V}|_\Gamma.$$

Here the notation ∇_Γ means the tangential gradient, which is defined as follows.

Definition 4.6 (Tangential gradient)**.** *Assume that Γ is a smooth orientable compact $(d-1)$ dimensional surface in $\mathbb{R}^d$ without boundary. Given $\phi \in C^2(\Gamma)$ and its smooth extension $\tilde{\phi} \in C^2(\mathbb{R}^d)$. The tangential gradient $\nabla_\Gamma\phi$ is defined to be*

$$\nabla_\Gamma\phi = (\nabla\tilde{\phi} - (\partial_\nu\tilde{\phi})\nu)|_\Gamma,$$

where ν denotes the unit normal vector to Γ and $\partial_\nu \tilde{\phi}$ is the normal derivative.

The concept of shape derivative is pivotal for calculating changes in quantities dependent on shape, as well as variations in shape energy functionals, with respect to deformations of the shape by specified velocity fields. The following discussion introduces a range of results from shape differential calculus as outlined in [32] without proof.

Theorem 4.1 (shape derivative for domain integral energy)**.** *Let $\phi = \phi(\Omega)$ be given so that the material derivative $\dot{\phi}(\Omega; \vec{V})$ and the shape derivative $\phi'(\Omega; \vec{V})$ exist. Then, the shape energy $J(\Omega) = \int_\Omega \phi(x, \Omega)dx$ is shape differentiable and its shape derivative at Ω in the direction of $\vec{V}$ is defined to be*

$$dJ(\Omega; \vec{V}) = \int_\Omega \phi'(\Omega; \vec{V})dx + \int_\Gamma \phi V dS.$$

Theorem 4.2 (shape derivative for boundary integral energy)**.** *Let $\phi = \phi(\Gamma)$ be given so that the material derivative $\dot{\phi}(\Gamma; \vec{V})$ and the shape derivative $\phi'(\Gamma; \vec{V})$ exist. Then, the shape energy $J(\Gamma) = \int_\Gamma \phi(x, \Gamma)dS$ is shape differentiable and its shape derivative at Γ in the direction of $\vec{V}$ is defined to be*

$$dJ(\Gamma; \vec{V}) = \int_\Gamma \phi'(\Gamma; \vec{V})dS + \int_\Gamma \kappa\phi V dS,$$

Moreover, if $\phi(\cdot, \Gamma) = \psi(\cdot, \Omega)|_\Gamma$, then we obtain

$$dJ(\Gamma; \vec{V}) = \int_\Gamma \psi'(\Omega; \vec{V})|_\Gamma dS + \int_\Gamma (\partial_\nu \psi + \kappa\psi) V dS.$$

Notice that for different free boundary problems, the formulation of the shape energies and the constraint PDEs vary significantly. The shape derivatives of the integrated objective functions ϕ' and ψ' need to be computed for specific applications. We will reference specific results in later sections when we have a more concrete example. In this section, our goal is

to introduce basic notations and the most general results in the field of shape optimization. Therefore, we conclude this section with a general property of shape derivatives, as expressed in the following theorem.

Theorem 4.3 (Hadamard structure theorem)**.** *For shapes with smooth boundaries, the shape gradient of a surface or domain energy always has a representation of the form*

$$dJ(\Gamma;\vec{V}) = \langle g, V\rangle_\Gamma \quad \text{or} \quad dJ(\Omega;\vec{V}) = \langle g, V\rangle_\Gamma,$$

where $\langle\cdot,\cdot\rangle$ *denotes a suitable duality pairing on the boundary, and* $V = \vec{V}\cdot\nu$ *is the normal component of* $\vec{V}$ *on the boundary.*

This theorem demonstrates that the shape derivative is concentrated on the boundary, indicating that only normal displacements of the boundary affect the value of the shape energy functional. It's important to note that this outcome holds true regardless of the specific form of the shape functional; both domain and boundary integral-defined shape functionals have their shape gradients defined on the boundary. For those interested in further details, we recommend consulting Chapter 9, Theorem 3.6 in [11] and Section 2.11, Theorem 2.27 in [32].

4.3 Gradient Descent on Shape Manifold

As indicates in Theorem 4.3, the shape derivative $dJ(\cdot;\vec{V})$ is solely dependent on V, the normal component of the velocity. This means that any vector field $\vec{V}$ that satisfies $\vec{V}\cdot\nu = V$ can be chosen as an extension of V beyond the boundary Γ without affecting the shape gradient. Therefore, without loss of generality, an extension of the normal velocity V that is locally constant in normal direction within a tubular neighborhood of the boundary can be readily used.

The primary motivation behind deriving the shape derivatives $dJ(\Gamma;\vec{V}), dJ(\Omega;\vec{V})$ of a given shape energy J is to derive algorithms aimed at minimizing this energy and calculating the optimal shape solution. In this section, we will provide a brief overview of how to develop gradient descent flows for shapes, utilizing shape derivatives for this objective.

In Theorem 4.3, it is noted that the shape derivative takes the form

$$dJ(\Gamma;\vec{V}) = \int_{\Gamma} g(\Gamma) V dS,$$

where $g(\Gamma)$ (or alternatively $g(\Omega)$) represents the shape gradient, which is dependent on the shape energy. By formally setting $V = -g(\Gamma)$, we can derive a gradient descent velocity, leading to

$$dJ(\Gamma;\vec{V}) = -\int_{\Gamma} g^2 dS \leq 0.$$

demonstrating a decrease in the shape energy, thus adhering to the principles of gradient descent.

The velocity $V = -g(\Gamma)$ is the most commonly used gradient descent velocity for solving shape optimization problems. Once a method for computing the gradient descent velocity V is established, minimization can proceed by starting with an initial surface Γ^0 and iteratively updating it, calculating the velocity V^{k+1} for the new shape Γ^{k+1} at each step

$$x^{k+1} = x^k + \tau_k \vec{V}^k, \quad \forall x^k \in \Gamma^k, \tag{4.1}$$

where $\tau_k > 0$ represents a step size parameter. This parameter may either be fixed or determined by a line search algorithm. The vector velocity $\vec{V}$ can be computed from the normal velocity V through a constant extension in the normal direction alone, as previously mentioned. This is because the tangential component of the velocity $\vec{V}$ does not change the

shape of the surface.

Iterative methods other than direct shape gradient descent can also be applied within the shape optimization framework, including accelerated gradient descent, the conjugate gradient algorithm, quasi-Newton, and Newton methods. However, when implementing any advanced iterative methods on the shape manifold, it's crucial to ensure that the updated shape remains on the manifold of shapes itself. Given the nonlinear nature of the shape manifold, careful implementation is required. Additionally, while the well-designed descent directions in more advanced iterative methods can significantly accelerate convergence, they may also substantially increase the computational cost per iteration. For instance, second-order iterative methods like Newton's method often require higher-degree shape derivatives, potentially introducing more approximation error and decreasing the numerical solutions' overall accuracy. While (semi-)implicit schemes, which offer enhanced stability for handling higher-degree derivatives (such as curvature), could be a viable alternative to the explicit scheme detailed in (4.1), exploring these schemes falls outside the scope of this thesis. Investigating these alternatives could be an avenue for future research. In this thesis, the main objective is to explore a viable strategy for broadening the application of the SEEM algorithm, with the aim of developing a high-order solver (in terms of grid refinement) for free boundary problems. Therefore, the focus is primarily on smooth problems where the shape gradient involves derivatives up to the first order, including normal vectors and normal derivatives. For these problems, employing a straightforward shape gradient formulation alongside an explicit scheme is sufficient to achieve competitive results.

4.4 Order of Convergence of Numerical Shape Gradients

In the previous sections, from 4.1 to 4.3, we have fully introduced the framework for the shape optimization problem using a continuous formulation. Regarding numerical implementation, various methods such as the finite element method are used to approximate solutions to the PDEs, along with the required differentiation and integration across the domain and its boundaries. However, when implementing shape gradients numerically, approximation errors arise in comparison to continuous analysis. This sparks an interest in the convergence rate of these numerical shape gradient approximations. The research presented in [19] shows that two analytically identical ways of representing shape gradients for PDE-constrained shape functionals, integrating their traces along the domain boundary versus evaluating integrals within the volume, become distinct for a numerical approximation with finite element solutions of BVPs. Through detailed convergence analysis, it has been determined that the volume-based expression for the shape gradient typically yields higher accuracy within a finite element framework. We will start by summarizing their key findings, then move on to use the SEEM to approximate shape gradients for comparison purposes. Finally, we aim to demonstrate that the convergence order of the boundary integral-defined shape gradient does not decrease when the SEEM is utilized as a BVP solver.

The Main Results of [19] can be summarized as follows. Consider the PDE constrained shape functional of the form

$$J(\Omega) = \int_{\Omega} j(u)dx,$$

where $j : \mathbb{R} \to \mathbb{R}$ possesses a locally Lipschitz continuous derivative j', and u is the solution

of the *state problem*, a scalar elliptic equation with Dirichlet boundary conditions

$$\begin{cases} -\Delta u + u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{4.2}$$

The formulas for shape gradients also contain p, the solution of the *adjoint problem*

$$\begin{cases} -\Delta p + p = j'(u) & \text{in } \Omega, \\ p = 0 & \text{on } \partial\Omega. \end{cases} \tag{4.3}$$

The shape gradient formulas expressed through both volume and boundary integrals are listed below, for a detailed derivation and understanding of these formulas, one should refer to [19].

- **Volume Integral for Dirichlet BC**

$$\begin{aligned} dJ(\Omega, u, p, \vec{V})^{\text{Vol}} = \int_\Omega \Big(& \nabla u \cdot (D\vec{V} + D\vec{V}^\top)\nabla p - f\vec{V} \cdot \nabla p \\ & + \text{div}\vec{V}(j(u) - \nabla u \cdot \nabla p - up) \\ & + (j'(u) - p)(\nabla g \cdot \vec{V}) - \nabla p \cdot \nabla(\nabla g \cdot \vec{V}) \Big) dx. \end{aligned}$$

- **Boundary Integral for Dirichlet BC**

$$dJ(\Omega, u, p, \vec{V})^{\text{Bdry}} = \int_{\partial\Omega} \vec{V} \cdot \nu \Big(j(u) + p_\nu (u - g)_\nu \Big) dS.$$

For exact solutions u^* and p^*, the shape gradients represented in both volume integral and boundary integral are equivalent

$$dJ(\Omega, \vec{V})^{\text{Exact}} = dJ(\Omega, u^*, p^*, \vec{V})^{\text{Vol}} = dJ(\Omega, u^*, p^*, \vec{V})^{\text{Bdry}}.$$

However, for the finite element solutions u_h and p_h, this equality may break down. The main theorems in [19] show that, if the Dirichlet BVPs for the Laplacian (4.2) and (4.3) are H^2-regular, The source function f and the boundary data g in (4.2) are restrictions of functions in $H^1(\mathbb{R}^d)$ and $H^3(\mathbb{R}^d)$ to Ω and $\partial\Omega$, respectively, the vector field $\vec{V}$ is in $W^{2,\infty}(\mathbb{R}^d)$, then the Ritz–Galerkin approximations u_h and p_h by piecewise linear Lagrangian finite elements satisfy

$$|dJ(\Omega, \vec{V})^{\text{Exact}} - dJ(\Omega, u_h, p_h, \vec{V})^{\text{Vol}}| \leq C(\Omega, u^*, p^*, f, g, \vec{V})h^2 = \mathcal{O}(h^2).$$

Moreover, if for some $p > d$, where d is the space dimension, the following assumption holds

$$\|u^*\|_{W^{2,p}(\Omega)} \leq C \|f\|_{L^p(\Omega)},$$

then

$$|dJ(\Omega, \vec{V})^{\text{Exact}} - dJ(\Omega, u_h, p_h, \vec{V})^{\text{Bdry}}| \leq Ch = \mathcal{O}(h).$$

Here h stands for the meshwidth, and $C > 0$ does not depend on h.

We compare the first test example in [19] using SEEM as the BVP solver instead of the finite element method. Specifically, consider the exact solution

$$u(x, y) = \cos(x)\cos(y),$$

the associated right hand side functions f and g can be calculated based on the specific operators and boundary conditions. The computational domain Ω is a disc with radius $\sqrt{\pi}$. The shape functional is quadratic with $j(u) = u^2$. This example is actually sufficiently smooth, however the numerical test from the original paper based on the finite element method verifies their predicted quadratic and linear convergence with respect to the mesh-width h for volume and boundary integral formulas respectively. Notice that the optimal order of convergence using linear element on this BVP is $\mathcal{O}(h^2)$ in L_2 norm, the shape gradient

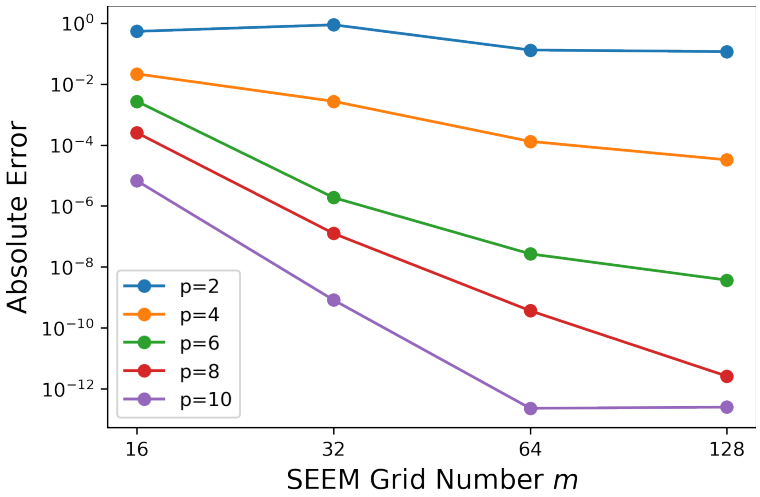


Figure 4.1: Approximation error of the boundary-formula shape gradient using p-th order SEEM solvers.

Reg Level	2	4	6	8	10
Order for dJ	2.751	4.373	6.135	8.423	11.805

Table 4.1: Order of convergence of the boundary-formula shape gradient approximation using p-th order SEEM solvers.

in volume integral formula preserves the order of convergence, whereas the boundary integral formula lose accuracy by 1 order.

As proven in Theorem 7.3 of [2], the order of convergence of the SEEM method with H^p-regularizer is $p-2$. However, for the sufficiently smooth solution $u \in H^{p+2}$, SEEM performs the p-th order convergence in numerical approximation. To assess whether the shape gradient defined by the boundary integral is less accurate compared to its volume integral counterpart under the SEEM approximation, it is enough to verify if the boundary integral formula ensures p-th order convergence when employing a p-th order SEEM solver, assuming the underlying problem possesses sufficient regularity.

Figure 4.1 and Table 4.1 demonstrate that the shape gradient, when represented by the boundary integral and solved with the SEEM solver, maintains the expected order of convergence. Therefore, we conclude that using the boundary integral formula in the development of straightforward and easily implementable shape optimization algorithms remains advan-

tageous. This is because it offers two significant benefits over the volume integral formula. Firstly, the boundary integral formula aligns directly with the Hadamard structure theorem (Theorem 4.3), facilitating the straightforward derivation of the explicit formula for the descent normal velocity in gradient descent iterations. Secondly, the explicit representation of normal velocity simplifies the shape evolution process through the perturbation of identity method (Definition 4.2), making it a direct and efficient approach.

Chapter 5

Main Algorithm

In previous chapter, an overview of different types of shape energy functional as well as their corresponding shape derivatives are introduced. However, for the simplicity of the algorithm introduction, we will standardize the notation for the algorithm in this chapter. Consider the free boundary problem formulated in the following general shape optimization framework

$$\begin{aligned} &\text{minimize } J(\Gamma) := j(u(\Gamma), \Gamma), \\ &\text{subject to } \mathrm{BVP}_{\text{state}}(u(\Gamma), \Gamma) = 0, \end{aligned} \tag{5.1}$$

with the analytical shape gradient represented in the Hadamard structure form as follows

$$dJ(\Gamma, V) = \int_{\Gamma} g(\Gamma, u(\Gamma), p(\Gamma)) V dS. \tag{5.2}$$

Here V is the normal velocity defined on the free boundary Γ, $p(\Gamma)$ is the solution to the adjoint boundary value problem $\mathrm{BVP}_{\text{adjoint}}(p(\Gamma), \Gamma) = 0$. The shape gradient descent algorithm, which utilizes numerical PDE solutions through SEEM, is proposed in Section 5.1. An introductory problem solved by the proposed algorithm will be presented in Section 5.2. It's important to note that while this test example is purely academic and may seem less

practical, it effectively showcases the advantages of our proposed method in overcoming the meshing and remeshing issues that plague finite element-based methods. Additionally, this test example also highlights the superior convergence rate of our proposed method when compared to finite element-based methods.

5.1 Description of the Algorithm

The gradient descent iteration begins with a set of N equally distributed discrete points along the arc length of a smooth curve. This curve is defined by an analytical parametric representation and serves as the initial approximation of the free boundary solution Γ, which we denote as Γ^0. In each iteration of the shape gradient descent, calculations are performed using the information from the current discrete curve Γ^k. At the end of each iteration, the discrete curve is updated to Γ^{k+1}, following the gradient descent flow. The detailed steps of the algorithm within a single iteration are outlined in Algorithm 1.

Several key points should be noted about this algorithm:

1. The initial guess for the discrete free boundary is characterized by N boundary points, a number that is maintained throughout the explicit shape gradient vector evaluation and the explicit curve evolution. Although these boundary points may be resampled to address any issues related to their quality, the total number of boundary points in the updated configuration should remain as N.

2. The boundary-domain ratio r_1, representing the ratio between the number of boundary points N and the SEEM grid points m, is typically set to be greater than 1. This approach leverages the high-order capabilities of SEEM solvers to minimize the overall computational cost, as SEEM can achieve sufficient accuracy with a coarser grid compared to the precision required for boundary approximation. In our experiments, we

Algorithm 1 SEEM-Based Gradient Descent Algorithm for Free Boundary Problems

Input: initial guess of the discrete free boundary $\Gamma^0 = \{x_0^0, \ldots, x_{N-1}^0\}$, max iteration number $k_{\max}$, boundary-domain ratio $r_1 = \frac{N}{m}$ between the number of boundary points and the SEEM grid points, SEEM boundary density level $\rho = \frac{4\pi N_m^{\text{SEEM}}}{Lm}$, regularization level l of SEEM, line search step length upper bound τ_0, tolerance ε of stopping criterion.

Output: numerical approximation of the discrete free boundary Γ^n.

1: **Initialization:** set $k = 0$, get the number of boundary points throughout the curve evolution $N = |\Gamma^0|$
2: **while** $k < k_{\max}$ **do**
3: Find level set representation $\phi(x)$ of the discrete curve Γ^k using regularized kernel-based boundary approximation.
4: Approximate the perimeter L of the current boundary curve.
5: Set grid number of SEEM by $m = \lceil \frac{N}{r_1} \rceil$. Set SEEM boundary number by $N_m^{\text{SEEM}} = \lceil \frac{\rho L m}{4\pi} \rceil$.
6: Obtain numerical solutions u_m and p_m of state and adjoint problems by SEEM.
7: Estimate shape gradient vector $G^k = g(\Gamma^k) = [g_0^k, \ldots, g_{N-1}^k]^\top$ in (5.2) with interpolation and boundary approximation techniques based on u_m, p_m, and $\phi(x)$.
8: Compute the normalized step length in gradient descent formula $\tau^k = \min\left\{\frac{\tau_0}{\|G^k\|_2}, 1\right\}$.
9: Get $\Gamma^{k+1} = \{x_j^{k+1}\}_{j=0}^{N-1}$ by the explicit gradient descent scheme $x_j^{k+1} = x_j^k - \tau^k g_j^k \nu_j^k$.
10: **if** $\|\Gamma^{k+1} - \Gamma^k\| = \sqrt{\sum_{j=0}^{N-1} |x_j^{k+1} - x_j^k|^2} < \varepsilon$ **then**
11: **return** Γ^{k+1}.
12: **end if**
13: Backtracking: Approximate the shape energy at Γ^{k+1} and Γ^k.
14: **if** $J(\Gamma^{k+1}) > J(\Gamma^k) + \beta^k$ **then**
15: Set $\tau^k = \tau^k/2$, repeat from step 8 or return Γ^k if $\tau^k < \varepsilon$.
16: **end if**
17: $k = k + 1$.
18: **end while**
19: **return** $\Gamma^{k_{\max}}$.

generally set r_1 to range from 2 to 4.

3. The SEEM boundary density, denoted as $\rho = \frac{4\pi N_m^{\text{SEEM}}}{Lm}$ and discussed in Section 3.3, quantifies the number of SEEM boundary points N_m^{SEEM} relative to the perimeter of the boundary curve L and the SEEM grid number m. It's important to note that SEEM boundary points differ from those used in curve evolution, with N_m^{SEEM} usually being much smaller than N. Analysis in Section 3.3 suggests that SEEM performs better with a lower boundary density ρ, to some extent, with $\rho = 0.5$ being a suitable choice for both regularized iterative and direct SEEM approaches. The SEEM boundary points N_m^{SEEM} in our algorithm are obtained through a point sampling algorithm applied to the level set function $\phi(x)$, as introduced in Section 2.4. N_m^{SEEM} can also be viewed as the dimension of the SEEM boundary operator, limiting the size of the Schur complement matrix to be inverted within SEEM. However, SEEM is capable of utilizing the full information from the N boundary points, including recognition of interior and exterior regions and interpolation on the boundary.

4. The approximation of the boundary in step 3 is a critical component of the iterative computations. The level set function $\phi(x)$, which encapsulates all the information from the N discrete boundary points, is extensively utilized in subsequent steps. These steps include but are not limited to resampling new boundary points, evaluating normal vectors, and recognizing domain regions. The regularization parameter for the regularized kernel-based method is set to 10^{-15} by default to ensure both accuracy and stability. It's important to note that the precise information always originates from the original N boundary points, regardless of how finely we resample the curve.

5. In step 4, the approximation of the perimeter L can be initially estimated by calculating the perimeter of the polygon formed by the discrete curve Γ^k. For more accurate results, the perimeter can also be computed based on the level set function $\phi(x)$.

6. In step 7, use interpolation, u_m, p_m, and level set representation $\phi(x)$ of the bound-

ary to find the approximation of the shape gradient formula $g(\Gamma, u(\Gamma), p(\Gamma))$ on the initial boundary points Γ^k, which is denoted as $G^k = [g_0^k, \ldots, g_{N-1}^k]$, where g_j^k is the approximation of $g(x_j^k)$, $j = 0, \ldots, N-1$.

7. The backtracking technique is employed to manage the reduction of the shape energy $J(\Gamma^k)$. However, during the evolution of the curve, particularly at the start, significant shape deformations may occasionally result in an acceptable increase in the shape energy functional. A widely used approach to address this situation is the application of the Armijo condition. For instance, in step 14, β^k could be set as $\beta^k = \beta\tau^k\|G^k\|_2$, where β is a constant within the range (0,1). This parameter is carefully chosen to strike a balance between minimizing the objective function and advancing towards the minimum efficiently. It helps to avoid excessively large steps that could bypass the minimum and excessively small steps that could decelerate the convergence process.

8. The algorithm terminates when any of the three stopping criteria is met. The first criterion, applied in step 10, evaluates the magnitude of the update: if the iteration results in a discrete curve that does not significantly differ from the previous step, as indicated by an L_2 norm of the update vector below a specified tolerance, then the algorithm stops. The second criterion, introduced in step 15, is related to backtracking: if the gradient descent step length τ^k falls below the tolerance and the shape energy does not effectively decrease, the algorithm halts, acknowledging that the current step's Γ^k, which leads to a smaller energy than Γ^{k+1}, is the output. The final stopping criterion is reached if neither of the first two criteria is met by the time the iteration count reaches the maximum number $k_{\max}$. However, in our experiments, this scenario has not occurred since the maximum number of iterations is usually set sufficiently high to allow the algorithm to progress fully.

5.2 An Introductory Test Problem

The shape gradient descent scheme, as outlined in Chapter 4 based on shape sensitivity analysis, is equally applicable to both continuous and discrete representations of the shape optimization problem. However, Section 4.4 mentions that this approach may not hold up when dealing with certain low-order numerical approximations, such as finite element methods that use piecewise linear elements. It's important to note that the preceding discussion primarily focuses on the approximation of the shape gradient in a single iteration. When considering the evolution of shape throughout the gradient descent process, the complexity increases. In the case of finite element methods, the domain Ω is typically represented by a computational mesh. As the shape evolves through the gradient descent iterations, and the boundary nodes are adjusted following the gradient descent flow, the interior nodes are subsequently shifted in response to the movements of the boundary nodes, following a certain nonlinear mapping. Numerous studies have identified a significant limitation of this direct approach: it often results in the degeneration of the computational mesh. Two primary sources of this degeneration are highlighted in [14]: deteriorating cell aspect ratios and mesh nodes moving into adjacent cells.

As introduced in Chapter 3, the SEEM method operates on a consistent domain grid that remains unaffected by the movement of boundary points during curve evolution. This approach eliminates the issues of meshing or remeshing within the interior domain discretization. Furthermore, by integrating SEEM with the stable and robust boundary approximation algorithm presented in Chapter 2, it effectively bypasses potential degeneration at the boundary points, showcasing its potential as an efficient solution to the primary challenges in shape optimization algorithms.

To evaluate the effectiveness of our methods in addressing mesh degeneration issues encountered in finite-element-based approaches, we examine a test example from [14]. This example

demonstrates how the direct application of the shape gradient, derived from the Hadamard structure theorem in the explicit shape gradient flow, leads to significant mesh degeneration.

Suppose $D \subset \mathbb{R}^d$ is some bounded and open hold-all domain. Find $\Omega \subset D$ open, such that

$$\int_\Omega u dx$$

is minimized, where u solves

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{5.3}$$

The shape derivative of above shape optimization problem in the direction of the perturbation field $\vec{V}$ is given by

$$dJ(\Omega, \vec{V}) = \int_{\partial\Omega} -\frac{\partial u}{\partial \nu}\frac{\partial p}{\partial \nu} V ds,$$

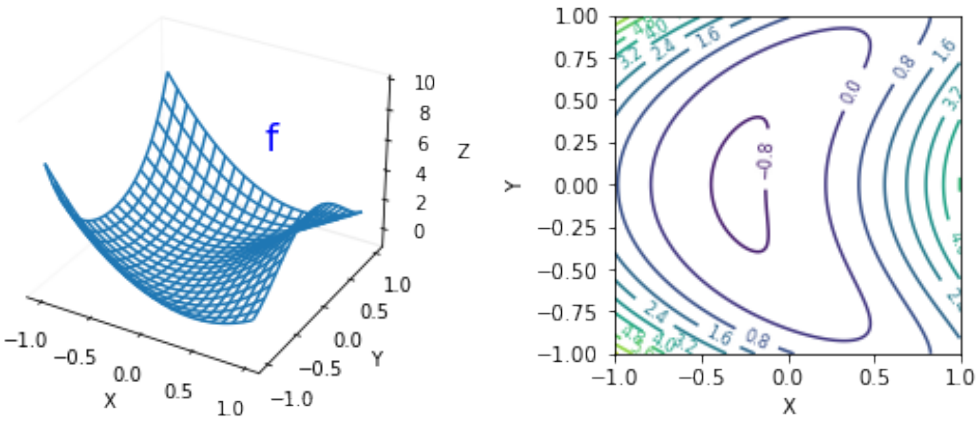
where ν denotes the outer unit normal vector along the boundary $\partial\Omega$ of Ω, $V = \vec{V} \cdot \nu$ is the normal component of the vector field $\vec{V}$, p represents the unique solution of the adjoint problem

$$\begin{cases} -\Delta p = -1 & \text{in } \Omega, \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

The derivation of the shape gradient above can be found in [23]. By the Hadamard structure theorem, a straightforward choice of the normal velocity in the gradient descent flow of the boundary $\partial\Omega$ is

$$V = \frac{\partial u}{\partial \nu}\frac{\partial p}{\partial \nu}.$$

Although the chosen example is primarily academic, it was selected to demonstrate that our method can achieve comparable convergence using the simplest shape gradient formula,



(a) Function graph. (b) Contour lines.

Figure 5.1: Visualization of the right-hand-side data function in the state problem (5.3).

bypassing any need for meshing and remeshing processes. This is in contrast to finite element methods, which require a carefully crafted shape gradient formula to sidestep issues related to mesh quality. More practical examples will be provided in the subsequent chapters to showcase the versatility and effectiveness of our algorithms across a wider range of PDE-constrained shape optimization problems.

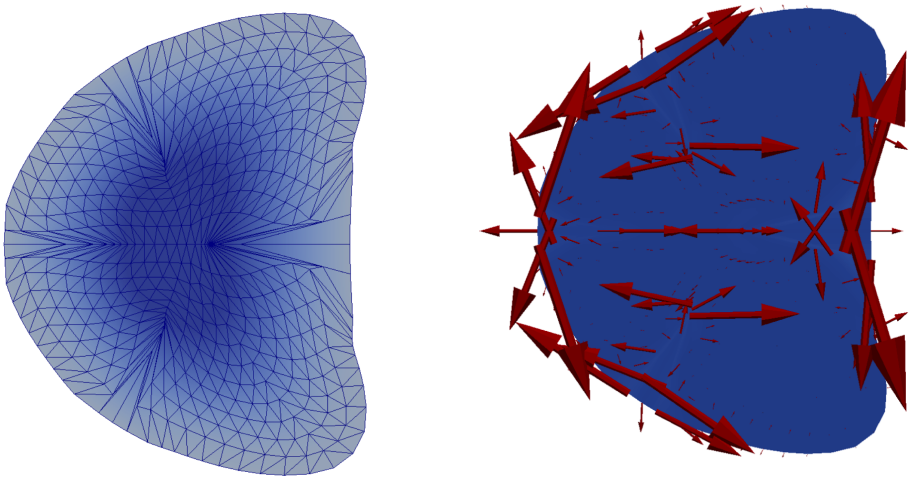
Consider the data function f in the state problem (5.3) as follows

$$f(x, y) = 2.5(x + 0.4 - y^2)^2 + x^2 + y^2 - 1.$$

Figure 5.1 provides a visualization of this function. The exact shape solution Ω of the shape optimization problem is unknown.

The classical gradient method, when applied with a piecewise linear finite element approximation, faces mesh degeneration issues during its gradient descent iterations. The degenerated domain mesh, observable after extended iteration periods, alongside the shape gradient field, are illustrated in Figure 5.2, originally sourced from [14].

The SEEM method maintains the same domain discretization throughout the curve evolu-



(a) Domain Mesh (b) Shape Gradient

Figure 5.2: The mesh condition and the shape gradient field in the last gradient descent iteration in classical finite element based shape gradient method.

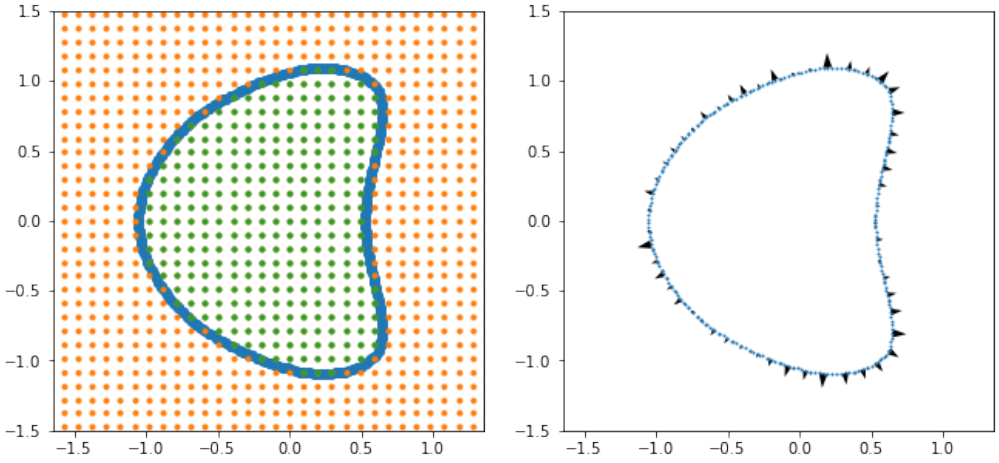
tion, as illustrated in Figure 5.3. This figure specifically showcases the conditions during the final iteration before the algorithm converges. In our approach, the shape gradient field performs as anticipated: it is smooth and exhibits a small magnitude.

Given the unknown exact solution to the problem, we assess the convergence of our proposed algorithms using the L_∞ norm of the shape gradient vectors. Figure 5.4 presents the plots of these gradient norms relative to the one-dimensional grid numbers m of SEEM.

Define the order of convergence p with respect to the mesh refinement, which in our algorithm is measured by the increase of the domain grid number m. Specifically,

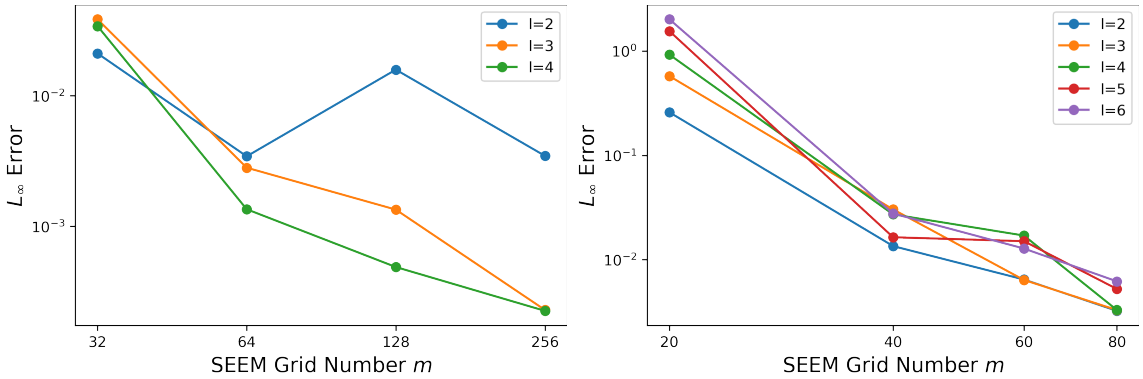
$$p = \frac{\log(e_{n+1}/e_n)}{\log(h_{n+1}/h_n)} = \frac{\log(e_{n+1}/e_n)}{\log(m_n/m_{n+1})},$$

where e_n is L_∞ norm of shape gradients, m_n is the number of SEEM grid points in one dimension used for solving BVP in curve evolution. Under this notation, the order of convergence of our algorithms are computed and listed in Table 5.1.



(a) Domain Grid (b) Shape Gradient

Figure 5.3: The domain grid and the shape gradient field in the last gradient descent iteration in SEEM based shape gradient method.



(a) SEEM-RPCG solver (b) SEEM-Direct solver

Figure 5.4: Plots of L_∞ norm in shape gradient vectors for different regularization levels of l using different types of solvers.

Regularization Level	Order of SEEM-RPCG	Order of SEEM-Direct
2	0.8681	3.1625
3	2.4641	3.7263
4	2.4142	4.0672
5	-	4.1064
6	-	4.1737

Table 5.1: Order of convergence in L_∞ norm of shape gradient vectors for different regularization levels of l using different types of solvers.

Chapter 6

Obstacle Problems

6.1 Problem Introduction

The obstacle problem is a prototypical free boundary problem, which describes the equilibrium position u of an elastic membrane loaded by the external force f, with boundary Σ held fixed at height 0, under a given obstacle $\varphi > 0$. Denote $\Omega =: \{x \mid u(x) < \varphi(x)\}$ as the non-contact set, the free boundary Γ is the interface between Ω and the contact set $I =: \{x \mid u(x) = \varphi(x)\}$. An illustration of the above problem is shown in Figure 6.1.

There are mainly two ways of formulating the obstacle problem mathematically [6]. Denote the set $D \subset \mathbb{R}^d$ as the hold-all domain with the smooth boundary $\Sigma = \partial D$. With the notations above, the set $D = I \cup \Omega$ is the union of both contact and non-contact sets. Assume functions $f \in H^{-1}(D)$, and $\varphi \in H^1(D)$, where $H^s(D)$ denotes the Sobolev space of order s. Denote the space K_0 as

$$K_0 = \{v \in H_0^1(D) | v \leq \varphi \text{ in the sense of } H^1(D)\}.$$

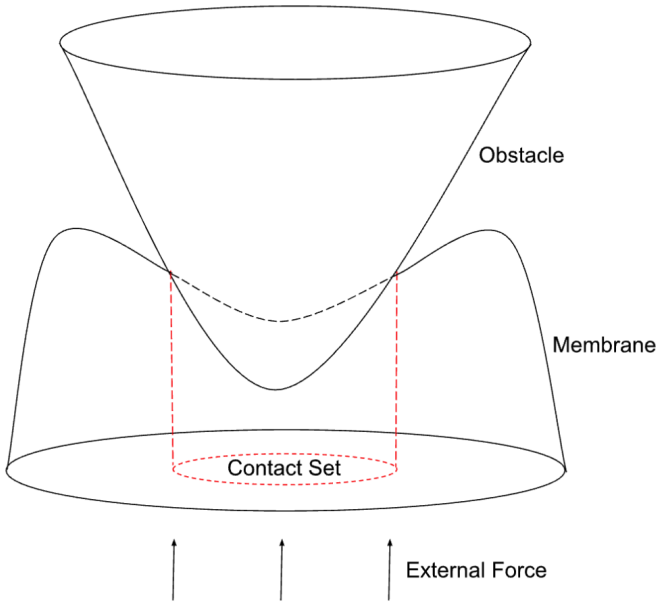


Figure 6.1: An illustration of a typical obstacle problem which models the displacement of an elastic membrane under a convex smooth obstacle due to an external force.

Then the obstacle problem can be formulated as the following *variational inequality*, where $u \in K_0$ to be found such that for $\forall v \in K_0$,

$$\int_D \nabla u \cdot \nabla (v-u) dx \geq \int_D f(v-u) dx. \tag{6.1}$$

The other formulation of the obstacle problem can be obtained if the free boundary Γ is sufficiently smooth. Then the above variational inequality (6.1) can be reformulated as an overdetermined boundary value problem

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = \varphi & \text{on } \Gamma \text{ (also in } I), \\ \partial_\nu u = \partial_\nu \varphi & \text{on } \Gamma, \\ u = 0 & \text{on } \Sigma, \end{cases} \tag{6.2}$$

where ∂_ν denotes the normal derivative on Γ.

Under the shape optimization framework, the above overdetermined BVP (6.2) can be reformulated as a shape energy functional minimization problem with PDE constraint. There are multiple choices of constructing the shape functionals. It's been shown in [31] that a straightforward choice of the shape energy functional

$$J(\Gamma) = \int_{\Gamma} (u - \varphi)^2 dS$$

subject to the mixed boundary value problem

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Sigma, \\ \partial_\nu u = \partial_\nu \varphi & \text{on } \Gamma. \end{cases}$$

has multiple stationary points. Therefore it is not an ideal choice for developing the numerical method to find the free boundary solution, as it may raise new difficulties in finding the the global minimizer of the shape functional. Notice that choosing a shape functional for computing the free boundary in the obstacle problem is a rather nontrivial task, since the main focus of this thesis is to extend the applications of the high-order SEEM algorithms, and to improve the order of convergence of some existing low order algorithms. Here we mainly follow the shape optimization framework suggested in [6] that is proven to possess a unique minimizer at the exact solution provided that some sufficient conditions on the problem data hold. Additionally, we will compare our SEEM-based algorithm with the work in [31] which is a more recent finite element based algorithm built under the same shape optimization framework.

The shape optimization framework in [6] can be formulated as follows. Denote F as a closed nonempty subset of the hold-all domain D with sufficiently smooth boundary $\Gamma_F = \partial F$,

consider the functional

$$J_t(F) = \frac{1}{2}\int_D |\nabla u_F|^2 dx + (t-1)\int_D f u_F dx, \quad t \in \mathbb{R}, \tag{6.3}$$

with the following PDE constraint

$$\begin{cases} -\Delta u_F = f & \text{in } \Omega_F = D/F, \\ u_F = \varphi & \text{in } F, \\ u_F = 0 & \text{on } \Sigma = \partial D, \end{cases} \tag{6.4}$$

where u_F is the solution to the constraint BVP, and the subscription F emphasizes this BVP depends on the set F. The parameter t in the shape functional (6.3) can be chosen as any real number in theory, however, special cases when $t = 0$ [35] and $t = 1$ [21] have already been proposed elsewhere, this choice of shape functional generalizes those existing results. In particular, the cases for $t \geq 1$ is of most interest.

Denote $\vec{V} : \mathbb{R}^2 \to \mathbb{R}^2$ as a sufficiently smooth vector field that vanishes outside D, the normal component of $\vec{V}$ on the boundary Γ_F is $V =: \vec{V} \cdot \nu$. According to [6], the shape derivative of $J_t(F)$ in the direction of $\vec{V}$ is as follows

$$dJ_t(F, \vec{V}) = \int_{\Gamma_F} (\partial_\nu u_F - \partial_\nu \varphi)\, B(t) V d\sigma, \tag{6.5}$$

where

$$B(t) = \frac{1}{2}(\partial_\nu u_F + \partial_\nu \varphi) - t\partial_\nu p + (t-1)\partial_\nu u_F, \tag{6.6}$$

and p is the unique solution of the adjoint BVP

$$\begin{cases} -\Delta p = 0 & \text{in } \Omega_F, \\ p = \varphi & \text{on } \Gamma_F, \\ p = 0 & \text{on } \Sigma. \end{cases} \tag{6.7}$$

6.2 Experiments with Known Exact Solutions

We test our algorithms using an example of obstacle problem (6.2) with known exact solutions. For circular domains and radically symmetric obstacles, the exact solution $u = u(r)$ is also radically symmetric, which is given in the following form

$$u(r) = \begin{cases} \varphi(r), & \text{for } r \le r_\Gamma, \\ -fr^2/4 + \alpha \log r + \beta, & \text{for } r_\Gamma \le r \le R, \end{cases}$$

If the obstacle is a flat plane parallel to and positioned Z units above the plane of D, i.e., the height function of the obstacle is $\varphi(r) = Z$. Then the parameters in the exact solution formula can be explicitly determined by the problem data in the following way:

$$\begin{cases} R = 1 \\ Z = f(1 - 3e^{-2})/4 \\ \varphi(r) = Z \end{cases} \implies \begin{cases} \alpha = fe^{-2}/2 \\ \beta = f/4 \\ r_\Gamma = e^{-1} \end{cases} \tag{6.8}$$

This is the test example used in [31], we choose it for better comparison between our methods and the existing ones. The initial curve for the gradient descent iteration is shown in Figure 6.2.

There are two kinds of error metrics used in [31] for measuring the approximation of the

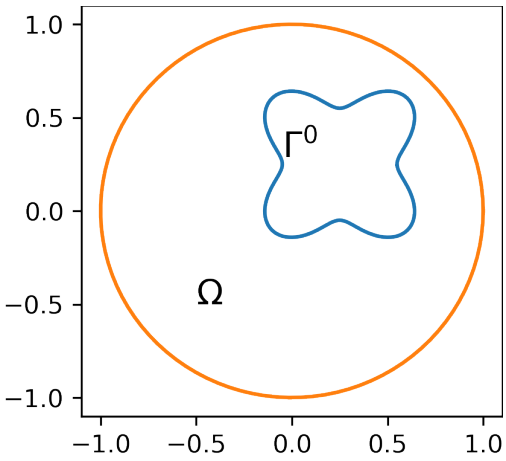


Figure 6.2: Initial curve used in the shape gradient descent iteration to detect the free boundary of the obstacle problem.

obstacle problems. We reformulate them in the notation of our algorithm as follows

- Root Mean Square Error E_{rms}

  The root mean square of the distances between the numerical solution Γ_N and the exact solution Γ is defined as

  $$E_{\text{rms}}(\Gamma_N) = \left(\frac{1}{N} \sum_{k=1}^{N} \|x_k - \Pi_\Gamma(x_k)\|^2 \right)^{1/2}.$$

  where x_k is the k-th point on Γ_N, and $\Pi_\Gamma(x_k)$ is the closest point projection of x_k onto Γ.

- L_2 Type Error E_{L_2}

  Denote P_N as the polygon generated by points on Γ_N, then the deviation of Γ_N from Γ in an L^2 sense is defined as

  $$E_{L_2}(\Gamma_N) = \left(\int_{P_N} \|x - \Pi_\Gamma(x)\|^2 ds \right)^{1/2}.$$

The order of convergence is computed with respect to the mesh refinement for both our algorithms and those in [31]. The finite element methods are used therein for solving the

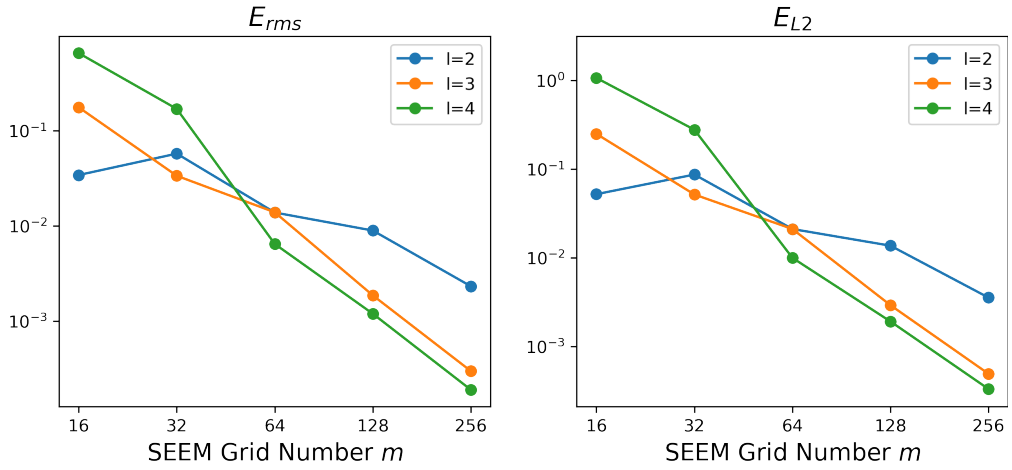


Figure 6.3: The errors with respect to SEEM grid number m in different regularization levels l of SEEM-RPCG solver applied to the obstacle problem.

Algorithm	Order of E_{rms}	Order of E_{L_2}
Algorithm in [31]	1.1014	0.8595
2nd-order SEEM-RPCG	0.9700	0.9687
3rd-order SEEM-RPCG	2.2998	2.2475
4th-order SEEM-RPCG	2.9357	2.9138

Table 6.1: Order of convergence in different errors using SEEM-RPCG solvers of varying orders.

BVPs defined in the domain Ω, and the free boundary Γ is approximated by the mesh nodes on Γ. As the mesh size h decreasing, finer triangulation induces more points on the free boundary, then the mesh refinement of the algorithm in [31] can be measured by the decrease of h. Equivalently, our algorithms use SEEM as the BVP solving algorithm, the mesh refinement can be measured by the increase of the grid number m. Now the specific formula of the order of convergence that are equivalent for both methods is as follows,

$$p = \frac{\log(e_n/e_1)}{\log(h_n/h_1)} = \frac{\log(e_n/e_1)}{\log(m_1/m_n)},$$

where e_k represents any of the approximation error metric defined above, h_k represents the mesh size of the finite element method used in [31], m_k represents the number of the spatial grid points used one dimension in SEEM, $k = 1, n$ respectively denotes the first and last record of data that is placed in order of mesh refinement.

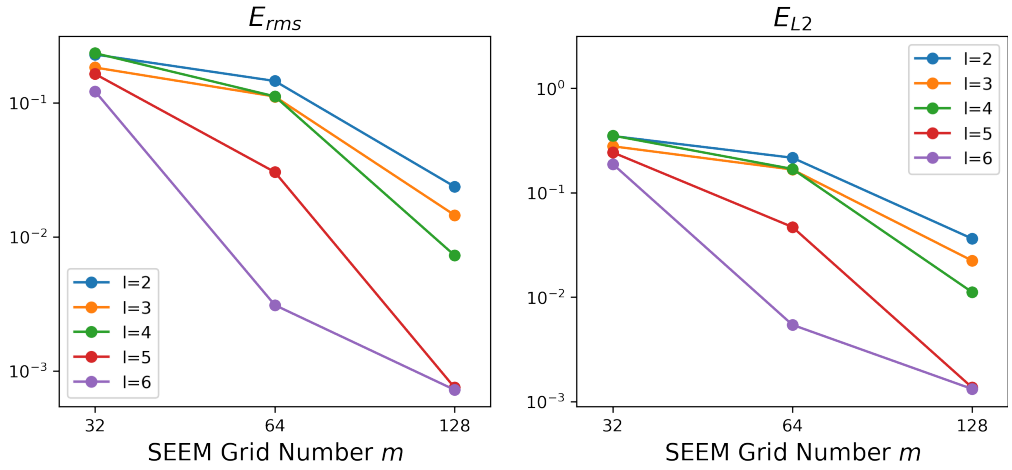


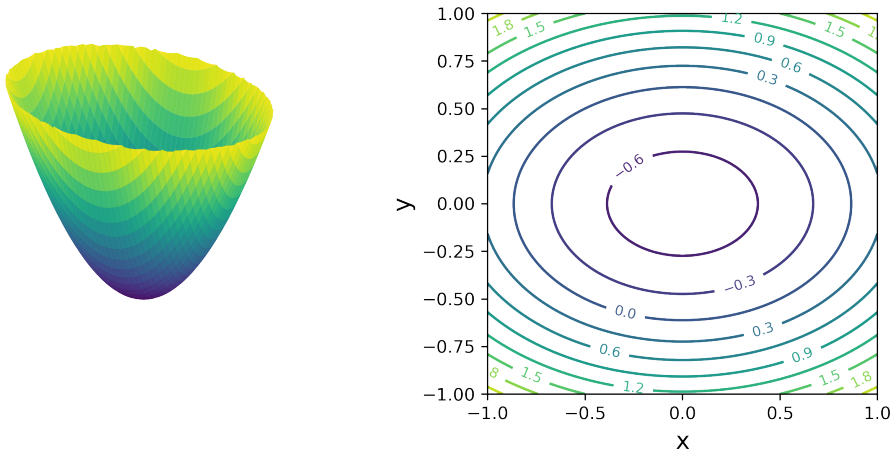
Figure 6.4: The errors with respect to SEEM grid number m in different regularization levels l of SEEM-Direct solver applied to the obstacle problem.

Figure 6.3 and Table 6.1 display the error plots and rate of convergence table of our algorithm applied to the obstacle problem with exact solution (6.8) using SEEM-RPCG solvers of different regularization levels.

Algorithm	Order of E_{rms}	Order of E_{L_2}
Algorithm in [31]	1.1014	0.8595
2nd order SEEM-Direct	1.6377	1.6322
3rd order SEEM-Direct	1.8294	1.8189
4th order SEEM-Direct	2.5058	2.4895
5th order SEEM-Direct	3.8862	3.7379
6th order SEEM-Direct	3.6988	3.5767

Table 6.2: Order of convergence in different errors using SEEM-Direct solvers of varying orders.

Figure 6.4 and Table 6.2 display the error plots and rate of convergence table of our algorithm applied to the obstacle problem with exact solution (6.8) using SEEM-Direct solvers of different regularization levels.



(a) 3D graph (b) Contour Plot

Figure 6.5: An elliptic paraboloid obstacle used for testing the SEEM-based gradient descent algorithm on an obstacle problem with no known exact solution.

Regularization Level	Order of SEEM-RPCG
2	0.8099731675905643
3	2.096281860836129
4	2.1089154406932495

Table 6.3: Order of convergence in L_∞ norm of shape gradient vector for different levels of l using SEEM-RPCG solver.

6.3 Experiments with Unknown Exact Solutions

Consider the obstacle problem (6.2) featuring an elliptic paraboloid obstacle described by the equation $z = x^2 + 2y^2 - 0.75$, as shown in Figure 6.5. The problem is sufficient smooth with a simple closed free boundary that is unknown analytically. We measure the performance of our algorithms by the L_∞ norm of the shape gradient vector evaluated on the discrete boundary solution. The initial curve for this example is taken as a circle with radius 0.75. Both SEEM-RPCG and SEEM-Direct solver are tested for this example.

Figure 6.6 and Table 6.3 show the results for SEEM-RPCG solver. Figure 6.7 and Table 6.4 are for SEEM-Direct solver.

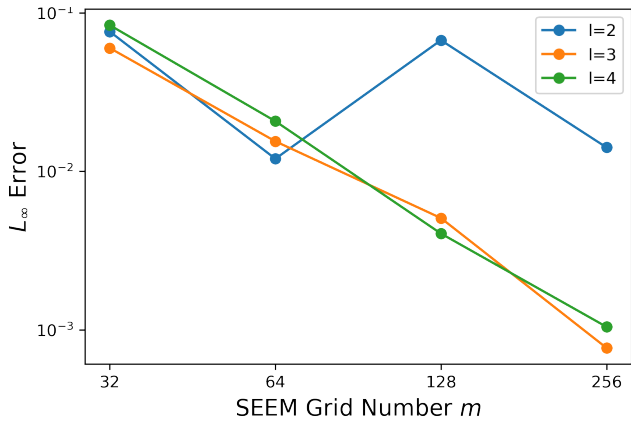


Figure 6.6: The L_∞ errors of the shape gradient vectors with respect to the SEEM grid number m in different regularization levels l using SEEM-RPCG solver.

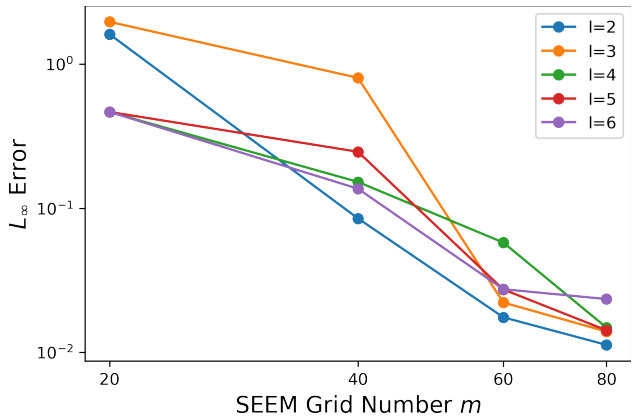


Figure 6.7: The L_∞ errors of the shape gradient vectors with respect to the SEEM grid number m in different regularization levels l using SEEM-Direct solver.

Regularization Level	Order of SEEM-Direct
2	3.579707665243948
3	3.5700885592783633
4	2.4835150383645166
5	2.5165530934708134
6	2.157117510663864

Table 6.4: Order of convergence in L_∞ norm of shape gradient vector for different levels of l using SEEM-Direct solver.

We conclude that our proposed SEEM-based gradient descent algorithm demonstrates a high order of convergence when applied to the obstacle problems. The regularized iterative method operates stably on a dense grid, whereas the direct method leverages the high-order nature to achieve competitive accuracy within a sparse grid, which significantly reduces computational costs.

Chapter 7

Conclusion

The primary contribution of this thesis is the expansion of applicability of SEEM to tackle free boundary problems within the framework of shape optimization. We have introduced a stable, high-order approach for free boundary problems by leveraging regularized SEEM techniques.

Specifically, this work addresses challenges faced by numerical shape optimization approaches in solving free boundary problems, where commonly employed boundary value problem solvers (BVP) exhibit low orders of convergence and struggle with mesh degeneration throughout the problem-solving process.

We propose the use of SEEM, an alternative BVP solving approach, within the optimization framework. SEEM stands out as a BVP solver due to its highly tunable order of convergence. It employs simple discretization across a larger hold-all domain and remains unaffected by boundary evolution, offering distinct advantages.

A key challenge in integrating SEEM into the shape optimization framework is its sensitivity to perturbations in the discrete representation of boundaries. We address this by employing

a regularized kernel-based level set method to stably represent discrete boundaries and extend the regularization concept to develop a regularized preconditioning approach for SEEM. This significantly reduces the condition number for rough boundaries by up to 70%, enhancing stability of SEEM in tackling boundary value problems on unknown boundaries and facilitating its use in shape optimization frameworks for free boundary problems.

We demonstrate that the regularized SEEM preserves high-order accuracy in shape derivative approximation within boundary integral formulations, whereas the same task using finite element approximation experiences a loss in convergence order. This enables the direct use of shape gradient representation in shape optimization algorithms.

Furthermore, the regularized SEEM method sidesteps the meshing complications in a direct and efficient way, which traditional finite element methods often have to expend additional efforts to overcome. Experimental results show that the regularized SEEM is capable of solving more applied problems, such as obstacle problems, while maintaining a high order of convergence.

Bibliography

[1] D. Agress, P. Guidotti, and D. Yan. The smooth extension embedding method with chebyshev polynomials. *Numerical Methods for Partial Differential Equations*, 39(3):2355–2377, 2023.

[2] D. J. Agress. *Seem: A Kernel-Based Fictitious Domain Method*. University of California, Irvine, 2020.

[3] D. J. Agress and P. Q. Guidotti. The smooth extension embedding method. *SIAM Journal on Scientific Computing*, 43(1):A446–A471, 2021.

[4] E. Bänsch, P. Morin, and R. H. Nochetto. A finite element method for surface diffusion: the parametric case. *Journal of Computational Physics*, 203(1):321–343, 2005.

[5] A. E. Berger, M. Ciment, and J. C. Rogers. Numerical solution of a diffusion consumption problem with a free boundary. *SIAM Journal on Numerical Analysis*, 12(4):646–672, 1975.

[6] A. Bogomolny and J. Hou. Shape optimization approach to numerical solution of the obstacle problem. *Applied Mathematics and Optimization*, 12:45–72, 1984.

[7] A. Bonito, R. H. Nochetto, and M. S. Pauletti. Geometrically consistent mesh modification. *SIAM Journal on Numerical Analysis*, 48(5):1877–1899, 2010.

[8] R. Brügger and H. Harbrecht. On the reformulation of the classical stefan problem as a shape optimization problem. *SIAM Journal on Control and Optimization*, 60(1):310–329, 2022.

[9] D. Bucur, G. Buttazzo, et al. Variational methods in shape optimization problems. *(No Title)*, 2000.

[10] J. Crank. Free and moving boundary problems. *(No Title)*, 1984.

[11] M. C. Delfour and J.-P. Zolésio. *Shapes and geometries: metrics, analysis, differential calculus, and optimization*. SIAM, 2011.

[12] G. Dogan, P. Morin, R. H. Nochetto, and M. Verani. Discrete gradient flows for shape optimization and applications. *Computer methods in applied mechanics and engineering*, 196(37-40):3898–3914, 2007.

[13] J. S. Dokken, S. W. Funke, A. Johansson, and S. Schmidt. Shape optimization using the finite element method on multiple meshes with nitsche coupling. *SIAM Journal on Scientific Computing*, 41(3):A1923–A1948, 2019.

[14] T. Etling, R. Herzog, E. Loayza, and G. Wachsmuth. First and second order shape optimization based on restricted mesh deformations. *SIAM Journal on Scientific Computing*, 42(2):A1200–A1225, 2020.

[15] P. Guidotti. Dimension independent data sets approximation and applications to classification. *arXiv preprint arXiv:2208.13781*, 2022.

[16] J. Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*, volume 33. Imprimerie nationale, 1908.

[17] J. Haubner, M. Siebenborn, and M. Ulbrich. A continuous perspective on shape optimization via domain transformations. *SIAM Journal on Scientific Computing*, 43(3):A1997–A2018, 2021.

[18] M. Hintermüller and W. Ring. A second order shape optimization approach for image segmentation. *SIAM Journal on Applied Mathematics*, 64(2):442–467, 2004.

[19] R. Hiptmair, A. Paganini, and S. Sargheini. Comparison of approximate shape gradients. *BIT Numerical Mathematics*, 55:459–485, 2015.

[20] J. B. Ho and C. Farhat. Aerodynamic shape optimization using an embedded boundary method with smoothness guarantees. In *AIAA Scitech 2021 Forum*, page 0280, 2021.

[21] J.-W. Hou. *Shape optimal design and free boundary value problems*. The University of Iowa, 1983.

[22] J. A. Iglesias, K. Sturm, and F. Wechsung. Two-dimensional shape optimization with nearly conformal transformations. *SIAM Journal on Scientific Computing*, 40(6):A3807–A3830, 2018.

[23] K. Ito, K. Kunisch, and G. H. Peichl. Variational approach to shape derivatives. *ESAIM: Control, Optimisation and Calculus of Variations*, 14(3):517–539, 2008.

[24] C. Maple. Geometric design and space planning using the marching squares and marching cube algorithms. In *2003 international conference on geometric modeling and graphics, 2003. Proceedings*, pages 90–95. IEEE, 2003.

[25] P. Morin, R. H. Nochetto, M. S. Pauletti, and M. Verani. Adaptive finite element method for shape optimization. *ESAIM: Control, Optimisation and Calculus of Variations*, 18(4):1122–1149, 2012.

[26] A. Nägel, V. Schulz, M. Siebenborn, and G. Wittum. Scalable shape optimization methods for structured inverse modeling in 3d diffusive processes. *Computing and Visualization in Science*, 17:79–88, 2015.

[27] S. Osher, R. Fedkiw, and K. Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004.

[28] D. W. Quinn and M. E. Oxley. The boundary element method applied to moving boundary problems. *Mathematical and Computer Modelling*, 14:145–150, 1990.

[29] L. Rao and H. Chen. The technique of the immersed boundary method: application to solving shape optimization problem. *Journal of Applied Mathematics and Physics*, 5(2):329–340, 2017.

[30] V. H. Schulz, M. Siebenborn, and K. Welker. Structured inverse modeling in parabolic diffusion problems. *SIAM Journal on Control and Optimization*, 53(6):3319–3338, 2015.

[31] A. Sharma and R. Rangarajan. A shape optimization approach for simulating contact of elastic membranes with rigid obstacles. *International Journal for Numerical Methods in Engineering*, 117(4):371–404, 2019.

[32] J. Sokolowski and J.-P. Zolésio. Introduction to shape optimization. In *Introduction to shape optimization*, pages 5–12. Springer, 1992.

[33] R. Udawalpola and M. Berggren. Optimization of an acoustic horn with respect to efficiency and directivity. *International journal for numerical methods in engineering*, 73(11):1571–1606, 2008.

[34] D. N. Wilke, S. Kok, and A. A. Groenwold. A quadratically convergent unstructured remeshing strategy for shape optimization. *International journal for numerical methods in engineering*, 65(1):1–17, 2006.

[35] J.-P. Zolésio. Domain variational formulation for free boundary problems. *Optimization of Distributed Parameter Structures-Volume II*, pages 1152–1194, 1981.

[36] P. Zunino. Multidimensional pharmacokinetic models applied to the design of drug-eluting stents. *Cardiovascular Engineering: An International Journal*, 4:181–191, 2004.