12-2016

A sharp interface isogeometric strategy for moving boundary problems

Tao Song
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Tao Song

Entitled

A Sharp Interface Isogeometric Strategy for Moving Boundary Problems

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Ganesh Subbarayan (Chair)	
Thomas Siegmund	
Kejie Zhao	
Ahmed Sameh	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Ganesh Subbarayan

Approved by: Jay P. Gore 12/2/2016

Head of the Departmental Graduate Program Date

A SHARP INTERFACE ISOGEOMETRIC STRATEGY FOR MOVING BOUNDARY PROBLEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Tao Song

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2016

Purdue University

West Lafayette, Indiana

To my parents and girlfriend,

for their love, support, and encouragement throughout my life

ACKNOWLEDGMENTS

I still remember the first time I set foot on the land of Purdue, the first time I met my advisor, Professor Ganesh Subbarayan, the first time I stepped into a graduate classroom, and the first time I presented in the group meeting, as if they just happened yesterday. I could see a nervous but excited, 22-year-old young man stood there, curious about his future and brave to meet any challenges. As I reach an important milestone in my life and look back, I would like to thank all the individuals who have encouraged me, supported me and influenced me since the beginning of my doctoral journey.

First of all, I feel honored and privileged to have had the chance to closely work with and be advised by, Professor Ganesh Subbarayan throughout my graduate study. His profound knowledge, sharp physical and engineering insights, proficient communication skills and genuine caring for his students have constantly inspired me towards a better individual in all aspects of my professional life. An ancient Chinese proverb says "*give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime*". I am extremely grateful to him for guiding me towards a fisherman.

I want to sincerely thank my esteemed committee members: Professor Thomas Siegmund, Professor Kejie Zhao, and Professor Ahmed Sameh for their invaluable and insightful suggestions. I also want to thank Professor Kejie Zhao and his doctoral student, Rong Xu for providing the nano-indentation facilities to validate my experimental ideas. I deeply appreciate the guidance from my internship supervisor at GLOBALFOUNDRIES Inc., Dr. Rod Augur, as well as my mentor, Dr. Dewei Xu. I am also grateful to the financial support from Purdue and Semiconductor Research Corporation (SRC).

I want to extend my acknowledgments to the members of the HiDAC lab for the precious memory and endless fun. I would like to thank lab-alumni – Chaitra and Abhishek for their help and support during my early days in the lab. I am thankful to lab-alumni – Hung-Yun for his guidance, Kritika for our collaboration, Subramanya for his brilliant thoughts, Anirudh for interesting discussions on integrity, responsibility and god, Mesut for his friendship and delicious Turkish dishes, and Yiran for his suggestions on career planning. I would like to thank my current labmates – Yuvraj for his sense of humor, Chun-pei for our effective collaboration and discussions, Travis for his enthusiasm and readiness to help, Pavan for his hardwork and sincerity, Maryam for her beautiful presence. I also want to thank the new members of the lab: Chung-shuo and Sukshitha.

I would like to thank my friends: Qingzi, Sheri, Yifei, Wuyang and Yan for their constant encouragement and support which made my life much more cheerful at West Lafayette. I would like to express my heartfelt gratitude to my parents for their unconditional, unwavering love, support and understanding of my choices. Last but not least, I want to thank my beautiful girlfriend, Heng, for sharing my happiness and sorrow, my smiles and tears, in the past, present and future.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

ABSTRACT

Song, Tao PhD, Purdue University, December 2016. A Sharp Interface Isogeometric Strategy for Moving Boundary Problems. Major Professor: Ganesh Subbarayan, School of Mechanical Engineering.

Moving boundary problems that include crack propagation, solidification and shape optimization occur in a variety of natural and engineered systems. A significant difficulty in solving the moving boundary problems, in addition to computationally modeling the moving interface, is to capture the interfacial behavior. Commonly, moving boundary problems are solved by diffuse interface techniques such as the phase field method that is challenged by high computational cost or by the level set method in which considerable care needs to be exercised to ensure stable evolution of the interface. The challenges with phase field method include the need to develop alternative (diffuse) mathematical forms of the governing equations that must be proven to converge to the sharp interface form. The phase-field equations are also usually non-linear and non-convex, and the diffuse transition region has to be very thin to converge to the physical solution. This last fact in turn requires the mesh to span several orders of magnitude in length scale and to be very refined near the interfacial region. In the case of level set method, the solution to the Hamilton-Jacobi equation needs stabilization to minimize the oscillation and an auxiliary velocity field equation needs to be posed and solved to extend the velocity from the interface into the domain. Finally, but most importantly, due to the geometrically implicit nature of these methods (the boundary is not explicitly represented, but inferred from the value of the phase field or level set parameter), the necessary geometric quantities such as normals and curvatures are difficult to compute with $\mathscr{C}^0$ continuous finite elements and can be accurately computed only in the limit of mesh refinement.

To circumvent these challenges, a sharp interface isogeometric formalism that enables efficient analysis and accurate capture of interfacial behavior is proposed. In this modeling strategy, the approximation of the underlying domain is isoparametrically enriched with lower-dimensional features such as domain boundaries, crack surfaces and phase interfaces, and the geometry of the enriching entities is explicitly tracked. The blending of the enrichment with the underlying approximation requires an estimate of distance to the enriching geometry from a quadrature point and the parametric value of the footpoint on the enriching geometry. In the present research, utilizing algebraic geometry concepts, purely algebraic estimates of distance coupled with an algebraic point projection are proposed. These algebraic techniques rely on implicitization of the parametric curve, and are shown to be more efficient and robust than Newton-Raphson iterations. Since, in the sharp interface isogeometric framework, the enriching geometry is immersed in the underlying domain and intersects with the domain mesh grid, the numerical integration of the regions created by the enrichment process is challenging. Here, a novel Kd-tree based adaptive quadrature scheme is developed to enhance integration accuracy and efficiency. The quadrature cells are generated through a smart subdivision process based on the signed distance of the endpoints and midpoints of parent cells, and then stored in a Kd-tree data structure. The proposed integration scheme effectively minimizes the inefficient and excessive number of quadrature cells resulting from classical quad-tree/oct-tree subdivision. Moreover, in many immersed boundary problems, including crack propagation and shape optimization, the behavioral field may exhibit a high local gradient (such as stress concentration) near the boundaries. An accurate solution of these problems necessitates a refinement of the underlying approximation. To this end, the truncated hierarchical B-splines (THB-splines), maintaining high smoothness of the isogeometric basis while enabling local refinement, are utilized to facilitate the analysis. Two efficient *a-priori* mesh refinement algorithms based on the signed and unsigned distance fields are developed to generate a hierarchical mesh adaptive to the immersed boundaries. It is also shown that the THB-splines may, theoretically, introduce a

large number of active basis functions at a quadrature point, which can degrade the efficiency during matrix assembly. This drawback is mitigated through a newly proposed all-at-once algorithm which calculates all the active THB-spline basis functions simultaneously.

The proposed methodology is first utilized to model stationary and propagating cracks. The crack face is enriched with the Heaviside function which captures the displacement discontinuity. Meanwhile, the crack tips are enriched with asymptotic displacement functions to reproduce the tip singularity. The enriching degrees of freedom associated with the crack tips are chosen as stress intensity factors (SIFs) such that these quantities can be directly extracted from the solution without *a-posteriori* integral calculation.

As a second application, the Stefan problem is modeled with a hybrid function/derivative enriched interface. Since the interface geometry is explicitly defined, normals and curvatures can be analytically obtained at any point on the interface, allowing for complex boundary conditions dependent on curvature or normal to be naturally imposed. Thus, the enriched approximation naturally captures the interfacial discontinuity in temperature gradient and enables the imposition of Gibbs-Thomson condition during solidification simulation.

The shape optimization through configuration of finite-sized heterogeneities is lastly studied. The optimization relies on the recently derived configurational derivative that describes the sensitivity of an arbitrary objective with respect to arbitrary design modifications of a heterogeneity inserted into a domain. The THB-splines, which serve as the underlying approximation, produce sufficiently smooth solution near the boundaries of the heterogeneity for accurate calculation of the configurational derivatives.

1. INTRODUCTION

The study of moving boundary problems, including crack propagation, solidification and shape optimization, is important to understanding many natural phenomena as well as guaranteeing the performance of engineered objects. Often, the behavioral characteristics associated with the immersed boundaries is known *a-priori*. Figure 1.1 illustrates different types of boundaries that may occur in engineering systems. A discontinuity can appear in the heat flux across the phase boundary, and in the normal strain across the material interface. The fracture is also characterized by a displacement jump across the crack face. In addition to the weak and strong discontinuities, the interface may also possess many geometry-dependent properties and boundary conditions. Therefore, the capability to capture the discontinuities across the interface, as well as an accurate geometric representation of the interface, plays an important role in modeling the moving boundary problems.

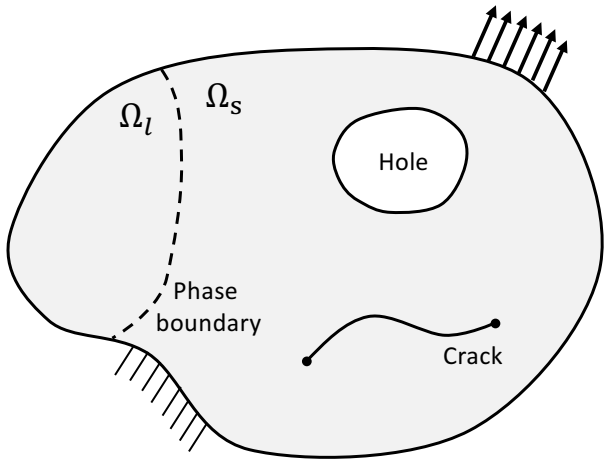


Figure 1.1. Schematic of typical immersed boundaries in engineering systems. Ω_l and Ω_s represents liquid and solid phases, respectively. The phase boundary, material interface and crack face possess specific behaviors involving strong or weak discontinuities.

To address the computational challenges, a straightforward approach is to force the mesh align with the interface. This in turn requires remeshing as the interface evolves. Alternatively, fixed mesh frameworks with enriching degrees of freedom (GFEM [1,2] or XFEM [3]) may be used. In these methods, the evolving interface is commonly tracked with an implicit geometry such as level-sets [4]. While the mesh regeneration is avoided in GFEM/XFEM, the enriched node set need to be updated during interface evolution. Another non-conforming mesh based technique is the phase-field method [5] where the interface is represented by a diffuse phase-field variable. Due to the implicit nature of the level-sets and phase-field, the geometric quantities such as interfacial normal and curvature are not explicitly known at a point on the interface. Since, the interface intersects with a non-conforming mesh, an accurate and efficient numerical integration over the cut elements remains a challenge.

In general, finite element based solutions also suffer from another drawback in that the classical Lagrangian elements are piecewise smooth but globally $\mathscr{C}^0$-continuous. The inherent element discontinuity would affects application of interface conditions, and gives rise to a large analysis error when the interface exhibiting weak discontinuities is too close to the element boundaries. In this case, a $\mathscr{C}^1$-continuous mesh is desired [6]. However, there are very few two-dimensional finite element implementations (e.g., Hermite elements) possessing $\mathscr{C}^1$-continuity and almost none in three-dimension.

Renken and Subbarayan [7] first exploited the geometric representation in computer aided dsegin (CAD) for analysis. To enable a seamless integration between design and analysis, Natekar et al. [8] proposed constructive solid analysis (CSA) in analogy to the constructive solid geometry (CSG) of CAD. The methodology relied on the same mathematical representation (e.g., non-uniform rational B-splines, or NURBS) for geometry and behavioral field, an idea that was later referred as isogeometric analysis (IGA) [9]. The NURBS, along with other spline variants, do not only enable a precise representation of complex geometries, but also provide a $\mathscr{C}^1$ or higher-order global continuity. Following the spirit of GFEM/XFEM, one can add

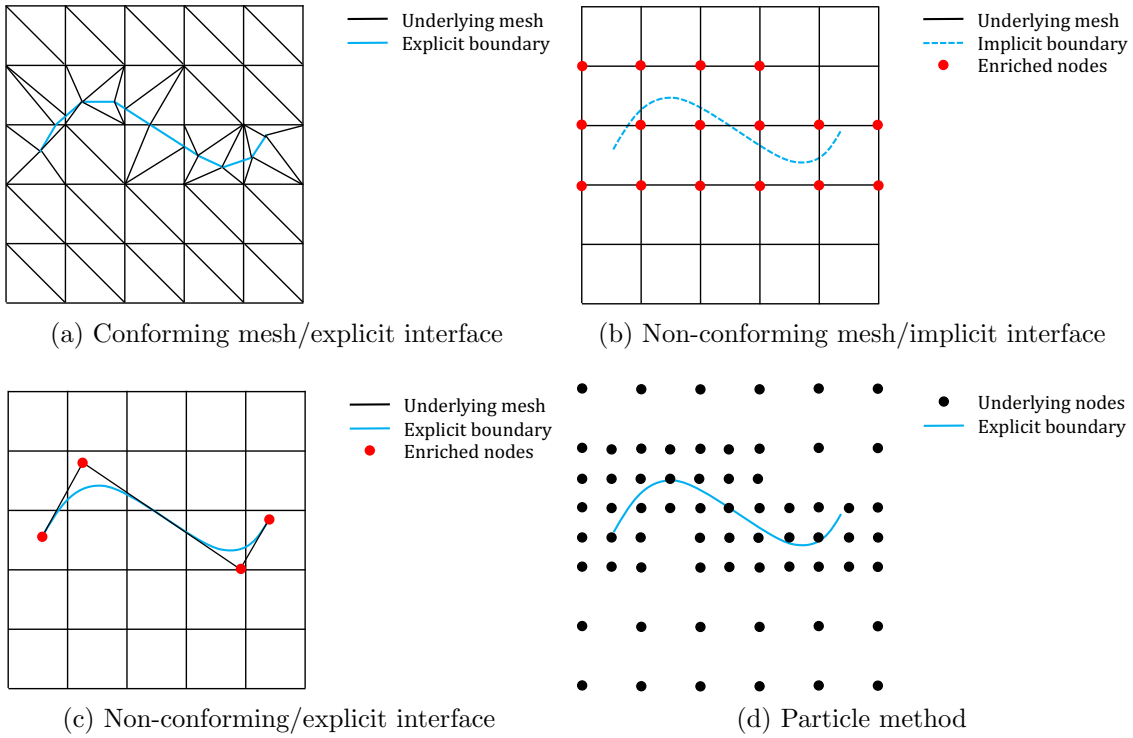
the enriching degrees of freedom to a fixed set of NURBS control points to model the moving boundaries without remeshing [10–12]. However, there is still a need for identifying and enriching the control points. In contrast, Tambat et al. [13] developed isogemetric enriched field approximation where the interface is explicitly represented by a lower-dimensional NURBS entity, and the additional degrees of freedom are directly associated with the control points of the interface. The methodology has been successfully applied to study crack propagation in layered structures [14]. Nevertheless, several numerical issues such as point projection, non-homogeneous interface conditions and local refinement, were not addressed in the early research.

1.1 Survey of Computational Techniques for Moving Boundary Problems

Based on the previous discussion, the computational modeling of moving boundary problems typically involves three challenges:

1. Accurate geometric representation of the underlying domain and the interface,
2. Construction of a solution space that accurately and efficiently captures the behavior near the interface, and
3. Ability to efficiently update and reanalyze the problem as the interface evolves.

The established techniques for modeling moving boundary problems can be first classified by the type of mesh used: Conforming mesh or non-conforming mesh or mesh-free (particle methods). The non-conforming mesh methods can be further categorized based on the representation of the interface. An interface is identified as *explicit* if its geometric quantities (e.g., normal, tangent, curvature, etc.) can be analytically obtained. In contrast, for an implicit interface, the material phase is known at a spatial point, but not the geometric quantities such as tangents or normals. The classification is pictorially illustrated in Figure 1.2 and further discussed below.



(a) Conforming mesh/explicit interface

(b) Non-conforming mesh/implicit interface

(c) Non-conforming/explicit interface

(d) Particle method

Figure 1.2. Classification of the techniques for modeling moving boundary problems: (a) Conforming mesh-explicit interface, e.g., finite element method (FEM); (b) non-conforming mesh/implicit interface, e.g., eXtended Finite Element Method (XFEM); (b) non-conforming mesh/explicit interface, e.g., Enriched Isogeometric Analysis (EIGA); (d) particle methods, e.g., Element Free Galerkin (EFG) method.

1.1.1 Conforming Mesh/Explicit Interface

A typical technique using the conforming mesh/explicit interface is the common implementation of the finite element method in commercial codes. The $\mathscr{C}^0$-continuous Lagrangian elements, despite causing a discontinuous gradient field across the element boundaries, may be used to represent weakly discontinuous behavior when the interface is aligned with the element boundaries. For instance, in the early literature, a moving mesh method [15, 16] was proposed to model the solidification problem (Stefan problem). According to the physics of the Stefan problem, the evolution speed of

the interface is proportional to the heat flux jump across the interface. Such a value can be readily calculated from the $\mathscr{C}^0$-continuous elements. Besides, the alignment of the interface with element boundaries also enables a strong imposition of the interface conditions.

However, the finite element approximation, with its inherent geometrical approximation, is accurate only in the limit of mesh refinement. The interface also becomes piecewise linear/planar while aligning with the element boundaries. In general, a great number of degrees of freedom are needed to model complex geometries and interfaces [17]. In addition, the tracking of the moving boundaries require remeshing at every step [18], and may cause a strong mesh distortion [19].

1.1.2 Non-conforming Mesh/Implicit Interface

The non-conforming mesh methods can avoid the inefficient mesh regeneration but require auxiliary techniques to capture the various interfacial behaviors. Babuska et al. developed the Partition of Unity Finite Element Method (PUFEM) which can incorporate *a-priori* knowledge of local behavior into solution space. Based on the PUFEM, Strouboulis et al. [2] proposed the Generalized Finite Element Method (GFEM) by adding behavior-dependent degrees of freedom to the underlying finite element nodes. A general expression of the GFEM approximation is given by

$$u_{\text{GFEM}} = \sum_{i \in I} N_i a_i + \sum_{j \in I_e} N_j \left(\sum_k g_k^{(j)} \psi_k^{(j)} \right) \tag{1.1}$$

where, I and I_e are the regular and enriched node sets, respectively. N_i is the classical FE basis function at node i and a_i is the corresponding nodal field. $\psi_k^{(j)}$ is the k-th enriching function at node j and $b_k^{(j)}$ represents the enriching degree of freedom associated with $\psi_k^{(j)}$. A variety of the enriching functions were provided in [20] for different behavioral fields. Belytschko et al. [3] proposed eXtended Finite Element Method (XFEM) to model the singular stress at crack tips by using asymptotic displacement

inspired enriching functions. Dolbow et al. [21] next added the Heaviside function to the enriching function space to model the displacement jump across the crack face. The level-sets [4], a tool to implicitize the immersed boundary, have been combined with XFEM to solve a variety of moving boundary problems including crack propagation [22], dislocation [23, 24] and phase evolution [25–27]. The level set method suffers from several drawbacks. First, the level sets are described by the Hamilton-Jacobi equation, which is a difficult to solve, first-order hyperbolic equation and needs stabilization to minimize the oscillation [28]. Further, due to the geometrically implicit nature, the interfacial geometric quantities are not analytically known. Since the enriching degrees of freedom are associated with the domain nodes, the interface conditions have to be (weakly) applied on mesh than the interface.

The Phase-field method in which the interface is represented by a thin region defined by the phase-field variable, has also been utilized to model moving boundary problems such as crack growth [29, 30] and solidification [31–34]. A significant challenge with phase field method is the need to develop alternative (diffuse) mathematical forms of the governing equations, which is typically much more complex that those of the sharp interface models. An comparison of the governing equations for the Stefan problem is shown in Table 1.1. The phase-field equations are often non-linear and non-convex, and the diffuse transition region has to be very thin to converge to the physical solution, which in turn requires the mesh to span several orders of magnitude in length scale and to be highly refined near the interfacial region. Thus, adaptive mesh refinement is strongly desired to improve the efficiency. Further, the introduction of phase-field variables often gives rise to a forth or higher order partial differential equation, which necessitates $\mathscr{C}^1$-continuous elements or other complicated numerical schemes.

Table 1.1.
Sharp and diffuse interface governing equations for the Stefan problem.

Sharp Interface	**Diffuse Interface** [32]
In the bulk :	In the bulk:
$\frac{\partial T}{\partial t} = \alpha \nabla^2 T$	$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + \frac{1}{2}\frac{\partial L(\phi)}{\partial t}$
On the phase boundary :	$\tau(\boldsymbol{n})\frac{\partial \phi}{\partial t} = [\phi - \lambda(1-\phi^2)](1-\phi^2)$
$[\![q_n]\!] = -\rho L v_n$	$+ \nabla \cdot [W^2(\boldsymbol{n})\nabla\phi]$
$T_m = T_m - \epsilon_c(\boldsymbol{n})\kappa - \epsilon_v(\boldsymbol{n})v_n$	$+ \frac{\partial}{\partial x}\left(\lvert\nabla\phi\rvert^2 W(\boldsymbol{n})\frac{\partial W(\boldsymbol{n})}{\partial \phi_{,x}}\right)$
	$+ \frac{\partial}{\partial y}\left(\lvert\nabla\phi\rvert^2 W(\boldsymbol{n})\frac{\partial W(\boldsymbol{n})}{\partial \phi_{,y}}\right)$

1.1.3 Non-conforming Mesh/Explicit Interface

In this work, a non-conforming mesh, explicit interface method, termed as *Enriched Isogeometric Analysis* (EIGA), is proposed for modeling moving boundary problems. The method retains the meshing convenience while preserving the geometric exactness of the interface with a explicit representation (see Figure 1.2c). The behavior-dependent, enriching degrees of freedom are directly associated with the interface than the underlying domain, avoiding cumbersome update of the enriched node set. More importantly, a strong imposition of interface conditions becomes possible. The advantages of the proposed method are compared against existing mesh-dependent techniques in Table 1.2.

The EIGA approximation is composed of the underlying approximation which has a continuous contribution over the problem domain, and the enriched approximation whose influence decays with distance. Both constitutive approximations are isogeometric to enable direct CAD&E integration and provide higher global smoothness. The influence of an interface on a spatial point can be estimated by constructing a distance field from the interface. Further, in order to specifically identify the active enriched nodes that influence the point, as illustrated Figure 1.3, a geometric point

Table 1.2.
Comparison of modeling techniques for moving boundary problems.

	Finite Element Method	**Level-set & Phase-field Method**	**Enriched Isogeometric Analysis**
Mesh Type	✗ Conforming mesh: • Remeshing at every step	✓ Non-conforming mesh: • No remeshing	✓ Non-conforming mesh: • optional remeshing • $\mathscr{C}^1$ or higher order continuity
Interface Type	✓ Explicit interface: • Direct imposition of interface conditions	✗ Implicit interface: • Computationally expensive • Hard-to-solve equations • Geometric quantities not known	✓ Explicit interface: • Direct imposition of interface conditions • Simpler governing equations • Accurate calculation of geometric quantities

projection becomes necessary. The non-conforming mesh, albeit providing flexibility in mesh generation, leads to a difficulty in numerical integration. To accurately integrate over the cells that are intersected by the interface, an efficient cell subdivision algorithm is desired. In addition, if the behavioral field exhibits high gradient in a region, according to the *a-posteriori* error estimator [35, 36], this region needs to be sufficiently refined to achieve a smaller computational error. However, the local refinement within isogeometric framework is still non-trivial.

1.1.4 Particle Methods

To eliminate the time-consuming mesh generation step, several particle methods, including Element Free Galerkin (EFG) method [37], Reproducing Kernel Particle Method (RKPM) [38] and Meshless Local Petrov-Galerkin (MLPG) method [39],

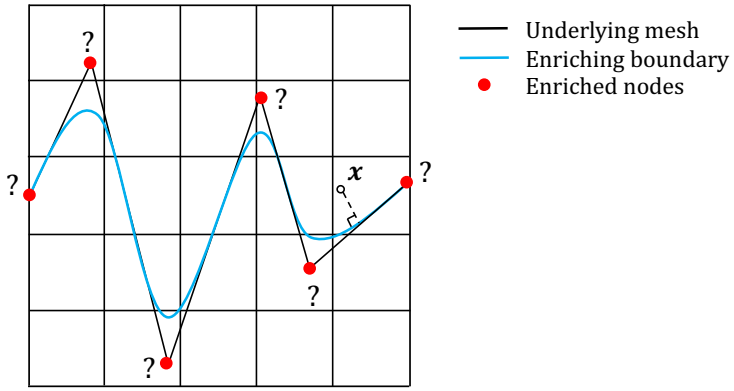


Figure 1.3. The enriched nodes that influence a given point $\boldsymbol{x}$ can be identified by point projection.

have been developed. These methods typically rely on a class of least-squares interpolants which do not require mesh connectivities. Thus, the nodes can be freely moved to account for complex moving boundaries. The particle methods have been successfully applied to model quasi-static and dynamic crack propagation [40–43]. But, the absence of mesh brings two major challenges: imposition of boundary conditions and numerical integration. Due to the non-interpolative nature of the particle approximations, the boundary conditions can only be weakly applied through Lagrange multiplier or penalty method [44]. Since, the domain integration by Gauss quadrature is very cumbersome in particle methods, many nodal integration techniques have been investigated [45–48]. Nevertheless, a stable and efficient numerical integration remains a challenge.

1.2 Research Objectives

The goal of this work is to develop a non-conforming mesh, sharp interface formalism for modeling moving boundary problems. The behavioral approximation is a weighted composition of the continuous approximation associated with the underlying domain and the enriching approximation associated with the interface. A variety of

enriching functions and degrees of freedom are designed to strongly impose interface conditions, and to capture different interfacial behaviors such as singularity, weak and strong discontinuity. Consistent with the isogeometric philosophy, the same mathematical representation is used for geometry as well as behavioral field, maintaining a tight connection between CAD models and engineering analysis. In order to describe the contribution of the enrichment to a spatial point in the domain, and in particular, the region of the enriching geometry that influences the point, there is a critical need to construct a distance field from the interface, and to calculate the parametric value of the footpoint onto the interface. The recently proposed algebraic distance field [49], providing highly efficient distance estimates without loss of geometric exactness, is utilized in the current work. Furthermore, extending the algebraic distance, a novel algebraic point projection method is next proposed. The proposed method is demonstrated to be faster and more robust than Newton-Raphson iterations. Next, to address the challenge in numerical integration of non-conforming mesh methods, a kd-tree based adaptive quadrature scheme is developed. The scheme relies on a smart, dimension-wise subdivision process which results in fewer quadrature sub-cells than classical quad-/oct-tree schemes with less overall computational cost. However, an accurate integration can not guarantee an accurate solution of the behavioral field involving high local gradients. The truncated hierarchical B-splines (THB-splines) [50], retaining $\mathscr{C}^1$ or higher-order smoothness while supporting local refinement, are used to produce accurate field solutions. Several novel algorithms for mesh generation and basis function evaluation are proposed to improve the efficiency during these steps. The enriched isogemetric analysis concept is implemented in a parallel computational framework termed OOF-HiDAC (Object-Oriented Fortran based Hierarchical Design and Analysis Code). The proposed methodology is first applied to model stationary and propagating cracks. The stress intensity factors are selected as the tip enriching degrees of freedom, such that they can be directly obtained from the solution without *a-posteriori* integral calculation. Next, the proposed methodology is utilized to model classical and dendritic Stefan problem. The enriched approximation naturally cap-

tures the interfacial discontinuity in temperature gradient and naturally enables the imposition of Gibbs-Thomson condition. Lastly, the shape optimization through configuration of finite sized heterogeneities is studied. The stress concentration resulting from material interfaces can be effectively represented by the THB-splines.

1.3 Outline

The rest of this dissertation is structured as the following. In Chapter 2, the mathematical form of the enriched isogeometric approximation is proposed. The continuous and enriching approximations that are composed together are formulated. In Chapter 3, the construction of the algebraic distance field, as well as an algebraic manipulation for fast calculation, is reviewed. In Chapter 4, a new algebraic point projection technique for two-dimensional parametric curves and three-dimensional parametric surfaces is developed. The technique provides exact on-curve solution and an accurate near-curve solution, and is faster and more robust than Newton-Raphson iterations. A novel kd-tree based adaptive quadrature scheme, aimed at accurately integrating the cells intersected by the immersed boundaries, is developed in Chapter 5. The truncated hierarchical B-splines, along with their advantages and challenges for engineering analysis, are discussed in Chapter 6. To address the challenges, several new mesh refinement and basis function evaluation algorithms are proposed. In Chapter 7, the implementation of the enriched isgoemetric framework and the supporting techniques in a Fortran code termed as OOF-HiDAC is described. The implementation relies on a hybrid OpenMP/MPI parallelism and is capable of solving problems with a large number of degrees of freedom. In the next three chapters, the proposed techniques are applied to three different types of moving boundary problems: crack modeling (Chapter 8), Stefan problem (Chapter 9) and shape optimization (Chapter 10), respectively. Different enriched approximations are constructed is designed in these problems to capture different interfacial behaviors. Finally, the thesis is summarized in Chapter 11 with the novel contribution and proposed future work.

2. ENRICHED ISOGEOMETRIC ANALYSIS

The general form of the enriched isogeometric approximation is developed in this chapter. The behavioral field is represented by a weighted composition of a continuous approximation on the underlying domain and enriched approximations associated with the moving boundaries. The approximations are usually chosen to be the same as those representing the geometry to enable a tight integration between the geometry construction and analysis.

2.1 Non-Uniform Rational B-Splines (NURBS) Based Approximation Spaces

In this study, both the underlying domain and the moving boundaries are modeled using Non-Uniform Rational B-Splines (NURBS). NURBS can represent complex free surfaces and conics exactly with fewer degrees of freedom, and are therefore widely used in the Computer-Aided Design (CAD) [51]. It has been shown that the NURBS basis is analysis-suitable and can yield higher convergence rates than classical linear basis functions [7–9]. The use of the same approximation (e.g., NURBS) to represent the geometry and the behavioral field enables a seamless CAD/CAE integration without a loss of the geometric accuracy. This methodology is usually referred as Isogeometric Analysis (IGA) [9].

The NURBS approximations for a three-dimensional domain and a behavioral field may be written as [51]:

$$\boldsymbol{x}(\xi,\eta,\zeta) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} R_{ijk}(\xi,\eta,\zeta)\bar{\boldsymbol{x}}_{ijk} \tag{2.1a}$$

$$f(\xi,\eta,\zeta) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} R_{ijk}(\xi,\eta,\zeta)\bar{f}_{ijk} \tag{2.1b}$$

respectively. $\bar{\boldsymbol{x}}_{ijk}$ is the ijk^{th} control point and $\bar{f}_{ijk}$ is the nodal field associated with $\bar{\boldsymbol{x}}_{ijk}$. n_i, n_j and n_k are the number of control points in the i^{th}, j^{th} and k^{th} directions, respectively. R_{ijk} is the rational basis function defined as the tensor product:

$$R_{ijk}(\xi,\eta,\zeta) = \frac{N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\zeta)w_{ijk}}{\sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k}N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\zeta)w_{ijk}} \tag{2.2}$$

where, $N_{i,p}$ is the degree p B-spline basis function defined on a set of non-uniform knot vectors written as:

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n_i+p+1}\}. \tag{2.3}$$

Based on the knot vectors, the B-spline basis functions are given by:

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.4a}$$

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}N_{i+1,p-1}(\xi). \tag{2.4b}$$

The B-spline basis functions have the following properties:

1. **Non-negativity**:

$$N_{i,p}(\xi) \geq 0 \quad \forall\, \xi. \tag{2.5}$$

2. **Partition of unity**:

$$\sum_{i=1}^{n_i} N_{i,p}(\xi) = 1. \tag{2.6}$$

3. **Local support**: $N_{i,p}(\xi)$ has a compact support and is non-zero in the interval $[\xi_i, \xi_{i+p+1})$.

4. **Smoothness**: $N_{i,p}(\xi)$ has continuous derivates of order $p - m$, with knot multiplicity of m.

The NURBS basis functions, which are normalized tensor products of the B-spline basis functions, possess similar properties.

2.2 Enriched Field Approximations

Based on the Partition of Unity Finite Element Method (PUFEM) [52], Subbarayan and co-workers [53, 54] developed the Hierarchical Partition of Unity Field Compositions (HPFC) theory where the global domain is a hierarchical composition of local sub-domains or primitives. The geometry, materials and the behavioral fields are constructed in the manner analogous to the Constructive Solid Geometry (CSG) procedure of CAD (see Figure 2.1). The HPFC theory states that at any given point in the domain, the following representation of the composed field is possible:

$$f(\boldsymbol{x}) = \sum_i w_i(\boldsymbol{x}) f_{\Omega_i}(\boldsymbol{x}) \tag{2.7}$$

where, in order to ensure the convergence of analysis, the weight function w_i satisfies the following conditions :

$$\sum_i w_i(\boldsymbol{x}) = 1 \tag{2.8a}$$

$$0 \leq w_i(\boldsymbol{x}) \leq 1 \tag{2.8b}$$

$$\|w_i(\boldsymbol{x})\|_\infty \leq C_\infty \tag{2.8c}$$

$$\|\nabla w_i(\boldsymbol{x})\|_\infty \leq \frac{C_G}{diam\Omega_i}. \tag{2.8d}$$

Tambat and Subbarayan [13] extended the HPFC theory to lower dimensional entities. The behavioral field can be constructed to form a partion of unity as follows:

$$f(\boldsymbol{x}) = \left(1 - \sum_{i=1}^{n_e} w_i\right) f_\Omega(\boldsymbol{x}) + \sum_{i=1}^{n_e} w_i f_{\Gamma_i}(\mathcal{P}(\boldsymbol{x})) \tag{2.9}$$

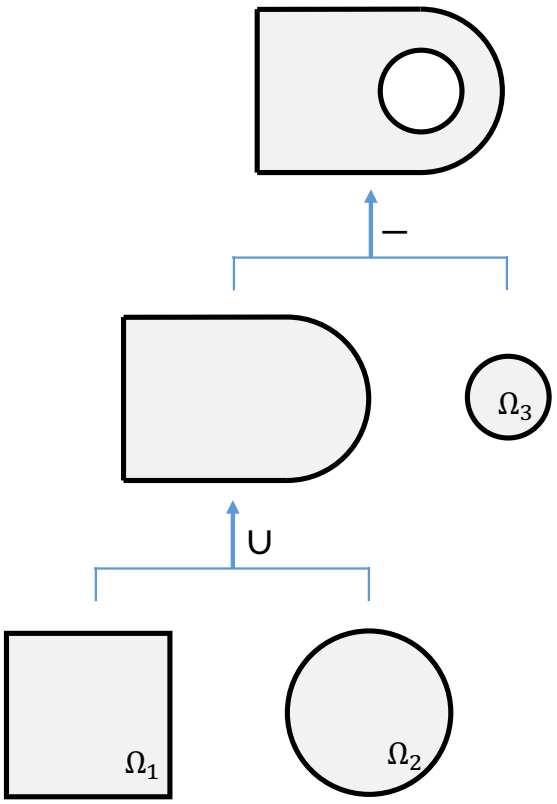


Figure 2.1. Hierarchical compositions of primitive geometrical entities in constructive solid geometry.

where, f_Ω is the continuous approximation associated with the underlying domain Ω and f_{Γ_i} is the enrichment approximation defined isoparametrically on the i^{th} external/internal boundary Γ_i. Not only can the enrichment f_{Γ_i} be a known function to apply boundary/interface conditions, it may also contain unknowns corresponding to the *apriori* knowledge of local behavior. Since each enrichment is defined on a lower-dimensional geometric entity $\boldsymbol{C}(u)$, a projection $u_f = \mathcal{P}(\boldsymbol{x})$ is necessary to map the spatial point $\boldsymbol{x}$ to the parameter u_f of footpoint on $\boldsymbol{C}(u)$. The methodology where the base approximation f_Ω is enriched with lower-dimensional approximations f_{Γ_i}, is termed as *Enriched Isogeometric Analysis* (EIGA). An enriched isogeometric

approximation with a single enrichment is illustrated in Figure 2.2. One approach to constructing the function w_i is to use the normalized inverse distance [55] as follows:

$$w_i(\boldsymbol{x}) = \frac{d_i^{-\mu}(\boldsymbol{x})}{\sum_{j=1}^{n_e} d_j^{-\mu}(\boldsymbol{x})} \tag{2.10}$$

where, $d_j(\boldsymbol{x})$ is the distance from the spatial point $\boldsymbol{x}$ to the boundary Γ_i. The function $w_i(\boldsymbol{x})$ is $\mu-1$ times differentiable at the boundary. Thus, an exponent $\mu > 1$ is chosen to satisfy Eq. (2.8d)

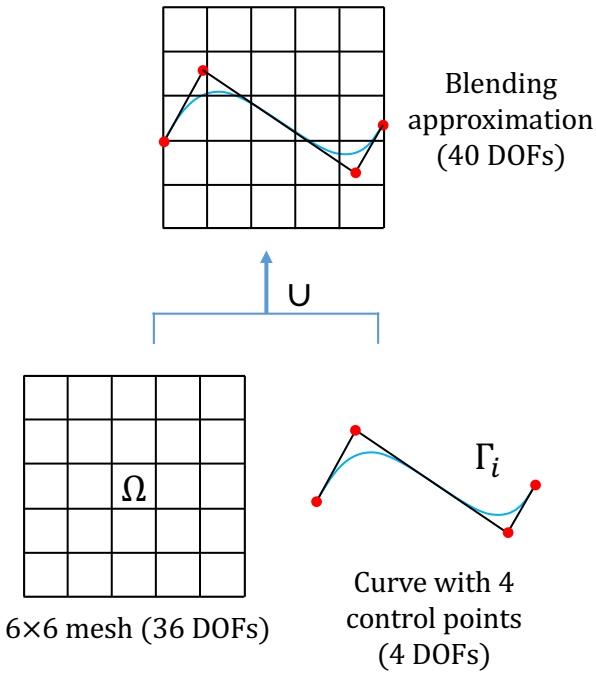


Figure 2.2. Illustration of a two-dimensional enriched isogeometric approximation.

If there is only one enrichment, the Eq. (2.9) can be reduced to:

$$f(\boldsymbol{x}) = (1 - w)\, f_\Omega(\boldsymbol{x}) + w f_\Gamma(\mathcal{P}(\boldsymbol{x})). \tag{2.11}$$

In this case, a natural candidate for w is the exponential function of $d(\boldsymbol{x})$:

$$w = w(d(\boldsymbol{x})) = e^{-\left|\frac{d(\boldsymbol{x})}{d_s}\right|^{\mu}} \tag{2.12}$$

where, d_s is a scaling factor. The weight function with different exponent μ is shown in Figure 2.3. Likewise, the exponent $\mu > 1$ is chosen to assure the differentiability of $w(\boldsymbol{x})$ at the boundary.

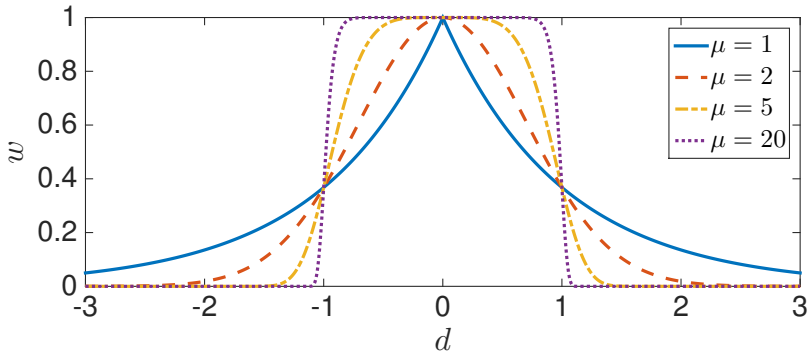


Figure 2.3. The exponential weight function (Eq. (2.12)) with different μ values. The scaling factor is chosen as $d_s = 1$.

3. ALGEBRAIC DISTANCE FIELD

As described by Eqs. (2.10) and (2.12), the construction of the weight functions necessitates a monotonic measure of distance from the boundaries. Since the distance needs to be evaluated at each quadrature point during numerical integration, efficient calculation of distance is critical. In this chapter, an approach relying on algebraic geometry to compute the distance is discussed. This technique possesses several useful properties to facilitate the enriched isogeometric analysis.

3.1 Introduction to Distance Measure

Given a parametric curve or surface entity $\boldsymbol{C}(u) \in \mathbb{R}^n$ (u is treated as a vector when the entity is a surface), the Euclidian distance function $d(\boldsymbol{x})$ is defined as the shortest distance from physical test point $\boldsymbol{x}$ to $\boldsymbol{C}(u)$ given by:

$$d(\boldsymbol{x}) = \inf\|\boldsymbol{x} - \boldsymbol{C}(u)\|. \tag{3.1}$$

The distance function $d(\boldsymbol{x})$ is continuous for all $\boldsymbol{x} \in \mathbb{R}^n$ and differentiable almost everywhere.

A classical approach to find continuously varying distance is the Newton-Raphson scheme [51, 56–58]. This scheme is very sensitive to smoothness and local curvature of the target curve or surface, as well as the initial guess. No only is the iterative scheme computationally expensive, the calculated distance field is often not sufficiently smooth for many engineering applications due to the imperfect geometry. The efficiency can be substantially improved by approximating the geometry with piecewise lines or planes. [59–61]. However, the resulted distance field is only piecewise continuous, and not exact on the boundary. Biswas and Shapiro [62] utilized R-functions [63] to construct a global distance field from the individual distances

to each linear segment, which provides a smooth solution but still compromises the exactness of the boundary.

Upreti et al [49] discussed in detail the efficient computation of algebraic distance estimates from curves and surfaces. The main idea is to implicitize the parametric entity and use the level set of the implicitized function as a measure of distance. Pre-processing by decomposing the NURBS entity into constituent Bezier patches and post-processing by blending using R-functions were utilized to generate the distance fields from the NURBS entities. The algebraic distance field has the following properties:

1. Exact locally near the surface
2. Monotonic function of exact distance
3. Sufficiently smooth for engineering applications
4. Efficiently obtained without numerical iterations

For the sake of completeness, we briefly review the computation of algebraic distance field and illustrate the procedure through simple examples.

3.2 Implicitization of a Parametric Curve

Given a rational parametric curve $\boldsymbol{C}(X(u), Y(u), W(u))$ of degree p with $x = \frac{X(u)}{W(u)}, y = \frac{Y(u)}{W(u)}$, one can construct two auxiliary polynomials:

$$g_1(x, u) = W(u)x - X(u) = 0 \tag{3.2a}$$

$$g_2(y, u) = W(u)y - Y(u) = 0. \tag{3.2b}$$

The above polynomial equations can be rearranged in descending power of u as follows:

$$g_1(u) = a_p u^p + a_{p-1} u^{p-1} + \cdots + a_1 u + a_0 \tag{3.3a}$$

$$g_2(u) = b_p u^p + b_{p-1} u^{p-1} + \cdots + b_1 u + b_0. \tag{3.3b}$$

From the above, the following resultant system may be obtained through algebraic manipulations [64]:

$$\begin{bmatrix} (a_p b_{p-1}) & \cdots & (a_p b_0) \\ \vdots & \ddots & \vdots \\ (a_p b_0) & \cdots & (a_1 b_0) \end{bmatrix} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ 1 \end{pmatrix} = \left[\mathbf{M}^B\right]_{p\times p} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ 1 \end{pmatrix} = 0 \tag{3.4}$$

where, $(a_i b_j) = a_i b_j - a_j b_i$, $\mathbf{M}^B$ is Bezout matrix and is a function of x and y having the following important property:

$$\mathbf{M}^B(x, y) = \mathbf{M}^B_x x + \mathbf{M}^B_y y + \mathbf{M}^B_w \tag{3.5}$$

where, $\mathbf{M}^B_x, \mathbf{M}^B_y$ and $\mathbf{M}^B_w$ depend on control point coordinates and weights. Therefore, these matrices can be pre-computed for a given rational parametric curve and re-used given any new physical point $\boldsymbol{x}$. The determinant, $\det(\mathbf{M}^B(\boldsymbol{x}))$, is defined as the Bezout resultant. Since all the allowable parameter values u for curve $\boldsymbol{C}(X(u), Y(u), W(u))$ are roots of Eq. (3.4), $\det(\mathbf{M}^B(\boldsymbol{x})) = 0$ gives the equation of the implicitized curve. Thus, the algebraic level sets corresponding to a rational parametric curve (e.g., Bezier curve) are given by:

$$\Gamma(\boldsymbol{x}) = \det(\mathbf{M}^B(\boldsymbol{x})). \tag{3.6}$$

An example of algebraic level sets is shown in Figure 3.1.

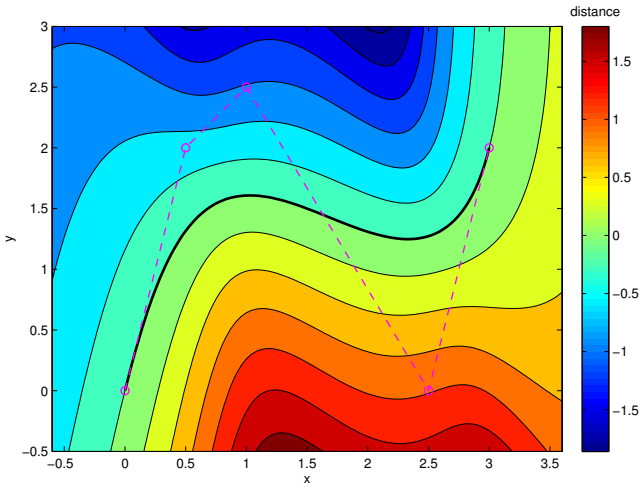


Figure 3.1. Implicitization of a quartic Bezier curve. Level set $\Gamma(\boldsymbol{x}) = \det(\mathbf{M}(\boldsymbol{x}))$ can be used as a measure of distance.

3.2.1 Boolean Operations by R-functions

As observed in Figure 3.1, the direct implicitization extends the parametric curve beyond its end points, and yields an invalid distance measure in the extended region. Therefore, it is desirable to trim the curve $\boldsymbol{C}(X(u), Y(u), W(u))$ within its parameter range $u \in [a, b]$. In related prior work, Biswas and Shapiro [62] constructed an approximate distance from a line segment as:

$$g = \sqrt{\Gamma^2 + \frac{(|\phi| - \phi)^2}{4}} \tag{3.7}$$

with Γ being the normal distance from the line, and ϕ being a set of points that are positive in a region formed by a circle circumscribing the line and negative outside. This form yields a smooth distance function across the boundary $\phi = 0$. Upreti et al. [49] extended the above idea by carrying out boolean operations on fields obtained on (individual segments of) an arbitrarily shaped parametric curve and an enclosing convex region using R-functions (Figure 3.2). The R-functions [63,65] enable a smooth

and purely algebraic boolean operation, and result in a continuous distance measure. Two specific R-functions used in this study are:

1. R-conjunction, equivalent to boolean intersection:

$$g_1 \wedge g_2 = g_1 + g_2 - \sqrt{g_1^2 + g_2^2}. \tag{3.8}$$

2. R-disjunction, equivalent to boolean union:

$$g_1 \vee g_2 = g_1 + g_2 + \sqrt{g_1^2 + g_2^2}. \tag{3.9}$$

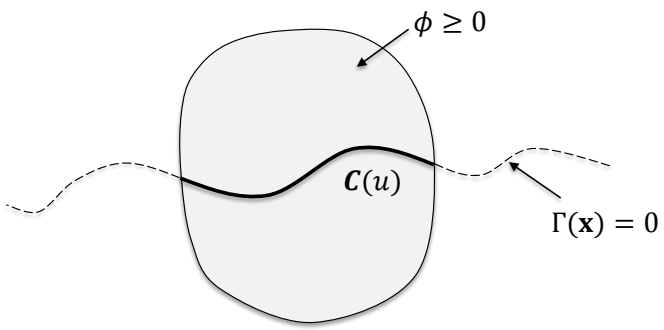


Figure 3.2. A convex region $\phi \geq 0$ is used to trim the implicitized curve $\Gamma(\boldsymbol{x}) = 0$ to a parametric curve $\boldsymbol{C}(u)$ within the allowable parameter range.

Upreti et al. [49] used the convex hull property of Bezier and NURBS curves to provide a natural convex region bounded by control points for curve trimming. Assume that the i-th hyper-plane of the convex hull is expressed as:

$$h_i(\boldsymbol{x}) = \boldsymbol{n}_i \cdot \boldsymbol{x} + b_i = 0 \tag{3.10}$$

where, $\boldsymbol{x} \in \mathbb{R}^n$ is a spatial point, $\boldsymbol{n}_i \in \mathbb{R}^n$ is inner normal and $b_i \in \mathbb{R}$ is offset. Thus, the exact distance from any point $\boldsymbol{x}$ to the hyper-plane $h_i(\boldsymbol{x}) = 0$ is $h_i(\boldsymbol{x})$. The

function ϕ can be obtained by applying R-conjunction operation of Eq. (3.8) to all $h_i(\boldsymbol{x}), i = 1, 2, ..., n$. An example of a cubic Bezier curve is shown in Figure 3.3.

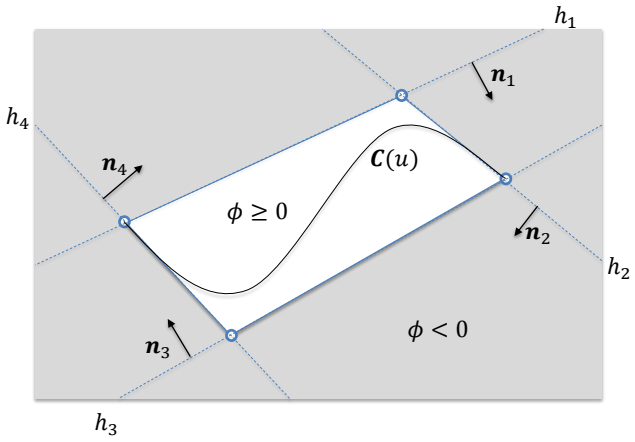


Figure 3.3. Control points of a cubic Bezier curve $\boldsymbol{C}(u)$ forms a convex hull consisting of four hyper-planes h_1, h_2, h_3 and h_4 with inner normals $\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3$ and $\boldsymbol{n}_4$ respectively. Boolean intersection of the four hyper-planes using R-function yields a trimming region $\phi \geq 0$.

3.3 Normalization and Composition of Algebraic Distance Fields

The aforementioned procedure generates a monotonic and continuous distance measure for a basic parametric curve such as a Bezier curve. Piecewise polynomial curves such as NURBS curves, on the other hand, require decomposition to Bezier segments and composition of distance sub-fields of the obtained segments. Further, normalization for each distance sub-field is desired to yield a monotonically varying composed field. Considering a physical footpoint $\boldsymbol{x}_f$, one can approximate $\Gamma(\boldsymbol{x}_f)$ to a first order using Taylor expansion:

$$\Gamma(\boldsymbol{x}_f) = \Gamma(\boldsymbol{x}) - \frac{\partial \Gamma(\boldsymbol{x})}{\partial \boldsymbol{n}} d. \tag{3.11}$$

Since the resultant has exact zero set on a parametric curve, i.e., $\Gamma(\boldsymbol{x}_f) = 0$. one can derive a normalized distance function as follows:

$$d = \frac{\Gamma(\boldsymbol{x})}{\frac{\partial \Gamma(\boldsymbol{x})}{\partial \boldsymbol{n}}} = \frac{\Gamma}{\|\nabla \Gamma\|}. \tag{3.12}$$

After obtaining normalized algebraic distance sub-fields for each decomposed Bezier segment, one can compose them using R-conjunction operation (Eq. (3.8)) and thereby generate the desired algebraic distance field. As demonstrated in [49], the R-conjunction operation preserves the normalization of individual Bezier segments. However, an implicitized curve obtained from a Bezier curve of degree p may have as many as $\frac{1}{2}(p-1)(p-2)$ self-intersections or double points [66]. Any double points inside the convex hull will affect the algebraic distance construction, and therefore need to be moved out by sub-divisions of the Bezier curve. The algorithm to carryout this process is discussed in reference [49]. Thus, for practical reasons of avoiding more than one double point while enabling sufficient generality in modeling complex geometries, the methodology is restricted to low degree NURBS curves ($p \leq 3$). Figure 3.4 shows an example of algebraic distance field of an open curve containing two points with $\mathscr{G}^0$ continuity.

3.4 Extension to NURBS Surface

The algebraic distance field construction can be extended to three-dimensional NURBS surfaces in a straightforward manner by implicitizing the rational parametric surface with the Dixon resultant [64]. Given a rational parametric surface

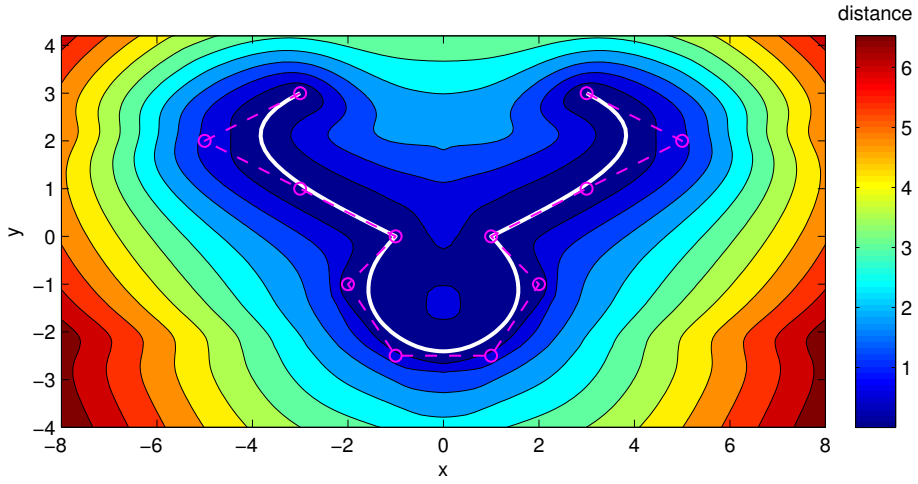


Figure 3.4. Algebraic distance field of a symmetric cubic curve. $\mathscr{G}^0$ continuity is present at $(x, y) = (-1, 0)$ and $(1, 0)$. The generated algebraic level sets retain the symmetry while ensuring the smoothness of the field.

$\boldsymbol{S}(X(u,v), Y(u,v), Z(u,v), W(u,v))$ of degree $p \times q$ with $x = \frac{X(u,v)}{W(u,v)}, y = \frac{Y(u,v)}{W(u,v)}$ and $z = \frac{Z(u,v)}{W(u,v)}$, three auxiliary polynomials can be formed as follows:

$$g_1(x, u, v) = W(u, v)x - X(u, v) = 0 \tag{3.13a}$$

$$g_2(y, u, v) = W(u, v)y - Y(u, v) = 0 \tag{3.13b}$$

$$g_3(z, u, v) = W(u, v)z - X(u, v) = 0. \tag{3.13c}$$

As before, using algebraic elimination theory, one can derive the corresponding resultant system for surface $\boldsymbol{S}$:

$$\left[\mathbf{M}^D\right]_{2pq \times 2pq} \begin{pmatrix} 1 & u & \cdots & u^{p-1} & \cdots & v^{2q-1} & uv^{2q-1} & \cdots & u^{p-1}v^{2q-1} \end{pmatrix}^T = 0 \tag{3.14}$$

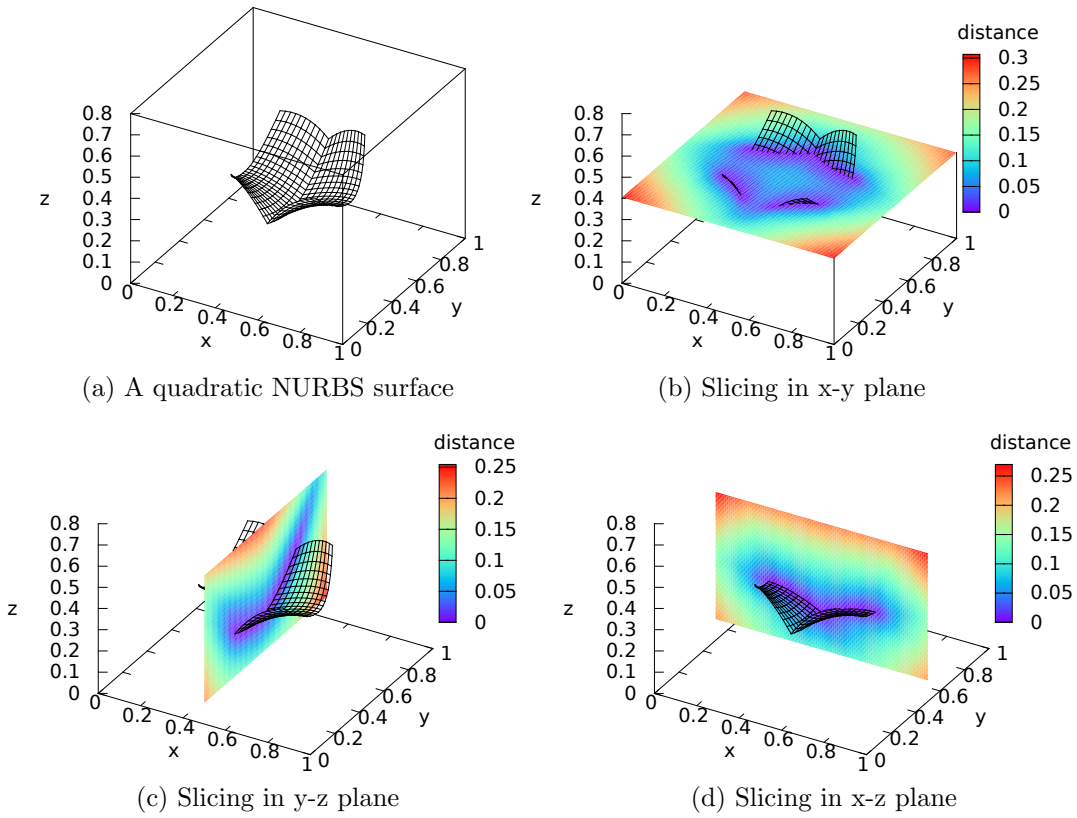
where, the vector is indexed lexicographically. $\mathbf{M}^D$ is the Dixon matrix which also possesses a property analogous to Eq. (3.5) of linearity with respect to x, y, and z:

$$\mathbf{M}^D(\boldsymbol{x}) = \mathbf{M}^D_x x + \mathbf{M}^D_y y + \mathbf{M}^D_z z + \mathbf{M}^D_w \tag{3.15}$$

where, as before, $\mathbf{M}_x^D, \mathbf{M}_y^D, \mathbf{M}_z^D$ and $\mathbf{M}_w^D$ depend on control point coordinates and weights. The determinant of the Dixon matrix is the Dixon resultant:

$$\Gamma(\boldsymbol{x}) = \det(\mathbf{M}^D(\boldsymbol{x})). \tag{3.16}$$

An example of the algebraic distance field from a free surface is illustrated in Figure 3.5. The pseudo-code in Alg. 1 shows the generic steps in algebraic distance computation for NURBS curves and surfaces. Both NURBS curves and surfaces are denoted by $\boldsymbol{C}(\boldsymbol{u})$ here for notational convenience, with the implicit understanding that $\boldsymbol{u} = (u)$ for curves and $\boldsymbol{u} = (u, v)$ for surfaces.



(a) A quadratic NURBS surface

(b) Slicing in x-y plane

(c) Slicing in y-z plane

(d) Slicing in x-z plane

Figure 3.5. Algebraic distance field from a symmetric quadratic NURBS surface. (a) The valley of the surface contains only a $\mathscr{G}^0$ continuity across the plane of symmetry. The distance field is plotted over three principal planes slicing the surface: (b) x-y plane (c) y-z plane and (d) x-z plane.

Algorithm 1 Algebraic Distance Field Algorithm

Input: NURBS curve or surface $\boldsymbol{C}(\boldsymbol{u})$ and given test point $\boldsymbol{x}$
Output: Algebraic distance d from $\boldsymbol{x}$ to the NURBS entity $\boldsymbol{C}(\boldsymbol{u})$

1: **function** ALGEBRAIC_DISTANCE($\boldsymbol{C}$, $\boldsymbol{x}$)
2: $\mathscr{B}(\boldsymbol{C}) \leftarrow$ Split NURBS entity $\boldsymbol{C}$ into a Bezier set with segments $B_i, i = 1, 2, \cdots, n$
3: **for** $i \leftarrow 1, n$ **do** ▷ Loops are independent and parallelizable
4: $h_i \leftarrow$ Create convex hull for $B_i \in \mathscr{B}(\boldsymbol{C})$
5: $d_i \leftarrow$ Carryout boolean union of distance fields of h_i obtained using Eq. (3.7) with B_i
6: **end for**
7: $d \leftarrow$ Carryout boolean intersection of distance sub-fields $d_i, i = 1, 2, \cdots, n$ using Eq. (3.8)
8: **end function**

4. ALGEBRAIC POINT PROJECTION

Point projection is an important need in Computer Aided Design, to estimate the parametric value of the nearest point on a curve or surface from a given spatial location. In the enriched isogeometric analysis, the influence of the enriching geometry to a spatial point can also be determined through point projection. In this chapter, a non-iterative, algebraic technique for point projection is presented for low degree NURBS curves and surfaces. The method is extended from the algebraic distance field, providing exact on-curve/surface solution and accurate near-curve/surface solution. Examples are presented to illustrate the efficiency and robustness of the developed method. The computational expense is demonstrated on the examples to be comparable or lower than that required for a *single* Newton-Raphson iteration. The method is shown to be robust and able to generate valid solutions even for curves and surfaces with high local curvature or $\mathscr{G}_0$ continuity – problems where the Newton-Raphson method fails due to discontinuity in the projected points or because the numerical iterations fail to converge to a solution, respectively.

4.1 Introduction to Point Projection

Given a test point and a parametric entity (curve or surface), the generalized point projection problem is to find the closest point (footpoint) on the entity as well as the corresponding parameter value. Since the footpoint is the closest point on the curve or surface, the line connecting the test point to the footpoint is normal to the curve or the surface [56]:

$$g(u) = \boldsymbol{C}'(u) \cdot (\boldsymbol{x} - \boldsymbol{C}(u)) = 0. \tag{4.1}$$

The footpoint of projection in parametric space u_f can be thus defined as

$$u_f = \mathcal{P}(\boldsymbol{x}) = \underset{\boldsymbol{C}'(u)\cdot(\boldsymbol{C}(u)-\boldsymbol{x})=0}{\arg\min} \|\boldsymbol{x} - \boldsymbol{C}(u)\|, \tag{4.2}$$

This problem is of importance in geometric modeling. For instance, while fitting a curve or surface to sampled data, one may need to compute corresponding parameter values and errors at data points since the error is the distance between the data point and the fitting curve or surface [67].

Point projection also plays an important role in computer aided engineering (CAE). Analysis of interaction with explicitly defined boundaries in a domain requires point projection to compute the influence of the domain point on the appropriate location of the inserted boundary (see Figure 4.1). For example, in solutions to mechanical contact problems [68, 69], point projection is needed to define the normal gap and tangential slip between two bodies. In fluid-structure interaction (FSI) problems, point projection has been utilized to transfer kinematic and traction data between non-matching fluid-structure interface [70]. Recently, point projection has been used to enrich the base approximations with those on lower-dimensional geometrical features such as crack surfaces and phase boundaries, enabling simulations of fracture propagation [13,14] and solidification [71]. A fast and robust point projection method is critical to efficiently solving these problems.

The use of Newton-Raphson (NR) iterations for solving Eq. (4.1) is well established at this time. These iterative methods mainly consist of two steps:

1. Seek an initial point or segment

2. Iterate by Newton-Raphson scheme until convergence

The robustness and the efficiency of Newton-Raphson scheme depends significantly on the initial guess. Therefore, to assure convergence of the second step, careful selection of initial position is needed.

To this end, a significant focus of the existing literature is on eliminating portions of the curve or surface where the solution cannot lie. Piegl and Tiller [67] developed

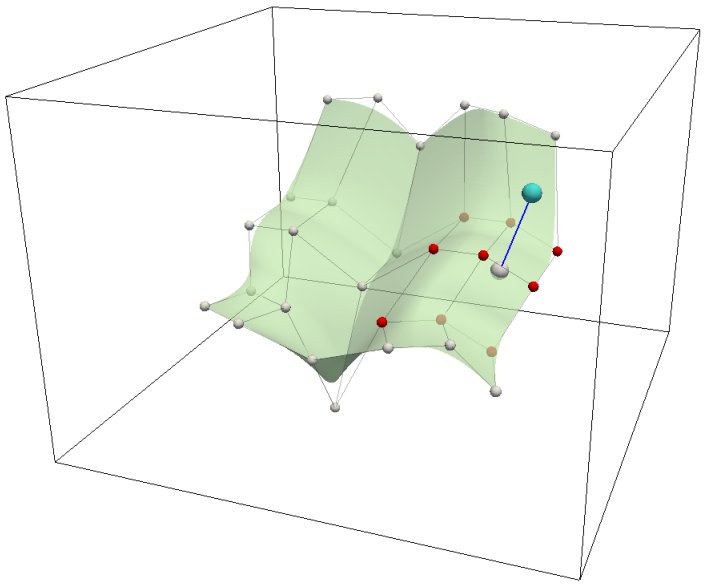
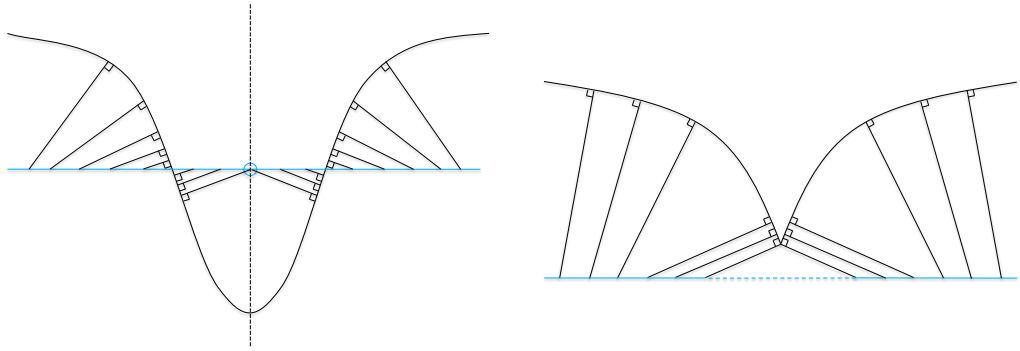


Figure 4.1. Mechanical analysis in the presence of complex free form embedded surface. The spatial point may only influence a local region of the surface with the highlighted control points, which can be identified by point projection.

a non-iterative, heuristic algorithm where a NURBS surface was decomposed into quadrilaterals and test points were projected onto the closest quadrilateral. Ma and Hewitt [57] described a search for the initial guess of the footpoint by recursively sub-dividing the Bezier segment associated with a valid control polygon. However, using the control polygon to eliminate segments of the curve may exclude the correct solution [72]. Selimovic [73] improved Ma and Hewitt's method using selective sub-division of the NURBS curve (surface) based on distance to the end (corner) point of the entity. Chen et al. [74] pointed out the need for all control points to lie on different sides of a hyperplane in Selimovic's algorithm and proposed a circular clipping method with a sufficiency condition for a curve to lie outside an elimination circle. Other iterative methods in the physical space have also been proposed for point projection including the geometric second order iteration method [75, 76].

On the other hand, $u_f(\boldsymbol{x})$ calculated from orthogonal projection (Eq. (4.2)) may be non-unique, discontinuous or non-existent as illustrated in Figure 4.2. The footpoint of a test point near the curve or surface segment with high local curvature can be non-unique, leading to discontinuity of point projection process as illustrated in Figure 4.2a. The non-existence is illustrated in Figure 4.2b, and occurs around points where $\boldsymbol{C}(u)$ has only $\mathscr{G}^0$ continuity.



(a) Illustration of the discontinuity of point projection

(b) Illustration of non-existence of point projection

Figure 4.2. Special cases encountered during point projection: (a) Exact projection from points on a straight line to a bell-shaped curve. Discontinuity occurs at the circled central point due to non-uniqueness of point projection. (b) Projection from points on a straight line to a cone-shaped curve. The dashed line segment has no footpoint on the curve.

To circumvent the challenges, a purely algebraic, and therefore non-iterative, non-recursive and efficient, near-orthogonal point projection technique for low degree two-dimensional (2D) NURBS curve and three-dimensional (3D) NURBS surface is developed. The projection solution is obtained in finite time since the solution does not rely on numerical iteration or recursion. The proposed technique preserves the geometric exactness of NURBS curve and surface, and therefore gives an exact on-curve/surface solution. In addition, the technique is robust for curves and surfaces with high local curvature or $\mathscr{G}_0$ continuity compared to techniques that rely on derivatives.

4.2 Algebraic Point Projection for NURBS Curve

As illustrated using Figure 4.2, iterative numerical solution for point projection may lead to a discontinuity in the projected point or may miss the correct solution. Hence, we develop in this section a purely algebraic point projection algorithm with the following properties:

1. Exact at any point on the curve or surface, i.e., exact point inversion
2. Controllable accuracy when projected from points near the curve or surface
3. Efficient, non-iterative and non-recursive solution procedure
4. Projected points are continuous even near curve segments with high curvature
5. Valid solutions even when projected onto curves with only $\mathscr{G}^0$ continuity

The present method consists of two steps: estimation of the footpoint in physical space and point inversion based on resultant system.

4.2.1 Projection in Physical Space

From Eq. (3.12), the gradient of normalized approximate distance function is derived as

$$\nabla d = \left(\mathbf{I} - \frac{\Gamma}{\|\nabla\Gamma\|^2}\mathbf{H} \right) \frac{\nabla\Gamma}{\|\nabla\Gamma\|} \tag{4.3}$$

where, $\mathbf{I}$ is the identity matrix and $\mathbf{H}$ is the Hessian of function $\Gamma(\boldsymbol{x})$. Using the above distance gradient, the physical footpoint $\boldsymbol{x}_f$ can now be approximately located as

$$\boldsymbol{x}_f = \boldsymbol{x} - d\frac{\nabla d}{\|\nabla d\|}. \tag{4.4}$$

To calculate d and ∇d using Eqs. (3.12) and (4.3), it would appear at first glance as though $\Gamma, \nabla\Gamma$ and $\mathbf{H}$ need to be evaluated for every test point. However, the following derivation as well as procedural detail show that d and ∇d can be computed efficiently without explicitly calculating $\Gamma, \nabla\Gamma$ or $\mathbf{H}$. One can express $\nabla\Gamma$ and $\mathbf{H}$ in terms of Bezout matrix $\mathbf{M}^B$ and its components $\mathbf{M}_x^B$ and $\mathbf{M}_y^B$ (the superscript B is dropped in the following for ease of reading):

$$\nabla\Gamma = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial x}\mathbf{M}\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial y}\mathbf{M}\right) \end{bmatrix} = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \end{bmatrix} = \Gamma\tilde{\boldsymbol{g}} \tag{4.5}$$

$$\mathbf{H} = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_x\right) & \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)^2\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) & \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)^2\right) \end{bmatrix} = \Gamma\tilde{\mathbf{H}}. \tag{4.6}$$

where, $\tilde{\boldsymbol{g}}$ and $\tilde{\mathbf{H}}$ are the vector/matrix multiplying $|\mathbf{M}|$ in the above equations, respectively. Substituting Eqs. (4.5) and (4.6) back into Eqs. (3.12) and (4.3), one obtains

$$d = \frac{1}{\|\tilde{\boldsymbol{g}}\|} \tag{4.7}$$

$$\nabla d = \frac{\tilde{\boldsymbol{g}}}{\|\tilde{\boldsymbol{g}}\|} - \frac{\tilde{\mathbf{H}}\tilde{\boldsymbol{g}}}{\|\tilde{\boldsymbol{g}}\|^3}. \tag{4.8}$$

The efficiency of algebraic point projection in two-dimensional physical space is summarized as follows:

1. Component matrices $\mathbf{M}_x$, $\mathbf{M}_y$ and $\mathbf{M}_w$ are constant for a given rational parametric curve. Therefore, they can be pre-computed and repeatedly used at a point $\boldsymbol{x}$.

2. Only matrix $\mathbf{M}$ needs to be factorized, and the procedure extensively reuses the products $\mathbf{M}^{-1}\mathbf{M}_x$ and $\mathbf{M}^{-1}\mathbf{M}_y$ when computing $\tilde{\mathbf{H}}$ and $\tilde{\boldsymbol{g}}$.

3. For a Bezout matrix $\mathbf{M}$ of the size $p \times p$, where p is the degree of the rational parametric curve, the typical computational cost is low since p is usually small in engineering applications.

Algebraic point projection in two-dimensional physical space is validated in Figure 4.3, where the developed method is compared against the Newton-Raphson iterations for various test distances. The developed method converges to the exact footpoint as the test distance decreases.

4.2.2 Inversion to Parametric Space

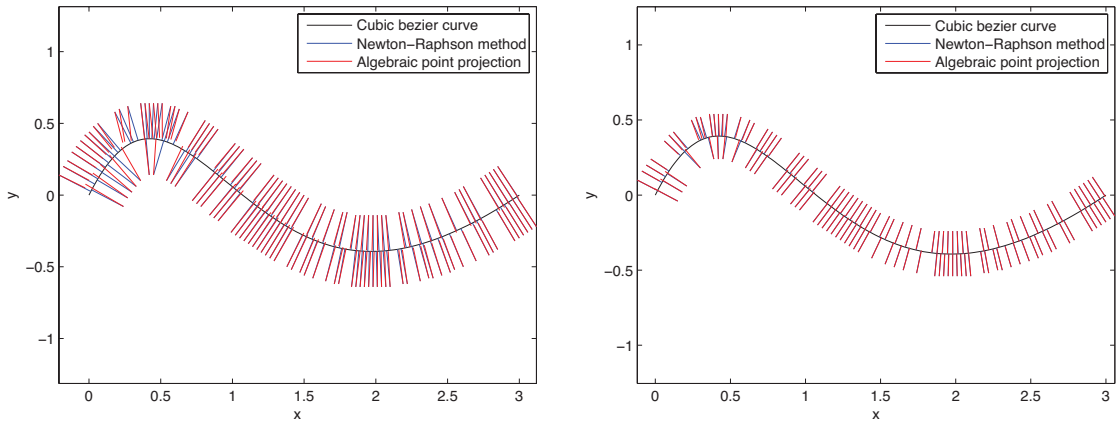
Given a footpoint $\boldsymbol{x}_f$ in physical space, finding a corresponding parameter u_f such that $\boldsymbol{C}(u_f) = \boldsymbol{x}_f$ is the point inversion problem. The direct approach to carrying out point inversion is by solving a system of polynomial equations, which may result in numerical tolerance related challenges, especially when $\boldsymbol{x}_f$ is not exactly on curve $\boldsymbol{C}(u)$ [51]. This drawback can be overcome by using the Bezout matrix [64] as shown in the following. Evaluate $\mathbf{M}^B(\boldsymbol{x}_f)$ with $\boldsymbol{x}_f = (x_f, y_f)$:

$$\mathbf{M}^B(\boldsymbol{x}_f) = \mathbf{M}^B_x x_f + \mathbf{M}^B_y y_f + \mathbf{M}^B_w = [m_{ij}]_{p\times p}. \tag{4.9}$$

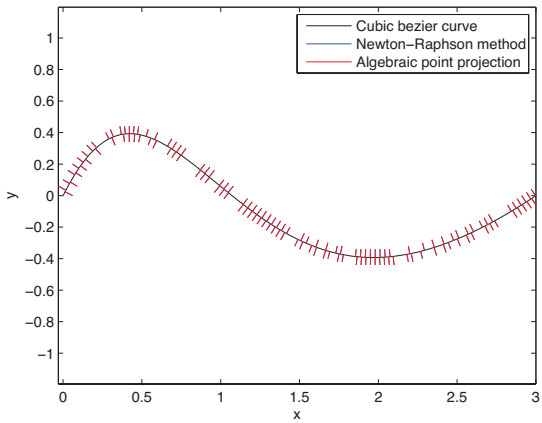
Then, Eq. (3.4) can be rewritten as the following over-constrained linear system:

$$A\tilde{\boldsymbol{u}} \equiv \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1(p-1)} \\ m_{21} & m_{22} & \cdots & m_{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots \quad \vdots \\ m_{p1} & m_{p2} & \cdots & m_{p(p-1)} \end{bmatrix} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ u \end{pmatrix} = - \begin{pmatrix} m_{1p} \\ m_{2p} \\ \vdots \\ m_{pp} \end{pmatrix}. \tag{4.10}$$

Matrix A is full ranked if Eq. (3.3) has only one common root, i.e., if $\boldsymbol{x}_f$ is not a double point [77]. Thus, u_f can be obtained by solving a linear least square problem resulting from Eq. (4.10), which requires bounded computational cost unlike numer-



(a) test distance $\bar{d} = 0.25$ (b) test distance $\bar{d} = 0.15$



(c) test distance $\bar{d} = 0.05$

Figure 4.3. Point projection in two-dimensional physical space using the developed algebraic method as well as Newton-Raphson iterations for (a) test distance $\bar{d} = 0.25$, (b) test distance $\bar{d} = 0.15$ and (c) test distance $\bar{d} = 0.05$

ical iterations using Newton-Raphson method. Also, if a physical test point $\boldsymbol{x}$ is initially on the curve, then $\boldsymbol{x}_f = \boldsymbol{x}$, and the point inversion can be directly applied.

In order to compute u_f on a NURBS curve, which is piece-wise polynomial, projection onto the appropriate Bezier segment is required. For this purpose, one can identify the closest Bezier segment using algebraic distance sub-fields, and apply algebraic point projection on the closest Bezier segment. Denoting the computed

parameter on the closest Bezier curve as u_f^B, the corresponding parameter on the original NURBS curve u_f^N can be obtained by purely linear scaling and offset. Unlike iterative or recursive schemes, the algebraic method guarantees the existence of a definite footpoint without needing to manipulate user-controlled parameters such as stop criterion or recursion limit in an effort to coax a solution. If a test point is close to the connection node of two adjacent Bezier segments, a result of $u_f^B < 0$ or $u_f^B > 1$ may be obtained. In this case, higher projection precision can be achieved when a second point projection to the adjacent Bezier segment is attempted (Figure 4.4).

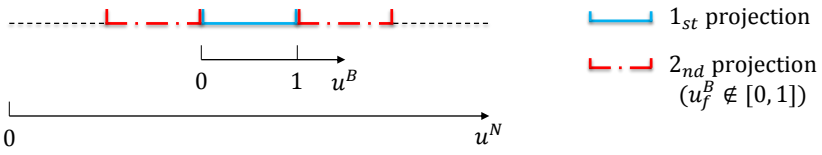


Figure 4.4. Illustration of the second projection onto an adjacent Bezier curve segment if the first projection yields an out-of-span solution.

4.3 Extension to NURBS surfaces

Analogous to the algebraic distance field in three-dimensions, the algebraic point projection can be naturally extended to three-dimensional Bezier and NURBS surfaces by replacing Bezout matrix $\mathbf{M}^B$ with Dixon matrix $\mathbf{M}^D$. Since the Dixon matrix also has the linearity property as given in Eq. (3.15), the basic procedure for point projection remains the same as outlined in Section 4.2.

4.3.1 Projection in Physical Space

Utilizing the linearity property (Eq. (3.15)), one can rewrite $\nabla\Gamma$ and $\mathbf{H}$ in Eq. (4.3) as follows (as before, superscript D is dropped for ease of reading):

$$\nabla\Gamma = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial x}\mathbf{M}\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial y}\mathbf{M}\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial z}\mathbf{M}\right) \end{bmatrix} = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \end{bmatrix} = \Gamma\tilde{\boldsymbol{g}} \tag{4.11}$$

$$\begin{aligned} \mathbf{H} = |\mathbf{M}| & \begin{bmatrix} \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_x\right) & \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) & \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)^2\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) & \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_y\right) & \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)^2\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) & \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) & \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right) & -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)^2\right) \end{bmatrix} \\ = \Gamma\tilde{\mathbf{H}}. \end{aligned} \tag{4.12}$$

Next, as before, Eqs. (4.4), (4.7) and (4.8) can be exploited to obtain the physical footpoint $\boldsymbol{x}_f$ in three-dimensional space. Earlier statements on efficiency of the algebraic point projection for rational parametric curves also apply to rational parametric surfaces except that the size of the Dixon matrix $\mathbf{M}^D$ is $2pq \times 2pq$, where p and q are the degrees of the rational parametric surface in each dimension. Algebraic point projection in three-dimensional physical space is demonstrated in Figure 4.5, where test points are projected onto a target Bezier surface using the proposed method and the Newton-Raphson iterations. Again, one can observe that the proposed method leads to accurate solutions for test points closer to the surface.

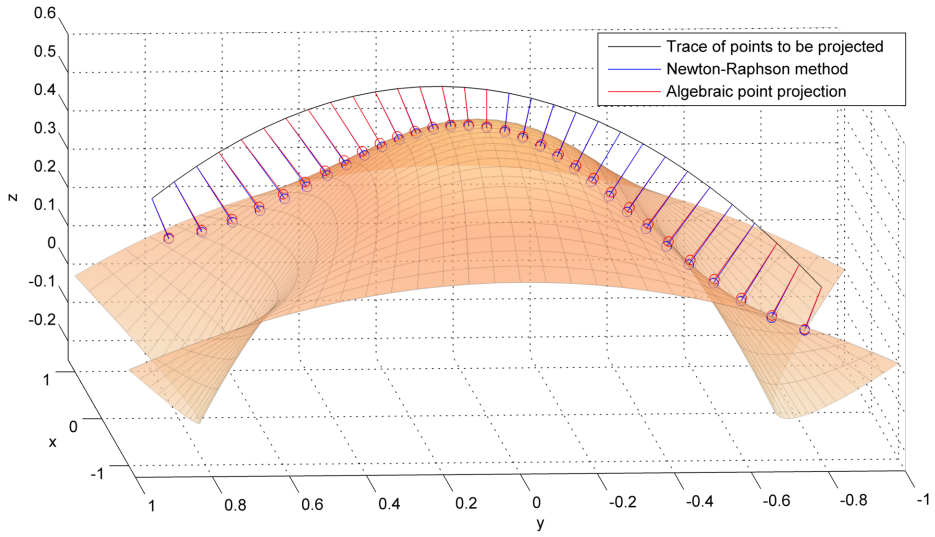


Figure 4.5. Point projection in 3D physical space using the proposed algebraic method and Newton-Raphson iterations.

4.3.2 Inversion to Parametric Space

The point inversion for rational parametric surfaces can be carried out using Dixon matrix as well. Substituting the physical coordinates of the footpoint $\boldsymbol{x}_f = (x_f, y_f, z_f)$ in $\mathbf{M}^D$ we get

$$\mathbf{M}^D(\boldsymbol{x}_f) = \mathbf{M}^D_x x_f + \mathbf{M}^D_y y_f + \mathbf{M}^D_z z_f + \mathbf{M}^D_w = [m_{ij}]_{2pq \times 2pq}. \tag{4.13}$$

Thus, as before, the homogeneous system Eq. (3.14) can be converted into an over-constrained non-homogeneous system as follows:

$$\begin{bmatrix} m_{12} & \cdots & m_{1(1+i+jp)} & \cdots & m_{1(2pq)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{(2pq)2} & \cdots & m_{(2pq)(1+i+jp)} & \cdots & m_{(2pq)(2pq)} \end{bmatrix} \begin{pmatrix} u \\ \vdots \\ u^i v^j \\ \vdots \\ u^{p-1} v^{2q-1} \end{pmatrix} = - \begin{pmatrix} m_{11} \\ \vdots \\ m_{(2pq)1} \end{pmatrix}. \tag{4.14}$$

The parameters (u_f, v_f) of the footpoint $\boldsymbol{x}_f$ can be computed by solving the above non-homogeneous system using QR factorization or singular value decomposition. As before, the computation of parameters (u_f^N, v_f^N) on a NURBS surface requires sub-division of the surface into a set of Bezier segments. One can apply algebraic point projection to the Bezier segment with shortest algebraic distance, and acquire (u_f^N, v_f^N) from the Bezier parameters (u_f^B, v_f^B) by simple linear scaling and offset. As illustrated in Figure 4.6, a second projection may be necessary when u_f^B or v_f^B is outside the range $[0, 1]$.

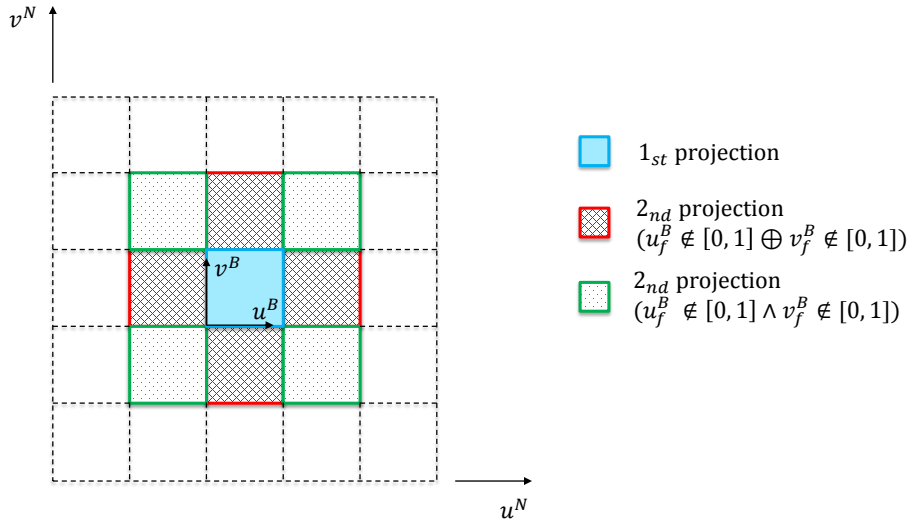


Figure 4.6. Illustration of the second projection onto an adjacent Bezier surface segment if the first projection yields an out-of-span solution.

The generic pseudo-code of algebraic point projection for both NURBS curves and surfaces is listed in Alg. 2. As can be seen from Alg. 1 and Alg. 2, algebraic distance field (ADF) and algebraic point projection (APP) are closely connected. Algebraic distance field provides the closest Bezier segment for the first point projection, but also restricts the target curve or surface to be low degree $(p, q \leq 3)$ so as to avoid double points.

Algorithm 2 Algebraic Point Projection Algorithm

Input: NURBS curve or surface $\boldsymbol{C}(\boldsymbol{u})$ and given point $\boldsymbol{x}$
Output: Parameter $\boldsymbol{u}_f^N$ of footpoint on NURBS entity $\boldsymbol{C}$.

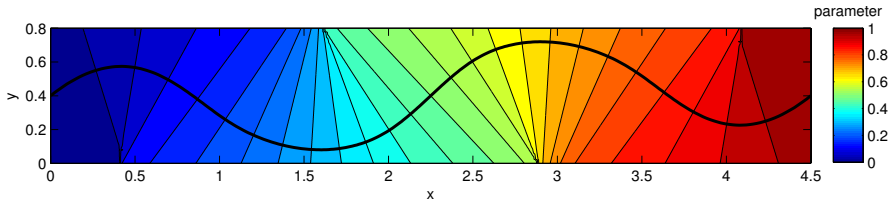
function ALGEBRAIC_POINT_PROJECTION($\boldsymbol{C}$, $\boldsymbol{x}$)
  $\mathscr{B}(\boldsymbol{C}) \leftarrow$ Split NURBS entity $\boldsymbol{C}$ into a Bezier set with segments $B_i, i = 1, 2, \cdots, n$
  for $i \leftarrow 1, n$ **do** ▷ Loops are independent, parallelization is possible
    $h_i \leftarrow$ Initialize convex hull for $B_i \in \mathscr{B}(\boldsymbol{C})$
    $d_i \leftarrow$ Carryout boolean union of distance fields of h_i and B_i by Eq. (3.7)
  end for
  $\boldsymbol{x}_f \leftarrow \boldsymbol{x} - d_{B_j}(\boldsymbol{x}) \frac{\nabla d_{B_j}(\boldsymbol{x})}{\left\|\nabla d_{B_j}(\boldsymbol{x})\right\|}$, where $j = \arg\min_{i \in [1,n]} d_i$
  $\boldsymbol{u}_f^B \leftarrow$ Solve Eq. (4.10) with $\mathbf{M}^B(\boldsymbol{x}_f)$ or Eq. (4.14) with $\mathbf{M}^D(\boldsymbol{x}_f)$
  if $\boldsymbol{u}_f^B$ is out of span **then**
    $\boldsymbol{u}_f^B \leftarrow$ Carryout the second projection based on Figure 4.4 or Figure 4.6
    if Second projection is infeasible or $\boldsymbol{u}_f^B$ is still out of span **then**
      $\boldsymbol{u}_f^B \leftarrow$ Compute corresponding parameter value on B_j boundary
    end if
  end if
  $\boldsymbol{u}_f^N \leftarrow$ Scale and offset $\boldsymbol{u}_f^B$ based on knot span of $\boldsymbol{C}$
end function

4.4 Numerical Examples of Algebraic Point Projection

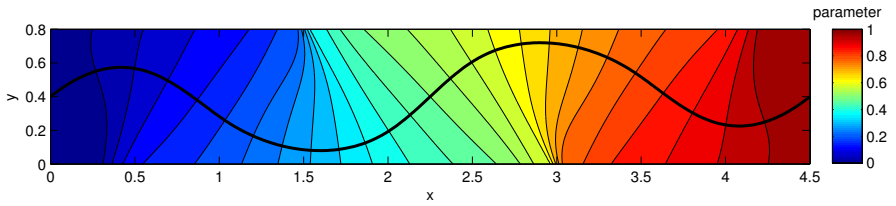
Four numerical examples are presented to demonstrate the algebraic point projection: two curve examples and two surface examples. The first curve example shows the performance of algebraic point projection in a simple test to estimate computational accuracy. In this example, it is shown that a quadratic convergence rate is achieved. The second curve example and both surface examples show the performance of the proposed method in complex tests involving discontinuous and/or non-existent footpoints. The computational efficiency of the developed method is demonstrated through a comparison with the Newton-Raphson iterations in all examples.

4.4.1 Curve Tests

The first curve example is illustrated in Figure 4.7, where physical points in the underlying domain are projected onto a given NURBS curve using both Newton-Raphson iterations as well as algebraic point projection. Contour levels indicate the value of parameter u_f^N of the predicted footpoint.



(a) Parameter values of the footpoints obtained through Newton-Raphson iterations



(b) Parameter values of the footpoints obtained through algebraic point projection

Figure 4.7. Parameter values of footpoints obtained using (a) Newton-Raphson method and (b) algebraic point projection. Parameter range of NURBS curve is $[0, 1]$.

As may be observed from Figure 4.7, the algebraic point projection provides an exact on-curve solution and a good approximate solution to the parameter values from points near the curve. A quadratic convergence rate was achieved as the distance between test points and curve decreased (Figure 4.8). Also, one may observe that there are two regions, at bottom-left and upper-right of Figure 4.7a respectively, where due to high curvature of nearby curve segments, a jump in parameter value occurs. Such discontinuities disappear when algebraic point projection is used. The computational cost per point of algebraic point projection, as listed in Table 4.1, is only 21% of that of the Newton-Raphson method. In fact, the computational cost of

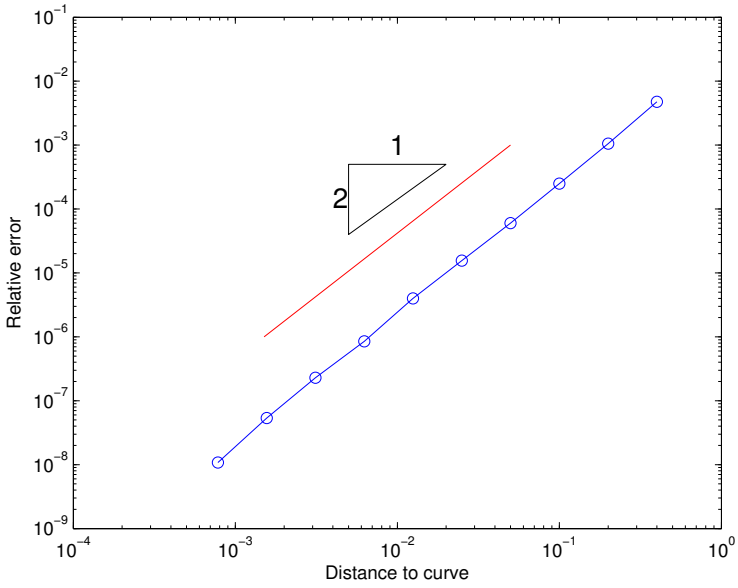


Figure 4.8. Relative error $\frac{|u_f^{NR} - u_f^{APP}|}{b-a}$ vs. normalized distance of test points $d(\boldsymbol{x})$, where u_f^{NR} and u_f^{APP} are parameter values of footpoints obtained using the Newton-Raphson method and the algebraic point projection respectively. $[a, b]$ is the parameter range of the NURBS curve. As the test point moves closer to the target curve, the projection error decreases quadratically.

the proposed method is comparable to or lower than the average cost per iteration when using the Newton-Raphson scheme.

Table 4.1.
The results of point projection for NURBS curves. The tolerance in Newton-Raphson iterations was chosen as $\epsilon = 10^{-8}$. The time required to find an initial guess for the Newton-Raphson method was excluded in the time per iteration calculation.

Example	**Newton-Raphson Iterations**			**Proposed Method**
Curve	Time per point (μs)	Number of iterations	Time per iteration (μs)	Time per point (μs)
#1 (Figure 4.7)	33.86	3.93	7.12	7.14
#2 (Figure 4.9)	251.6	15.03	16.74	8.11

The second curve example shown in Figure 4.9 demonstrates the robustness of the algebraic point projection when the footpoint is either discontinuous or non-existent. One can observe from Figure 4.9b that at the discontinuous point, the Newton-Raphson method causes a large jump in parameter value, whereas the algebraic method yields a continuous parameter value. In general, the projected solution is smoother (fewer oscillations) and matches the Newton-Raphson solution well. Finally, since the Newton-Raphson iterations failed because maximum allowed iterations were reached for $u_{trace} \in [0.53, 0.77]$, in this example, the computational cost of the algebraic method (see Table 4.1) was only 3% of the cost of the Newton-Raphson method, and significantly smaller than that required for a single Newton-Raphson iteration.

4.4.2 Surface Tests

The first and second surface examples demonstrate the robustness of the algebraic point projection for NURBS surfaces involving discontinuous and non-existent footpoints, respectively. In the first example (Figure 4.10), the discontinuous projection occurs again when the Newton-Raphson method is applied on a surface segment with high mean curvature. In the second example (Figure 4.11), the Newton-Raphson method failed in the regions where the mathematical footpoints do not exist. Not only does the algebraic point projection overcome those problems, but it also produces an accurate and efficient solution. According to Table 4.2, the computational cost per point of the proposed method is only 22% and 8.9% of that of the Newton-Raphson method in the first and second surface examples, respectively.

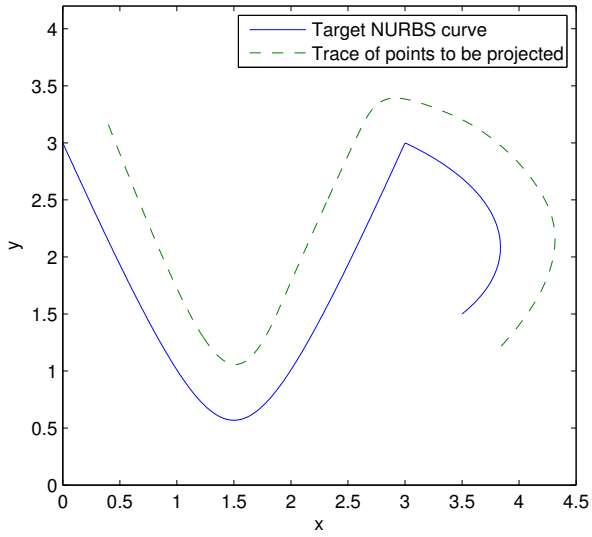
4.5 Concluding Remarks

In the present study, a novel algebraic projection method for low degree two-dimensional NURBS curves and three-dimensional NURBS surfaces was proposed. The procedure utilizes the recently developed algebraic distance field construction

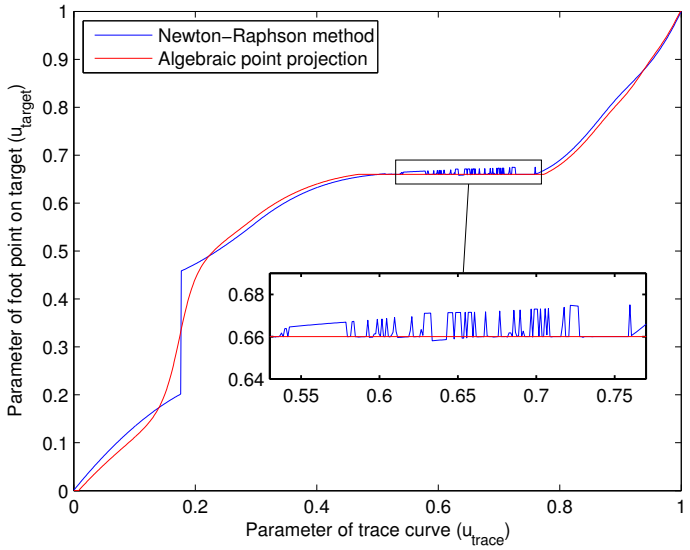
Table 4.2.
The results of point projection for NURBS surfaces. The tolerance in Newton-Raphson iterations was chosen as $\epsilon = 10^{-8}$. Note that the time of finding an initial point is excluded in the time per iteration.

Example	**Newton-Raphson Iterations**			**Proposed Method**
Surface	Time per point (μs)	Number of iterations	Time per iteration (μs)	Time per point (μs)
#1 (Figure 4.10)	94.74	5.00	18.33	20.89
#2 (Figure 4.11)	328.8	11.26	27.89	29.25

methodology. As a first step, the procedure exploits the differential property of the resultant matrix to obtain the footpoint in physical space. Next, the parameter value of the footpoint is computed by solving the over-constrained resultant system. Algebraic point projection eliminates inefficient iterative/recursive computations and challenging search for an initial guess by providing an exact on-curve solution and good approximate solution for points near the curve. Through numerical examples, the algebraic method is demonstrated to be faster and more robust than Newton-Raphson iterations. The computational cost of the developed method is comparable or lower than that required for a single Newton-Raphson iteration. While the algebraic point projection has many advantages, for points far away from the target NURBS entity, the estimated footpoint may not be accurate. For at least isogeometric analysis using explicitly defined surfaces [13, 71], inaccuracy in estimating footpoints from Gauss points far from the entity is less important relative to robustness of solution and the overall computational cost of projection from a very large number of quadrature points.

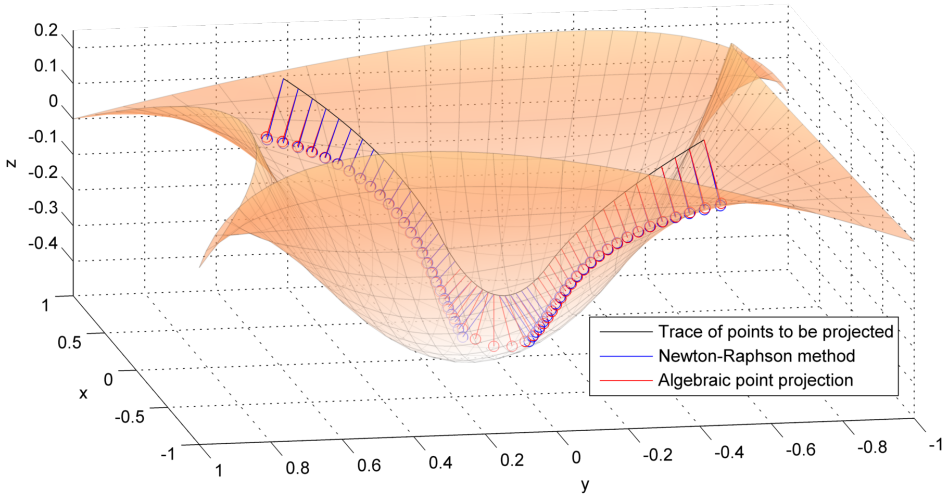


(a) Trace of test points and target curve. The trace was roughly offset relative to the target.

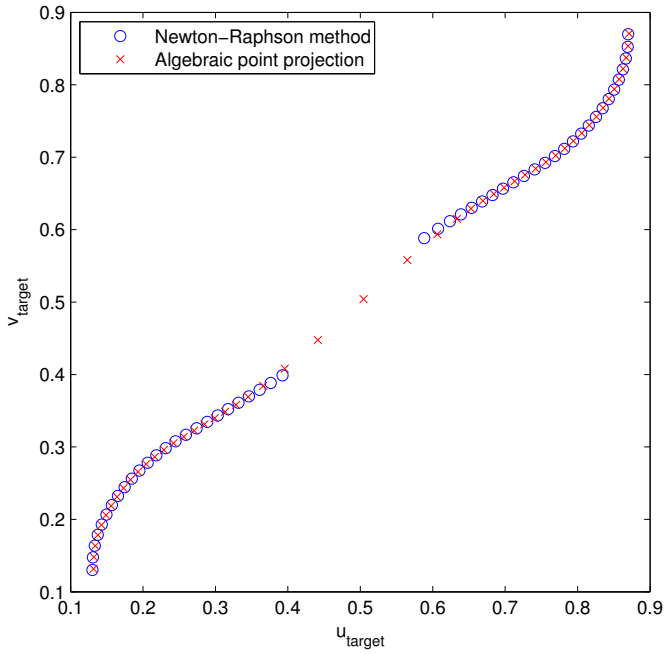


(b) Parameter values of footpoints vs. those of the trace curve.

Figure 4.9. Illustration of the robustness of the 2D algebraic point projection for the NURBS curve. (a) Trace of points that were projected onto target curve. (b) Solution parameter of footpoints on target curve vs parameter of trace curve for the two methods. Parameter discontinuity in Newton-Raphson solution occurs due to non-uniqueness of the footpoint near the local minimum at $u_{trace} \approx 0.176$, and the solution does not exist when $u_{trace} \in [0.53, 0.77]$ as shown in the inset magnified region.

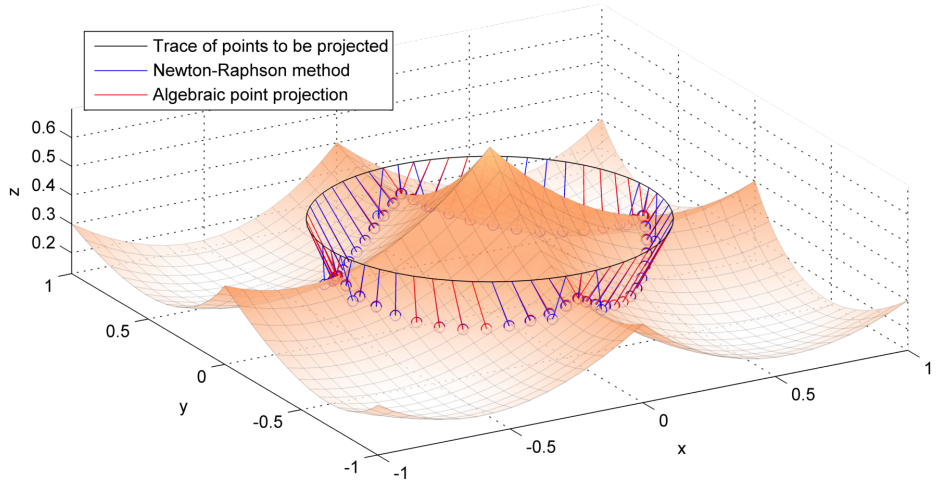


(a) Trace of test points, bowl-shaped target surface and the identified footpoints.

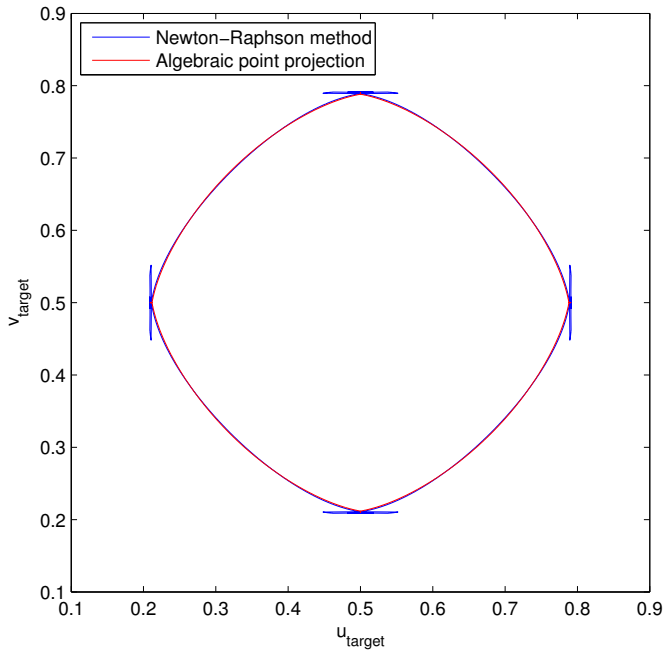


(b) Parameter values (u_{target}, v_{target}) of footpoints shown in (a).

Figure 4.10. Illustration of the robustness of the 3D algebraic point projection algorithm involving discontinuous footpoints. (a) Trace of points that were projected onto bowl-shaped target surface using the proposed algebraic method and the Netwon-Raphson method. (b) Parameters of footpoints on the target obtained by both the methods. Discontinuity occurs due to non-unique footpoints for test points near the bottom of the surface.



(a) Trace of test points, mountain-shaped target surface and consequent footpoints.



(b) Parameter values (u_{target}, v_{target}) of footpoints.

Figure 4.11. Illustration of the robustness of the 3D algebraic point projection algorithm involving test points whose mathematical footpoints do not exist. (a) Trace of points which are projected onto mountain-shaped target surface using both methods. (b) Parameters of footpoints on target. The solution does not exist near the four mountain ridges of $\mathscr{G}^0$ continuity as shown in the four corner regions of (b).

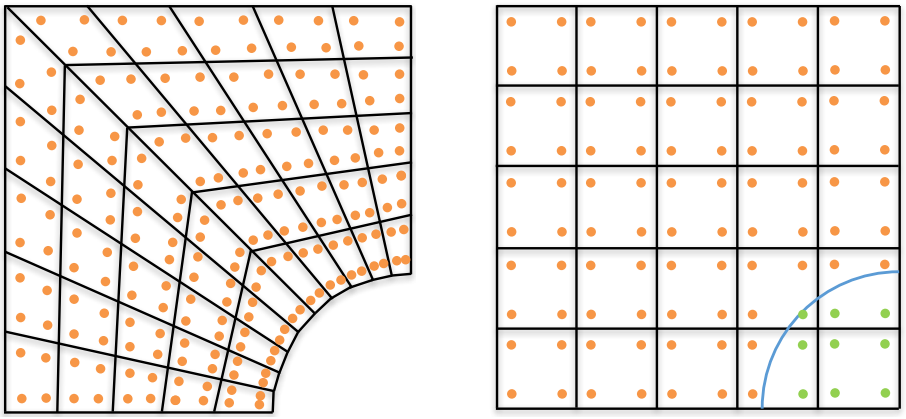
5. EFFICIENT ALGORITHMS FOR IMMERSED BOUNDARY PROBLEMS. PART I: KD-TREE BASED ADAPTIVE QUADRATURE

In the enriched isogeometric analysis, the underlying domain mesh does not necessarily conform to the boundaries. While the non-conforming mesh provides great flexibility and convenience for mesh generation, this in turn requires a sophisticated scheme to integrate over the elements that are intersected by the boundaries. In this chapter, a novel kd-tree based adaptive quadrature algorithm is proposed to carry out the numerical integration in the cut elements/cells. This technique is demonstrated to be cheaper as well as producing fewer quadrature sub-cells than classical quad-/oct-tree based schemes.

5.1 Introduction to Adaptive Quadrature

The modeling of immersed boundary problems, using a mesh that does not necessarily fit to the boundaries, has been extensively studied in the computational fluid dynamics (CFD) community [78–81]. Many immersed boundary methods based on the finite element approximation were proposed to solve the fluid-structure interaction problems [82–86]. The generation of the non-conforming mesh is, in general, much faster than that of a boundary-fitted mesh. However, in the former case, many elements are intersected by the boundaries, and therefore the traditional element-wise numerical integration is no longer feasible for these elements. As illustrated in Figure 5.1, the immersed boundary problems necessitate a separate integration on each side of the boundary, but an accurate integration over such cut elements is challenging.

To this end, many subdivision based adaptive quadrature algorithms were developed to generate boundary-fitted quadrature sub-cells in the cut elements. Rüberg



(a) Boundary-fitted mesh

(b) Non-conforming mesh in the presence of an immersed boundary

Figure 5.1. Different types of analysis mesh: (a) The mesh conforming to the boundary, and (b) the mesh grid intersected by a immersed boundary.

and Cirak [85] planarized the immersed surfaces and then subdivided the hexahedral elements into boundary-fitted tetrahedral sub-cells. Chen and Fries [87] approximated the boundary segment within each cut element with a polynomial, followed by a subdivision of the cut element into a few serendipity elements. Kudela et al. [88] developed a directly integratable blending formulation for the curved triangles or quadrilaterals resulting from the cut elements. While the curved-element methods yield very few quadrature sub-cells and preserve the geometric accuracy, a robust implementation accounting for all the special cases is still non-trivial. Furthermore, these methods can not be extended to three-dimensional (3D) problems.

Another popular subdivision scheme relies on the space tree, i.e., quad-tree (2D) and oct-tree (3D). This scheme works robustly for any geometry without complicated case-by-case solution, and has been extensively applied to quadrilateral and hexahedral elements, such as the patches in isogeometric analysis [13, 53, 89–91]. Recently, the space tree based subdivision scheme was employed to facilitate the numerical integration in finite cell method (FCM) [92–94]. However, the classical quad-/oct-tree subdivision introduces superfluous quadrature cells and significantly increases the

computational time during matrix assembly. A quad-tree based adaptive quadrature example is shown in Figure 5.2, where the total number of quadrature cells increases 276% after a three-level subdivision. The computational performance is even worse if more elements are intersected by the immersed boundaries, or if a higher level of subdivision is desired.

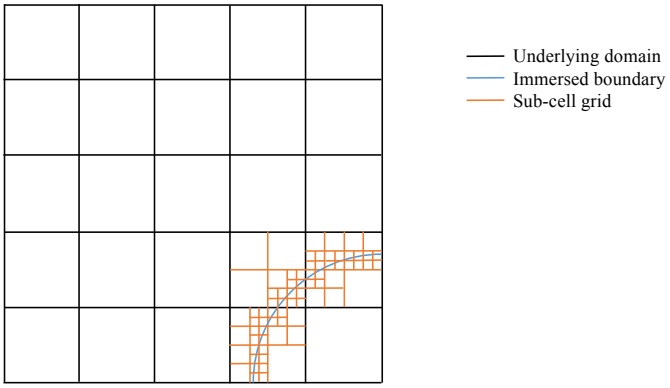


Figure 5.2. Illustration of the quad-tree based adaptive quadrature in the presence of a circular boundary. The initial number of quadrature cells is 25. After a three-level quad-tree subdivision, the number increases to 94.

Recently, several non-subdivision approaches, including the adaptive weight method [95] and the variational collocation method [96, 97], were proposed. The former approach yields much fewer Gauss points than the subdivision based methods, but require handling of a large number of special cases. The variational collocation method produces a Galerkin solution by carefully choosing the collocation points (termed as *Cauchy-Galerkin points*), and thus avoids explicit numerical integration. However, analytical solution for the Cauchy-Galerkin points only exists for uniform meshes and non-singular problems.

5.2 Kd-tree Based Adaptive Quadrature Algorithm

The existing adaptive quadrature schemes either introduce unnecessary Gauss points or require special rules or evaluation points. To circumvent the two challenges, a novel adaptive quadrature algorithm based on the kd-tree data structure, is developed.

5.2.1 Motivation

The magnified bottom right 2×2 elements in Figure 5.2 is shown in Figure 5.3. It can be observed that many adjacent, non-intersected quad-tree sub-cells (separated by dash lines) can be combined into one cell. By removing these dash lines, the number of sub-cells is reduced without loss of quadrature accuracy. Thus, for certain cells, a complete subdivision into four (2D) or eight (3D) sub-cells is not necessary. A partial subdivision can be carried out by either splitting the cells direction-wise or combining the sub-cells during post-processing. As shown in Figure 5.3, the total number of quadrature sub-cells decreases by 17% with partial subdivisions. This method is analogous to the kd-tree data structure in computer geometry, and is therefore termed as *Kd-Tree based Adaptive Quadrature* (KTAQ).

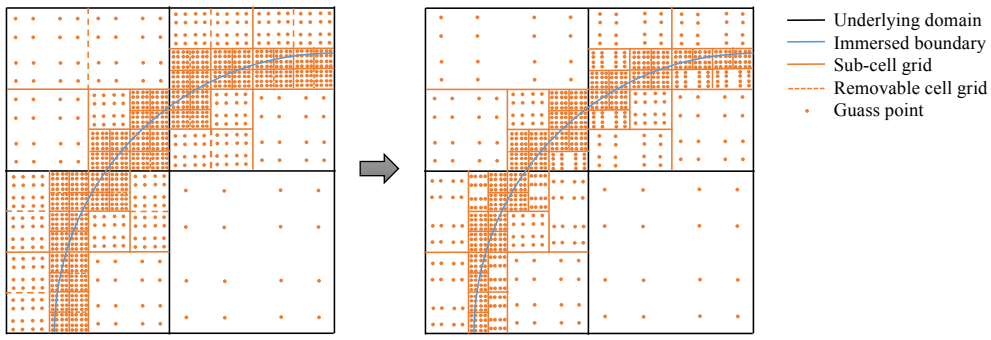


Figure 5.3. Illustration of the sub-cell coalescence by removing non-intersected cell grids.

5.2.2 Algorithm

k-dimensional tree, also referred as *kd-tree*, is a binary space tree that partitions the space dimension-wise, and is also a superset of the quad-tree and oct-tree. The kd-tree structure has a long tradition in computational geometry [98] to solve range searching problems. Specifically, given a set of spatial points, the kd-tree can store the points in an orderly manner so that quick range searching can be achieved. In this work, the kd-tree is used to store the quadrature sub-cells instead of spatial points. A two-dimensional example is shown in Figure 5.4.

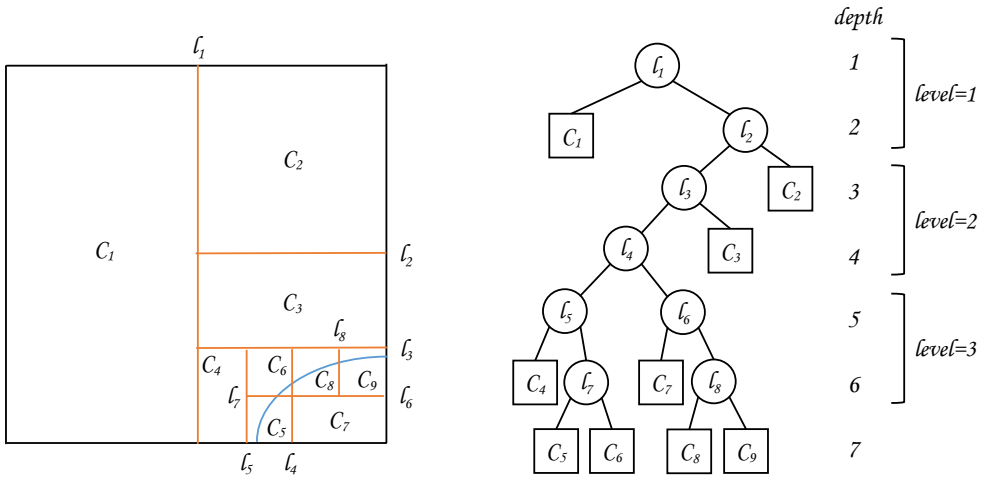


Figure 5.4. Illustration of the kd-tree based adaptive quadrature. The maximum level illustrated here is three. Each level consists of two depths that represent different splitting directions. l_i (stored in nodes) and C_i (stored in leaves) denote a splitting line and a sub-cell, respectively.

In the standard kd-tree, each level has k depths, i.e., k times of splitting (e.g., 2d-tree splits in x and y directions successively whereas 3d-tree splits in x, y and z directions). As for the kd-tree based adaptive quadrature, extra flexibility is provided to the splitting procedure by enabling an arbitrary order of the splitting direction within each level. First, we explore all the possible splitting directions at each depth, and choose the direction in which the splitting line/plane l does not intersect with the

boundary. If the intersection can not be avoided in any direction, a default splitting direction will be chosen. Each split produces two new sub-cells. If any of the sub-cells is not intersected by the boundary, a corresponding leaf will be created in the kd-tree. Otherwise, the split continues until the maximum level is reached. A pseudocode of the proposed algorithm is described in Alg. 3. The KTAQ is only necessary for cut elements. The regular elements can handled with the standard Gaussian quadrature.

Remarks:

1. Once the kd-tree is constructed for a given cut element, a tree traversal algorithm such as Depth First Search (DFS) or Breadth First Search (BFS) can be employed to extract all the leaves of the kd-tree and obtain the quadrature sub-cells stored therein. The tree traversal step is, in general, much faster than the tree construction.

2. The intersection between the splitting line/plane and the boundary can be identified by checking the signed algebraic distance [99] between them. A line or plane is assumed to intersect the boundary if the bounding vertices of the line/plane have opposite signs. The algebraic distance field is exact to the NURBS boundary, enabling an accurate judgment of the intersection.

5.2.3 Comparison with Quad-/Oct-tree

The kd-tree structure is compared with quad-tree and oct-tree in the following three aspects:

1. *Number of Quadrature Sub-cells Generated*: Table 5.1 summarizes the worst- and best-case ratios of the number of sub-cells generated by kd-tree to that by quad-/oct-tree. In the worst-case scenario, the immersed boundary is sufficiently complex such that all the cells have to be completely subdivided in each level. Therefore, the number of sub-cells generated by the kd-tree subdivision

Algorithm 3 Kd-tree Based Adaptive Quadrature Algorithm

Input: Cell C, depth of tree *depth*, splitting direction vector *dir* and auxiliary flag *nocheck*

Output: Kd-tree v containing quadrature sub-cells in leaves

```
1: function BuildKdTree(C, depth, dirs, nocheck)
2:     if depth mod Dim = 0 then        ▷ Reinitialize at the first depth of each level
3:         dir ← [x, y](2D) or [x, y, z](3D)
4:         nocheck ← false
5:     end if
6:     SplitDir ← dir(1)        ▷ Set the first element in dir as the default splitting direction
7:     if nocheck = false then
8:         if C not intersect boundary or depth = MaxDepth then
9:             return leaf containing C
10:        else if length(dir) ≥ 2 then        ▷ No further check is needed if only one direction is available
11:            for i ← 1, length(dir) do
12:                nocheck ← true
13:                if Splitting line(2D)/plane(3D) in dir(i) not intersect boundary then
14:                    nocheck ← false
15:                    SplitDir ← dir(i)
16:                end if
17:            end for
18:        end if
19:    end if
20:    C'_half, C''_half ← Split C in SplitDir
21:    dir_new ← Exclude SplitDir from dir
22:    v_left ← BuildKdTree(C'_half, depth+1, dir_new, nocheck)
23:    v_left ← BuildKdTree(C''_half, depth+1, dir_new, nocheck)
24:    return v with two branches v_left and v_right
25: end function
```

is the same as that by quad-tree and oct-tree. The best-case ratio is proved in Theorem 5.2.1.

Theorem 5.2.1 *Given an arbitrary immersed boundary, the number of quadrature sub-cells generated by the kd-tree subdivision is no smaller than 2/3 of that by a quad-tree, and 3/7 of that by an oct-tree.*

Table 5.1.
Worst- and best-case ratios of the number of sub-cells generated by kd-tree to that by quad-tree and oct-tree.

Problem Dimension	**Worst Case**	**Best Case**
2D ($N_{\mathrm{2d-tree}}/N_{\mathrm{quad-tree}}$)	1	2/3
3D ($N_{\mathrm{3d-tree}}/N_{\mathrm{oct-tree}}$)	1	3/7

Proof The minimum kd-tree splits of a 2D as well as a 3D cell are illustrated in Figure 5.5. A complete, single-level 2D subdivision operation increases the number of sub-cells by $4 - 1 = 3$ whereas an incomplete, single-level 2D subdivision only increases it by $3 - 1 = 2$. Assume the total number of complete and incomplete subdivision operations is given by S_c and S_i, respectively. The total number of sub-cells in a 2d-tree is given by

$$N_{\mathrm{2d-tree}} = 1 + 3S_c + 2S_i. \tag{5.1}$$

Next, assume the total number of subdivision is S_{total}. If $S_c = S_{\mathrm{total}}$ and $S_i = 0$, the kd-tree is degenerated to a quad-tree, i.e.,

$$N_{\mathrm{quad-tree}} = 1 + 3S_{\mathrm{total}}. \tag{5.2}$$

Given an arbitrary immersed boundary, the $N_{2d-tree}/N_{quad-tree}$ ratio is given by

$$\frac{N_{\mathrm{2d-tree}}}{N_{\mathrm{quad-tree}}} = \frac{1 + 3(S_{\mathrm{total}} - S_i) + 2S_i}{1 + 3S_{\mathrm{total}}} \geq \frac{1 + 2S_{\mathrm{total}}}{1 + 3S_{\mathrm{total}}} > \frac{2}{3}. \tag{5.3}$$

Likewise, a complete, single-level 3D subdivision operation adds $8 - 1 = 7$ sub-cells. However, the number of sub-cells added by an incomplete 3D subdivision may vary from 3 to 6. If a real number $c \in [3, 6]$ is used to represent the average

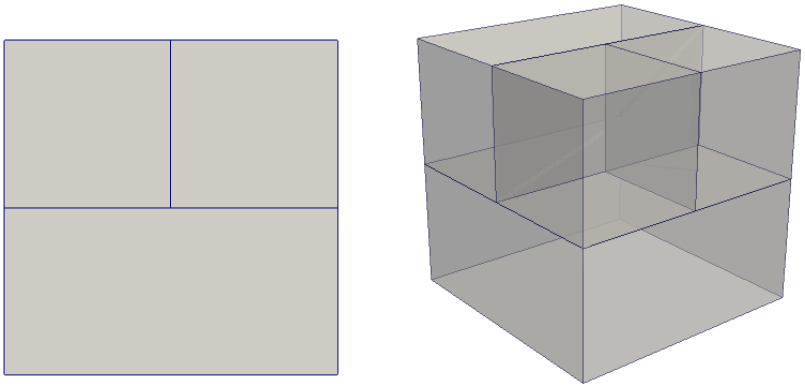
sub-cell increment per subdivision, the total number of sub-cells in a 3d-tree can be expressed as

$$N_{\text{3d-tree}} = 1 + 7S_c + cS_i. \tag{5.4}$$

The $N_{\text{3d-tree}}/N_{\text{oct-tree}}$ ratio can be next obtained as follows:

$$\frac{N_{\text{3d-tree}}}{N_{\text{oct-tree}}} = \frac{1 + 7(S_{total} - S_i) + cS_i}{1 + 7S_{total}} \geq \frac{1 + cS_{\text{total}}}{1 + 7S_{\text{total}}} > \frac{c}{7} \geq \frac{3}{7}. \tag{5.5}$$

■



(a) Minimum 2d-tree splits. (b) Minimum 3d-tree splits.

Figure 5.5. Minimum kd-tree splits within a single level. (a) The 2d-tree and (b) 3d-tree splits produce three and four sub-cells, respectively.

2. *Computational Cost in Tree Construction*: Given a quadrature cell to be subdivided, the vertices that need to be checked in the kd-tree are a subset of those in quad-/oct-tree. Furthermore, the kd-tree yields fewer number of sub-cells in each level (see Figure 5.5). Therefore, the kd-tree results in a faster algorithm than quad-/oct-tree.

3. *Aspect Ratio of Generated Sub-cells*: Table 5.2 lists the worst- and best-case aspect ratios of the sub-cells generated by quad-, oct- and kd-tree subdivisions.

Since the cell splits are symmetric and the splitting direction changes at different depths of each level, no aspect ratio worse than 1:2 (2d-tree) or 1:1:2 (3d-tree) would occur.

Table 5.2.
Aspect ratios of the sub-cells generated by quad-, oct- and kd-tree subdivision. The initial element shape is assumed to be square (2D) or cubic (3D).

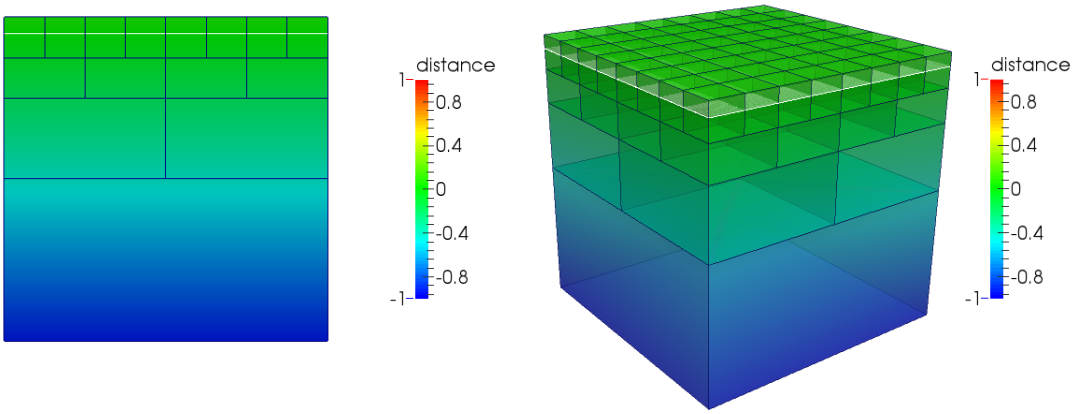
Tree Type	**Worst Case**	**Best Case**
Quad-tree	1:1	
Oct-tree	1:1:1	
2d-tree	1:2	1:1
3d-tree	1:1:2	1:1:1

5.3 Numerical Examples

Four numerical examples are presented to demonstrate the kd-tree based adaptive quadrature. The proposed technique is robust and applicable for both 2D and 3D immersed boundary problems without the need for deal with cumbersome special cases. It is further shown that the kd-tree based scheme effectively reduces the number of quadrature sub-cells while taking less amount of time than the classical quad-/oct-tree based schemes.

5.3.1 Hyper-planar Boundary

The first example, as shown in Figure 5.6, involves a hyper-plane immersed boundary (line in 2D and plane in 3D). A hierarchical sub-cell structure can be observed in this example. The number of generated sub-cells is listed in Table 5.3.



(a) Immersed line at $y = 0.95$.

(b) Immersed plane at $z = 0.95$.

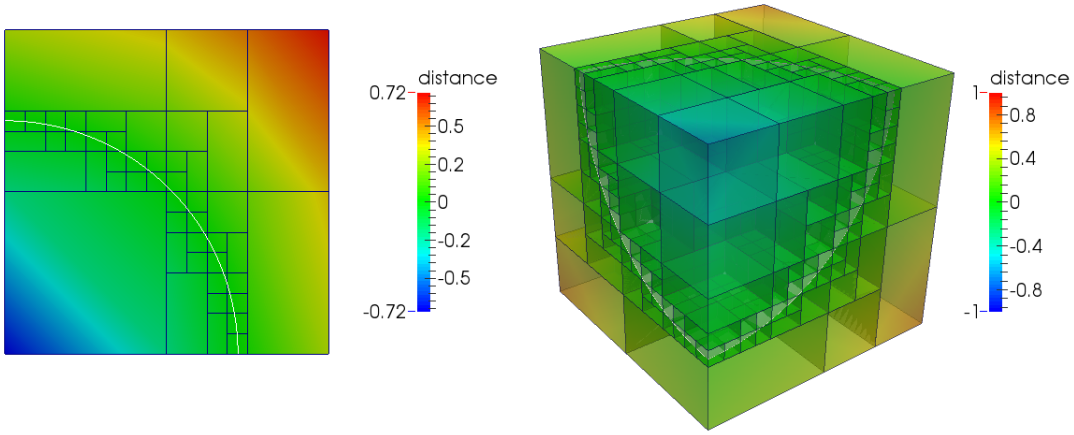
Figure 5.6. Kd-tree subdivision of a unit cell in the presence of (a) a 2D immersed line and (b) a 3D immersed plane. The maximum level is three in both examples. The domain color represents a signed distance to the immersed boundary.

Table 5.3.
Comparison of the kd-tree and quad-/oct-tree subdivision in the presence of a hyper-planar boundary as shown in Figure 5.6.

Tree Type	$N_{subcell}$	**Ratio of** $N_{subcell}$
2d-tree	15	0.682
Quad-tree	22	
3d-tree	85	0.574
Oct-tree	148	

5.3.2 Hyper-spherical Boundary

The second example is illustrated in Figure 5.7, where a hyper-spherical boundary is embedded in the domain. The corresponding number of created sub-cells are summarized in Table 5.4



(a) Immersed circle ($R = 0.72$) (b) Immersed spherical surface ($R = 0.72$)

Figure 5.7. Kd-tree subdivision of a unit cell in the presence of (a) a quadrant and (b) an 1/8 spherical surface. The hyper-spheres are centered at a corner and have a radius of R. The maximum level is four in both examples. The domain color represents a signed distance to the immersed boundary.

Table 5.4.
Comparison of the kd-tree and quad-/oct-tree subdivision in the presence of a hyper-spherical boundary as shown in Figure 5.7.

Tree Type	$N_{subcell}$	**Ratio of** $N_{subcell}$
2d-tree	48	0.787
Quad-tree	61	
3d-tree	521	0.722
Oct-tree	722	

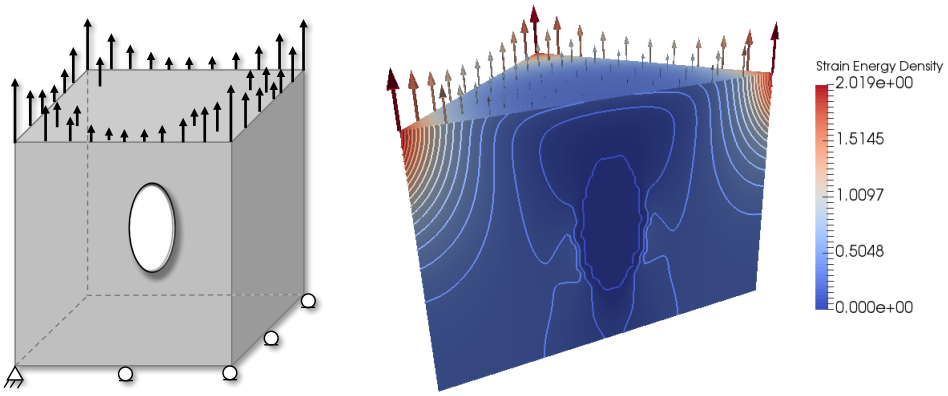
6. EFFICIENT ALGORITHMS FOR IMMERSED BOUNDARY PROBLEMS. PART II: TRUNCATED HIERARCHICAL B-SPLINES AND LOCAL REFINEMENT

For the immersed boundary problems that involve a high local gradient of the behavioral field, an accurate numerical integration is usually insufficient to assure an accurate solution. In addition, a refinement of the underlying (continuous) approximation is also desired. In this chapter, Truncated Hierarchical B-Splines (THB-splines) are utilized to enable local refinement in the enriched isogeometric analysis. Two new *a-priori* mesh generation algorithms based on the signed and unsigned distance field are first developed. To accelerate the stiffness/mass matrix assembly, we next develop an efficient, all-at-once algorithm to evaluate all the active THB-spline basis functions at a given point.

6.1 Introduction to Isogeometric Local Refinement

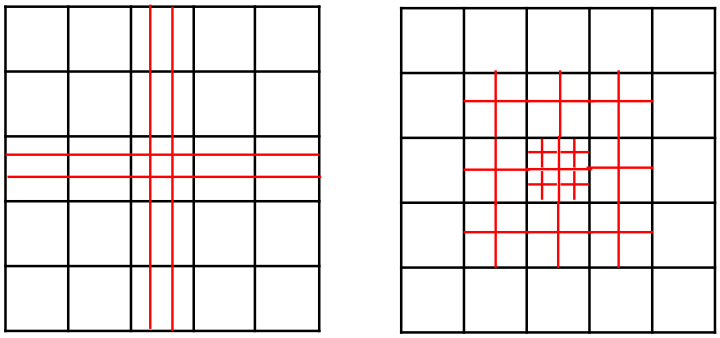
In many immersed boundary problems, the behavioral fields may vary rapidly or even discontinuously across the boundaries. As illustrated in Figure 6.1, a coarse, uniform approximation of the underlying domain may lead to an inaccurate solution to these problems. A global refinement (Figure 6.2a) of the continuous approximation may improve the accuracy, but also introduces redundant degrees of freedom (DOFs). Alternatively, a local refinement (Figure 6.2b) requires fewer degrees of freedom while achieving the same accuracy of solution.

There are many well-established approaches to realizing local refinement in finite element methods, e.g., hanging nodes and graded meshes. However, in the isogeometric framework, the tensor product NURBS surfaces only support global refinement (*p-refinement* [9, 51]) through knot insertion. To address the challenge of isogeo-



(a) Problem description (b) Strain energy density field

Figure 6.1. Elastostatic Analysis of a cubic domain with a ellipsoidal hole. (a) The schematic of the problem. A quadratically distributed load is applied to the top surface of the domain while the bottom surface is fixed. (b) The strain energy density field obtained from a coarse, uniform B-spline approximation. The domain is clipped diagonally for a better view of the hole region. It can be observed from the contours that the resulting strain energy density is oscillatory near the hole.

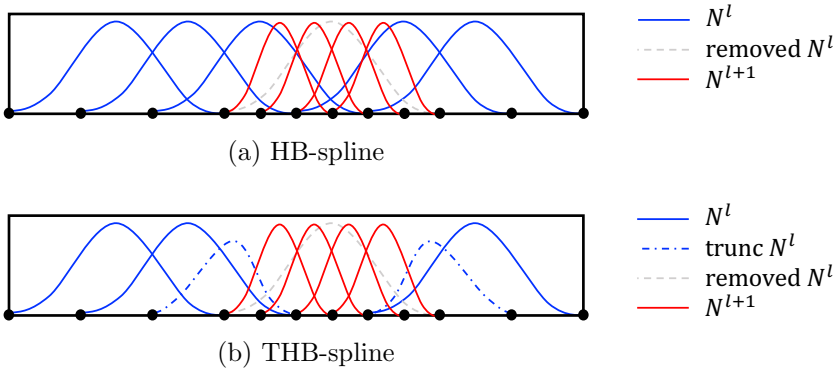


(a) Global refinement (b) Local refinement

Figure 6.2. Schematics of (a) global refinement and (b) local refinement. The goal is to refine the central element.

metric local refinement, many extensions of tensor-product representations, such as T-splines [100, 101] and THB-splines [50], were developed. The T-splines, which can represent complex CAD models with fewer control points, has become a powerful

enabling tool for advanced geometric modeling. Bazilevs et al. [102] first applied the T-splines to isogeometric analysis. Nevertheless, the linear independence of the T-spline basis functions, a condition that may be relaxed for CAD but required by CAE, is only guaranteed in a specific subset of T-splines [103, 104]. Although several algorithms [105, 106] were proposed to generate analysis-suitable T-splines, these procedures may cause a propagation of the refinement regions. Another technique for isogeometric local refinement is Hierarchical B-splines (HB-splines, see Figure 6.3a). Similar to the T-splines, the HB-splines were originally proposed for computer aided geometric design [107], and were later applied to analysis problems [108]. A critical drawback of the HB-splines is that the corresponding basis functions do not form a partition of unity. To circumvent the issue, Giannelli et al. [50] developed truncated hierarchical B-splines (Figure 6.3b), which retain the partition of unity property and also have a smaller support than the HB-splines. The properties of T-splines and THB-splines are summarized in Table 6.1.



(a) HB-spline

(b) THB-spline

Figure 6.3. The basis functions of a two-level (a) hierarchical B-spline and (b) truncated hierarchical B-spline. The N^l in dash line can be represented by a linear combination of N^{l+1}, and is therefore removed to avoid linear dependence.

The THB-splines are chosen for local refinement in the enriched isogeometric framework. However, as will be shown in Section 6.4, the number of active THB-spline basis functions at a point can be much larger than those of the original B-splines or

Table 6.1.
Typical properties of T-splines and THB-splines. In general, the T-splines provide more flexibility from the prospectives of non-uniformity and rationality, whereas the THB-splines enable a simpler and more robust implementation.

Spline Type	**Non-uniformity**	**Linear Independence**	**Partition of Unity**	**Refinement Propagation**
T-spline	Yes	Partial	Partial	Medium
THB-spline	No	Yes	Yes	Little

NURBS. Therefore, an efficient algorithm for evaluating all the active basis functions is desired.

6.2 Mathematical Description of Truncated Hierachical B-splines

Define a sequence of k-variate B-spline bases $B^l, l = 0, 1, \ldots, L-1$ satisfying the nesting relation:

$$\operatorname{span} \mathcal{B}^0 \subset \operatorname{span} \mathcal{B}^1 \subset \cdots \subset \operatorname{span} \mathcal{B}^{L-1}. \tag{6.1}$$

Thus, any lower-level B-spline basis function $N_i^l \in \mathcal{B}^l, l = 0, 1, \ldots, L-2$ can be represented by a linear combination of higher-level B-spline basis functions $N_j^{l+1} \in \mathcal{B}^{l+1}$ as follows:

$$N_i^l = \sum_{\operatorname{supp} N_j^{l+1} \subset \operatorname{supp} N_i^l} \alpha_j N_j^{l+1}. \tag{6.2}$$

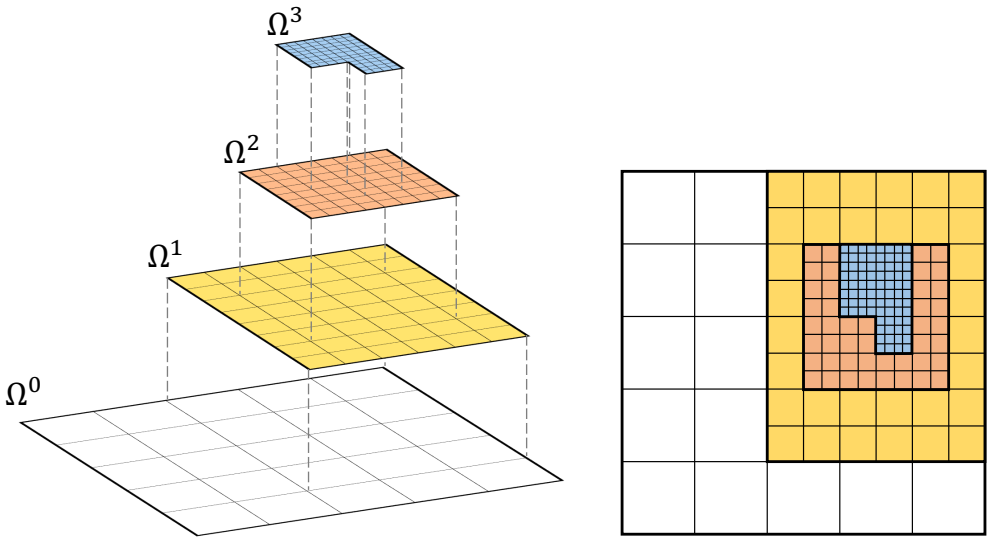
For the case of one-dimensional dyadic refinement (i.e., the knot vectors are uniform and each knot span is halved from V^l to V^{l+1}), Eq. (6.2) takes the form:

$$N_{i,p} = 2^{-p} \sum_{j=0}^{p+1} \binom{p+1}{j} N_{2i-1+j,p}^{l+1} \tag{6.3}$$

where, p the degree of the B-spline. We further define a sequence of nested domains:

$$\Omega^0 \supseteq \Omega^1 \supseteq \cdots \supseteq \Omega^{L-1} \tag{6.4}$$

where, each $\Omega^l \in \mathbb{R}^k$ represents a k-dimensional refinement region at level l and its boundary $\partial\Omega^l$ is aligned with the knot grid of $\mathcal{B}^l$. A two-dimensional hierarchical mesh example is shown in Figure 6.4.



(a) Nested domains (b) Overlaid hierarchical mesh

Figure 6.4. A two-dimensional, four-level dyadic hierarchical mesh: (a) The nested domains contain level-wise sub-meshes. (b) The sub-meshes are overlaid to generate the hierarchical mesh.

The hierarchical B-spline basis $\mathcal{H}$ is defined as:

$$\mathcal{H} = \bigcup_{l=0}^{L-1} \left\{ N^l \,\middle|\, N^l \in \mathcal{B}^l \wedge \operatorname{supp} N^l \subseteq \Omega^l \wedge \operatorname{supp} N^l \nsubseteq \Omega^{l+1} \right\}. \tag{6.5}$$

As illustrated in Figure 6.3a, the hierarchical B-spline basis functions do not satisfy the partition of unity property. Giannelli et al. [50] proposed a truncation operator to remove the higher-level B-spline components from a current-level basis function:

$$\operatorname{trunc}^{l+1} N_i^l = N_i^l - \sum_{\substack{\operatorname{supp} N_j^{l+1} \subset \operatorname{supp} N_i^l \\ \operatorname{supp} N_j^{l+1} \subseteq \Omega^{l+1}}} \alpha_j N_j^{l+1} \qquad \text{(Subtractive Representation)} \tag{6.6a}$$

$$= \sum_{\substack{\operatorname{supp} N_j^{l+1} \subset \operatorname{supp} N_i^l \\ \operatorname{supp} N_j^{l+1} \not\subseteq \Omega^{l+1}}} \alpha_j N_j^{l+1} \qquad \text{(Additive Representation).} \tag{6.6b}$$

Figure 6.5 illustrates the subtractive and additive representation of a THB-spline basis function. The general THB-spline basis $\mathcal{T}$ can be constructed as follows:

$$\mathcal{T} = \bigcup_{l=0}^{L-1} \left\{ \widehat{N}^l \,\middle|\, \widehat{N}^l = \operatorname{trunc}^{L-1} \cdots \operatorname{trunc}^{l+2} \operatorname{trunc}^{l+1} N^l \,, \right.$$
$$\left. N^l \in \mathcal{B}^l \wedge \operatorname{supp} N^l \subseteq \Omega^l \wedge \operatorname{supp} N^l \not\subseteq \Omega^{l+1} \right\}. \tag{6.7}$$

6.3 Adaptive Mesh Generation

The THB-splines are defined over a sequence of nested domains which form a hierarchical mesh. An efficient representation of the hierarchical mesh as well as a robust mesh generator plays an important role in the performance of the THB-spline based local refinement.

6.3.1 Kd-tree based Mesh Representation

Kiss et al. [109] proposed a quad-tree data structure to represent two-dimensional THB-spline meshes. As will be shown later, the quad-tree results in an exponential space complexity in the worst case. Recently, an alternative binary tree data struc-

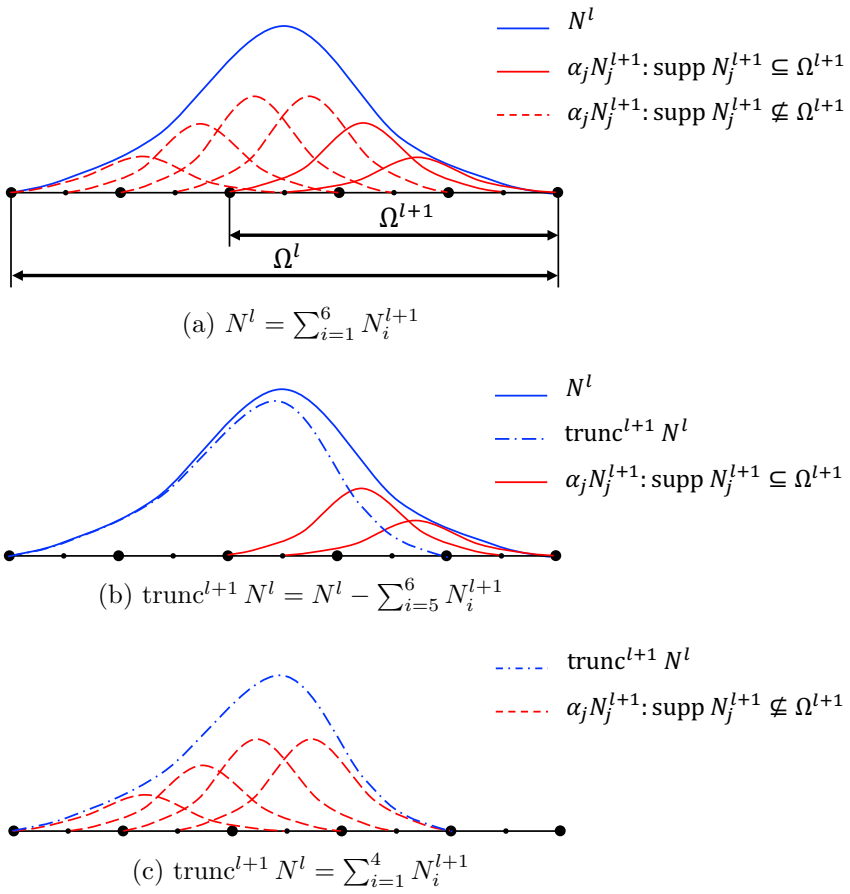
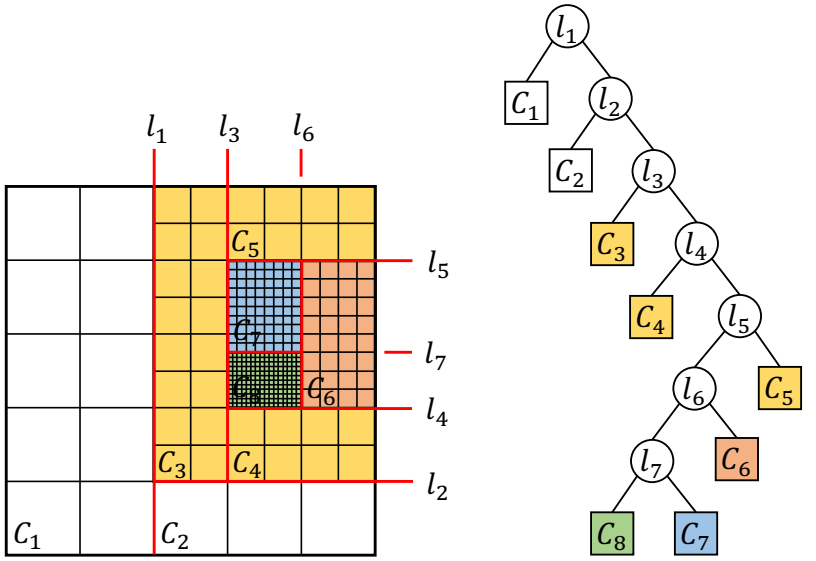


Figure 6.5. Illustration of the truncation operator: (a) The non-truncated B-spline basis function N^l, and the truncated N^l in a (b) subtractive representation and a (c) additive representation.

ture was utilized [110] for multi-dimensional meshes. The internal nodes of the tree store splitting lines during domain subdivision whereas the leaves of the tree contains homogeneous pieces of the domain. The nature of the new data structure is, in fact, a kd-tree. Figure 6.6 illustrates an example of the kd-tree representation for a two-dimensional hierarchical mesh.

A significant advantage of the kd-tree representation over the quad-/oct-tree is that the former approach enables unequal subdivision of the domain, which results in fewer splits during tree construction. Consequently, the number of leaves generated by a kd-tree subdivision is smaller than that generated by a quad-tree or oct-tree.



(a) Splitting lines and cells in the hierarchical mesh

(b) Kd-tree representation

Figure 6.6. Kd-tree representation of a two-dimensional hierarchical mesh. (a) The domain is subdivided into homogeneous cells of which each only belongs to one hierarchical level. (b) The splitting lines and the cells are stored in the internal nodes and leaves of a kd-tree data structure, respectively.

Figure 6.7 illustrates an instance of successive stripe refinement on a square domain. Since the boundaries of the refinement regions are not aligned with bisection planes, a quad-tree representation will contain $3 \cdot 2^n - 3$ internal nodes and $9 \cdot 2^n - 8$ leaves, leading to a space complexity of $O(2^n)$. In contrast, its kd-tree counterpart only needs n internal nodes and $n + 1$ leaves, of which the space complexity is $O(n)$.

6.3.2 Mesh Refinement Algorithms

The kd-tree data structure is chosen in current work due to its lower space complexity. However, the overall computational performance also depends on the efficiency of the tree construction algorithm. The classical mesh refinement schemes were based on *a-posteriori* error estimators [35,36], which requires a trial solution be-

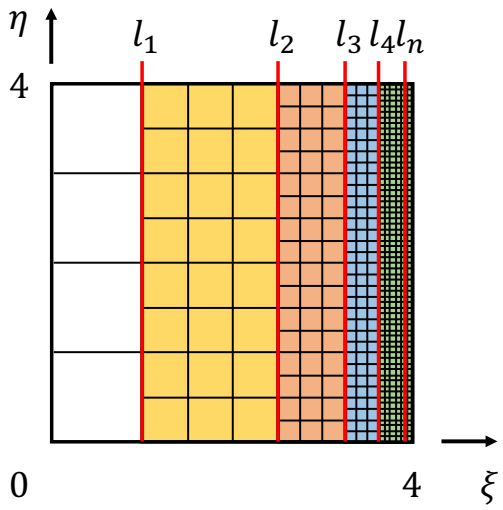
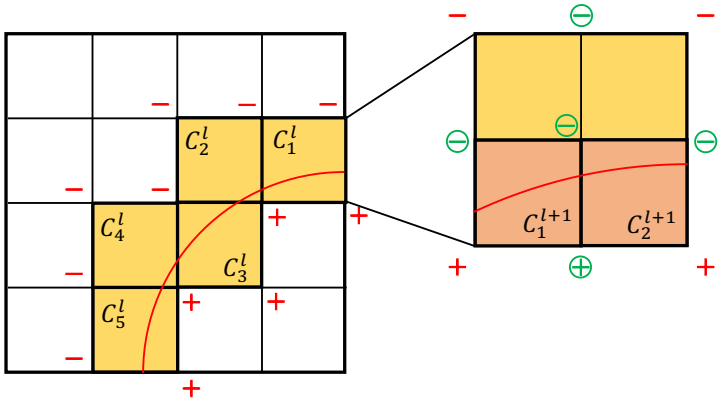


Figure 6.7. n-stripe refinement on a 4×4 domain. The left boundary of each refinement region Ω^i is given by $\xi = 4 - 3 \cdot 2^{1-i}$ and the right boundaries coincide at $\xi = 4$. The splitting lines during kd-tree subdivision are labeled with $l_i, i = 1, 2, \cdots, n$.

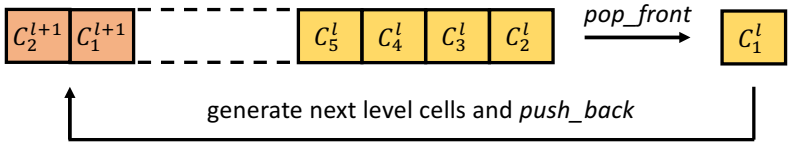
fore the adaptive mesh is generated. In the context of immersed boundary problems, it is expected that the behavioral fields vary rapidly near the boundaries. Therefore, given the position of the immersed boundaries, *a-priori* mesh refinement is possible. Two adaptive refinement algorithms based on signed [99] and unsigned [49] distance field, are proposed as follows:

1. *Sign-based Refinement Algorithm* (SRA): The cut cells and their neighbors are recursively subdivided until the maximum level is reached. The cut cells are identified by checking the signed distance of the cell vertices to the immersed boundaries. As illustrated in Figure 6.8, a cell is marked as a cut cell if its vertices have opposite signs. To minimize the number of sign checkings, the vertex signs of the cut cells (uncircled in Figure 6.8a) are stored in memory. As the subdivision continues, the cut cells in the next level can be determined by additionally checking the signs of edge, face and cell centers (circled in Figure 6.8a). A first-in-first-out (FIFO) queue data structure (Figure 6.8b) is utilized to man-

age the cells. The cut cells in the current level are first popped out from the front of the queue. After sign checking, the cut cells in the next level are generated and then pushed into the back of the queue. The checking and new cell generation stop when the maximum level is reached.



(a) Sign-based cell subdivision with respect to a circular boundary



(b) Cell queue corresponding to (a)

Figure 6.8. Illustration of the sign-based refinement algorithm. (a) The cut cells in level l are subdivided into four (2D) or eight (3D) candidate subcells. The new cut cells in level $l+1$ are identified by the newly calculated (circled) signs and the previously obtained (uncircled) signs, and (b) then pushed to the back of a queue and wait for the next level subdivision.

2. *Distance-based Refinement Algorithm* (DRA): The algorithm is similar to the SRA except that the cut cells are determined by checking the magnitude of the distance instead of the sign. This is particularly useful when the feature size d_{feature} of the immersed boundaries is smaller than the cell size d_{cell}, in which

case the sign-based algorithm may fail to detect a cut cell (see Figure 6.9a). In the distance-based refinement, a cell is marked as a cut cell if:

$$\max_i |d_i| < \sqrt{k} d_{\text{cell}} \tag{6.8}$$

where, k is the dimension of the domain and d_i is the distance of the i^{th} vertex of the cell to the closest boundary. As shown in Figure 6.9b, the missing cell can be found by DRA. Nevertheless, the criterion of the DRA is looser than that of the SRA, which can result in many redundant cut cells. To mitigate the problem, it is suggested to dynamically switch between the two algorithms based on d_{cell}. The distance-based criterion is chosen if $d_{\text{cell}} > d_{\text{feature}}$. Otherwise, the sign-based criterion is preferred. The inset magnified picture of Figure 6.9b shows a sign-based subdivision in the cut cell detected by the distance-based algorithm.

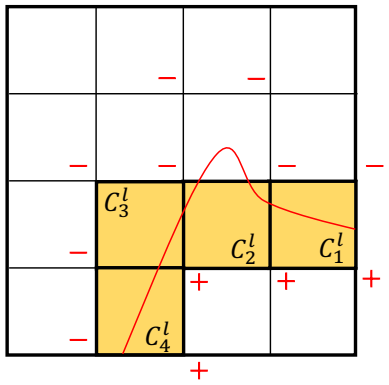
6.3.3 Numerical Examples

The efficiency and robustness of the proposed algorithms are demonstrated through several numerical examples. Figure 6.10 shows a seven-level hierarchical refinement of a square domain with respect to a rectangular immersed boundary. The computational cost as a function of the number of levels is listed in Table 6.2. The hierarchical mesh generation only takes tens of microseconds even for very large number of levels.

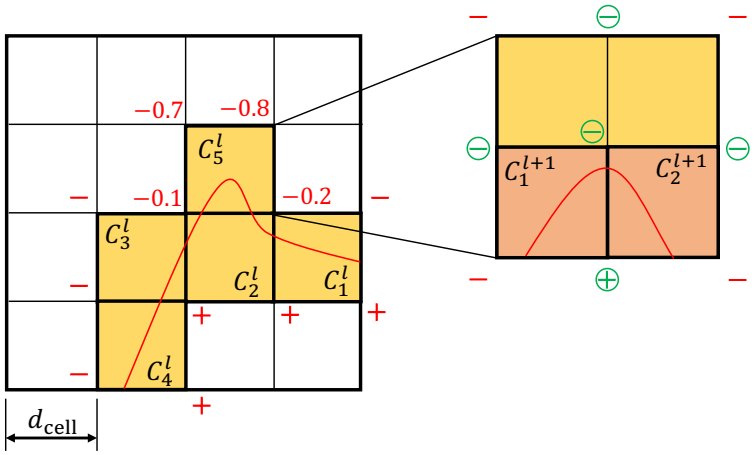
Table 6.2.
Computer time for the hierarchical refinement of the geometry shown in Figure 6.10.

Number of Levels	1	2	3	4	5	6	7
Time Cost (μs)	0.1	0.7	1.4	3.2	6.8	14.8	36.7

In order to have a larger local refinement region, it may be desired to not just refine the cut cells but also their i^{th} neighbors. As illustrated in Figure 6.11, the order i controls the bandwidth of a single level mesh. In contrast to the level-wise hierarchical



(a) Sign-based refinement for a complex boundary

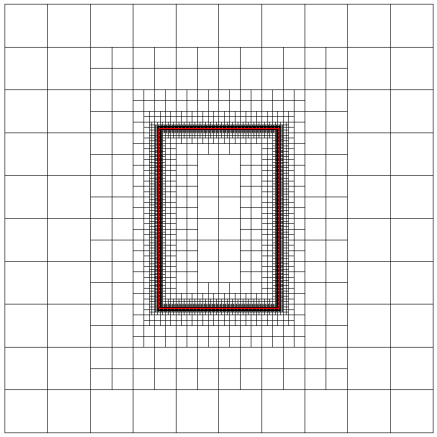


(b) Combination of distance-based and sign-based refinement

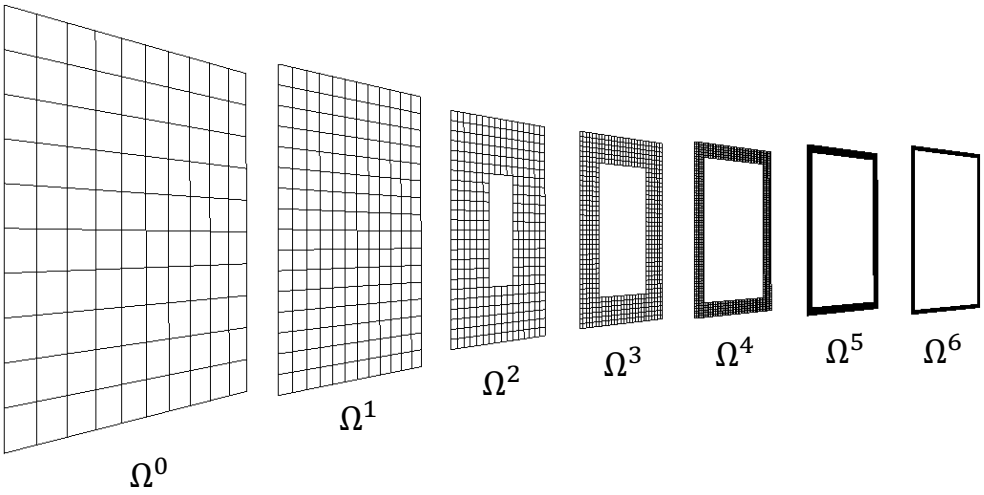
Figure 6.9. (a) Determination of the level l cut cells by the sign-based criterion. A cut cell is missed due to the small feature size. (b) Determination of the level l cut cells by the distance-based criterion, followed by a sign-based subdivision which generates the level $l+1$ cut cells. Given a cell size of $d_{\text{cell}} = 1$, the distances of the C_5^l vertices to the boundary are annotated in the figure.

refinement, the refinement within a level is referred as *Horizontal Refinement*. A two-dimensional example of horizontal refinement is shown in Figure 6.12.

The mesh refinement algorithms and the kd-tree data structure can be directly extended to three-dimensional problems due to their dimension independence. Figure 6.13 shows a six-level hierarchical refinement of a cubic domain in the presence of an ellipsoidal boundary. The corresponding computational cost is summarized in



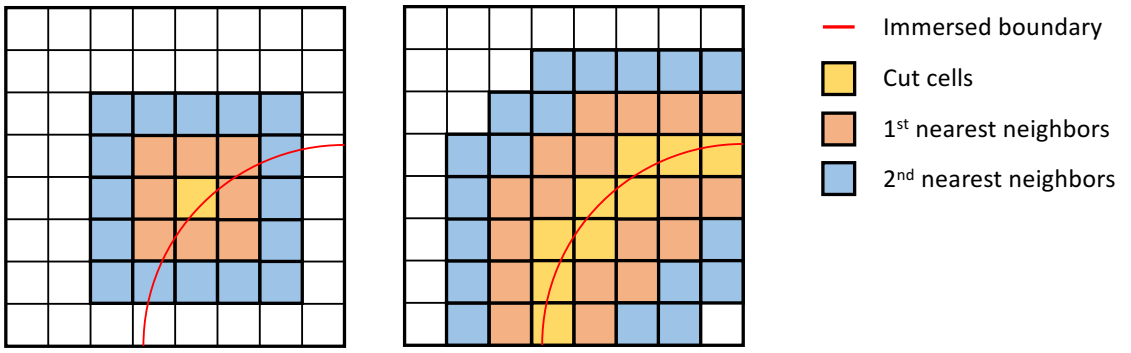
(a) Hierarhical mesh around the immersed boundary



(b) Exploded view of the nested sub-meshes

Figure 6.10. A two-dimensional, seven-level hierarchical refinement of a square domain with respect to a rectangular immersed boundary: (a) The hierarchical mesh and (b) the sub-meshes in each level.

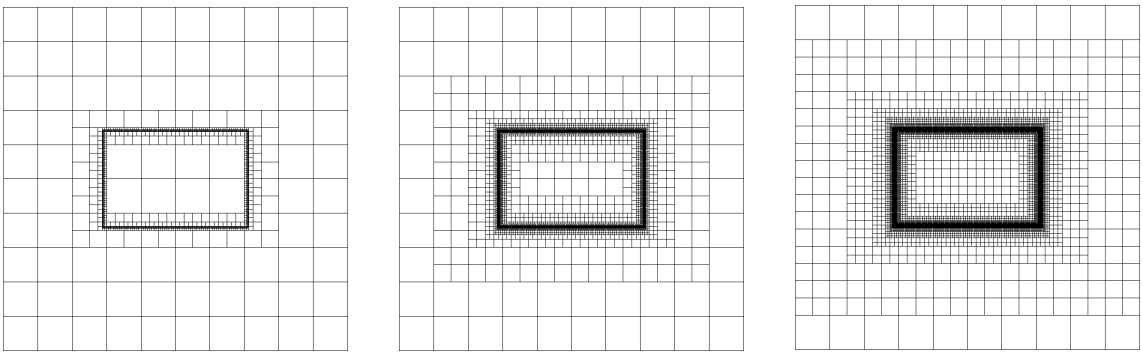
Table 6.3. Compared to the time cost during matrix assembly and system solution, the sub-second mesh generation time is almost negligible.



(a) A single cut cell and its first and second nearest neighbors

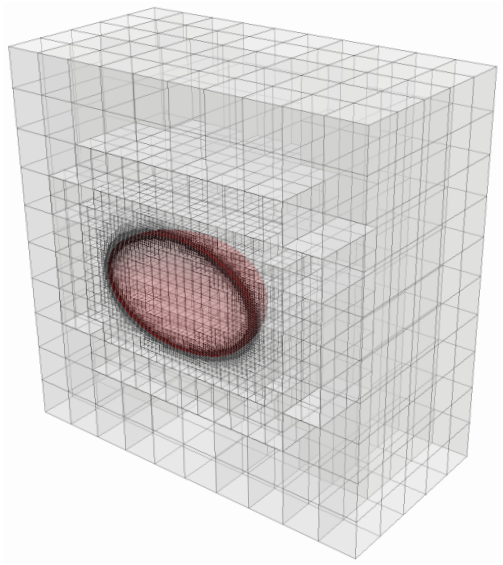
(b) All the cut cells and the union of their first and second nearest neighbors

Figure 6.11. Schematic of horizontal refinement. (a) The neighbors of a cut cell provide a larger refinement region. (b) The union of the neighbors forms a refinement band.

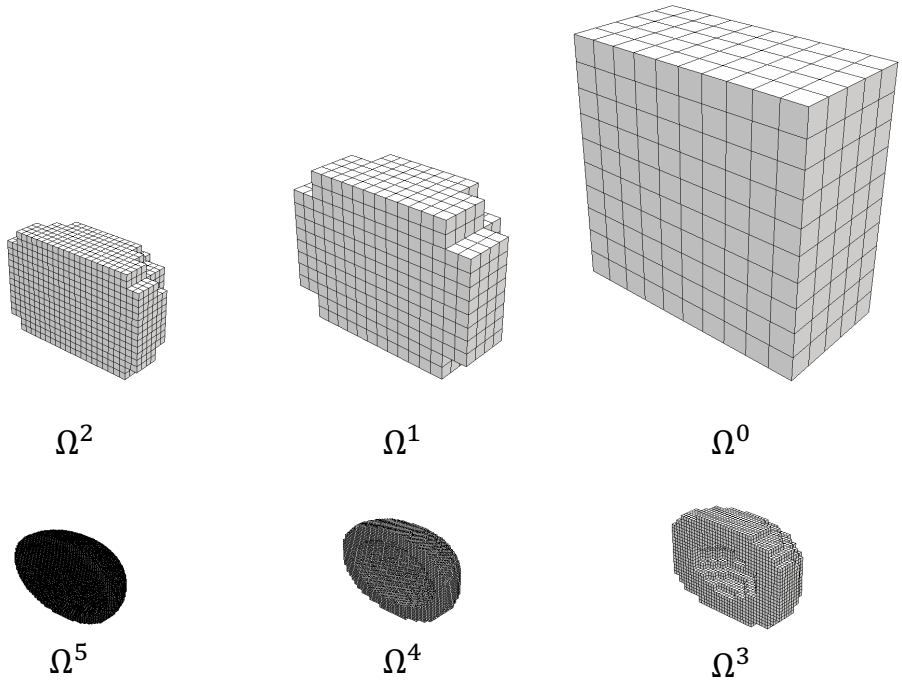


(a) $i = 0$ (b) $i = 1$ (c) $i = 2$

Figure 6.12. Horizontal refinement with different bandwidth. (a) Only the cut cells are refined. (b) First order refinement: The cut cells and their first nearest neighbors are refined. (c) Second order refinement: The cut cells and up to the second nearest neighbors are refined.



(a) Hierarhical mesh around the immersed boundary



(b) Exploded view of the nested sub-meshes

Figure 6.13. A three-dimensional, six-level hierarchical refinement of a cubic domain with respect to an ellipsoidal boundary: (a) The hierarchical mesh and (b) the sub-meshes in each level.

Table 6.3.
Computer time for the hierarchical refinement of the geometry shown in Figure 6.13.

Number of Levels	1	2	3	4	5	6
Time Cost (μs)	0.1	2.4	8.6	47.9	166.3	434.8

6.4 Maximum Number of Active THB-spline Basis Functions at a Point

The active THB-spline basis at a given point $\boldsymbol{x} \in \Omega^0$ is defined as

$$\mathcal{A}(\boldsymbol{x}) = \left\{ \widehat{N} \,\middle|\, \widehat{N} \in \mathcal{T} \wedge \widehat{N}(\boldsymbol{x}) \neq 0 \right\} \tag{6.9}$$

where, $\mathcal{T}$ is the general THB-spline basis given by Eq. (6.7). The B-splines and NURBS possess an important property that the number of active basis functions (n_{B} for B-splines and n_{N} for NURBS) at any point is a constant:

$$n_{\mathrm{B}} = n_{\mathrm{N}} = \prod_{i=1}^{k} (p_i + 1) \tag{6.10}$$

where, p_i is the degree of the spline in i^{th} direction. However, this property does not hold for THB-splines. As illustrated in Figure 6.14, the number of active THB-spline basis functions (n_{THB}) may vary between different knot spans. Consequently, if the THB-spline framework is utilized as an analysis tool, the size of the elemental matrix, as well as the number of non-zeros in each row of the stiffness/mass matrix, also varies. The varied size of the elemental matrix necessitates a dynamic memory allocation during matrix assembly, which is computationally very expensive [111]. Therefore, an estimate of the maximum number of active THB-spline basis functions ($\max\{n_{\mathrm{THB}}\}$) , which enables a preallocation of the memory, is of great importance to reduce the computational overhead. The range of the $\max\{n_{\mathrm{THB}}\}$ is discussed in Theorem 6.4.1.

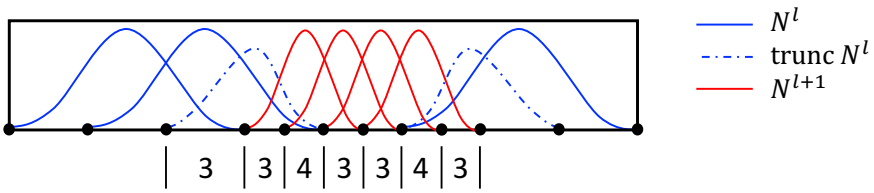


Figure 6.14. Number of non-zero THB-spline basis functions in each knot span.

Theorem 6.4.1 *The maximum number of active basis functions of a k-dimensional, L-level THB-spline satisfies*

$$\prod_{i=1}^{k}(p_i+1)+(L-1)\left[\prod_{i=1}^{k}p_i-\prod_{i=1}^{k}\left\lfloor\frac{p_i}{2}\right\rfloor\right]\leq\max\{n_{\mathrm{THB}}\}<L\prod_{i=1}^{k}(p_i+1)\,. \tag{6.11}$$

Proof

Upper bound: Neglecting the truncation and removal of B-spline basis functions, Eq. (6.10) must be satisfied in every level. For a L-level THB-spline, we have

$$\max\{n_{\mathrm{THB}}\}\leq L\prod_{i=1}^{k}(p_i+1)\,. \tag{6.12}$$

However, if any higher level B-spline basis is complete at a point, all the lower level B-spline basis functions covering that point will be truncated, and therefore the equality in Eq. (6.12) can not be achieved.

Lower bound: To prove the lower bound, we simply need to find a special case that can produce the bound with arbitrary p_i, k and L. The construction of such case is described as follows:

1. Define the base level: $\Omega^0=\bigotimes_{i=1}^{k}[-p_1-1,+\infty)$,
2. Define the l^{th} level, $l=1,2,\cdots,L-1$: $\Omega^l=\bigotimes_{i=1}^{k}[-p_i2^{-l},+\infty)$,
3. Define the knot interval in each level and direction: $\Delta\xi_i^l=2^{-l}$.

In this manner, $\forall \boldsymbol{x} \in \bigotimes_{i=1}^{k}[-p_i 2^{-L}, 0]$,

$$n_{\mathrm{THB}}^{l}(\boldsymbol{x}) = \begin{cases} \prod_{i=1}^{k}(p_i+1) - \prod_{i=1}^{k}\left\lfloor \frac{p_i}{2} \right\rfloor & l = 0 \\ \prod_{i=1}^{k} p_i - \prod_{i=1}^{k}\left\lfloor \frac{p_i}{2} \right\rfloor & l = 1, 2, \cdots, L-2 \\ \prod_{i=1}^{k} p_i & l = L-1 \end{cases} . \tag{6.13}$$

Thus,

$$n_{\mathrm{THB}}(\boldsymbol{x}) = \prod_{i=1}^{k}(p_i+1) + (L-1)\left[\prod_{i=1}^{k} p_i - \prod_{i=1}^{k}\left\lfloor \frac{p_i}{2} \right\rfloor\right]. \tag{6.14}$$

Figures 6.15 and 6.16 show two examples based on the proposed construction procedure. ■

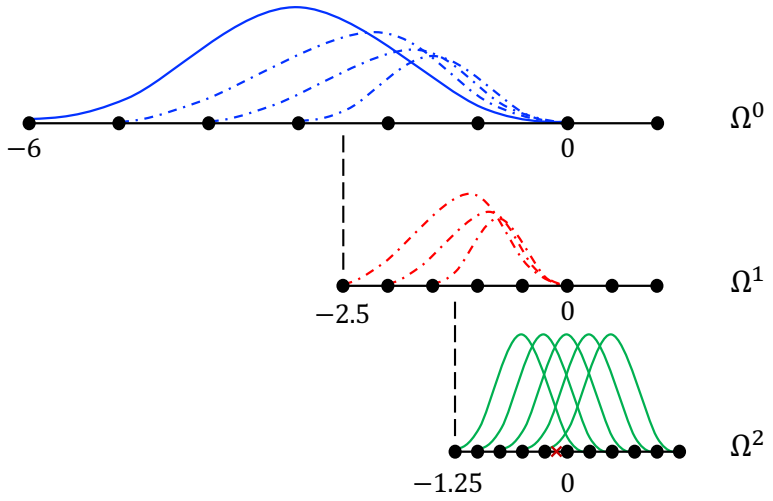


Figure 6.15. Construction of an one-dimensional, three-level quintic THB-spline based on the procedure described in the proof of Theorem 6.4.1. The solid lines and dash lines represent non-truncated and truncated basis functions, respectively. The $\max\{n_{\mathrm{THB}}\} = 12$ occurs in the knot span where the cross point resides.

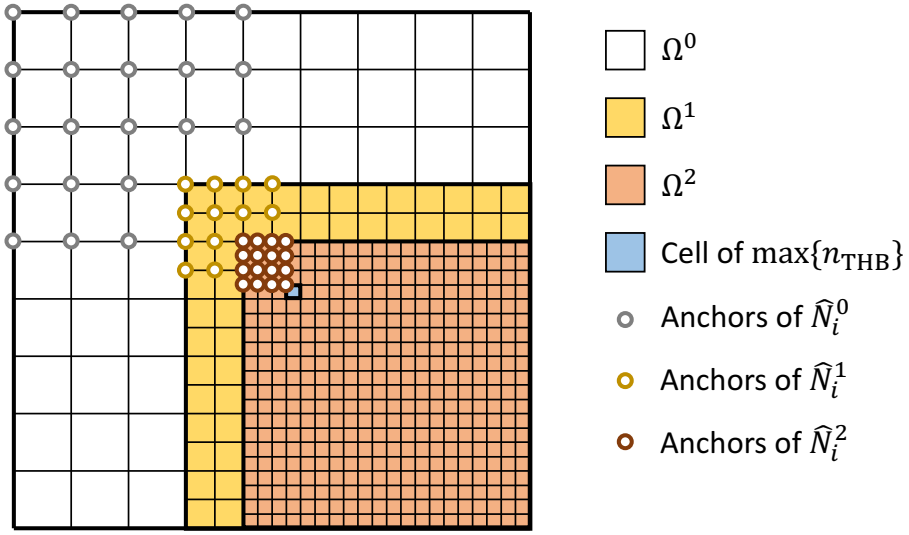


Figure 6.16. Construction of a two-dimensional, three-level quartic THB-spline based on the procedure described in the proof of Theorem 6.4.1. The $\max\{n_{\text{THB}}\} = 49$ occurs in the blue cell. The anchors of all the active basis functions in the cell are marked with circles.

6.5 Evaluation of the Active Basis Functions

The THB-spline framework has already shown its potential as a powerful approximation technique for isogemetric analysis [94, 110, 112]. Compared to uniform B-splines, the THB-splines require far fewer degrees of freedom to achieve the same accuracy of solution. However, according to Theorem 6.4.1, the $\max\{n_{\text{THB}}\}$ increases exponentially with the dimension k. Consider a typical scenario that $k = 3, p_i = 3, L = 5$, the n_{THB} can be as large as 168, whereas the corresponding n_{B} for B-splines is only 64. This would adversely influence both the matrix assembly and the system solving as follows:

1. In the matrix assembly phase, the THB-splines may need to evaluate a much larger number of basis functions at a gauss point.

2. Given the same number of degrees of freedom, the matrix system created by a THB-spline approximation is much denser than that from a B-spline approx-

imation. Therefore, in the system solving phase, the THB-splines cost more memory space and computational time than the B-splines.

The second issue, due to the nature of the THB-splines, is difficult to resolve. Nevertheless, the first issue may be mitigated through an efficient algorithm for basis function evaluation. To this end, we first review two existing algorithms and next propose a new all-at-once algorithm that can evaluate all the active THB-spline basis functions simultaneously.

6.5.1 Naive Algorithm

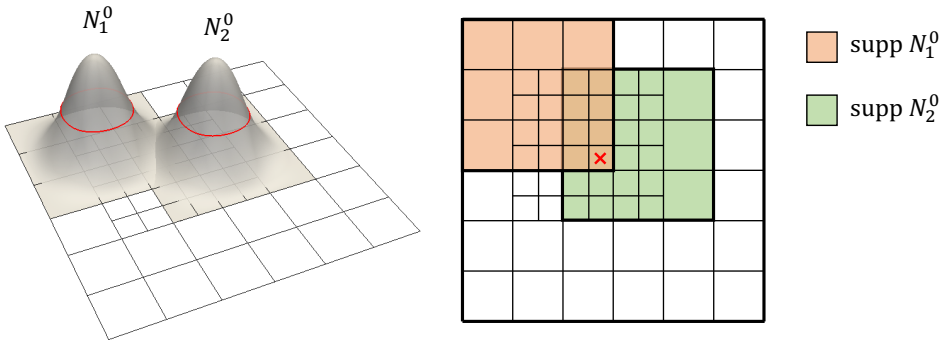
The naive algorithm is based on the subtractive representation (Eq. (6.6a)). Given a point, the algorithm caches the active B-spline basis functions in all levels and then subtracts the higher level basis functions from the lower level ones. The implementation of this algorithm is straightforward, but it suffers from two drawbacks:

1. In the worst case, it is necessary to evaluate $L\prod_{i=1}^{k}(p_i+1)$ B-spline basis functions at a point.

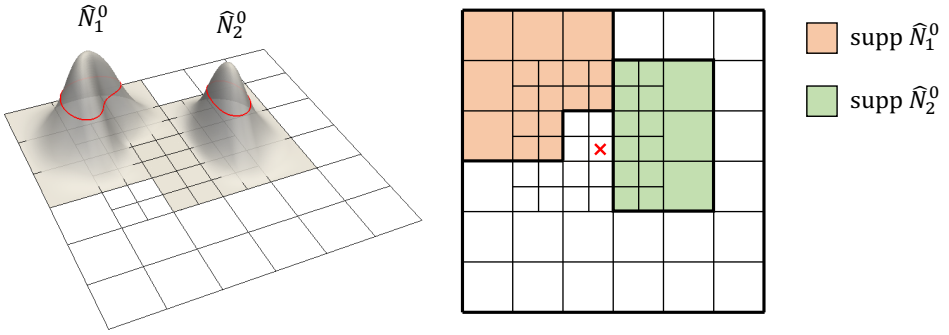
2. It is difficult to determine the support of the THB-spline basis functions since the support of a THB-spline basis function can only be known after the subtraction is completed. In the instance shown in Figure 6.17, although $\widehat{N}_1^0$ and $\widehat{N}_2^0$ are not active at the cross point, the values of N_1^0 and N_2^0 will still be calculated by this algorithm.

6.5.2 Giannelli's Algorithm

To *a-priori* identify the support of THB-spline basis functions as well as reducing the number of B-spline function evaluated, Giannelli et al [110] proposed an algorithm based on the additive representation Eq. (6.6b). For any $\widehat{N}^l \in \mathcal{T}$, there exists a



(a) B-spline basis functions and their support



(b) THB-spline basis functions and their support

Figure 6.17. Illustration of the support change after truncation. (a) Two active B-spline basis functions at the cross point and (b) the corresponding THB-spline basis functions, which do not cover the point any longer.

representation level $l_r \leq L-1$ such that $\widehat{N}^l$ can be represented by a linear combination of the level l_r B-spline basis functions, i.e.,

$$\widehat{N}^l = \sum_j \alpha_j N_j^{l_r}. \tag{6.15}$$

In Eq. (6.7), a level l THB-spline basis function is defined as a successive truncation of a level l B-spline basis function up to the maximum level. In fact, it is not necessary to truncate the function to the maximum level but just its representation level l_r, i.e.,

$$\widehat{N}^l = \mathrm{trunc}^{l_r} \cdots \mathrm{trunc}^{l+2}\, \mathrm{trunc}^{l+1}\, N^l. \tag{6.16}$$

The Giannelli's algorithm stores the representation level l_r and coefficients α_j for all $\widehat{N}^l \in \mathcal{T}$. When the value of a THB-spline basis function is needed, one only needs to calculate the active B-spline functions in the representation level and substitute them into Eq. (6.15). This algorithm has two major advantages:

1. Since many THB-spline basis functions possess the same representation level, the average number of evaluated B-spline basis functions is smaller than that of the naive algorithm.

2. The support of a THB-spline basis function can be easily determined from the additive representation.

However, in the worst case, if the active THB-spline basis functions at a point have distinct representation levels, the algorithm still needs to evaluate $L\prod_{i=1}^{k}(p_i+1)$ B-spline basis functions at the point.

6.5.3 All-at-Once Algorithm

Due to the subdivisability of B-spline (Eq. (6.2)), Eq. (6.15) can be extended to any level $l \geq l_r$. Given a point $\boldsymbol{x} \in \Omega^0$, there exist an integer set $\mathcal{S}$ containing the representation levels of all the $\widehat{N} \in \mathcal{A}(\boldsymbol{x})$. Next let

$$l_{\mathcal{A}} = \max_{l_r \in \mathcal{S}} l_r \tag{6.17}$$

be the representation level of $\mathcal{A}(\boldsymbol{x})$. Then all the $\widehat{N} \in \mathcal{A}(\boldsymbol{x})$, no matter how many there are, can be expressed by a linear combination of the $\prod_{i=1}^{k}(p_i+1)$ B-spline basis functions in the level $l_{\mathcal{A}}$. Based on this idea, an all-at-once algorithm can be developed to evaluate all the $\widehat{N} \in \mathcal{A}(\boldsymbol{x})$ simultaneously.

An arbitrary B-spline function $\beta \in \operatorname{span}\mathcal{B}^l$ can be written as

$$\beta = \sum_i N_i^l u_i^l = \left[\mathbf{N}^l\right]\left\{\boldsymbol{u}^l\right\} = \left[\mathbf{N}^{l+1}\mathbf{R}^{l+1}\right]\left\{\boldsymbol{u}^l\right\} \tag{6.18}$$

where, $\mathbf{R}^{l+1}$ is the refinement matrix which can be determined through the binomial relation (Eq. (6.3)) for a dyadic domain, or a knot insertion algorithm for a general domain. The level $l+1$ truncation of β can be expressed as

$$\text{trunc}^{l+1}\,\beta = \left[\mathbf{N}^{l+1}\left(\mathbf{I}^{l+1} - \mathbf{X}^{l+1}\right)\mathbf{R}^{l+1}\right]\left\{\boldsymbol{u}^{l}\right\} \tag{6.19}$$

where, $\mathbf{I}^{l+1}$ is identity matrix and $\mathbf{X}^{l+1}$ is the characteristic matrix [110] given by

$$\mathbf{X}^{l+1} = \text{diag}(x_i^{l+1}), \quad x_i^{l+1} = \begin{cases} 1 & \text{if } N_i^{l+1} \text{ is active} \\ 0 & \text{otherwise} \end{cases}. \tag{6.20}$$

Therefore, the level $l+1$ truncation of the active B-spline basis matrix can be defined as

$$\text{trunc}^{l+1}\,\mathbf{N}^{l} = \mathbf{N}^{l+1}\left(\mathbf{I}^{l+1} - \mathbf{X}^{l+1}\right)\mathbf{R}^{l+1}. \tag{6.21}$$

If the THB-spline has only two levels: level l and $l+1$, the $\mathcal{A}(\boldsymbol{x})$ should include all the active, truncated basis functions in the level l and the active, non-truncated basis functions in the level $l+1$ as follows:

$$\begin{aligned} \widehat{\mathbf{N}} &= \left[\text{trunc}^{l+1}\,\mathbf{N}^{l}\widetilde{\mathbf{X}}^{l} \quad \mathbf{N}^{l+1}\widetilde{\mathbf{X}}^{l+1}\right] \\ &= \left[\mathbf{N}^{l+1}\left(\mathbf{I}^{l+1} - \mathbf{X}^{l+1}\right)\mathbf{R}^{l+1}\widetilde{\mathbf{X}}^{l} \quad \mathbf{N}^{l+1}\widetilde{\mathbf{X}}^{l+1}\right] \\ &= \mathbf{N}^{l+1}\left[\left(\mathbf{I}^{l+1} - \mathbf{X}^{l+1}\right)\mathbf{R}^{l+1}\widetilde{\mathbf{X}}^{l} \quad \widetilde{\mathbf{X}}^{l+1}\right] \\ &\equiv \mathbf{N}^{l+1}\mathbf{C}^{l+1} \end{aligned} \tag{6.22}$$

where, $\widetilde{\mathbf{X}}^{l}$ is a sub-matrix that contains all the non-zero column vectors of $\mathbf{X}^{l}$. A general algorithm for multi-level THB-splines is formulated in Alg. 4.

Remarks:

1. The all-at-once algorithm inherits all the merits of Giannelli's algorithm, for instance, the easy determination of function supports. Furthermore, at any

Algorithm 4 All-at-Once Algorithm for Evaluating all the Active THB-spline Basis Functions

Input: Characteristic matrices $\mathbf{X}^l, l = 0, 1, \cdots, l_{\mathcal{A}}$, refinement matrices $\mathbf{R}^l, l = 1, 2, \cdots, l_{\mathcal{A}}$ and B-spline basis matrix in the level $l_{\mathcal{A}}$, i.e., $\mathbf{N}^{l_{\mathcal{A}}}$

Output: Active THB-spline basis matrix $\widehat{\mathbf{N}}$

1: **function** ACTIVE_THB_SPLINE_BASIS($\mathbf{X}^l$, $\mathbf{R}^l$, $\mathbf{N}^{l_{\mathcal{A}}}$)
2: $\quad\mathbf{C}^0 \leftarrow \widetilde{\mathbf{X}}^0$
3: $\quad$**for** $l \leftarrow 1, l_{\mathcal{A}}$ **do**
4: $\quad\quad\mathbf{C}^l \leftarrow \begin{bmatrix}\left(\mathbf{I}^l - \mathbf{X}^l\right)\mathbf{R}^l\mathbf{C}^{l-1} & \widetilde{\mathbf{X}}^l\end{bmatrix}$
5: $\quad$**end for**
6: $\quad\widehat{\mathbf{N}} \leftarrow \mathbf{N}^{l_{\mathcal{A}}}\mathbf{C}^{l_{\mathcal{A}}}$
7: **end function**

point of interest, only $\prod_{i=1}^{k}(p_i+1)$ B-spline basis functions need to be evaluated regardless of the number of active basis functions at that point.

2. In the case of k-dimensional dyadic refinement, there are only 2^k distinct refinement matrices, which can be computed *a-priori* and cached to improve the efficiency.

3. The refinement matrix $\mathbf{C}^{l_{\mathcal{A}}}$ keeps constant in each element of the hierarchical mesh.

Example 6.5.1 Given a THB-spline shown in Figure 6.18, the refinement matrix $\mathbf{C}^2$ at the cross point can be derived as follows:

1. *Input*:

$$\mathbf{X}^0 = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}, \mathbf{X}^1 = \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}, \mathbf{X}^2 = \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

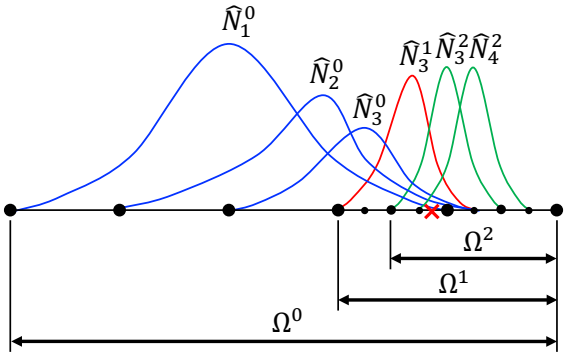


Figure 6.18. An one-dimensional, three-level cubic THB-spline. There are six active THB-spline basis functions at the cross point.

$$\mathbf{R}^1 = \mathbf{R}^2 = \frac{1}{8}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}.$$

2. *Initialization*:

$$\mathbf{C}^0 = \widetilde{\mathbf{X}}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

3. *First Iteration*:

$$(\mathbf{I}^1 - \mathbf{X}^1)\mathbf{R}^1\mathbf{C}^0 = \frac{1}{8}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & 6 & 1 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

Thus,

$$\mathbf{C}^1 = \frac{1}{8}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 0 & 0 & 8 \\ 0 & 0 & 4 & 0 \end{bmatrix}.$$

4. *Second Iteration*:

$$(\mathbf{I}^2 - \mathbf{X}^2)\mathbf{R}^2\mathbf{C}^1 = \frac{1}{64}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 0 & 0 & 8 \\ 0 & 0 & 4 & 0 \end{bmatrix}$$
$$= \frac{1}{64}\begin{bmatrix} 1 & 30 & 25 & 8 \\ 0 & 16 & 16 & 32 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus,

$$\mathbf{C}^2 = \frac{1}{64}\begin{array}{c} \begin{matrix} \widehat{N}_1^0 & \widehat{N}_2^0 & \widehat{N}_3^0 & \widehat{N}_3^1 & \widehat{N}_3^2 & \widehat{N}_4^2 \end{matrix} \\ \begin{bmatrix} 1 & 30 & 25 & 8 & 0 & 0 \\ 0 & 16 & 16 & 32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 64 \end{bmatrix} \end{array}.$$

It may be noticed that the row sum of $\mathbf{C}^2$ is equal to one. Therefore, when a B-spline basis matrix $\mathbf{N}^2$ is multiplied with $\mathbf{C}^2$, every basis function in $\mathbf{N}^2$ is partitioned into parts of the THB-spline basis functions.

7. IMPLEMENTATION: HIERARCHICAL DESIGN AND ANLYSIS CODE

The techniques and algorithms described earlier in Chapters 2 to 6 are implemented in an Object Oriented Fortran Based Hierarchical Design and Anlysis Code (OOF-HiDAC). The OOF-HiDAC relies on the Fortran 2008 standard [113], with its advantages derived from the high efficiency of compiled languages and the code reusability of object-oriented programming. This chapter describes the outstanding features and architecture of the code. A typical analysis flow using the code is also provided.

7.1 Code Features

Fortran, considered to be the first widely used programming language in the world, remains a popular choice for scientific programming due to several reasons. First, as a compiled language, Fortran usually generates faster and more efficient code than the interpreted languages. Compared to C/C++, Fortran has built-in vector/matrix operations, which is very appealing for finite element and isogeometric analysis. Moreover, modern Fortran standards (Fortran 2003/2008) introduce many object-oriented features, which provide more flexibility in programing and produces more succinct code.

The OOF-HiDAC contains more than 30,000 lines of Fortran 2008 code, and has been used to solve many moving boundary problems including crack propagation, solidification and shape optimization. Several features of the code are highlighted as follows:

1. *Problem Dimension Independence*: All the classes and functions are generically implemented to support one-, two- and three-dimensional problems.

2. *Polymorphism*: Sibling types are encapsulated with a unified function interface. For instance, both the `NURBS_Approximation` and the `THBS_Approximation` in-

herit from the same abstract type `Approximation`, and therefore share the same function interface for higher level modules such as *Domain*. The Domain module can freely switch between these approximations without concerning possible interface changes.

3. *Hybrid OpenMP/MPI Parallelism*: In order to have the capability to solve large problems, parallel computing [114] is utilized in matrix assembly and system solution. In the matrix assembly phase, the calculation of each elemental matrix is independent and therefore can be parallelized. Two different parallel programing paradigms, Open Multi-Processing (OpenMP) [115] and Message Passing Interface (MPI) [116], are employed to facilitate the matrix assembly. The OpenMP provides many user-friendly directives for quick implementation but only works for shared memory machines, whereas the MPI is capable for large distributed memory environments such as supercomputers. As for the system solving phase, the OOF-HiDAC relies on the MUltifrontal Massively Parallel sparse direct Solver (MUMPS) [117, 118] which is based purely on the MPI.

4. *Visualization Support*: A VTK writer is included in the code for the convenience of visualizing two- and three-dimensional meshes, geometries and behavioral fields. The writer generates unstructured, XML format, binary VTK files which can be directly read by many visualization softwares such as *Paraview* or *Techplot*.

7.2 Architecture

The architecture of OOF-HiDAC is schematically shown in Figure 7.1. The key modules are briefly described below.

1. *Approximation*: The Approximation module contains several spline types, including Bezier spline, NURBS and THB-spline. These splines provide ge-

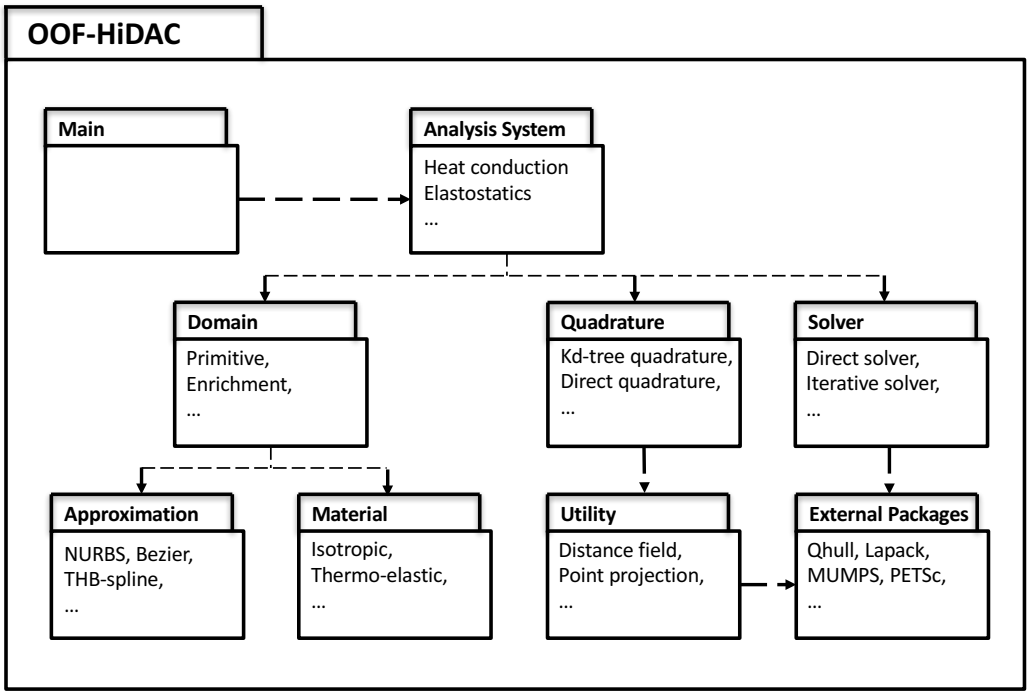


Figure 7.1. Architecture and design of OOF-HiDAC.

ometry representation and behavioral approximation space for the isogeometric analysis. Furthermore, they are extended from the same abstract type `Approximation` which defines the following methods:

- getValue
- getDerivative
- getJacobian

Two additional methods are implemented for NURBS and THB-splines to support matrix assembly:

- getNmatrix
- getBmatrix

2. *Material*: Material types and properties are handled in this module. The module defines an abstract type `Material` which is next extended to a variety of materials (e.g., elastic, thermal, etc.) by adding different material properties.

3. *Domain*: This module defines an analysis domain consisting of primitives and enrichments. In the context of enriched isogeometric analysis, the first primitive provides a base approximation of the underlying domain, meanwhile, the enrichments contain behavior-dependent enriching degrees of freedom. Additional primitives can be added to the domain to model possible heterogeneities such as holes and stiffeners.

4. *Utility*: The proposed algebraic distance field and point projection are implemented in this module. The algebraic distance calculation requires a robust convex hull algorithm. Benefiting from the interoperability with C [113], the OOF-HiDAC is directly linked to the Qhull [119] C library for convex hull construction. This module also contains the aforementioned VTK writer and many other mathematical functions.

5. *Quadrature*: The Quadrature module mainly provides two numerical integration techniques – the standard Gaussian quadrature and the kd-tree based adaptive quadrature. The latter method requires distance calculation which is enabled by the Utility module.

6. *Solver*: The linear system for a boundary value problem is assembled and solved in this module. There are two solvers available in the current implementation – Lapack [120] and MUMPS. While the Lapack introduces less amount of computational overhead, it uses dense matrices and is only suitable for small problems ($n_{\text{DOF}} < 20000$). The MUMPS has a specific representation for sparse matrices and is therefore suitable for much larger problems. To enable a seamless connection to other popular solvers such as PETSc [111] in the future, an abstract type `Solver` is defined in the module as a portal for all numerical solvers. The following abstract methods are declared in the `Solver` type:

- initialize
- assemble
- solve
- postprocess
- finalize

7. *Analysis System*: All the problem-dependent systems are implemented in this module. The currently available systems include heat conduction, phase transition, elastostatics, fracture and sensitivity analysis. Different systems consist of different boundary conditions, materials, and enrichment types. To communicate with the Solver module through a unified interface, we designed another abstract type `Analysis` as well as the associated abstract methods as follows:

   - getInitializationInfo
   - getElementalMatrix
   - getElementalRHS
   - updateField

   These methods are separately implemented in each system based on the physics of that system.

7.3 Analysis Flow

Figure 7.2 shows a typical analysis flow for heat conduction problem. Given a problem domain, an appropriate approximation is first constructed. The user specified material properties are assigned to the domain. Meanwhile, enrichments are generated for each immersed and external boundary, followed by assignment of boundary conditions. The domain and enrichments are next composed into an augmented domain. If a THB-spline is chosen as the approximation, a hierarchical mesh generation is also carried-out in this step. Once the mesh, material properties and boundary

conditions are well defined, a linear system for the problem is parallelly assembled and solved. It may be desirable to use the kd-tree based adaptive quadrature to improve the integration accuracy during matrix assembly. Eventually, the results are output to a VTK file and displayed through visualization softwares.

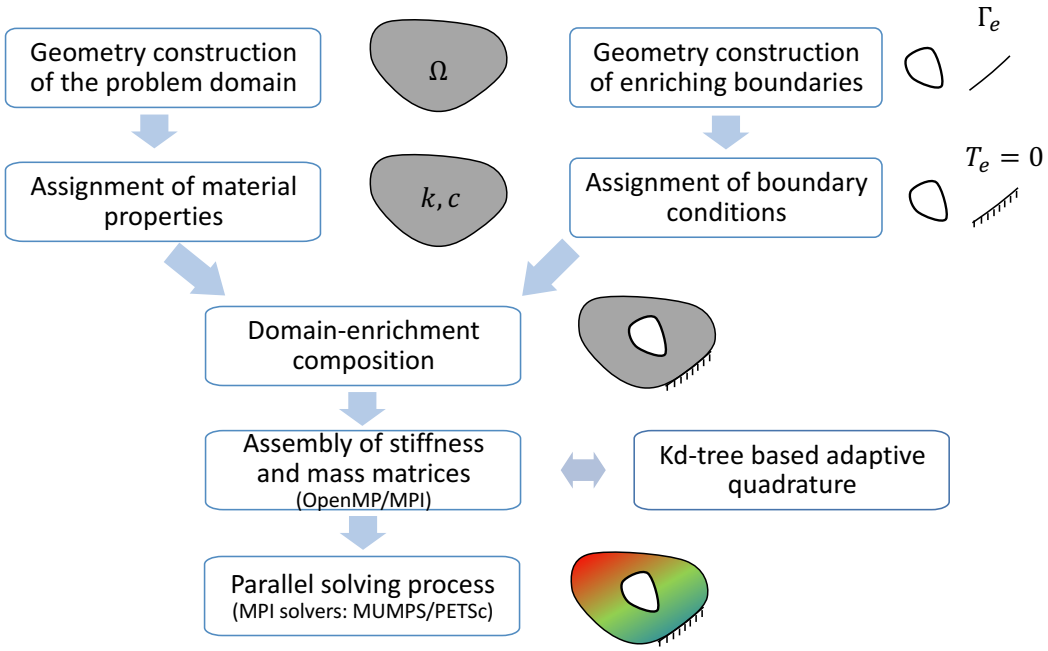


Figure 7.2. Analysis flow for heat conduction problem.

8. MODELING OF STATIONARY AND GROWING CRACKS

The proposed enriched isogeometric analysis is first applied to crack modeling in this chapter. The crack face is enriched with the Heaviside function to represent the displacement discontinuity, while the crack tips are enriched with asymptotic displacement field functions to reproduce the tip singularity. The stress intensity factors (SIFs) are chosen as tip degrees of freedom such that the SIFs can be directly obtained without post-processing. The proposed method is demonstrated through several static and quasi-static crack growth problems.

8.1 Introduction to Crack Modeling

The analysis of failure modes, including fracture, is of great importance to assess the performance of many engineering materials and structures. Crack modeling has been an active research topic in the computational mechanics, and particularly, using the finite element method. The classical finite element method requires the crack face be aligned with the element edges and the mesh be regenerated as the crack propagates. To alleviate the issues, many mesh-free methods, including Element Free Galerkin (EFG) method [37], Reproducing Kernel Particle Method (RKPM) [38] and Meshless Local Petrov-Galerkin (MLPG) method [39] were proposed. Despite the avoidance of a structured mesh, the mesh-free methods introduce new challenges in the numerical integration and application of boundary conditions. Belytschko et al. [3] developed the eXtended Finite Element Method (XFEM) in which the mesh is retained for convenience of integration, while enriching degrees of freedom are added locally to capture the crack behavior. In this manner, only the additional degrees of freedom rather than the mesh need to be updated during crack growth.

Recently, several crack modeling techniques were developed within the isogeometric framework. Verhoosel et al. [121] utilized locally discontinuous T-splines to model cohesive cracks. The discontinuity was created by knot insertion and needed to be updated at every propagation step. Borden et al. [29, 30] combined IGA with phase field method to model quasi-static and dynamic brittle fracture. While the phase-field method naturally enables crack branching and coalescence, the governing equations are usually non-convex and non-linear, and are therefore difficult to solve. Following the philosophy of XFEM, the eXtended Isogeometric Analysis (XIGA) method [10–12] was proposed by adding enriching degrees of freedom to local domain control points. Since the additional degrees of freedom are on the domain, the necessity to identify new enriching nodes still exists in the crack propagation.

In the enriched isogemetric framework, the crack-dependent degrees of freedoms are directly associated with crack control points such that no enriching node identification is needed. Tambat and Subbarayan [13, 14] successfully modeled stationary cracks by enriching the crack face with the Heaviside function. However, the suggested approximation only allows homogeneous boundary conditions. In the current study, a new enriched isogeometric approximation is developed to account for mixed-mode cracks subjected to non-homogeneous boundary conditions. Furthermore, inspired by the research work of Liu et al. [122], we use the SIFs as the tip degrees of freedom and modify the William's crack tip solution [123] to form the tip enriching functions. Thus, the SIFs can be directly extracted from the problem solution without *a-posteriori* calculation of the path independent integrals [124].

8.2 Isogeometric Formulation

The enriched isogeometric approximation is derived based on linear elastic fracture mechanics (LEFM). The enriching functions that capture the crack face discontinuity and crack tip asymptotic field are first described. The approximation is next discretized to enable numerical analysis.

8.2.1 Governing Equations

Consider a problem domain Ω with an internal crack Γ_c as illustrated in Figure 8.1. The equilibrium equation is given by:

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \mathbf{0} \qquad \text{in } \Omega \tag{8.1}$$

where, $\boldsymbol{\sigma}$ and $\boldsymbol{b}$ are stress and body force, respectively. Classical Dirichlet and Neumman boundary conditions are enforced on the domain boundary and the crack face:

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \qquad \text{on } \Gamma_u \tag{8.2a}$$

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \qquad \text{on } \Gamma_t \tag{8.2b}$$

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \mathbf{0} \qquad \text{on } \Gamma_c^+ \text{ and } \Gamma_c^- \tag{8.2c}$$

where, $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{t}}$ are prescribed displacement and traction, respectively. $\boldsymbol{n}$ is the outward normal of surface. The weak form of Eq. (8.1), i.e., the principle of virtual work, is written as:

$$\int_\Omega \boldsymbol{\sigma} : \delta\boldsymbol{\epsilon} \, \mathrm{d}\Omega = \int_\Omega \boldsymbol{b} \cdot \delta\boldsymbol{u} \, \mathrm{d}\Omega + \int_{\Gamma_t} \bar{\boldsymbol{t}} \cdot \delta\boldsymbol{u} \, \mathrm{d}\Gamma \tag{8.3}$$

where, $\delta\boldsymbol{\epsilon}$ and $\delta\boldsymbol{u}$ are compatible virtual strain and displacement, respectively.

8.2.2 Construction of the Enriched Approximation

As described earlier in Eq. (2.11), the behavioral field in the presence of one enrichment can be constructed as follows:

$$f(\boldsymbol{x}) = (1 - w) f_\Omega(\boldsymbol{x}) + w f_\Gamma(\mathcal{P}(\boldsymbol{x})). \tag{8.4}$$

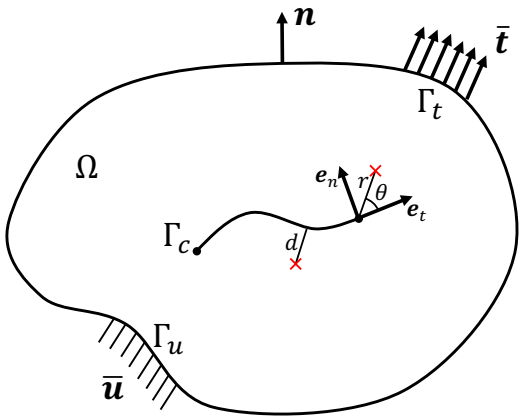


Figure 8.1. Definition of the problem domain and boundaries.

In the proposed method, the displacement field is approximated as:

$$\boldsymbol{u} = \begin{Bmatrix} u_x \\ u_y \end{Bmatrix} = \boldsymbol{u}^c(\boldsymbol{x}) + (w^e - w^t)\mathcal{H}(\boldsymbol{x})\boldsymbol{c}^e(\mathcal{P}(\boldsymbol{x})) + w^t\boldsymbol{k}^t(r,\theta) \tag{8.5}$$

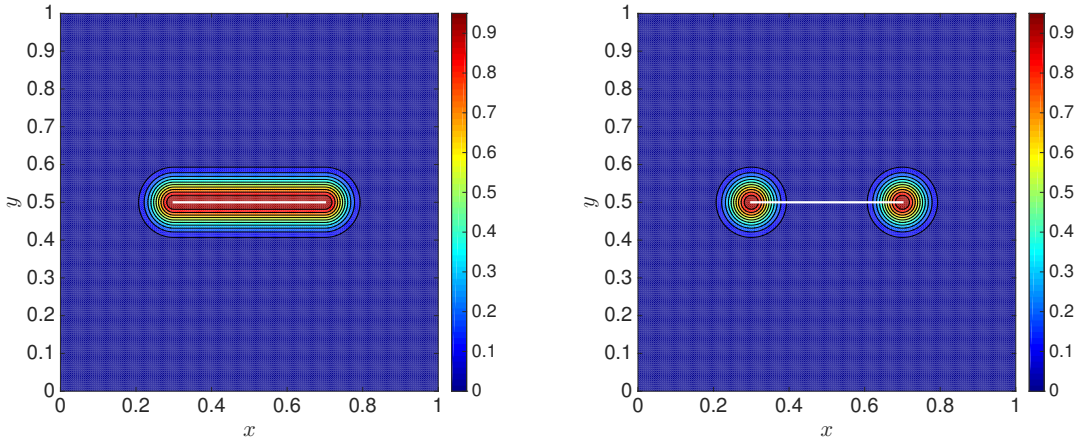
where, the continuous approximation $\boldsymbol{u}^c$ is non-vanishing on the crack to account for all possible rigid body modes as well as the deformation modes including the T-stress. The enriching approximation $\boldsymbol{c}^e$, along with the Heaviside function $\mathcal{H}(\boldsymbol{x})$, models the strong displacement discontinuity across the crack face. w^e and w^t are the weight functions depending on the distance to the crack face (d) and the crack tips (r), respectively. As suggested in Eq. (2.12), the weight functions can take the exponential forms:

$$w^e = w^e(d(\boldsymbol{x})) = e^{-\left|\frac{d(\boldsymbol{x})}{d_s}\right|^\mu} \tag{8.6a}$$

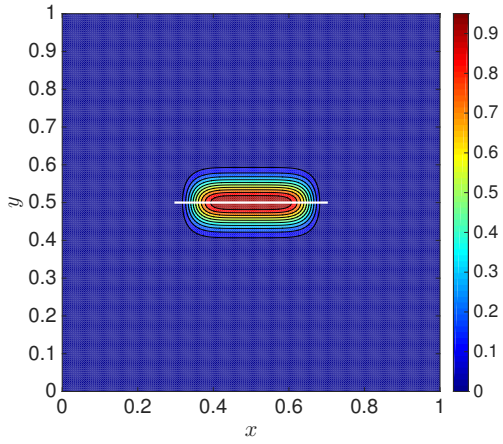
$$w^t = w^t(r(\boldsymbol{x})) = e^{-\left|\frac{r(\boldsymbol{x})}{d_s}\right|^\mu}. \tag{8.6b}$$

The Eq. (8.6a) for a line crack is shown in Figure 8.2. It may be noticed that the effective weight associated with the crack face is chosen to be $w^e - w^t$ such that it

vanishes at the crack tips (see Figure 8.2c). Therefore, the enrichment term $\mathcal{H}\boldsymbol{c}^e$ is active near the crack face but excluding the tip region.



(a) w^e (b) w^t



(c) $w^e - w^t$

Figure 8.2. Weight functions (a) w^e, (b) w^t and (c) the effective weight $w^e - w^t$ for the crack face. The given line crack spans from $(0.3, 0.5)$ to $(0.7, 0.5)$. $d_s = 0.06$ and $\mu = 2$ are chosen in the plots.

The enriching function $\boldsymbol{k}^t$ represents the tip asymptotic displacement field. A common choice of the asymptotic field function space is given by [3]:

$$\{g_i(r,\theta)\}_{i=1}^{4} \equiv \left\{ \sqrt{r}\sin\left(\frac{\theta}{2}\right), \sqrt{r}\cos\left(\frac{\theta}{2}\right), \sqrt{r}\sin(\theta)\sin\left(\frac{\theta}{2}\right), \sqrt{r}\sin(\theta)\cos\left(\frac{\theta}{2}\right) \right\}. \tag{8.7}$$

However, the degrees of freedom associated with the enriching functions in Eq. (8.7) are not directly related to the stress intensity factors. In this manner, the calculation of SIFs necessitates an additional post-processing step based on the path independent integrals is necessary. Liu et al. [122] proposed a new class of tip enriching functions where the SIFs themselves are chosen as the degrees of freedom. The proposed asymptotic displacement field takes the form:

$$\boldsymbol{k}^t = \sum_{j=1}^{n_{\text{tip}}} \sum_{i=1}^{n} \begin{bmatrix} g_{11,i}^{(j)} & g_{12,i}^{(j)} \\ g_{21,i}^{(j)} & g_{22,i}^{(j)} \end{bmatrix} \begin{Bmatrix} K_{\text{I},i}^{(j)} \\ K_{\text{II},i}^{(j)} \end{Bmatrix}, \tag{8.8}$$

where, n_{tip} is the number of crack tips and n is the order of expansion. Based on the William's crack tip field [123, 125], the angular functions g_{11i}, g_{12i}, g_{21i} and g_{22i} are given by:

$$\begin{Bmatrix} g_{11,i} \\ g_{12,i} \\ g_{21,i} \\ g_{22,i} \end{Bmatrix} = \frac{r^{i/2}}{2\mu i\sqrt{2\pi}} \begin{Bmatrix} \left[\kappa + \frac{i}{2} + (-1)^i\right]\cos\left(\frac{i}{2}\theta\right) - \frac{i}{2}\cos\left(\frac{i-4}{2}\theta\right) \\ \left[\kappa + \frac{i}{2} - (-1)^i\right]\sin\left(\frac{i}{2}\theta\right) - \frac{i}{2}\sin\left(\frac{i-4}{2}\theta\right) \\ \left[\kappa - \frac{i}{2} - (-1)^i\right]\sin\left(\frac{i}{2}\theta\right) + \frac{i}{2}\sin\left(\frac{i-4}{2}\theta\right) \\ -\left[\kappa - \frac{i}{2} + (-1)^i\right]\cos\left(\frac{i}{2}\theta\right) - \frac{i}{2}\cos\left(\frac{i-4}{2}\theta\right) \end{Bmatrix}. \tag{8.9}$$

Following this procedure, the SIFs can be directly obtained from the problem solution. As shown in [125], the $K_{\text{I},1}$ and $K_{\text{II},1}$ represent the mode I and II SIFs in homogeneous, isotropic materials.

8.2.3 Discretized Equations

The discretization of $\boldsymbol{u}^c$ and $\boldsymbol{c}^e$ are chosen to be of the following form:

$$\boldsymbol{u}^c = \sum_{i=1}^{n^c} N_i^c \boldsymbol{u}_i^c = [\mathbf{N}^c]\{\boldsymbol{\upsilon}^c\} \tag{8.10a}$$

$$\boldsymbol{c}^e = \sum_{i=1}^{n^e} N_i^e \boldsymbol{c}_i^e = [\mathbf{N}^e]\{\boldsymbol{\gamma}^e\} \tag{8.10b}$$

where, N_i^c and N_i^e are the NURBS basis functions defined on the underlying domain and the enriching lower-dimensional entity, respectively. The $\boldsymbol{k}^t$ is self-discretized, and can be rearranged into a grand matrix-vector multiplication as follows:

$$\boldsymbol{k}^t = \begin{bmatrix} f_{11,1}^{(1)} & f_{12,1}^{(1)} & f_{11,2}^{(1)} \cdots & f_{11,r}^{(1)} & f_{12,n}^{(1)} & f_{11,1}^{(2)} & \cdots & f_{12,n}^{(2)} \\ f_{21,1}^{(1)} & f_{22,1}^{(1)} & f_{21,2}^{(1)} \cdots & f_{21,r}^{(1)} & f_{22,n}^{(1)} & f_{21,1}^{(2)} & \cdots & f_{22,n}^{(2)} \end{bmatrix} \begin{Bmatrix} K_{\mathrm{I},1}^{(1)} \\ K_{\mathrm{II},1}^{(1)} \\ K_{\mathrm{I},2}^{(1)} \\ \vdots \\ K_{\mathrm{I},n}^{(1)} \\ K_{\mathrm{II},n}^{(1)} \\ K_{\mathrm{I},1}^{(2)} \\ \vdots \\ K_{\mathrm{II},n}^{(2)} \end{Bmatrix} \equiv [\mathbf{N}^t]\{\boldsymbol{\kappa}^t\}. \tag{8.11}$$

With the discretization given in Eqs. (8.10) and (8.11), Eq. (8.5) can be rewritten in a matrix form as

$$\boldsymbol{u} = \begin{bmatrix} \mathbf{N}^c & (w^e - w^t)\mathcal{H}\mathbf{N}^e & w^t\mathbf{N}^t \end{bmatrix} \begin{Bmatrix} \boldsymbol{\upsilon}^c \\ \boldsymbol{\gamma}^e \\ \boldsymbol{\kappa}^t \end{Bmatrix} \equiv [\mathbf{N}]\{\boldsymbol{d}\}. \tag{8.12}$$

The corresponding strain field is given by

$$\boldsymbol{\epsilon} = \nabla_s \boldsymbol{u} \equiv [\mathbf{B}]\{\boldsymbol{d}\} \tag{8.13}$$

where, ∇_s is the symmetric gradient operator defined as

$$\nabla_s = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \tag{8.14}$$

The $\mathbf{B}$ matrix takes the following form:

$$\mathbf{B} = \begin{bmatrix} [\nabla_s \mathbf{N}^c]^T \\ (w^e - w^t)\mathcal{H}[\nabla_s \mathbf{N}^e]^T + \mathcal{H}[\mathbf{N}^e]^T[\nabla_s (w^e - w^t)]^T \\ w^t[\nabla_s \mathbf{N}^t]^T + [\mathbf{N}^t]^T[\nabla_s w^t]^T \end{bmatrix}^T . \tag{8.15}$$

The final discrete form of Eq. (8.3) can be written as

$$\mathbf{K}\boldsymbol{d} = \boldsymbol{f}_t + \boldsymbol{f}_b \tag{8.16}$$

where,

$$\mathbf{K} = \int_\Omega [\mathbf{B}]^T[\mathbf{D}][\mathbf{B}] \, \mathrm{d}\Omega \tag{8.17a}$$

$$\boldsymbol{f}_t = \int_{\Gamma_t} [\mathbf{N}]^T \{\bar{\boldsymbol{t}}\} \, \mathrm{d}\Gamma \tag{8.17b}$$

$$\boldsymbol{f}_b = \int_\Omega [\mathbf{N}]^T \{\boldsymbol{b}\} \, \mathrm{d}\Omega. \tag{8.17c}$$

8.3 Numerical Examples

The developed methodology is first validated by calculating the SIFs of an inclined crack under uniaxial tension. Next, the domain with a curved crack is analyzed with the proposed algebraic distance and point projection techniques. The last example

shows a quasi-static crack propagation where the crack growth direction can be readily determined by the enriching degrees of freedom.

8.3.1 Inclined Crack under Uniaxial Tension

Figure 8.3 illustrates a square cracked plate with side $W = 100$ and crack length $a/W = 0.06$. The plate is subjected to a uniform tension $\sigma = 1$, and possesses a Young's modulus and Poisson's ratio of $E = 100$ and $\nu = 0.3$, respectively. For an infinite plate under uniaxial loading, the analytical solutions of the Mode I and Mode II SIFs are given by [126]

$$K_{\mathrm{I}} = \sigma\sqrt{a\pi}\cos^2(\beta) \tag{8.18a}$$

$$K_{\mathrm{II}} = \sigma\sqrt{a\pi}\sin(\beta)\cos(\beta) \tag{8.18b}$$

where, β is the incline angle of the crack to the horizontal.

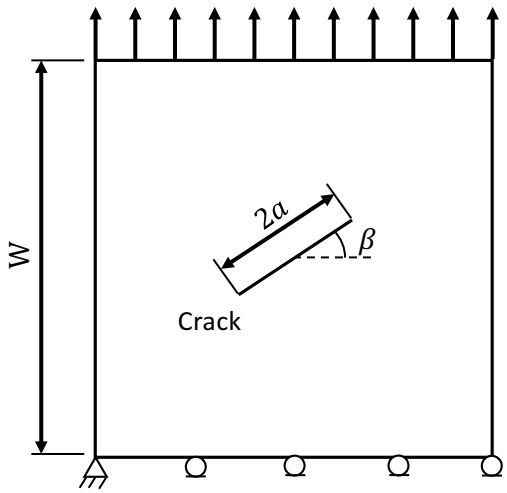


Figure 8.3. An inclined crack under uniaxial tension.

A two-dimensional, seven-level quadratic THB-spline approximation was used to solve the problem (see Figure 8.4). The base level mesh size was chosen as 20×20.

A second order horizontal refinement was carried-out to strengthen the local control. Only the leading terms in the asymptotic solution (Eq. (8.8)) were considered in this problem. The numerical results are shown in Figure 8.5. It can be noticed that a large K_{I} error occurs at small incline angles. This may be mitigated by considering the higher order terms in Eq. (8.8). A good agreement with the analytical solution can be observed when $\beta > 20^{\circ}$.

8.3.2 Plate with a Curved Crack

In this example, a curved crack is present in the center of a square plate with side $W = 20$ (see Figure 8.6). The geometry of the crack is given by

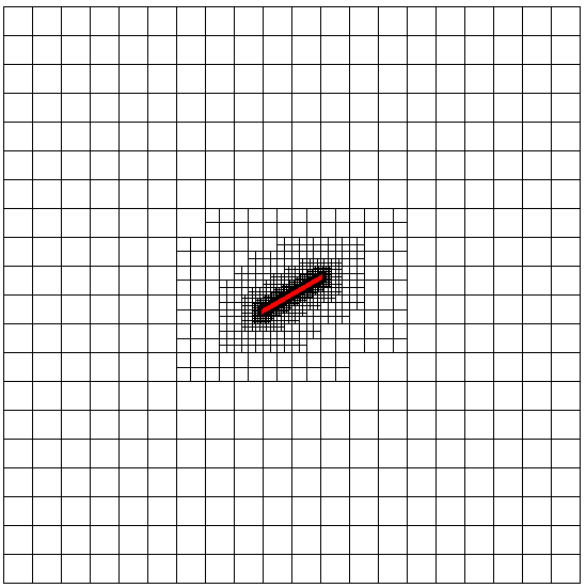
$$y = 10 + \sin\left(\frac{\pi}{4}x\right), \quad x \in [6, 14]. \tag{8.19}$$

A uniaxial loading of $\sigma = 1$ is applied to the plate. The Young's modulus and Poisson's ratio are taken as 100 and 0.3, respectively.

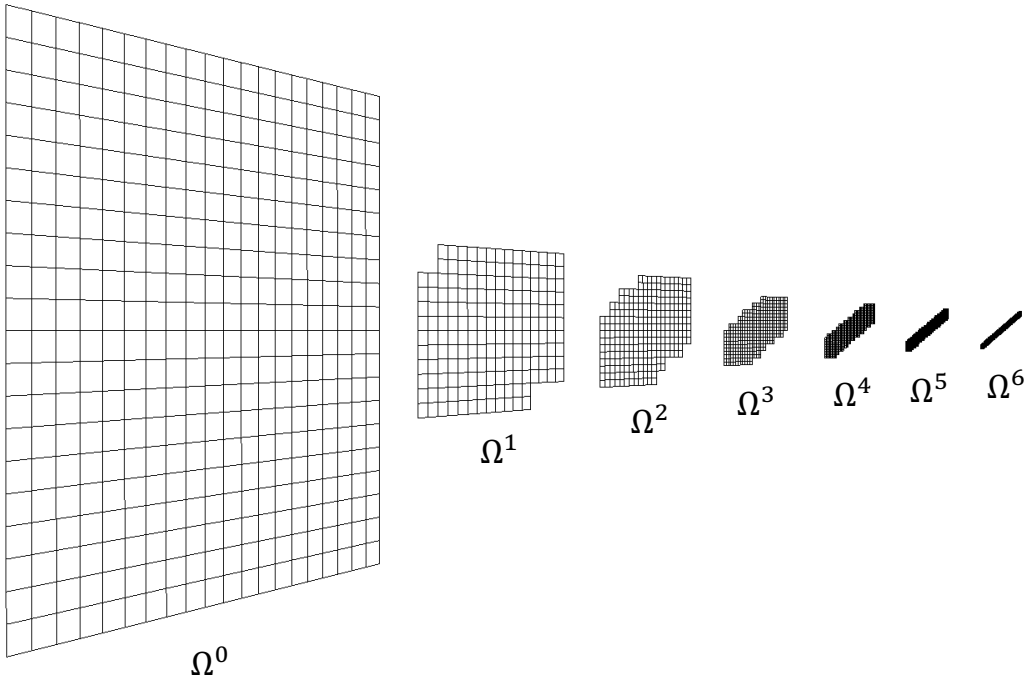
This problem was also solved using a seven-level quadratic THB-spline approximation with a base discretization of 20×20. Figure 8.7 illustrates the hierarchical mesh adaptive to the crack geometry. In addition, the algebraic distance field and point projection are utilized to facilitate the matrix assembly phase. The solved displacement field along y axis is shown in Figure 8.8, where a very smooth displacement field and open crack surfaces can be observed.

8.3.3 Quasi-static Crack Propagation

In this example, a crack in a square plate propagates in a quasi-static state. As shown in Figure 8.9, the plate with side $W = 20$ is subjected to a uniaxial loading

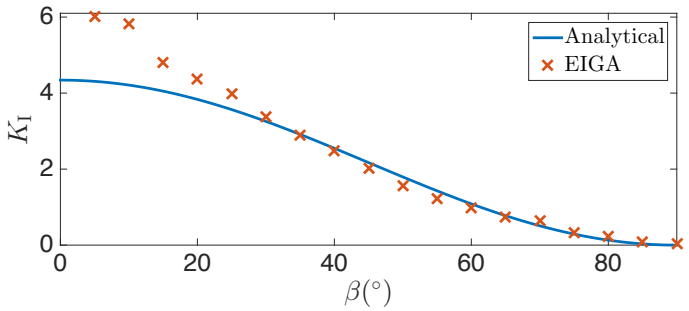


(a) Hierarhical mesh around the crack

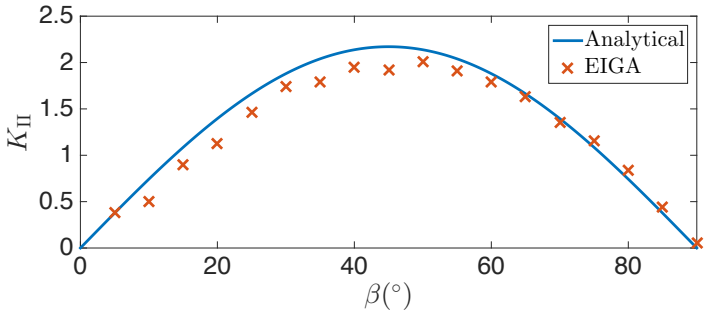


(b) Exploded view of the nested sub-meshes

Figure 8.4. A two-dimensional, seven-level hierarchical refinement of a square domain with an inclined crack: (a) The hierarchical mesh and (b) the sub-meshes in each level. The incline angle β was chosen as 30°.



(a) K_{I}



(b) K_{II}

Figure 8.5. (a) Mode I and (b) Mode II stress intensity factors as a function the crack angle.

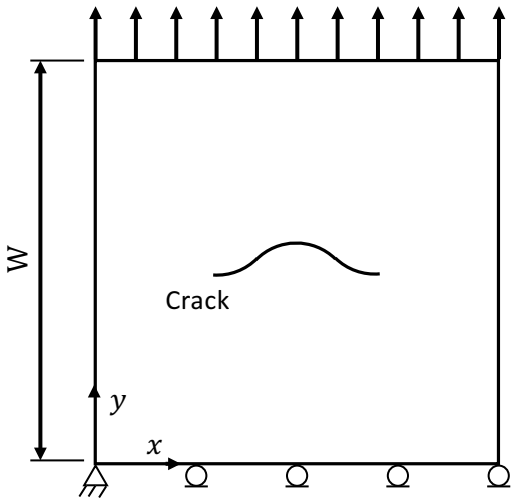
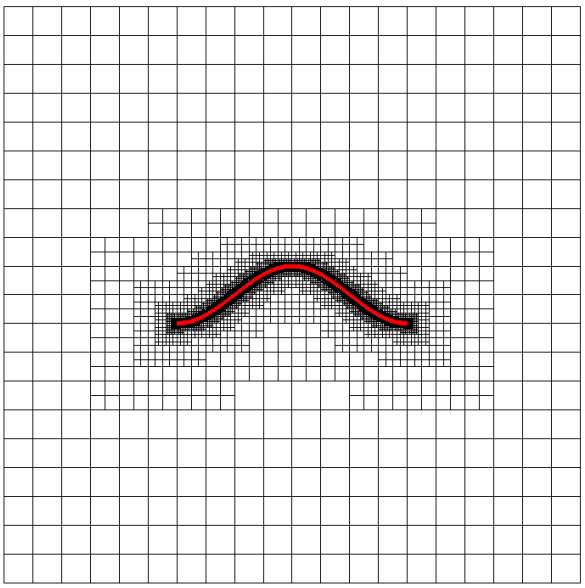
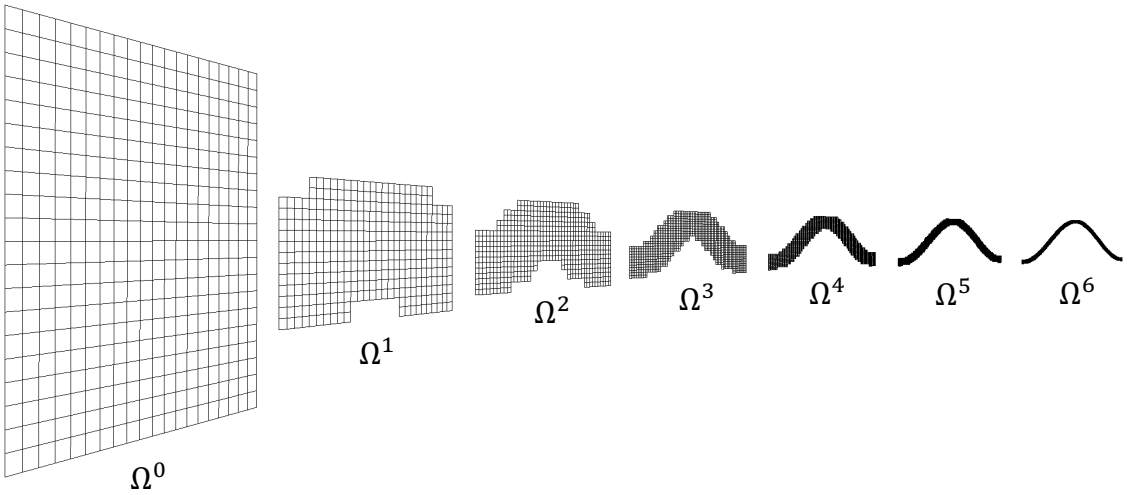


Figure 8.6. A square plate with a sinusoidal crack.



(a) Hierarhical mesh around the crack

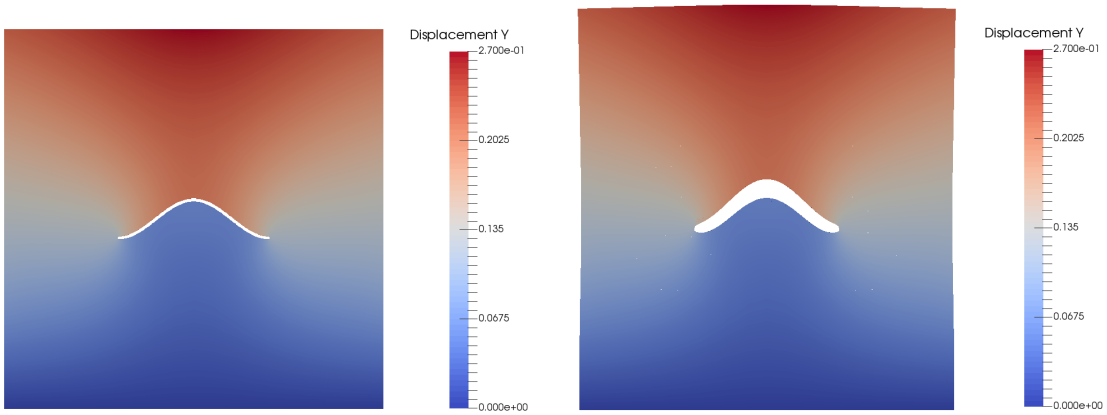


(b) Exploded view of the nested sub-meshes

Figure 8.7. A two-dimensional, seven-level hierarchical refinement of a square domain with a sinusoidal crack: (a) The hierarchical mesh and (b) the sub-meshes in each level.

of $\sigma = 1$. The same Young's modulus ($E = 100$) and Poisson's ratio ($v = 0.3$) are considered. The initial geometry of the crack is given by

$$y = 10 - \cos\left(\frac{\pi}{4}x\right), \quad x \in [7.2, 12.8]. \tag{8.20}$$



(a) Undeformed body (b) Deformed body

Figure 8.8. y-displacement of the problem described in Figure 8.6. The displacement field is shown on (a) an undeformed body and (b) a deformed body.

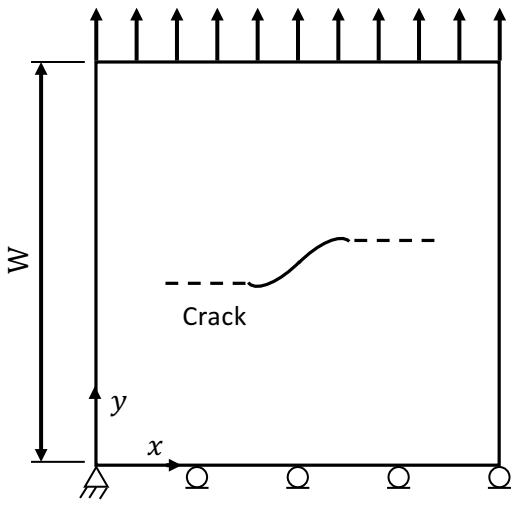
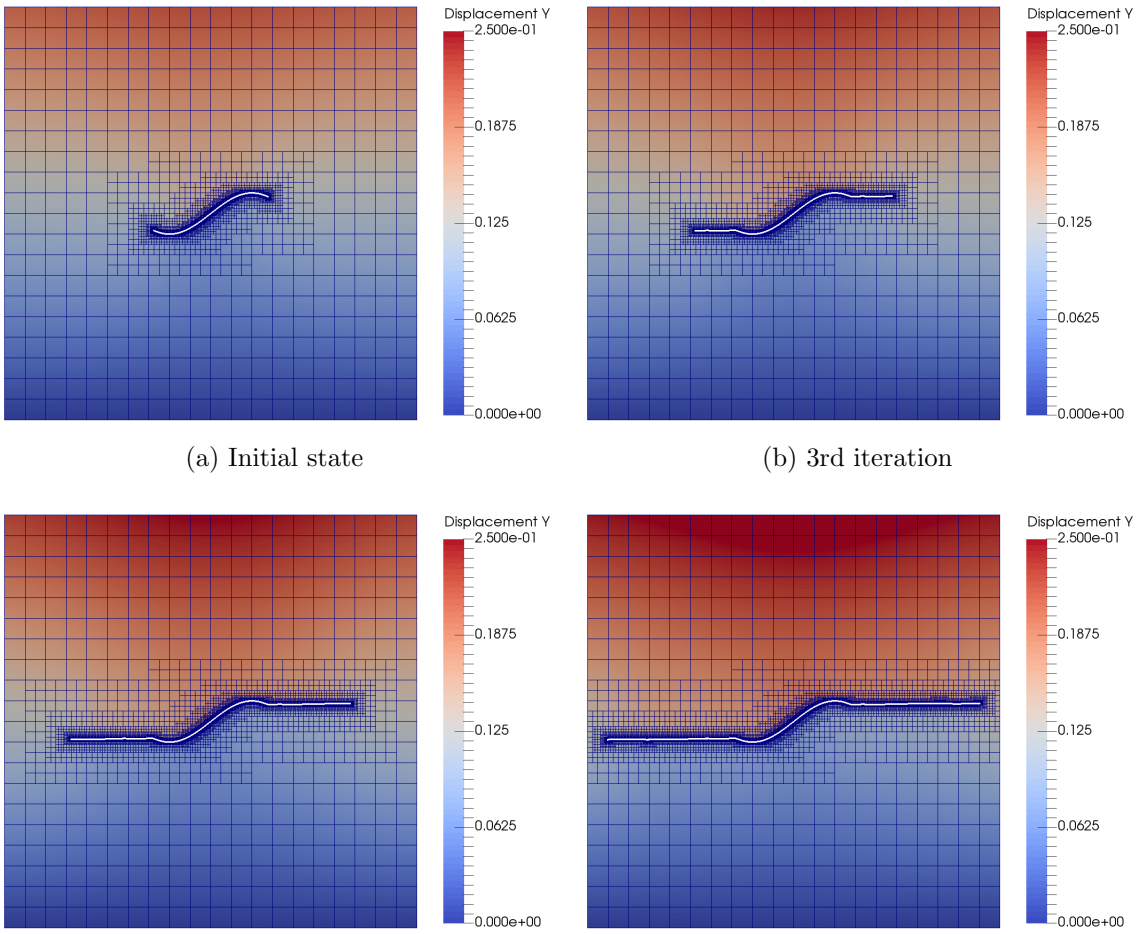


Figure 8.9. A propagating crack in a square plate under uniform tension.

The deflection angle at every propagation step is determined by the maximum tensile stress criterion [126] as follows:

$$\Delta\theta = 2\tan^{-1}\left(\frac{1 \pm \sqrt{1 + 8\left(K_{\mathrm{II}}/K_{\mathrm{I}}\right)^2}}{4K_{\mathrm{II}}/K_{\mathrm{I}}}\right) \tag{8.21}$$

where, the stress intensity factors K_{I} and K_{II} can be directly extracted from the solution without the need for path-independent integral calculation. The simulation totally took 25 steps, and was terminated when the crack reached the plate boundary. Figure 8.10 illustrates the y-displacement and adaptive mesh at four different steps. As expected, the crack propagated horizontally under the model I loading. The von Mises stress on a deformed body is shown in Figure 8.11, where the singular stress is accurately captured with the THB-spline based local refinement.



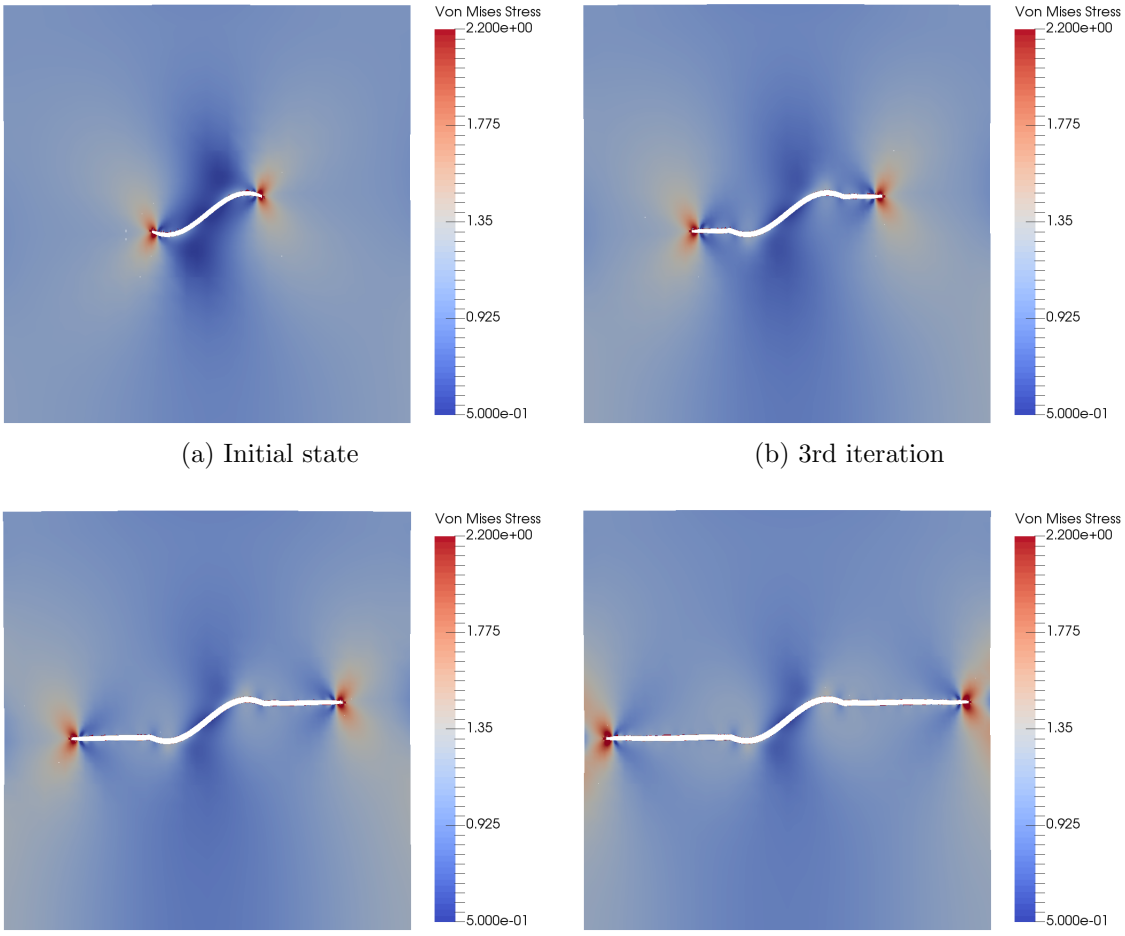
(a) Initial state

(b) 3rd iteration

(c) 7th iteration

(d) Final configuration

Figure 8.10. y-displacement and adaptive mesh at (a) initial state, (b) 8th step, (c) 16th step, and (d) final state (25th step) of the crack propagation. The behavioral field was approximated using a six-level quadratic THB-spline.



(a) Initial state

(b) 3rd iteration

(c) 7th iteration

(d) Final configuration

Figure 8.11. Von Mises stress at (a) initial state, (b) 8th step, (c) 16th step, and (d) final state (25th step) of the crack propagation.

9. SIMULATION OF THE STEFAN PROBLEM

In this chapter, the Stefan problem is solved by enriching an underlying NURBS-based isogeometric approximation with an explicitly defined (sharp) interface on which a hybrid function/derivative condition is isoparametrically described. Since the geometry of the enrichment is explicitly defined, normals and curvatures are explicitly computed at any point on the interface. Thus, the enriched approximation naturally captures the interfacial discontinuity in temperature gradient and naturally enables the imposition of Gibbs-Thomson condition. The blending of the enrichment with the underlying approximation requires an estimate of distance to the enriching geometry from a quadrature point and the parametric value of the footpoint on the enriching geometry. These quantities are computed efficiently using the algebraic distance and point projection methods described in Chapters 3 and 4. Procedures for adaptive time stepping, refinement and coarsening of geometry are developed to increase the stability and efficiency of the developed methodology. Several numerical examples of classical and dendritic Stefan problem are presented to demonstrate the methodology.

9.1 Introduction to the Stefan Problem

The Stefan problem, mathematically describing solidification or melting [127], is a moving boundary problem of importance in many engineering applications. A significant difficulty in solving the Stefan problem, in addition to computationally modeling the moving interface, is in applying the two interface conditions: the Stefan condition and the Gibbs-Thomson condition. The first condition requires numerical approximation to accurately calculate the jump in temperature gradient across the interface, while the second condition requires the accurate computation of curvature

and normal of the phase interface to account for the effects of surface tension and kinetic mobility.

Early literature on computationally modeling the Stefan problem was based on moving mesh methods [15, 16], which may be classified as being geometrically and behaviorally explicit since the mesh is updated to conform to the explicitly (sharply) defined phase interface at each step. Since the interface coincides with element boundary, the Gibbs-Thomson condition can be prescribed on the interface and the governing equations can be solved separately in each phase. However, a significant drawback of these methods is that remeshing is necessary at every time step. In addition, significant distortion of elements typically occurs as interface evolves making it challenging to simulate complex evolution. Diffuse interface methods, or methods that are both geometrically and behaviorally implicit, are often proposed as an alternative to the moving mesh methods to overcome the above challenges. These are briefly reviewed below.

The level set method (LSM, [4]), which is widely used to solve moving boundary problems, was first introduced to solve Stefan problem in [128, 129] in combination with finite difference solution to the underlying field. Ji et al. [6] and Chessa et al. [25] used level sets to evolve solidification fronts defined on a fixed underlying mesh using the eXtended Finite Element Method (XFEM). Zabaras et al. [130] advanced this hybrid XFEM/LSM scheme to solve dendritic Stefan problem. However, there are inherent disadvantages to level set method. First, level set method requires additional, difficult to solve, equation that describe the evolution of the level set: the Hamilton-Jacobi equation is a first-order hyperbolic equation and needs stabilization to minimize the oscillation [28]. In the Stefan problem, the speed field is known only on the phase interface and has to be extended to the rest of the domain necessitating even more governing equations. Often, an auxiliary velocity field equation is posed for this purpose [131]. Further, due to the geometrically implicit nature of the diffuse interface methods, the computed geometric quantities such as normals and curvatures are accurate only in the limit of mesh refinement. Due to its behaviorally implicit na-

ture (being a non-interpolative approximation), additional constraints on the degrees of freedom are needed to enforce the Stefan condition or the Gibbs-Thomson condition [6, 25, 130]. Finally, when piecewise smooth but globally $\mathscr{C}^0$-continuous linear elements are used with level sets, they yield an inaccurate solution when the phase interface is close to element boundaries. This is because the discontinuity in temperature gradient across element boundaries may affect the interfacial Stefan condition, necessitating a $\mathscr{C}^1$-continuous Hermite mesh for accurate solution [6].

The Phase-field method in which the interface is represented by a thin region defined by the phase-field variable, has also been utilized to simulate the dendritic Stefan problem [31–34]. The phase-field model for Stefan problem can be derived from the thermodynamics of phase transition. While, in the phase-field method, the Stefan condition and Gibbs-Thomson condition are typically incorporated into the diffuse governing equations, this does increase the complexity of the governing equations. The phase-field equations are often non-linear and non-convex, and the diffuse transition region has to be very thin to converge to the physical solution, which in turn requires the mesh to span several orders of magnitude in length scale and to be highly refined near the interfacial region. A large number of additional degrees of freedom are required to represent the geometry relative to a sharp interface model in which the boundary is described by an explicit parametrized geometric entity. This makes adaptive mesh refinement as well as efficient computational solution strongly desired. Recently, several adaptive meshing techniques were introduced to alleviate these issues in the simulation of interfacial fluid dynamics: Ceniceros et al. [132] proposed a block-structured, hierarchical mesh refinement method, Yue et al. [133] and Zhou et al [134] introduced an unstructured, grading meshing method with gradually refinement and coarsening. However, either a complex non-linear system or a few linear equations have to be solved in these methods.

In general, the ability to naturally generate $\mathscr{C}^1$ or higher-order continuity approximations is an important and powerful feature of isogeometric analysis (IGA [8,9]). Related to the computational techniques for phase transition problems, in the isogeomet-

ric literature, Gomez et al. [135] developed a phase-field solution to the Cahn-Hilliard equation to effect phase separation. Recently, Tambat and Subbarayan [13] developed explicit lower-dimensional enrichments to impose Dirichlet, Neumann boundary conditions as well as to model discontinuities and singularities. They demonstrated the use of these enrichments for simulating crack propagation. Extending this isogeometric enrichment philosophy, in this work, a hybrid function/derivative enrichment is proposed for solving the Stefan problem. This sharp interface strategy retains and evolves a geometrically explicit interface, eliminates additional level set or phase field variables as well as their associated evolution equations. Also, curvature and normal are readily obtained from the explicitly defined interface geometry. The developed enriched approximation, however, retains the desirable features of behaviorally implicit methods by not requiring conforming discretization or remeshing. Approximations of $\mathscr{C}^1$ or higher order continuity are adopted to assure continuous temperature gradient in underlying domain other than the phase interface. The Dirichlet enrichment enables direct imposition of prescribed Gibbs-Thomson condition, while the Neumann enrichment enables the discontinuity in temperature gradient across phase interface as required by the Stefan condition.

9.2 Governing Equations

The problem domain Ω consists of liquid sub-domain Ω_l and solid sub-domain Ω_s, between which is the phase interface Γ_{int} (see Figure 9.1). $\mathbf{n}_s$ is the outward normal of solid sub-domain. The goal of the Stefan problem is to determine the domain temperature field and to track with time the position of the phase interface [127].

The governing equation for each of the phases is

$$\rho\frac{\partial}{\partial t}(cT) = \nabla\cdot(k\nabla T) + s, \quad \boldsymbol{x}\in\Omega_l\cup\Omega_s \tag{9.1}$$

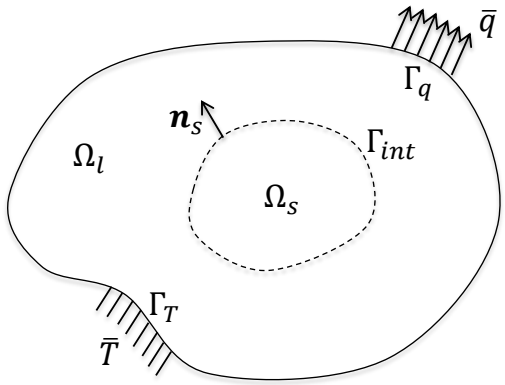


Figure 9.1. Definition of the Stefan problem.

where, parameter ρ, c and k represent material density specific heat and thermal conductivity respectively. c and k are discontinuous across Γ_{int}, whereas ρ was chosen to be the same for both phases to avoid analysis of volume change.

Classical Dirichlet and Neumann boundary conditions are to be enforced on the domain boundary:

$$T = \bar{T}, \quad \boldsymbol{x} \in \Gamma_T \tag{9.2a}$$

$$-k\nabla T \cdot \mathbf{n} = \bar{q}, \quad \boldsymbol{x} \in \Gamma_q. \tag{9.2b}$$

To satisfy energy balance of phase transition, the Stefan condition on the interface also needs to be enforced:

$$[\![q_n]\!] = -\rho L v_n, \quad \boldsymbol{x} \in \Gamma_{int} \tag{9.3}$$

where, L is the latent heat, v_n is normal velocity of interface. The sign of v_n needs care in its definition. Using the convention given in [25], v_n in Eq. (9.3) is defined as

$$v_n = -\frac{[\![q_n]\!]}{\rho L} = -\frac{q_{n_s} + q_{n_l}}{\rho L} = \frac{(k_s \nabla T_s - k_l \nabla T_l) \cdot \boldsymbol{n}_s}{\rho L} \tag{9.4}$$

where, k_s and k_l are the thermal conductivities of the solid and liquid phases respectively. Direction of v_n is defined in reference to $\boldsymbol{n}_s$. That is, the interface moves in the direction $\boldsymbol{n}_s$ with positive v_n (solidifying system) or in the direction $-\boldsymbol{n}_s$ with negative v_n (melting system). Finally, the generalized Gibbs-Thomson condition must be satisfied on the interface [136]:

$$T - T_m + \frac{T_m(c_l - c_s)}{L}\left(T\ln\frac{T}{T_m} + T_m - T\right) + \frac{\gamma(\boldsymbol{n})T_m}{L}\kappa + \epsilon_v(\boldsymbol{n})v_n = 0, \quad \boldsymbol{x} \in \Gamma_{int} \tag{9.5}$$

where, T_m is the melting temperature, $\gamma(\boldsymbol{n})$ is the capillary length indicating surface tension effect and κ is the mean surface curvature. To model the inherent non-equilibrium nature of phase transition, the $\epsilon_v(\boldsymbol{n})v_n$ term is appended to the equation, where $\epsilon_v(\boldsymbol{n})$ is the kinetic mobility coefficient. Assuming $c_l = c_s = c$ and $\epsilon_c(\boldsymbol{n}) = \frac{\gamma(\boldsymbol{n})T_m}{L}$, Eq. (9.5) is reduced to the classical Gibbs-Thomson condition Eq. (9.6a) for the dendritic Stefan problem. Further assumption of $\epsilon_c(\boldsymbol{n}) = \epsilon_v(\boldsymbol{n}) = 0$ gives the constant temperature condition Eq. (9.6b) for the classical Stefan problem.

$$T = T_m - \epsilon_c(\boldsymbol{n})\kappa - \epsilon_v(\boldsymbol{n})v_n, \quad \boldsymbol{x} \in \Gamma_{int} \tag{9.6a}$$

$$T = T_m, \qquad \boldsymbol{x} \in \Gamma_{int}. \tag{9.6b}$$

9.3 Isogeometric Formulation

The Stefan problem raises significant challenges to achieving computational accuracy and efficiency. Here, to accurately capture the discontinuity in temperature gradient across the interface and to directly impose Gibbs-Thomson condition, an isogeometric approximation with hybrid function/derivative enrichment is proposed. This enriched approximation is a smooth blending of $\mathscr{C}^1$ or higher order continuous isogeometric approximation of underlying domain enriched with a $\mathscr{C}^0$-continuous local approximation. Since the enriching field is isogeometrically described on the

enriching geometrical entity, auxiliary governing equations to indirectly describe the evolution of the interface are not necessary.

9.3.1 Construction of the Enriched Approximation

As introduced earlier in Eq. (2.11), the enriched behavioral field can be constructed to form a partition of unity as follows:

$$f(\boldsymbol{x}) = (1-w)f_\Omega(\boldsymbol{x}) + w f_\Gamma(\mathcal{P}(\boldsymbol{x}), d(\boldsymbol{x})) \tag{9.7}$$

where, the weight field w, chosen as Eq. (2.12), satisfies on Γ_{int}:

$$w = 1 \tag{9.8a}$$

$$\|\nabla w\| = 0. \tag{9.8b}$$

In the proposed method, the temperature field is approximated as

$$T(\boldsymbol{x}) = (1-w)T^c(\boldsymbol{x}) + w\left[T^e(\mathcal{P}(\boldsymbol{x})) + G_i^e(\mathcal{P}(\boldsymbol{x}))d(\boldsymbol{x})\right] \tag{9.9}$$

$$G_i^e(\mathcal{P}(\boldsymbol{x})) = \begin{cases} G_l^e(u_f)|_{u_f=\mathcal{P}(\boldsymbol{x})} & \boldsymbol{x} \in \Omega_l \\ G_s^e(u_f)|_{u_f=\mathcal{P}(\boldsymbol{x})} & \boldsymbol{x} \in \Omega_s \end{cases}. \tag{9.10}$$

where, T^c is the continuous temperature field of the underlying domain and T^e is the enriching temperature field defined by

$$T^e(u) = T|_{\Gamma_{int}} = T(\boldsymbol{C}(u)), \tag{9.11}$$

such that Eq. (9.6) is automatically enforced. The subdomain functions G_l^e and G_s^e represent the interfacial temperature gradient on liquid and solid side respectively. Thus, the jump function G_i^e can capture the discontinuity in temperature gradient across the interface. Its value depends on location of $\boldsymbol{x}$ relative to the interface, which can be readily discerned from the signed distance field $d(\boldsymbol{x})$. Per the usual

convention, the sign of $d(\boldsymbol{x})$ is positive for the liquid phase and negative for the solid phase, i.e., $\nabla d|_{\Gamma_{int}} = \boldsymbol{n}_s$. In present work, all the functions with superscript e imply the extension from lower dimensional entity Γ_{int} to domain Ω by point projection $\mathcal{P}(\boldsymbol{x})$. A two-dimensional example of isogeometric enrichment functions is shown in Figure 9.2. The enrichment with both temperature function T^e and gradient-like function G_i^e is termed here as a *hybrid function/derivative enrichment*.

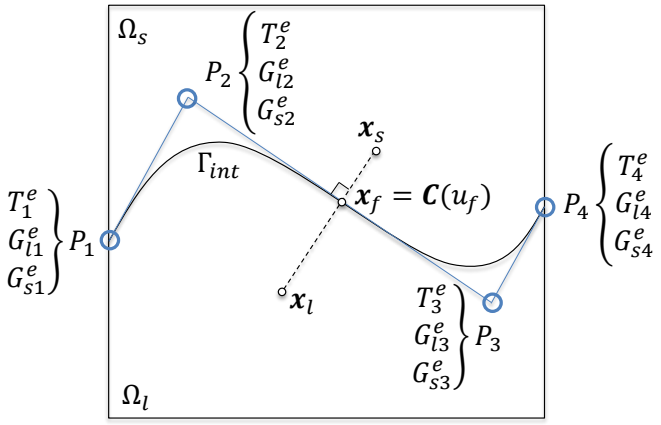


Figure 9.2. Illustration of isogeometric enrichment functions $T^e = \sum_{I=1}^4 N_I^e(u) T_I^e$, $G_l^e = \sum_{I=1}^4 N_I^e(u)(G_l^e)_I$ and $G_s^e = \sum_{I=1}^4 N_I^e(u)(G_s^e)_I$, where T_I^e, $(G_l^e)_I$ and $(G_s^e)_I$ are enriching degrees of freedoms attached to control point P_I and N_I^e is the shape function. $\boldsymbol{x}_f$ is the footpoint of both $\boldsymbol{x}_s$ and $\boldsymbol{x}_l$ on the curve.

The gradient ∇T is obtained from the enriched approximation of Eq. (9.10):

$$\nabla T(\boldsymbol{x}) = (1-w)\nabla T^c + w(\nabla T^e + d\nabla G_i^e + G_i^e \nabla d) + \nabla w(T^e + G_i^e d - T^c). \quad (9.12)$$

Thus, the heat flux on interface is obtained by applying Eq. (9.8):

$$q_n|_{\Gamma_{int}} = -k\nabla T \cdot \boldsymbol{n}|_{\Gamma_{int}} = -k(\nabla T^e + d\nabla G_i^e + G_i^e \nabla d) \cdot \boldsymbol{n}|_{\Gamma_{int}} = -kG_i^e \boldsymbol{n}_s \cdot \boldsymbol{n}. \quad (9.13)$$

Here, $d = 0$ and $\nabla d = \boldsymbol{n}_s$ on the interface were used to arrive at the final result. Also, on the interface, it is shown in [71] that $\nabla T^e \cdot \boldsymbol{n} = 0$ on a parametric curve or surface. Particularly, $\nabla T^e = \mathbf{0}$ can be obtained for the classical Stefan problem with condition Eq. (9.6b).

Finally, one can obtain the normal speed of interface from Eq. (9.4) and Eq. (9.13) as

$$v_n = \frac{k_s G_s^e \boldsymbol{n}_s \cdot \boldsymbol{n}_s + k_l G_l^e \boldsymbol{n}_s \cdot \boldsymbol{n}_l}{\rho L} = \frac{k_s G_s^e - k_l G_l^e}{\rho L}. \tag{9.14}$$

9.3.2 Discretized Equations

The discretization of T^c, T^e and G^e are chosen to be of the following form:

$$T^c = \sum_{I=1}^{n} N_I^c T_I^c = [\mathbf{N}^c]\{\boldsymbol{T}^c\} \tag{9.15a}$$

$$T^e = \sum_{I=1}^{n_e} N_I^e T_I^e = [\mathbf{N}^e]\{\boldsymbol{T}^e\} \tag{9.15b}$$

$$G_i^e = \begin{cases} \sum_{I=1}^{n_e} N_I^e (G_l^e)_I = [\mathbf{N}^e]\{\boldsymbol{G}_l^e\} & \boldsymbol{x} \in \Omega_l \\ \sum_{I=1}^{n_e} N_I^e (G_s^e)_I = [\mathbf{N}^e]\{\boldsymbol{G}_s^e\} & \boldsymbol{x} \in \Omega_s \end{cases}. \tag{9.15c}$$

where, N_I^c and N_I^e are the NURBS basis functions for the underlying domain and the enriching lower-dimensional entity respectively. N_I^c is required to be $\mathscr{C}^1$ or higher order continuous function to ensure continuous temperature gradient in the underlying domain. With the discretization given in Eq. (9.15), Eq. (9.10) can be rewritten in matrix form as

$$\begin{aligned} T(\boldsymbol{x}) &= \begin{bmatrix} (1-w)\mathbf{N}^c & w\mathbf{N}^e & wd\mathbf{N}_l^e & wd\mathbf{N}_s^e \end{bmatrix} \begin{bmatrix} \boldsymbol{T}^c \\ \boldsymbol{T}^e \\ \boldsymbol{G}_l^e \\ \boldsymbol{G}_s^e \end{bmatrix} \\ &= [\mathbf{N}]\{\boldsymbol{d}\} \end{aligned} \tag{9.16}$$

where,

$$\mathbf{N}_l^e = \begin{cases} \mathbf{N}^e & \boldsymbol{x} \in \Omega_l \\ \mathbf{O} & \boldsymbol{x} \in \Omega_s \end{cases}, \qquad \mathbf{N}_s^e = \begin{cases} \mathbf{O} & \boldsymbol{x} \in \Omega_l \\ \mathbf{N}^e & \boldsymbol{x} \in \Omega_s \end{cases}. \tag{9.17}$$

Thus, Eq. (9.12) for temperature gradient can be written in discretized form as

$$\nabla T(\boldsymbol{x}) = [\mathbf{B}]\{\boldsymbol{d}\} \tag{9.18}$$

where,

$$[\mathbf{B}] = \begin{bmatrix} (1-w)[\nabla \mathbf{N}^c]^T - [\mathbf{N}^c]^T[\nabla w]^T \\ w[\nabla \mathbf{N}^e]^T + [\mathbf{N}^e]^T[\nabla w]^T \\ wd[\nabla \mathbf{N}_l^e]^T + [\mathbf{N}_l^e]^T[\nabla (wd)]^T \\ wd[\nabla \mathbf{N}_s^e]^T + [\mathbf{N}_s^e]^T[\nabla (wd)]^T \end{bmatrix}^T. \tag{9.19}$$

Now, the weak form of energy equation (Eq. (9.1)) is:

$$\int_\Omega \delta T \rho \frac{\partial}{\partial t}(cT)\,\mathrm{d}\Omega + \int_\Omega \nabla(\delta T)\cdot(k\nabla T)\,\mathrm{d}\Omega = -\int_{\Gamma_q} \delta T \bar{q}\,\mathrm{d}\Gamma + \int_\Omega \delta T s\,\mathrm{d}\Omega. \tag{9.20}$$

Using the above equation, one can derive the discrete form of the energy equation using a backward Euler time integration scheme [25]:

$$\left(\frac{1}{\Delta t}\mathbf{M} + \boldsymbol{K}\right)\boldsymbol{T}^{n+1} = \frac{1}{\Delta t}\mathbf{M}^*\boldsymbol{T}^n + \boldsymbol{f}_q^{n+1} + \boldsymbol{f}_s^{n+1} \tag{9.21}$$

where,

$$\mathbf{M} = \int_{\Omega} \rho c^{n+1} [\mathbf{N}^{n+1}]^T [\mathbf{N}^{n+1}] \, \mathrm{d}\Omega \tag{9.22a}$$

$$\mathbf{M}^* = \int_{\Omega} \rho c^{n+1} [\mathbf{N}^{n+1}]^T [\mathbf{N}^{n}] \, \mathrm{d}\Omega \tag{9.22b}$$

$$\boldsymbol{K} = \int_{\Omega} k^{n+1} [\mathbf{B}^{n+1}]^T [\mathbf{B}^{n+1}] \, \mathrm{d}\Omega \tag{9.22c}$$

$$\boldsymbol{f}_q^{n+1} = -\int_{\Gamma_q} [\mathbf{N}^{n+1}]^T \bar{q}^{n+1} \, \mathrm{d}\Gamma \tag{9.22d}$$

$$\boldsymbol{f}_s^{n+1} = \int_{\Omega} [\mathbf{N}^{n+1}]^T s^{n+1} \, \mathrm{d}\Omega. \tag{9.22e}$$

The above integrals need extra care due to location-dependent definition of $\mathbf{N}_l^e$ and $\mathbf{N}_s^e$ in Eq. (9.17). It is more convenient to integrate separately in the solid and liquid sub-domains Ω_s and Ω_l respectively. To enable this the integration is carried in the manner illustrated for the mass matrix (Eq. (9.22a)) below in Eq. (9.23). For notational simplicity, superscript $n+1$ is neglected here. The remaining integrals can also be computed in a similar manner.

$$\begin{aligned}
\mathbf{M} = &\int_{\Omega_l} \rho c_l \begin{bmatrix} (1-w)^2[\mathbf{N}^c]^T[\mathbf{N}^c] & w(1-w)[\mathbf{N}^c]^T[\mathbf{N}^e] & wd(1-w)[\mathbf{N}^c]^T[\mathbf{N}^e] & 0 \\ w(1-w)[\mathbf{N}^e]^T[\mathbf{N}^c] & w^2[\mathbf{N}^e]^T[\mathbf{N}^e] & w^2 d[\mathbf{N}^c]^T[\mathbf{N}^e] & 0 \\ wd(1-w)[\mathbf{N}^e]^T[\mathbf{N}^c] & w^2 d[\mathbf{N}^e]^T[\mathbf{N}^c] & (wd)^2[\mathbf{N}^c]^T[\mathbf{N}^e] & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathrm{d}\Omega \\
&+ \int_{\Omega_s} \rho c_s \begin{bmatrix} (1-w)^2[\mathbf{N}^c]^T[\mathbf{N}^c] & w(1-w)[\mathbf{N}^c]^T[\mathbf{N}^e] & 0 & wd(1-w)[\mathbf{N}^c]^T[\mathbf{N}^e] \\ w(1-w)[\mathbf{N}^e]^T[\mathbf{N}^c] & w^2[\mathbf{N}^e]^T[\mathbf{N}^e] & 0 & w^2 d[\mathbf{N}^c]^T[\mathbf{N}^e] \\ 0 & 0 & 0 & 0 \\ wd(1-w)[\mathbf{N}^e]^T[\mathbf{N}^c] & w^2 d[\mathbf{N}^e]^T[\mathbf{N}^c] & 0 & (wd)^2[\mathbf{N}^c]^T[\mathbf{N}^e] \end{bmatrix} \mathrm{d}\Omega.
\end{aligned} \tag{9.23}$$

Consistent with the fixed mesh philosophy, the control net of the underlying domain is fixed during solution process. The moving interface defining the integration sub-domains together with the non-linearity of the enriching fields cause challenges to accurate integration. To resolve this problem, the kd-tree based adaptive quadrature,

as described in Chapter 5, is utilized to integrate over the cells that are intersected by interface. The point containment problem of identifying a point with the solid or liquid subdomain is easy to perform once the signed distance field is algebraically computed.

To obtain the speed at the control points of the enriching entity, the velocity defined by Eq. (9.14) is discretized as follows:

$$v_n = \frac{k_s \sum_{I=1}^{n_e} N_I^e (G_s^e)_I - k_l \sum_{I=1}^{n_e} N_I^e (G_l^e)_I}{\rho L} = \sum_{I=1}^{n_e} N_I^e \frac{k_s (G_s^e)_I - k_l (G_l^e)_I}{\rho L} = \sum_{I=1}^{n_e} N_I^e \bar{v}_I \tag{9.24}$$

where,

$$\bar{v}_I = \frac{k_s (G_s^e)_I - k_l (G_l^e)_I}{\rho L}. \tag{9.25}$$

In order to recover the control point velocity from the scalar speed, the following unconstrained least square problem of lower dimension is solved to determine velocity at control points:

$$\text{minimize} \qquad \int_{\Gamma_{int}} \left\| \sum_{I=1}^{n_e} N_I^e \left(\boldsymbol{v}_I - \bar{v}_I \boldsymbol{n}_s \right) \right\|_2^2 \mathrm{d}\Gamma \tag{9.26}$$

where, $\boldsymbol{v}_I = (v_{Ix}, v_{Iy}, v_{Iz})$. The control point velocities, $\boldsymbol{v}_I$, can be obtained by solving following normal equation:

$$\left(\int_{\Gamma_{int}} [\mathbf{N}^e]^T [\mathbf{N}^e] \, \mathrm{d}\Gamma \right) [\boldsymbol{V}] = \int_{\Gamma_{int}} [\mathbf{N}^e]^T [\mathbf{N}^e] \{ \bar{\boldsymbol{v}} \} \boldsymbol{n}_s \, \mathrm{d}\Gamma \tag{9.27}$$

where,

$$[\boldsymbol{V}] = \begin{bmatrix} v_1^x & v_1^y & v_1^z \\ v_2^x & v_2^y & v_2^z \\ \vdots & \vdots & \vdots \\ v_{n_e}^x & v_{n_e}^y & v_{n_e}^z \end{bmatrix}, \quad \{\bar{\boldsymbol{v}}\} = \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \vdots \\ \bar{v}_{n_e} \end{bmatrix}. \tag{9.28}$$

The normal equation given in Eq. (9.27) is of one dimension lower than the governing equation, Eq. (9.21), since it involves unknowns only on the control points defining the enriching entity. Since the degrees of freedom on the interface $n_e \ll n$, computational cost of solving Eq. (9.27) is negligible compared with solving the governing equation, Eq. (9.21).

9.3.3 Validation of Approximation

The enriched approximation was validated by solving three-dimensional melting system with a planar moving interface. A uniform cube of solid was initially set to the melting temperature $T_m = 0$ as illustrated in Figure 9.3. The constant melting temperature condition, Eq. (9.6b), was enforced on the interface. The temperature on the upper face was raised to a constant value of $T_w = 1$ at time $t = 0$. The latent heat value was chosen as $L = 10$ and all other parameters were set to a unit value.

The analytical and numerical solutions for the position of the interface in z direction as a function of time are shown in Figure 9.4. A good agreement with the analytical solution was observed even when the interface was close to the bottom face. Further, temperature fields corresponding to three time instants are illustrated in Figure 9.5. As can be seen, the spatial variation of the temperature at each instant is also in very good agreement with the analytical solution. In Figure 9.6, an analysis of convergence under four different discretizations is presented. Nearly quadratic rate of convergence was observed.

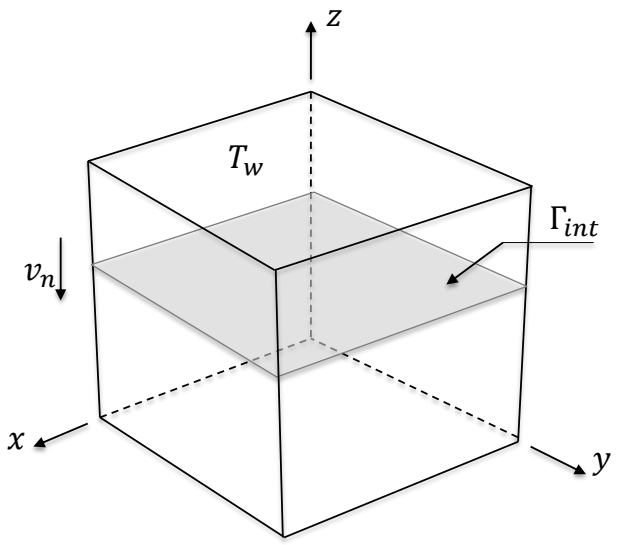


Figure 9.3. Illustration of 3D planar melting problem.

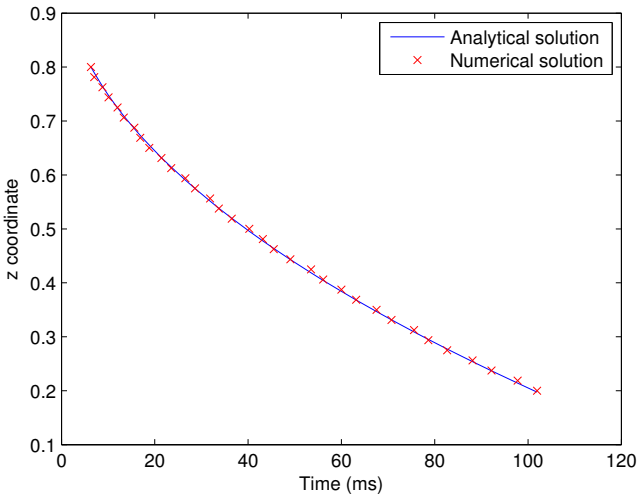
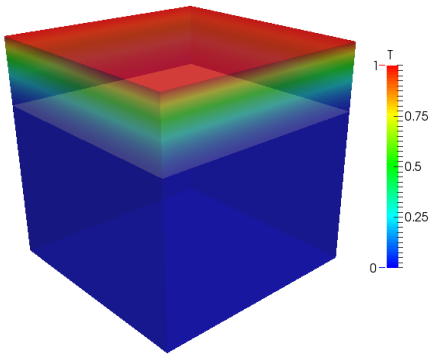
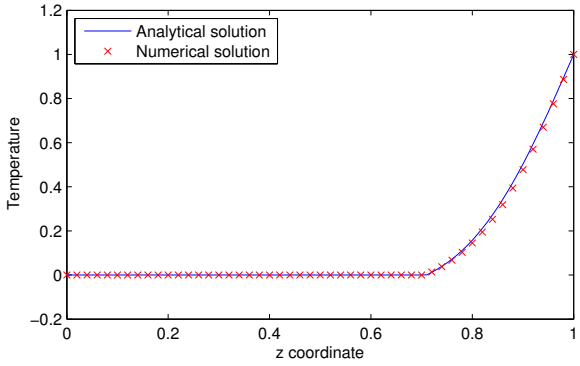


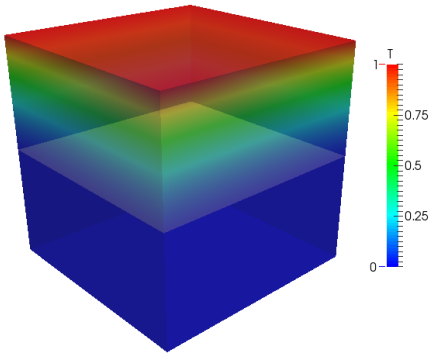
Figure 9.4. Comparison of interface position obtained using the developed method against the Neumann analytical solution. Discretization size of $40 \times 40 \times 40$ control points was used.



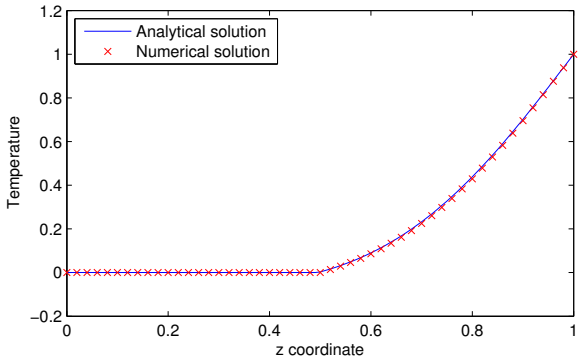
(a) 3D temperature field at $t = 13$ms



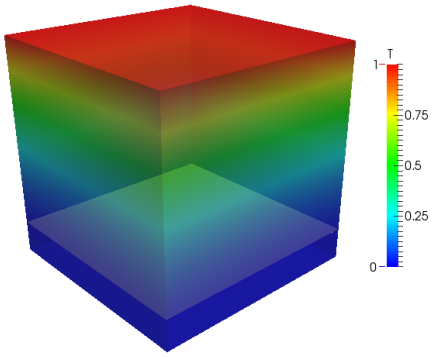
(b) Temperature along z axis at $t = 13$ms



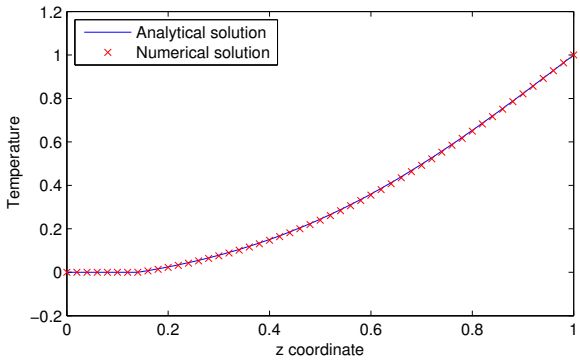
(c) 3D temperature field at $t = 40$ms



(d) Temperature along z axis at $t = 40$ms



(e) 3D temperature field at $t = 117$ms



(f) Temperature along z axis at $t = 117$ms

Figure 9.5. 3D Temperature distribution (a),(c),(e) as well as plots of temperature along z axis compared to the Neumann analytical solution (b),(d),(f) at $t = 13, 40, 117$ms. The plane shown in the interior of the cube indicates the physical position of interface at that instant.

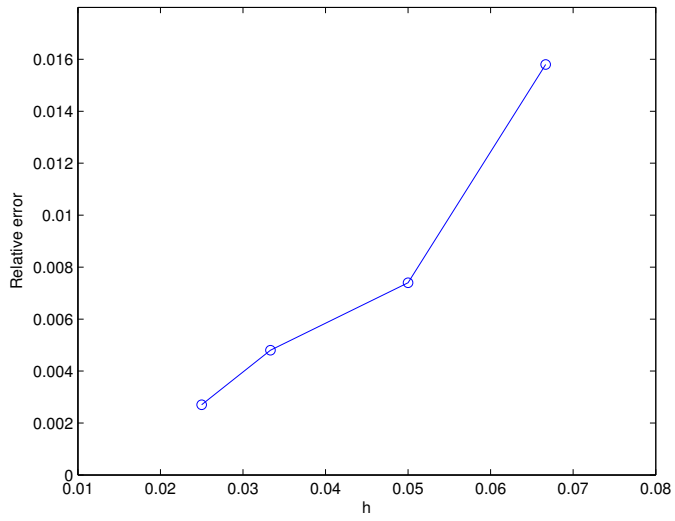


Figure 9.6. Convergence analysis in 3D planar interface example. Nearly quadratic convergence rate was achieved.

9.4 Adaptive Time Stepping and Enrichment

In the Stefan problem, the behavioral field near the interface may undergo significant changes during the course of the interface evolution, causing the interface shape to undergo correspondingly large changes. Thus, there is a need to adaptively refine the representation of the interface during the course of the interfacial shape evolution. In this work, a two-dimensional adaptive refinement and coarsening algorithm is proposed to model the complex evolving interface. An auxiliary adaptive time stepping scheme is developed to ensure geometrical stability and computational robustness.

9.4.1 Adaptive Refinement and Coarsening of Enrichment

In the finite element method, a common adaptive refinement scheme is based on gradient jump across element boundary [35, 137, 138]. Inspired by these studies, an

aposteriori refinement criteria based on jump in heat flux at the interface may be developed as

$$\eta^i = \frac{\max [\![q_n^i]\!] - \min [\![q_n^i]\!]}{\|[\![q_n]\!]\|_\infty} \tag{9.29}$$

where, superscript i represents the index of the knot whose span is $[u_i, u_{i+1}]$. Using the Stefan condition Eq. (9.4) and the convex hull property of the NURBS entity, Eq. (9.29) can be simplified as

$$\eta^i = \frac{\max v_n^i - \min v_n^i}{\|v_n\|_\infty} \geq \frac{\max v_n^i - \min v_n^i}{\|\bar{\boldsymbol{v}}\|_\infty}. \tag{9.30}$$

During adaptive refinement, the knot value $\bar{u} = \frac{1}{2}(u_i + u_{i+1})$ will be inserted into the knot span $[u_i, u_{i+1}]$ when

$$\frac{\max v_n^i - \min v_n^i}{\|\bar{\boldsymbol{v}}\|_\infty} \geq C_g \tag{9.31}$$

where, $C_g \in (0, 1]$ is a refinement coefficient. The smaller the value of the chosen C_g, the more knot spans where new knots are inserted, and therefore more refined the enriching entity becomes. This criteria is equivalent to refining knot spans where large differences in heat flux jump and, therefore, large differences in the evolving speed of the interface occur.

Adaptive coarsening becomes necessary when self-intersection of the enriching entity is imminent. As illustrated in Figure 9.7, a large time step than acceptable results in the self-intersection of the NURBS curve. To eliminate this possibility, a two-dimensional geometrical non-self-intersection (NSI) condition is introduced.

In general, the self-intersection of parametric curve is difficult to detect. Therefore, it is challenging to predict the Δt at which self-intersection would occur. However, for a NURBS entity, due to the convex hull property, the self-intersection becomes possible to detect. In general, two segments of a NURBS curve will not intersect if their corresponding control hulls do not intersect each other. The problem of detecting

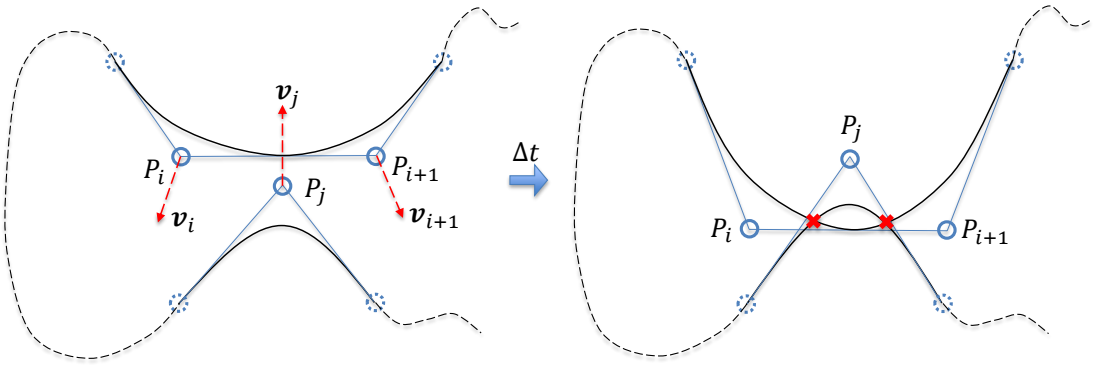


Figure 9.7. Detection of self-intersection. If the control net of a quadratic NURBS curve changes from the left configuration to the one on the right right after Δt, self-intersection will occur. Crosses in the right figure indicate points of self-intersection of the curve. $\boldsymbol{v}_i$, $\boldsymbol{v}_{i+1}$ and $\boldsymbol{v}_j$ are moving velocities of P_i, P_{i+1} and P_j respectively.

the self-intersection of a NURBS curve can therefore be converted into detecting the self-intersection of its control net. In Figure 9.7, consider the two adjacent control points $P_i(x_i, y_i)$ and $P_{i+1}(x_{i+1}, y_{i+1})$ with nodal velocities $\boldsymbol{v}_i = (v_i^x, v_i^y)$ and $\boldsymbol{v}_{i+1} = (v_{i+1}^x, v_{i+1}^y)$ respectively. After time step Δt, another control point $P_j(x_j, y_j)$ intersects the segment connecting P_i and P_{i+1}. One can then derive the equations of motion:

$$k(x_i + v_i^x \Delta t_{ij}) + (1-k)(x_{i+1} + v_{i+1}^x \Delta t_{ij}) = x_j + v_j^x \Delta t_{ij} \tag{9.32a}$$

$$k(y_i + v_i^y \Delta t_{ij}) + (1-k)(y_{i+1} + v_{i+1}^y \Delta t_{ij}) = y_j + v_j^y \Delta t_{ij} \tag{9.32b}$$

where, $k \in [0, 1]$ is the partition coefficient and $\boldsymbol{v}_j = (v_j^x, v_j^y)$. Thus, from Eq. (9.32), one can obtain:

$$\Delta t_{ij} = -\frac{k(x_i - x_{i+1}) + (x_{i+1} - x_j)}{k(v_i^x - v_{i+1}^x) + (v_{i+1}^x - v_j^x)} = -\frac{k(y_i - y_{i+1}) + (y_{i+1} - y_j)}{k(v_i^y - v_{i+1}^y) + (v_{i+1}^y - v_j^y)}. \tag{9.33}$$

Therefore, the two-dimensional non-self-intersection condition is:

$$\Delta t \leq \Delta t_{NSI} \equiv \min_{\substack{i,j \\ \Delta t_{ij}>0}} \{\Delta t_{ij}\}. \tag{9.34}$$

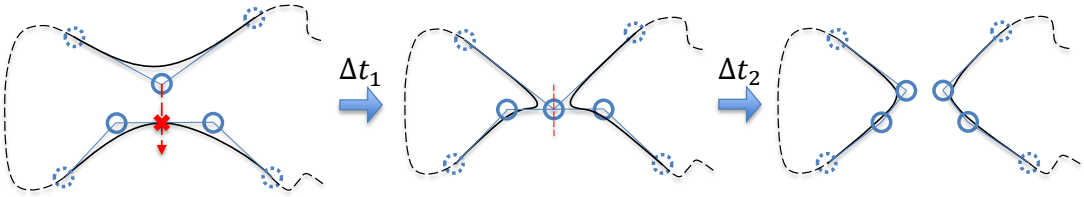
The computational cost of determining Δt_{NSI} is $O(n_e^2)$, and is insignificant since $n_e \ll n$, with n_e being the degrees of freedom of the enriching entity and n that of the underlying domain. If potential self-intersection is detected, then a time step $\Delta t = \Delta t_{NSI}$ is chosen in Eq. (9.34). Note the general intersection event could cause three different types of rearrangement of control nets as well as NURBS curves as illustrated in Figure 9.8:

1. Interfacial rupture increases the genus of the enrichment by one
2. Interfacial coalescence decreases the genus of the enrichment by one
3. The genus is unchanged, but a small number of control points are isolated and removed

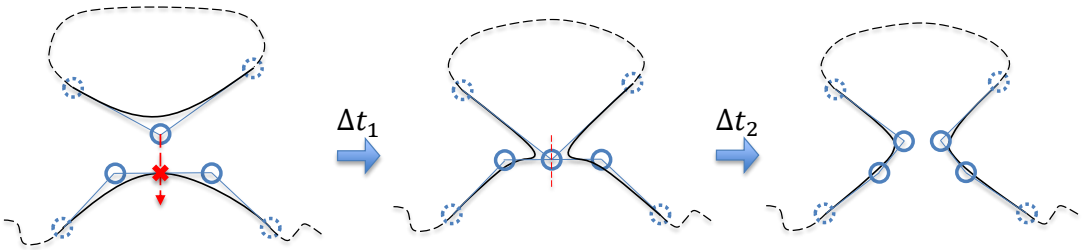
The first and second cases involve multiple enrichments and introduce topological changes. The proposed method can be extended to these cases by enrichment-wise approximation:

$$T(\boldsymbol{x}) = (1 - w_k)T^c(\boldsymbol{x}) + w_k \left[T^{e_k}(P_k(\boldsymbol{x})) + G_i^{e_k}(\mathcal{P}_k(\boldsymbol{x}))d_k(\boldsymbol{x})\right] \tag{9.35}$$

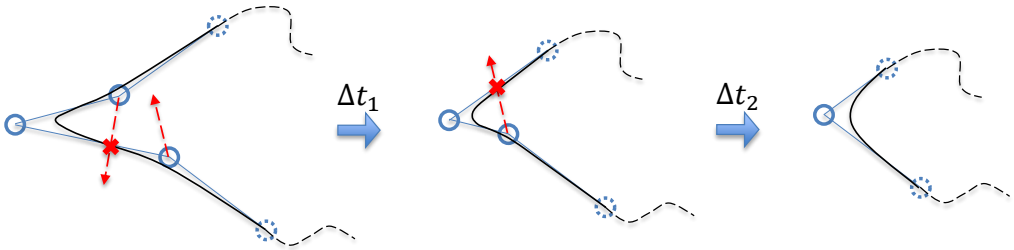
where, subscript k is the index of dominant enrichment for given point $\boldsymbol{x}$, T^{e_k} and $G_i^{e_k}$ are the enriching temperature and gradient fields attached to k^{th} enrichment. w_k, $\mathcal{P}_k$ and d_k are the weight, projection and distance functions with regards to k^{th} enrichment respectively. Although second case is not self-intersection, the condition Eq. (9.34) is still applicable. The third case, frequently encountered when the adjacent control points move close, could reduce the number of control points and trigger adaptive coarsening. We focus on the third case where adaptive coarsening could effectively improve robustness of interfacial evolution in present work, and postpone the implementation of the cases with topological changes to our future work.



(a) Illustration of genus increase



(b) Illustration of genus decrease



(c) Illustration of unchanged genus

Figure 9.8. Three different circumstances in the rearrangement of control net/NURBS curve: (a) Genus increased by interfacial rupture, (b) Genus decreased by interfacial coalescence and (c) Genus unchanged under adaptive coarsening (number of control points decreases).

A change in number of control points requires a change of number of knots. In the present work, centripetal method [51, 139] with averaging technique is used to generate the knots. The procedure is as follows. Define

$$l = \sum_{i=2}^{n_e} \sqrt{\|P_i - P_{i-1}\|_2}. \tag{9.36}$$

The centripetal parameters corresponding to each control point are:

$$\begin{aligned} &\bar{u}_1 = 0, \qquad \bar{u}_{n_e} = 1, \\ &\bar{u}_i = \bar{u}_{i-1} + \frac{\sqrt{\|P_i - P_{i-1}\|_2}}{l}, \ i = 2, \cdots, n_e - 1. \end{aligned} \tag{9.37}$$

Next, for the open curve, the following averaging of centripetal parameters is used to identify the new knot values:

$$\begin{aligned} &u_0 = \cdots = u_p = 0, \qquad u_{n_e} = \cdots = u_{n_e+p} = 1, \\ &u_{i+p} = \tfrac{1}{p} \textstyle\sum_{j=i+1}^{i+p} \bar{u}_j, \ i = 1, \cdots, n_e - p - 1. \end{aligned} \tag{9.38}$$

9.4.2 Adaptive Time Stepping and Update Procedure

When solving a hyperbolic partial differential equation with forward Euler time stepping, the CFL condition needs to be imposed [4, 140]:

$$\Delta t \leq \Delta t_{CFL} \equiv C_t \min_i \left\{ \frac{h}{|\bar{v}_i|} \right\} \tag{9.39}$$

where, h is the discretization size and $C_t \in (0, 1)$ is the Courant coefficient. Although hyperbolic partial differential equation is not involved in the present work, the CFL condition is used to increase the stability of the interface evolution. Taking account of both NSI condition of Eq. (9.34) and the CFL condition of Eq. (9.39), the final time step length is chosen as

$$\Delta t = \min\{\Delta t_{NSI}, \Delta t_{CFL}\}. \tag{9.40}$$

The update procedure is summarized in Alg. 5.

9.5 Numerical Examples

The developed method is first validated by modeling the infinite corner solidification problem which has an analytical solution. Next, complex benchmark problems of

Algorithm 5 Update Procedure for Solving the Stefan problem with Enriched IGA

Input: Phase interface $\boldsymbol{C}^n$ and Solution $\boldsymbol{T}^n$ at n^{th} step
Output: Phase interface $\boldsymbol{C}^{n+1}$ and Solution $\boldsymbol{T}^{n+1}$ at $(n+1)^{\text{th}}$ step

function UPDATE_STEFAN_PROBLEM($\boldsymbol{C}^n$, $\boldsymbol{T}^n$)
    $\bar{v} \leftarrow$ Obtain nodal speed using Eq. (9.25)
    $\boldsymbol{V} \leftarrow$ Fit nodal velocity by solving normal equation Eq. (9.27)
    $\Delta t \leftarrow \min\{\Delta t_{NSI}, \Delta t_{CFL}\}$, given Δt_{NSI} from Eq. (9.34) and Δt_{CFL} from Eq. (9.39)
    if $\Delta t == \Delta t_{NSI}$ **then**
        $\boldsymbol{C}^{n+1} \leftarrow$ Apply adaptive coarsening on $\boldsymbol{C}^n$
    else
        $\boldsymbol{C}^{n+1} \leftarrow$ Apply adaptive refinement based on Eq. (9.31)
    end if
    $\boldsymbol{T}^{n+1} \leftarrow$ Given $\boldsymbol{C}^{n+1}$, compute matrices and vectors to solve Eq. (9.21)
end function

isotropic crystal solidification and anisotropic dendritic solidification in supercooled liquid are solved. Unlike piece-wise linear approximation of the interface in the level set method, the NURBS representation of the interface achieves sufficient smoothness both in the geometrical shape of the interface and in the solved temperature field during evolution.

9.5.1 Infinite Corner Solidification Problem

An infinite quarter-space corner, shown in Figure 9.9, contains liquid initially at the temperature $T_i \geq T_m$, where T_m is the melting temperature. At time $t = 0$, the temperature of corner surface is decreased to $T_w < T_m$, leading to solidification of the quarter-space.

Analytical solution for cases with constant melting temperature (Eq. (9.6b)) and equal diffusivities $\alpha_l = \alpha_s = \alpha$ was given by Rathjen and Jiji [141]:

$$y^* = \left(\lambda + \frac{C}{x^{*m} - \lambda^m}\right)^{\frac{1}{m}} \tag{9.41}$$

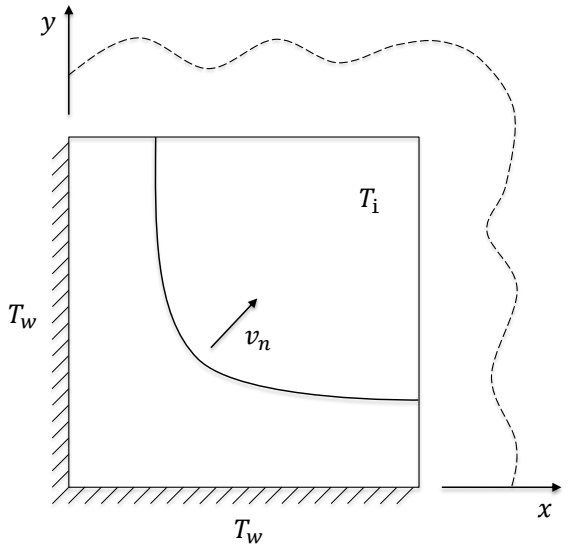
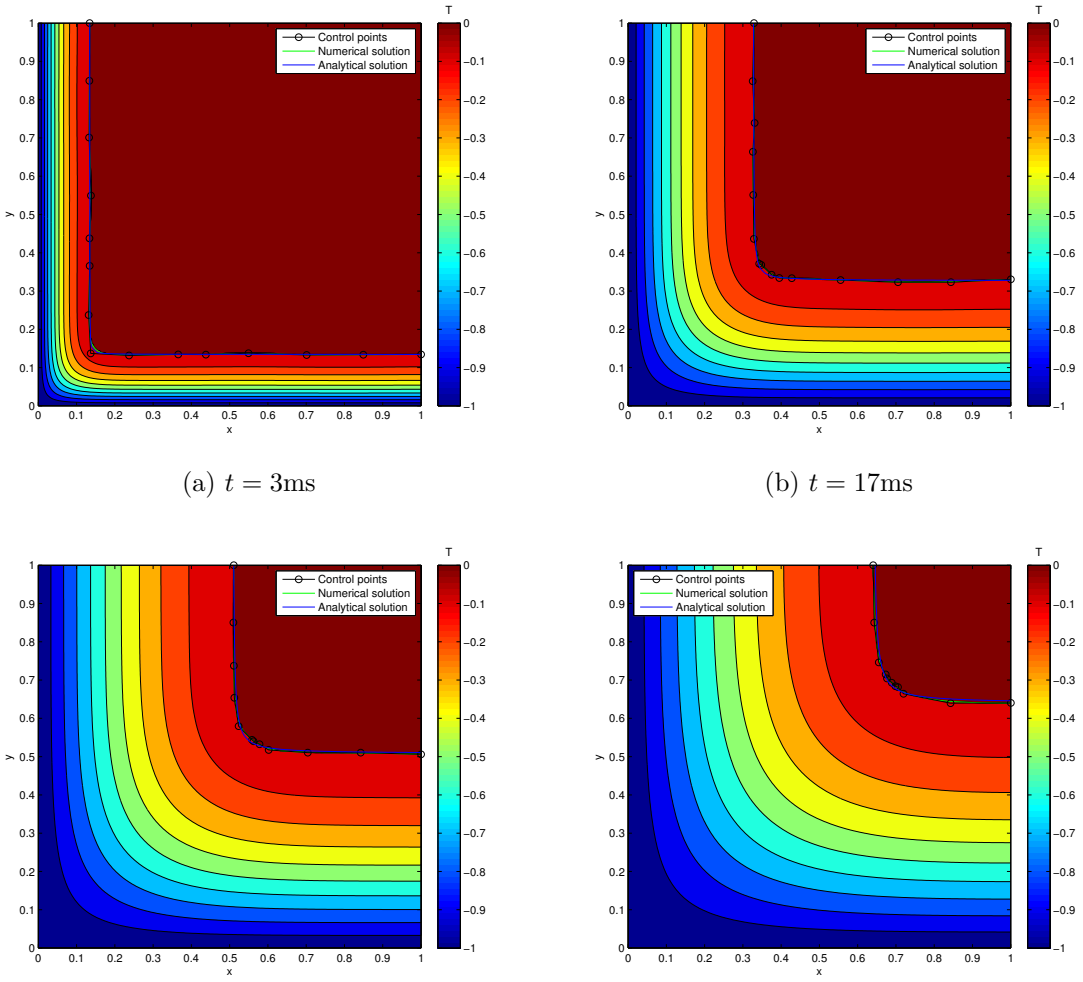


Figure 9.9. Definition of two-dimensional infinite corner solidification problem [141].

where, $x^* = \frac{x}{\sqrt{4\alpha t}}$ and $y^* = \frac{y}{\sqrt{4\alpha t}}$ are dimensionless coordinates of x and y, λ is the stationary interface position, C and m are parameters depending on T_i, T_m, T_w and L. The problem was simulated in a unit square region with $T_i = T_m = 0$ and $T_w = -1$. The parameters k, ρ, c and L were chosen to have unit values. The adaptivity related coefficients were chosen as $C_g = 0.25$ and $C_t = 1$. The temperature field and interface of four transient states are illustrated in Figure 9.10, where the numerically determined interface positions are compared against the analytical solution. Excellent agreement was observed during whole solidification process. As a result of the adaptive refinement and coarsening, the corner arc of interface was approximated well by ensuring a high density of control points during interface evolution. The interface position along the diagonal of the domain versus time is shown in Figure 9.11.



(a) $t = 3$ms

(b) $t = 17$ms

(c) $t = 41$ms

(d) $t = 65$ms

Figure 9.10. The solved temperature field at (a) $t = 3$ms, (b) $t = 17$ms, (c) $t = 41$ms and (d) $t = 65$ms. Numerical solution is compared to the analytical solution. Control points of the NURBS numerical interface are shown to illustrate adaptive refinement and coarsening. Discretization size of 100×100 was used.

9.5.2 Crystal Solidification in a Supercooled Liquid

In this example, a small solid seed is placed in a region containing a supercooled liquid, leading to crystal solidification beginning from the seed site. Several researchers have previously solved this problem using the level-set method [128, 142, 143]. The

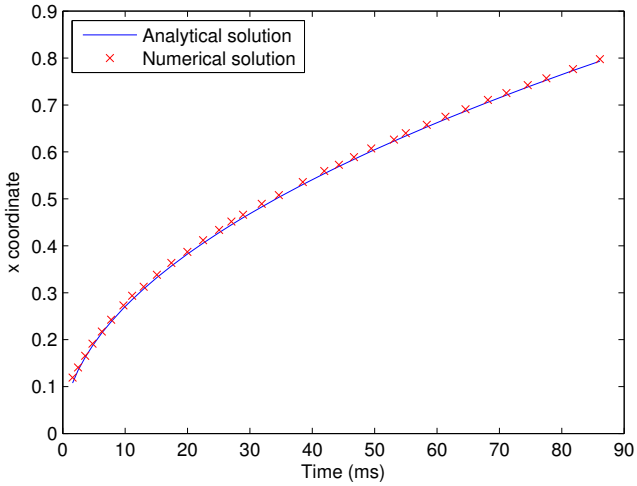


Figure 9.11. Comparison of interface position along diagonal for proposed method and Rathjen's analytical solution.

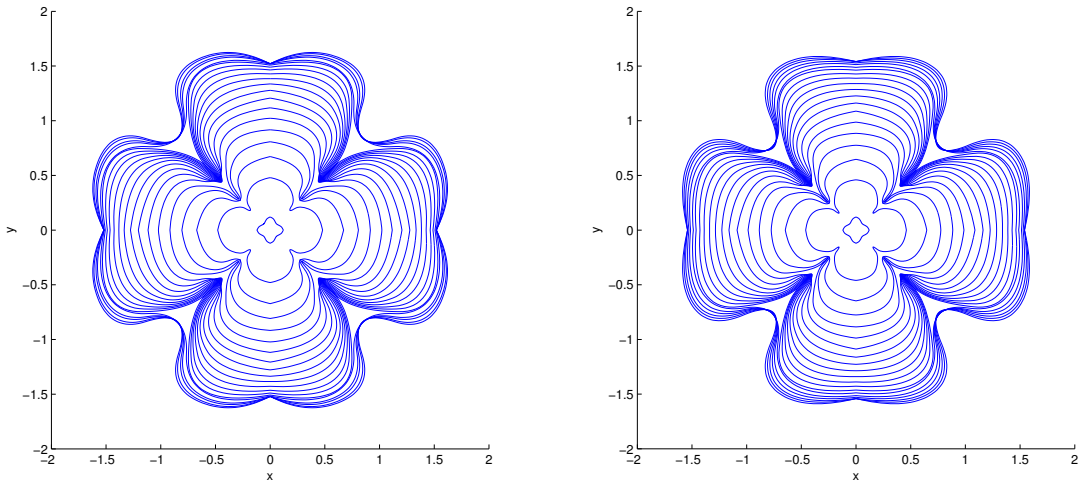
Gibbs-Thomson condition (Eq. (9.6a)) with constant coefficients ϵ_c and ϵ_v was enforced as the interface temperature condition. Due to the explicit geometrical representation of the interface, the curvature κ was obtained readily and without approximation. The simulation domain was $[0, 2] \times [0, 2]$ quarter-space that was insulated on all sides. Melting temperature and supercooled initial temperature were set as $T_m = 0$ and $T_i = -0.5$ respectively. The parameters in the Gibbs-Thomson condition were chosen as $\epsilon_c = 5 \times 10^{-3}$ and $\epsilon_v = 2 \times 10^{-2}$. These parameters correspond to isotropic crystal growth. The remaining parameters, namely k, ρ, c and L, were chosen to have unit values. At $t = 0$, a small petal-shaped solid seed was added at the origin of the computational domain. The geometry of the seed was

$$x(u) = (R + P\cos(8\pi u))\cos(2\pi u) \tag{9.42a}$$

$$y(u) = (R + P\cos(8\pi u))\sin(2\pi u) \tag{9.42b}$$

where, $R = 0.1$ and $P = 0.02$. In the implementation of this example, two different discritization sizes (100×100 and 200×200) with same refinement and Courant

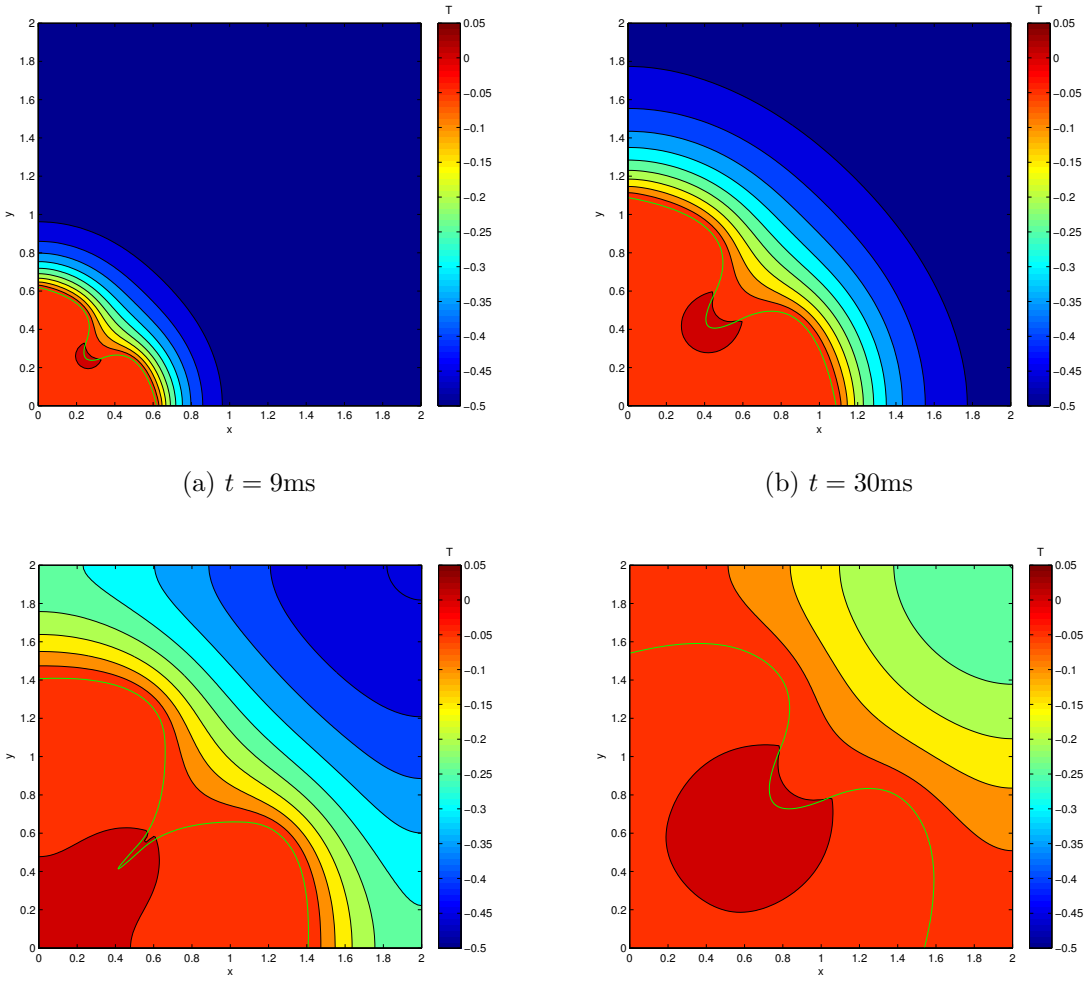
coefficients $C_g = C_t = 0.5$ were tested. Results are presented in Figure 9.12. In the figure, the area of final solid region is exactly one half of the computational domain. These results match well with the solutions in the literature [128, 143]. They also demonstrate a solution that is not dependent on the level of discretization.



(a) Mesh 100×100 (b) Mesh 200×200

Figure 9.12. Simulations of crystal growth with effect of isotropic surface tension and kinetic mobility solved using two different discretizations (a) 100×100 and (b) 200×200. The profiles correspond to time increments of 0.05 ending with a final time of 1.0.

To study the supercooling phenomenon influenced by surface tension and kinetic mobility, the temperature fields of four intermediate states are shown in Figure 9.13. Unlike the corner solidification case (Figure 9.10), the phase interface doesn't coincide with zero temperature level due to supercooling. In all sub-figures, the temperature around the interface with negative curvature are positive because of the effect of surface tension. As time increases, the region with positive temperature increases because the evolution speed decreases and the supercooling effect of kinetic mobility is weakened.



(a) $t = 9$ms

(b) $t = 30$ms

(c) $t = 57$ms

(d) $t = 97$ms

Figure 9.13. Obtained temperature fields during crystal solidification at (a) $t = 9$ms, (b) $t = 30$ms, (c) $t = 57$ms and (d) $t = 97$ms. The computed interface is also shown overlaid on the temperature contour plots.

9.5.3 Steady-state Dendritic Solidification

Another class of anisotropic dendritic problem is one that has steady-state features of dendritic growth including the characteristic dendritic shape and tip velocity. This dendritic problem was also studied by several researchers using the level-set method [129, 130] as well as the phase-field method [32, 33, 144]. To model this problem, the

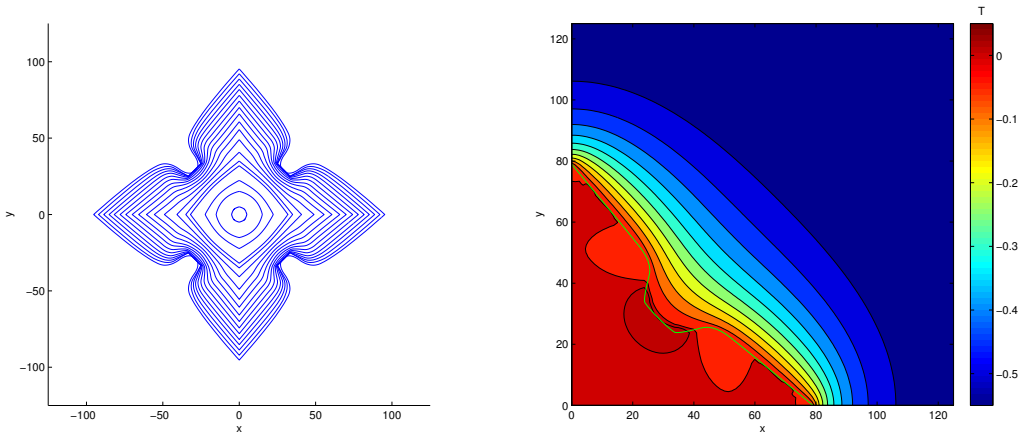
usual assumptions of $T_m = 0$, $\epsilon_v(\boldsymbol{n}) = 0$ and $\epsilon_c(\boldsymbol{n}) = a(1 - 15b\cos(4\theta))$ were applied as the Gibbs-Thomson condition, resulting in the interface temperature condition:

$$T = -a(1 - 15b\cos(4\theta))\kappa, \ \boldsymbol{x} \in \Gamma_{int} \tag{9.43}$$

where, a is the capillary scaling factor, b is anisotropic strength and θ is the angle between interfacial normal vector and x-axis. Due to anisotropy, an initially circular seed grows into a dendrite with four-fold axis of symmetry.

One quarter of a domain of size 250×250 units was simulated exploiting symmetry. At $t = 0$, a circular seed with radius $R = 5$ was placed at the center of the domain which was initially at a supercooled temperature of $T_i = -0.55$. The parameters in Eq. (9.43) were chosen as $a = 0.4$ and $b = 0.05$. As before, the parameters k, ρ, c and L were chosen to have unit values. Discretization size of 100×100 and parameters $C_g = 0.6$ and $C_t = 0.4$ were used in the simulation. The evolution of the interface and the contours of steady-state temperature are illustrated in Figure 9.14, where time is non-dimensionalized as $\tau = \frac{t\alpha}{a^2}$.

One can again observe that in Figure 9.14b, the undercooled region near the interface where curvature is positive (for example, the dendritic tips). A similar observation was made earlier in [130]. Tip velocity is plotted in Figure 9.15, where the velocity is non-dimensionalized as $\xi_{tip} = \frac{v_{tip}a}{\alpha}$. According to microscopic solvability theory, the tip velocity, which is much larger than at other points of the dendritic interface, would converge to an asymptotic limit. The dimensionless velocity is plotted against dimensionless time in Figure 9.15. In order to investigate the influence of adaptivity, two cases with different initial number of control points (n_e^0) were tested. It may be observed that the velocity reduction is not monotonic and that two distinct local minima exist for each case. This was due to extensive adaptive refinement during time $\tau \in [2200, 2700]$ for case $n_e^0 = 4$ and during time $\tau \in [3600, 4400]$ for case $n_e^0 = 6$, which drastically changed the number of control points and therefore affected the tip velocity. Such adaptive refinement can occur at any time but do not affect



(a) Evolution of interface

(b) Temperature field

Figure 9.14. Steady-state dendritic solidification with four-fold growth axis of symmetry: (a) evolution of interface with step length $\Delta\tau = 500$ and (b) steady-state temperature field at $\tau = 6150$. Initial number of control points $n_e^0 = 4$ was used.

the steady-state tip velocity. In fact, during time steps before as well as after the adaptive refinement, the tip velocity decreases monotonically towards the asymptotic limit.

9.6 Summary

The Stefan problem was modeled in this chapter with a geometrically explicit evolving interface isogeometrically enriched with appropriate hybrid function/derivative values. This method combines the fixed mesh characteristic of XFEM with $\mathscr{C}^1$ or higher smoothness naturally enabled by isogeometric analysis, while allowing one to explicitly compute normals and curvature at any point on the interface. The function enrichment allows direct imposition of the Gibbs-Thomson condition, while the derivative enrichment allows one to model the temperature gradient discontinuity arising from the Stefan condition. The recently developed algebraic distance estimations

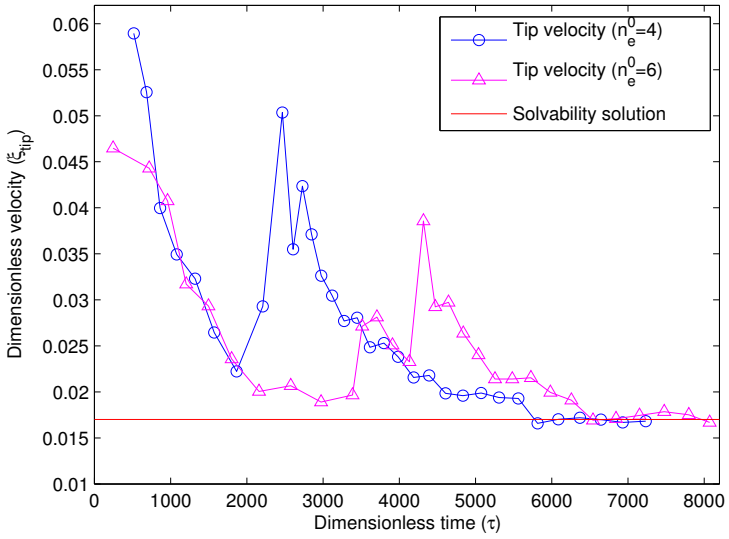


Figure 9.15. Convergence of tip velocity to solution of microscopic solvability theory for $n_e^0 = 4$ and $n_e^0 = 6$. The asymptotic limit is $\bar{\xi}_{tip} = 0.017$ and sampling step length is $\Delta\tau \approx 250$.

was coupled with a newly developed algebraic point projection method to efficiently carryout quadrature. Adaptive strategies for both interface geometry refinement and time stepping were employed to improve efficiency and robustness. The accuracy of the proposed method was demonstrated on validation problems that possessed an analytical solution. Further, complex benchmark problems of isotropic crystal solidification and anisotropic dendritic solidification in supercooled liquid were presented to demonstrate the power of the developed method.

10. SHAPE OPTIMIZATION USING CONFIGURATIONAL DERIVATIVE

In this chapter, a method for shape optimization through configuration of finite sized heterogeneities is demonstrated. The method relies on the recently derived configurational derivative that describes the sensitivity of an arbitrary objective with respect to arbitrary design modifications of a heterogeneity inserted into a domain with moving boundary. The configurational derivative takes special forms when the heterogeneity is subject to translation, rotation and uniform scaling. The configurational derivative may be further simplified for special objectives such as compliance. The computational implementation is based on the THB-splines which can accurately represent the behavioral field near the heterogeneity boundaries. Several two- and three-dimensional numerical examples are presented to demonstrate the methodology.

10.1 Introduction to Configurational Optimization

Many engineering problems, including the design of mechanical parts, necessitate an optimal placement of *finite-sized*, *regular-shaped* geometries within the structure. A widely used technique for topological design of structures is Solid Isotropic Material with Penalization (SIMP) method [145, 146], which attempts to obtain the optimal topology by distributing material in a fixed region. However, the SIMP method requires a sophisticated numerical scheme to avoid the checkerboad problem [147] and often leads to a skeletal or irregular shaped machinable structure. In contrast, the shape optimization [148, 149] is aimed at determining the optimal external/internal boundary shapes of (typically homogeneous) bodies with given constraints (minimum mass, etc.). Extensive studies [150–152] have been conducted on integrating topology and shape optimization, and in particular, automating transition between them.

In practice, the availability of analytical design sensitivities plays an important role in the efficiency of topology and shape optimization. Shape optimization by placing finite-sized heterogeneities within a homogeneous domain necessitates a sensitivity analysis of arbitrary functionals with respect to translation, rotation and scaling of the heterogeneities. Such sensitivities of arbitrary functionals for the configuration of the heterogeneities appear to have been investigated only in [153]. The derivation was founded on the conservation rules of elasticity. While the conservation rules have served as a powerful tool in fracture mechanics [124, 154–156], the configurational sensitivities do not appear to have been applied to shape optimization. Very recently, a new technique for optimal topological design of solids, termed as *configurational optimization*, was proposed [157]. The derived sensitivities of arbitrary functionals to translation, rotation or scaling of a finite-sized heterogeneity were shown to be a generalization of the classical topological derivative, and were exploited to determine the optimal configuration of the heterogeneities inserted into the solid structure.

Since the position, orientation or size of the inserted heterogeneities are changed at every iteration of the configurational optimization, ans since the accuracy of solution can not be assured without an appropriate mesh refinement near the heterogeneity boundaries (see Figure 6.1), an adaptive, easy-to-regenerate mesh is strongly desired in the procedure. However, the regeneration of a boundary-fitted finite element mesh is in general very time-consuming. Thus, there are relatively few numerical examples in the literature that realize shape optimization through the growth of explicitly defined heterogeneities. In this work, the THB-splines, which allows efficient local refinement as described in Chapter 6, is utilized to facilitate the configurational optimization. The methodology is demonstrated through a series of examples.

10.2 Mathematical Description

The configurational optimization, originally proposed in Lin's doctoral research [158], is briefly reviewed here. The configurational optimization problem for arbitrary

objectives is first posed. The material time derivative of the objective with respect to arbitrary boundary modifications, referred as Configurational Derivative, is presented and further simplified for special design velocities (i.e., translation, rotation and scaling).

10.2.1 Configurational Optimization Problem

Given a homogeneous linear elastic solid Ω as shown in Figure 10.1a, the principle of virtual work is given by

$$\int_{\Omega} \boldsymbol{\epsilon}^0 : \boldsymbol{C}^0 : \boldsymbol{\epsilon}^{a0} \, \mathrm{d}\Omega = \int_{\Gamma_t} \boldsymbol{t}^0 \cdot \boldsymbol{u}^{a0} \, \mathrm{d}\Gamma. \tag{10.1}$$

where, $\boldsymbol{\epsilon}^0$ is infinitesimal strain, $\boldsymbol{C}^0$ is isotropic material tensor and $\boldsymbol{t}^0$ is prescribed traction. $\boldsymbol{\epsilon}^{a0}$ and $\boldsymbol{u}^{a0}$ are compatible virtual strain and displacement, respectively. The body force term is neglected in Eq. (10.1). It is also assumed that $\boldsymbol{u}^{a0} = \boldsymbol{0}$ on Γ_u. Next, we choose an arbitrary subdomain Ω_s bounded by Γ_s (see Figure 10.1b) for design purposes. The corresponding Dirichlet and Neumann boundary conditions are defined on Γ_{su} and Γ_{st}, respectively. Likewise, the principle of virtual work takes the form:

$$\int_{\Omega_s} \boldsymbol{\epsilon}^0 : \boldsymbol{C}^0 : \boldsymbol{\epsilon}^{a0} \, \mathrm{d}\Omega = \int_{\Gamma_s} \boldsymbol{t}^{s0} \cdot \boldsymbol{u}^{a0} \, \mathrm{d}\Gamma. \tag{10.2}$$

A *design transformation* along with a pseudo design time t is defined within the homogeneous subdomain as follows:

$$\boldsymbol{x}^0 = \boldsymbol{x}^0(\boldsymbol{X}^0, t) \qquad \boldsymbol{X}^0 \in \Omega_s, t \in [0, +\infty) \tag{10.3}$$

where, $\boldsymbol{X}^0$ is the position of a material point at $t = 0$. Correspondingly, the *design velocity* can be defined as

$$\boldsymbol{v}^0(\boldsymbol{x}^0, t) = \frac{\partial \boldsymbol{x}^0}{\partial t}. \tag{10.4}$$

Next, a heterogeneity Ω_p bounded by Γ_p is introduced to Ω_s at the point $\boldsymbol{x}_p$ (see Figure 10.1c). The corresponding principle of virtual work is stated as

$$\int_{\Omega_s} \boldsymbol{\epsilon} : \boldsymbol{C} : \boldsymbol{\epsilon}^a \, \mathrm{d}\Omega = \int_{\Gamma_s} \boldsymbol{t}^s \cdot \boldsymbol{u}^a \, \mathrm{d}\Gamma \tag{10.5}$$

where, compared to Eq. (10.2), the superscript 0 is removed to represent the quantities in the heterogeneous subdomain.

Arbitrary objectives defined on the homogeneous and heterogeneous subdomains are now posed as

$$f^0(t) = \int_{\Omega_s} \psi^0(t, \epsilon^0) \, \mathrm{d}\Omega + \int_{\Gamma_s} \phi^0(t, \boldsymbol{n}, \boldsymbol{x}^0, \boldsymbol{u}^0, \boldsymbol{t}^0) \, \mathrm{d}\Gamma \tag{10.6a}$$

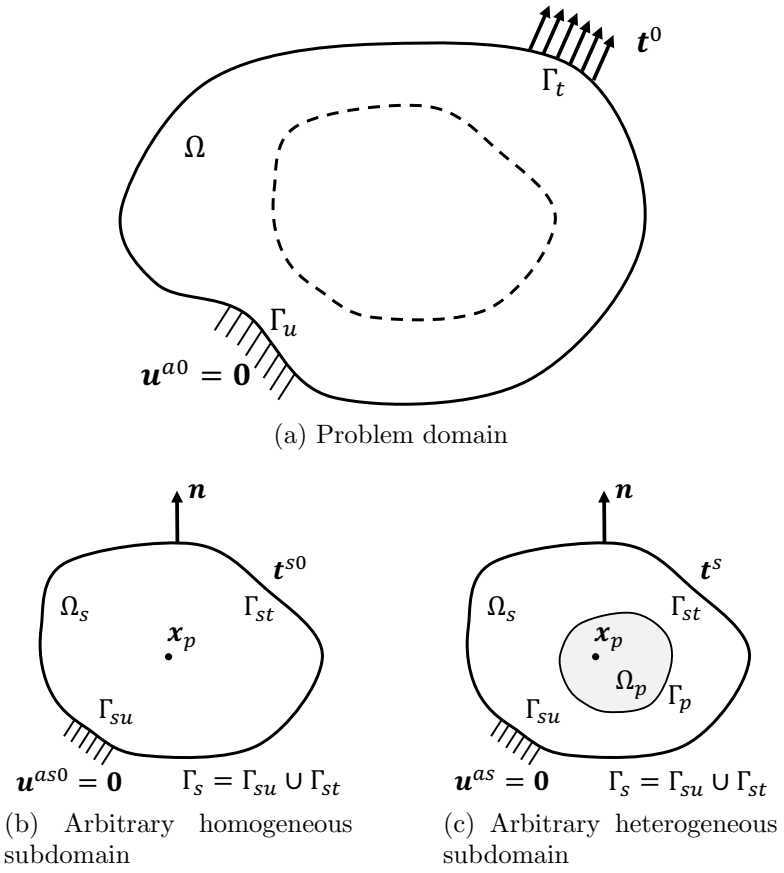
$$f(t) = \int_{\Omega_s} \psi(t, \epsilon) \, \mathrm{d}\Omega + \int_{\Gamma_s} \phi(t, \boldsymbol{n}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{t}) \, \mathrm{d}\Gamma \tag{10.6b}$$

respectively. $\boldsymbol{n}$ is outward normal to Γ_s and therefore identical for both subdomains. Given the material density $\rho^0(t, \boldsymbol{x}^0)/\rho(t, \boldsymbol{x})$ in the homogeneous/heterogeneous subdomains, the total mass can be written as

$$m^0(t) = \int_{\Omega_s} \rho^0(t, \epsilon^0) \, \mathrm{d}\Omega \tag{10.7a}$$

$$m(t) = \int_{\Omega_s} \rho(t, \epsilon) \, \mathrm{d}\Omega \tag{10.7b}$$

It is worth noting that the heterogeneity can be stiffer $\left(\frac{m}{f} > \frac{m^0}{f^0}\right)$ or softer $\left(\frac{m}{f} < \frac{m^0}{f^0}\right)$ than the homogeneous subdomain. The limit of a soft heterogeneity $\left(\frac{m}{f} \to 0\right)$ is a



(a) Problem domain

(b) Arbitrary homogeneous subdomain

(c) Arbitrary heterogeneous subdomain

Figure 10.1. Definition of the configurtional optimization problem: (a) Original problem domain, (b) homogeneous subdomain of interest and (c) corresponding heterogeneous subdomain by introducing an arbitrary heterogeneity to (b).

hole. The configurational optimization problem is to find $\boldsymbol{x}_p, \boldsymbol{n}_p$ and optimized shape of the heterogeneity to

$$\begin{aligned}
\text{minimize} \quad & g(t) = (1-w)\left[m(t) - m^0(t))\right] + w\left[m(t) - m^0(t))\right], w \in [0,1] \\
\text{subject to} \quad & c^0(t) = \int_{\Omega_s} \boldsymbol{\epsilon}^0 : \boldsymbol{C}^0 : \boldsymbol{\epsilon}^{a0} \, \mathrm{d}\Omega - \int_{\Gamma_s} \boldsymbol{t}^{s0} \cdot \boldsymbol{u}^{a0} \, \mathrm{d}\Gamma = 0 \\
& c(t) = \int_{\Omega_s} \boldsymbol{\epsilon} : \boldsymbol{C} : \boldsymbol{\epsilon}^{a} \, \mathrm{d}\Omega - \int_{\Gamma_s} \boldsymbol{t}^{s} \cdot \boldsymbol{u}^{a} \, \mathrm{d}\Gamma = 0.
\end{aligned} \tag{10.8}$$

where, w is the weight of the pareto-optimal problem which reflects the designer's preference between the performance and mass terms. Without loss of generality, it is assumed that the design transformation, as well as the design velocities, is identical in both subdomains, i.e., $\boldsymbol{x} = \boldsymbol{x}^0$ and $\boldsymbol{v} = \boldsymbol{v}^0$.

10.2.2 Configurational Derivative

The material derivative of the Lagrangian $G(t)$ corresponding to Eq. (10.8) can be derived as [158]

$$\begin{aligned}
\dot{G}(t) =&(1-w)\left\{\left[\int_{\Gamma_p}[\![\boldsymbol{n}\cdot\boldsymbol{\Sigma}]\!]\cdot\boldsymbol{v}\,\mathrm{d}\Gamma+\int_{\Gamma_s}\boldsymbol{n}\cdot\boldsymbol{\Sigma}\cdot\boldsymbol{v}\,\mathrm{d}\Gamma\right.\right.\\
&+\int_{\Gamma_s}\left[\left(\phi+\boldsymbol{t}\cdot\boldsymbol{u}^a\right)\left(\nabla\cdot\boldsymbol{v}-\boldsymbol{n}\cdot\nabla\boldsymbol{v}\cdot\boldsymbol{n}\right)+\nabla\phi\cdot\boldsymbol{v}+\phi_{,\boldsymbol{n}}\cdot\dot{\boldsymbol{n}}\right]\,\mathrm{d}\Gamma\\
&\left.+\int_{\Gamma_{st}}\dot{\boldsymbol{t}}^s\cdot\left(\phi_{,\boldsymbol{t}}+\boldsymbol{u}^a\right)\,\mathrm{d}\Gamma+\int_{\Gamma_{su}}\left(\phi_{,\boldsymbol{u}}\cdot\dot{\boldsymbol{u}}^s+\boldsymbol{t}\cdot\dot{\boldsymbol{u}}^{as}\right)\,\mathrm{d}\Gamma\right]\\
&-\left[\int_{\Gamma_s}\boldsymbol{n}\cdot\boldsymbol{\Sigma}^0\cdot\boldsymbol{v}\,\mathrm{d}\Gamma\right.\\
&+\int_{\Gamma_s}\left[\left(\phi^0+\boldsymbol{t}^0\cdot\boldsymbol{u}^{a0}\right)\left(\nabla\cdot\boldsymbol{v}-\boldsymbol{n}\cdot\nabla\boldsymbol{v}\cdot\boldsymbol{n}\right)+\nabla\phi^0\cdot\boldsymbol{v}+\phi^0_{,\boldsymbol{n}}\cdot\dot{\boldsymbol{n}}\right]\,\mathrm{d}\Gamma\\
&\left.\left.+\int_{\Gamma_{st}}\dot{\boldsymbol{t}}^{s0}\cdot\left(\phi^0_{,\boldsymbol{t}^0}+\boldsymbol{u}^{a0}\right)\,\mathrm{d}\Gamma+\int_{\Gamma_{su}}\left(\phi^0_{,\boldsymbol{u}^0}\cdot\dot{\boldsymbol{u}}^{s0}+\boldsymbol{t}^0\cdot\dot{\boldsymbol{u}}^{as0}\right)\,\mathrm{d}\Gamma\right]\right\}\\
&+w\int_{\Gamma_p}[\![\rho(\boldsymbol{v}\cdot\boldsymbol{n})]\!]\,\mathrm{d}\Gamma. \qquad (10.9)
\end{aligned}$$

where, $\boldsymbol{\Sigma}$ is the *configurational tensor* defined as

$$\boldsymbol{\Sigma}=\left(\psi-\boldsymbol{\sigma}:\boldsymbol{\epsilon}^a\right)\boldsymbol{I}+\boldsymbol{\sigma}^a\cdot\nabla\boldsymbol{u}^T+\boldsymbol{\sigma}\cdot\nabla\boldsymbol{u}^{aT}. \qquad (10.10)$$

The $\dot{G}(t)$ is termed as *configurational derivative*. Given three special design velocities corresponding to translation, rotation and scaling of the heterogeneity boundary Γ_p (see Figure 10.2):

$$\boldsymbol{v} = \bar{\boldsymbol{v}} \tag{10.11a}$$

$$\boldsymbol{v} = \bar{\omega} \times \boldsymbol{r}_p \tag{10.11b}$$

$$\boldsymbol{v} = \alpha(t)\boldsymbol{r}_p. \tag{10.11c}$$

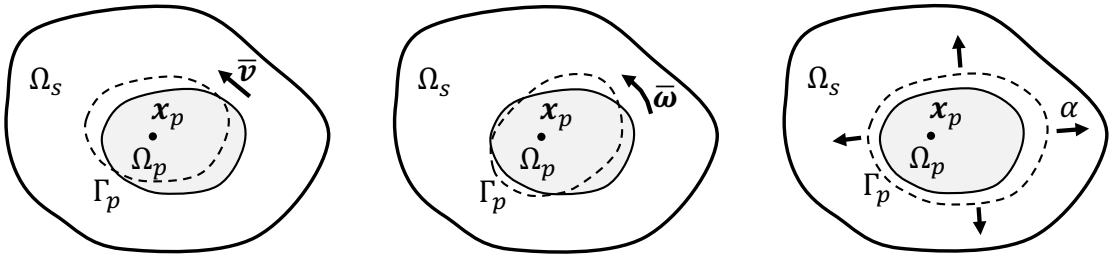
Eq. (10.9) can be simplified to the following path-independent integral forms:

$$\begin{aligned}
\dot{G}^T(t) =& -(1-w)\left\{\int_{\Gamma_s} \boldsymbol{n}\cdot\Sigma \,\mathrm{d}\Gamma\right\}\cdot\bar{\boldsymbol{v}} \\
\equiv& -(1-w)I^T\cdot\bar{\boldsymbol{v}} && (10.12a)\\
\dot{G}^R(t) =& -(1-w)\left\{\int_{\Gamma_s} \left[\boldsymbol{n}\cdot(-\boldsymbol{\Sigma}\times\boldsymbol{r}_p)+\boldsymbol{t}^a\times\boldsymbol{u}+\boldsymbol{t}\times\boldsymbol{u}^a+\boldsymbol{t}\right]\,\mathrm{d}\Gamma\right\}\cdot\bar{\boldsymbol{\omega}} \\
\equiv& -(1-w)I^R\cdot\bar{\boldsymbol{\omega}} && (10.12b)\\
\dot{G}^S(t) =& -(1-w)\left\{\int_{\Gamma_s}\left[\boldsymbol{n}\cdot\Sigma\cdot\boldsymbol{r}_p+\left(\frac{k-p}{p}\right)\boldsymbol{t}^a\boldsymbol{u}+\left(\frac{qk-q-k}{q}\right)\boldsymbol{t}\boldsymbol{u}^a\right]\,\mathrm{d}\Gamma\right\}\alpha \\
&+\left(w\int_{\Gamma_p}[\![\rho\,(\boldsymbol{n}\cdot\boldsymbol{r}_p)]\!]\,\mathrm{d}\Gamma\right)\alpha \\
\equiv& -(1-w)I^S\alpha+\left(w\int_{\Gamma_p}[\![\rho\,(\boldsymbol{n}\cdot\boldsymbol{r}_p)]\!]\,\mathrm{d}\Gamma\right)\alpha && (10.12c)
\end{aligned}$$

where, k is problem dimension and Γ_s is an arbitrary surface that encloses Γ_p. Implicit in the Eq. (10.12c) is the assumption that the bulk objective function ψ is a q^{th} order function of strain $\boldsymbol{\epsilon}$, i.e., $\psi(c\boldsymbol{\epsilon}) = c^q\psi(\boldsymbol{\epsilon})$.

A common choice of the design objective is the structural compliance given by

$$\Theta(t) = \int_{\Omega_s} \boldsymbol{\epsilon} : \boldsymbol{C} : \boldsymbol{\epsilon}\,\mathrm{d}\Omega = \int_{\Omega_s} 2U\,\mathrm{d}\Omega \tag{10.13}$$



(a) Translation (b) Rotation (c) Scaling

Figure 10.2. Three special design velocities: (a) Translation, (b) rotation around an axis passing through $\boldsymbol{x}_p$ and (c) uniform scaling with respect to $\boldsymbol{x}_p$.

where, $U = \frac{1}{2}\boldsymbol{\sigma} : \boldsymbol{\epsilon}$ is strain energy density. Comparing Eq. (10.13) with Eq. (10.6), the bulk and surface objective functions for structural compliance can be written as

$$\psi(\boldsymbol{\epsilon}) = \boldsymbol{\epsilon} : \boldsymbol{C} : \boldsymbol{\epsilon} \quad \text{in } \Omega_s \tag{10.14a}$$

$$\phi(\boldsymbol{t}, \boldsymbol{u}) = 0 \qquad \text{on } \Gamma_s. \tag{10.14b}$$

As shown in [158], the general forms of Eqs. (10.12a) to (10.12c) can be further simplified for the compliance objective to the following:

$$\dot{G}^T_\Theta(t) = (1-w)\left\{2\int_{\Gamma_s}\left[U\boldsymbol{n} - \boldsymbol{t}\cdot\nabla\boldsymbol{u}^T\right]\,\mathrm{d}\Gamma\right\}\cdot\bar{\boldsymbol{v}} \tag{10.15a}$$

$$\dot{G}^R_\Theta(t) = (1-w)\left\{2\int_{\Gamma_s}\left[U(\boldsymbol{r}_p\times\boldsymbol{n}) - \boldsymbol{r}_p\times(\boldsymbol{t}\cdot\nabla\boldsymbol{u}^T) + \boldsymbol{u}\times\boldsymbol{t}\right]\,\mathrm{d}\Gamma\right\}\cdot\hat{\boldsymbol{\omega}} \tag{10.15b}$$

$$\begin{aligned}\dot{G}^S_\Theta(t) = (1-w)&\left\{2\int_{\Gamma_s}\left[U(\boldsymbol{n}\cdot\boldsymbol{r}_p) - \boldsymbol{t}\cdot\nabla\boldsymbol{u}^T\cdot\boldsymbol{r}_p - \left(\frac{d-2}{2}\right)(\boldsymbol{t}\cdot\boldsymbol{u})\right]\,\mathrm{d}\Gamma\right\}\alpha \\ &+ w\left(\int_{\Gamma_p}[\![\rho\,(\boldsymbol{n}\cdot\boldsymbol{r}_p)]\!]\,\mathrm{d}\Gamma\right)\alpha.\end{aligned} \tag{10.15c}$$

Again, the boundary Γ_p, as long as it encloses the heterogeneity, can be chosen arbitrarily for the convenience of numerical integration.

10.3 Numerical Examples

The configurational derivative described above has been implemented in the OOF-HiDAC framework. The problem domain is discretized using THB-splines to obtain a smooth behavioral field near the heterogeneity boundaries. The optimization direction is next determined through a steepest descent algorithm [159]. The methodology is first validated by optimizing the location of a circular hole within a square domain. Next, the optimal location of different types of heterogeneity in a cracked plate is studied. A three-dimensional example, involving simultaneous optimization of position and orientation of an ellipsoidal hole, is finally presented.

10.3.1 Optimal Location of a Circular Hole

As illustrated in Figure 10.3, a circular hole of radius $R = 1.5$ is inserted in a square plate with side $W = 10$. The plate is subjected to a quadratic load given by

$$\sigma = \left(1 - \frac{2x}{W}\right)^2 \tag{10.16}$$

A plane stress state, with Young's modulus $E = 1$ and Poisson's ratio $\nu = 0.3$ was considered. The position of the hole, starting from $(3, 3)$, was optimized to minimize the total structural compliance of the domain.

The stopping criterion was chosen as

$$\frac{\left\|I^T\right\|}{\Theta_{\max}} \leq 10^{-7}, \tag{10.17}$$

and it took 31 iterations to converge. The von Mises stress field at initial configuration, 10th, 20th and 31st iterations, along with the corresponding hole locations, is shown in Figure 10.4. As can be observed, the hole moves towards a low stress location. Figure 10.5 illustrates the change in the objective during optimization. The structural compliance of the final configuration is 11.3% less than that of the initial configuration.

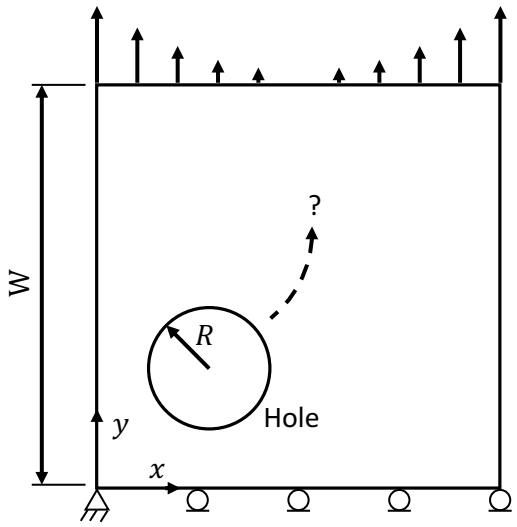
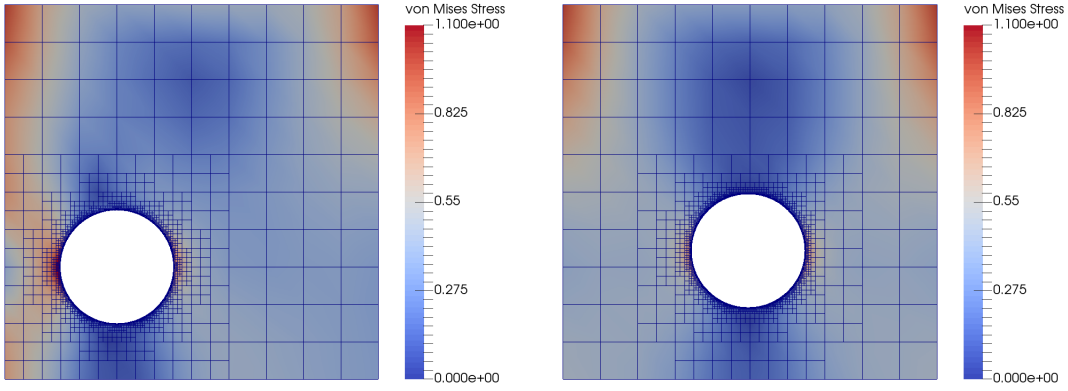


Figure 10.3. Optimization of location of a circular hole within a square plate subjected to a quadratic loading.

10.3.2 Optimal Location of a Heterogeneity in the Presence of a Crack

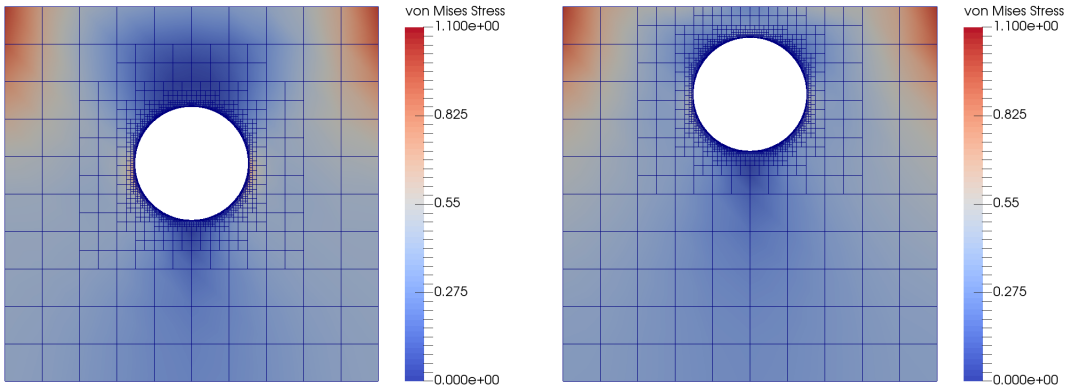
In this example, the location of two different types of heterogeneities (hole and stiffener) is optimized within a cracked plate to minimize the system structural compliance. As shown in Figure 10.6, the plate has a side length of $W = 20$ and the crack spans from $(0.7W, 0.36W)$ to $(W, 0.36W)$. The radius of the heterogeneity is $R = 0.12W$. The problem is solved under plain stress state. The Young's modulus of the plate and the stiffener are taken as 1 and 100, respectively, while the Poisson's ratio is chosen as 0.3 for the whole problem domain. A uniform tension of $\sigma = 1$ is applied to the plate.

Due to the different nature of the heterogeneities, the hole is initially placed in the vicinity of the crack whereas the stiffener is placed far away from the crack. The von Mises stress field in the presence of a hole and a stiffener is illustrated in Figures 10.7 and 10.8, respectively. It can be observed that, during the configurational optimization, the hole moves away from the crack tip to avoid interaction with the singular stress. On the contrary, the stiffener tends to move towards the crack tip to mitigate



(a) Initial configuration

(b) 10th iteration



(c) 20th iteration

(d) Final configuration

Figure 10.4. Von Mises stress field and hole location at (a) initial configuration, (b) 10th iteration, (c) 20th iteration and (d) final configuration (31st iteration). The behavioral field was approximated using a seven-level quadratic THB-spline.

the singular stress. The objective change for both problems is shown in Figure 10.9, where the structural compliance is shown to reduce by 5% in both cases. It may be noted that the final configuration of the stiffener is not a converged solution since the iteration was terminated to avoid collision with the crack. This example shows the potential of configurational derivative as a powerful tool for fracture-resistant design.

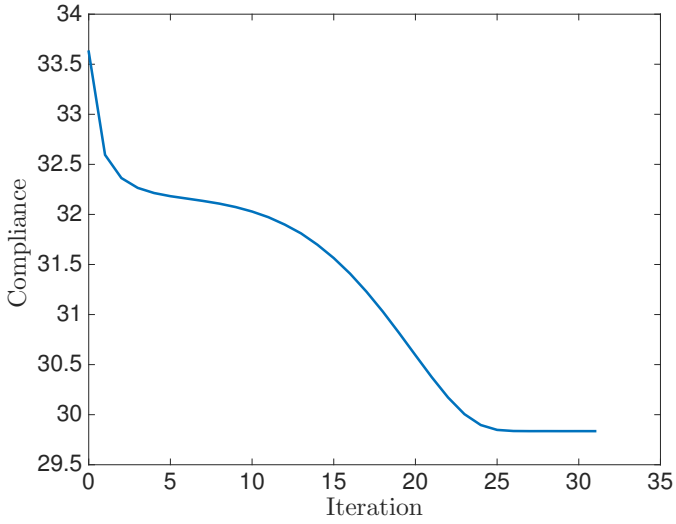
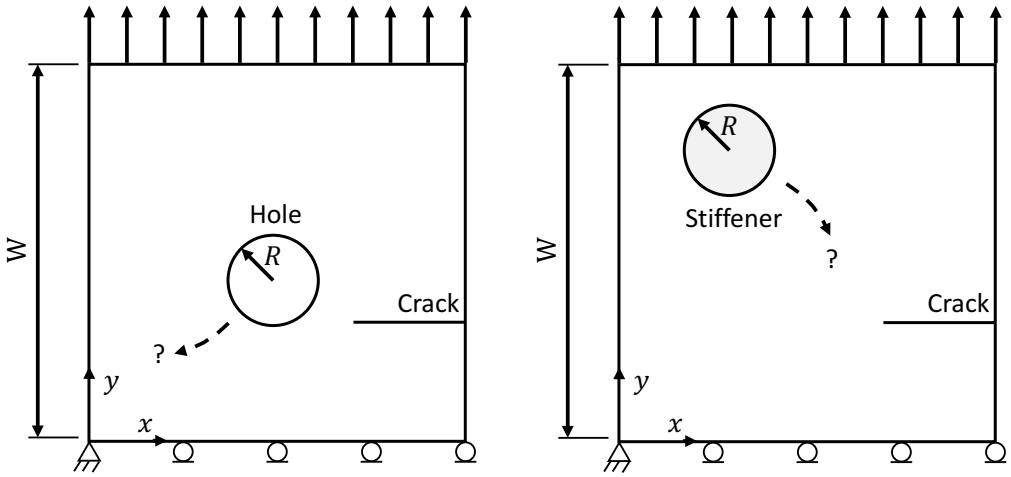
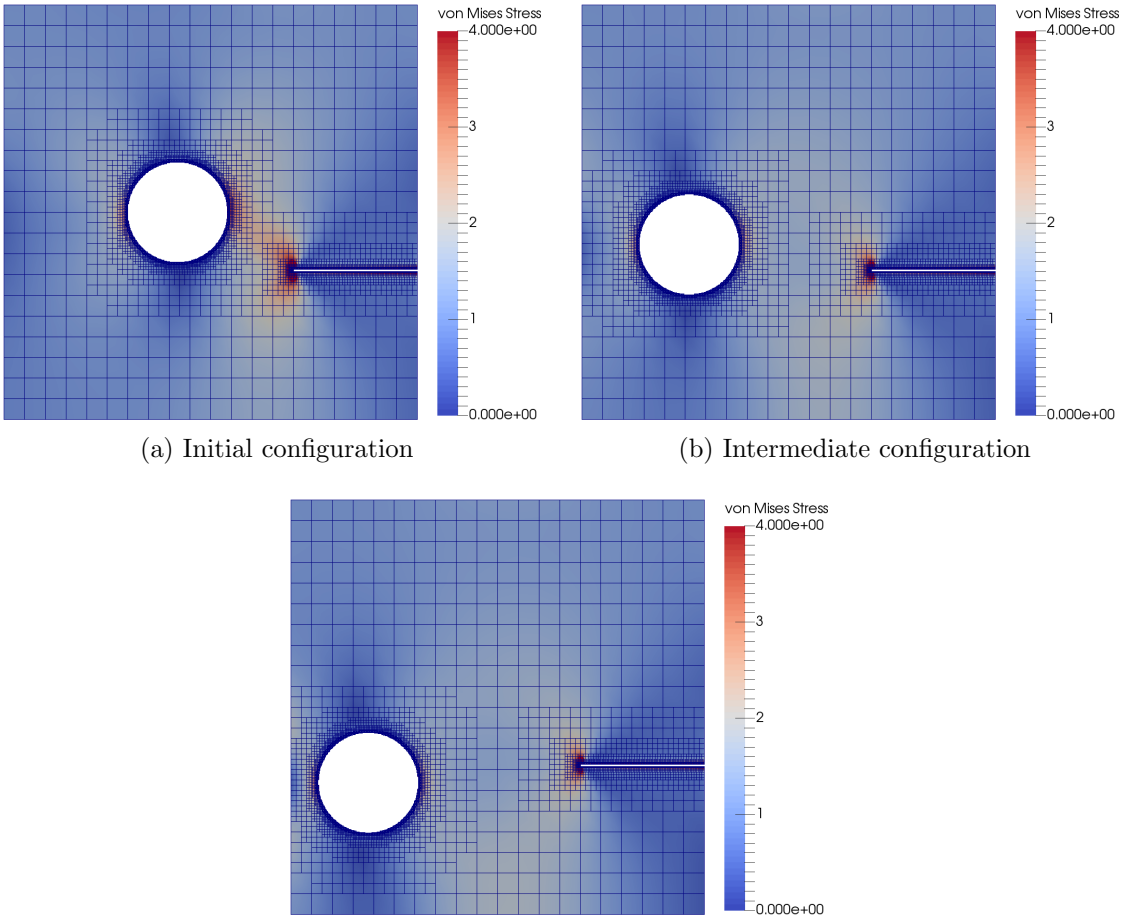


Figure 10.5. Structural compliance of the plate shown in Figure 10.3 against iteration count.



(a) Hole (b) Stiffener

Figure 10.6. Optimization of location of (a) a hole and (b) a stiffener within a cracked plate. Different initial locations are chosen for different heterogeneities.

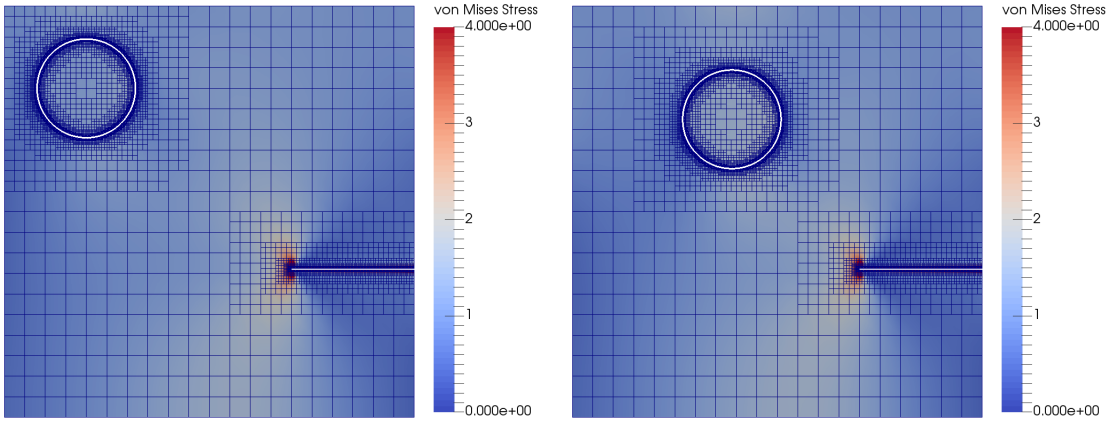


(a) Initial configuration

(b) Intermediate configuration

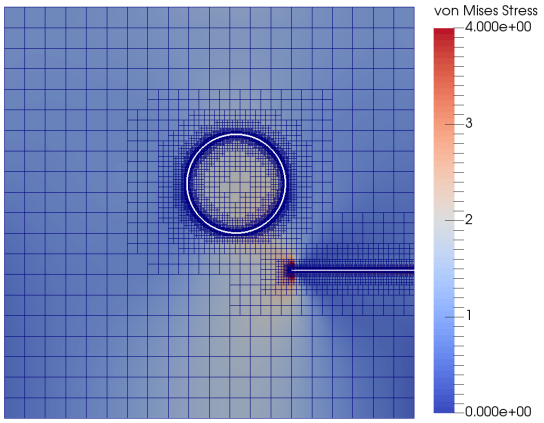
(c) Final configuration

Figure 10.7. Von Mises stress field and hole location at (a) initial configuration, (b) intermediate configuration (10th iteration) and (c) final configuration (49th iteration). The behavioral field was approximated using a six-level quadratic THB-spline.



(a) Initial configuration

(b) Intermediate configuration



(c) Final configuration

Figure 10.8. Von Mises stress field and stiffener location at (a) initial configuration, (b) intermediate configuration (30th iteration) and (c) final configuration (49th iteration). The behavioral field was approximated using a six-level quadratic THB-spline.

10.3.3 Simultaneous Optimization of Location and Orientation of an Ellipsoidal Hole

This example involves an ellipsoidal hole whose location and orientation are simultaneously updated at every iteration. The problem domain, as shown in Figure 10.10, has a side length of $W = 10$ and is subjected to a quadratic load given by

$$\sigma = \left(1 - \frac{2x}{W}\right)^2 + \left(1 - \frac{2y}{W}\right)^2. \tag{10.18}$$

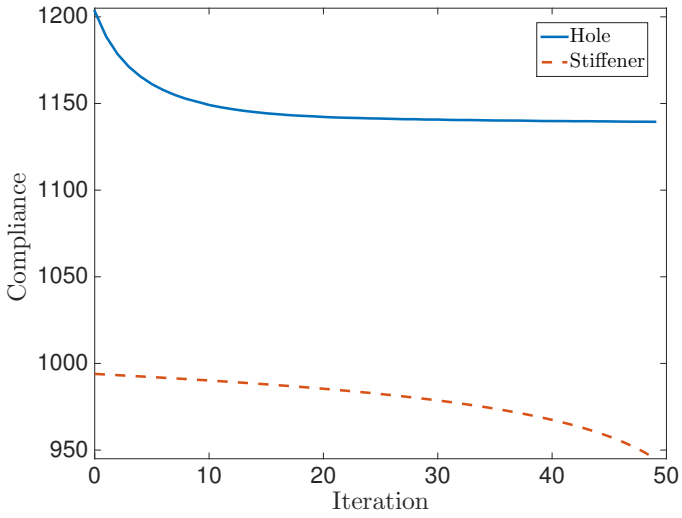


Figure 10.9. Structural compliance of the plate in the presence of a heterogeneity as shown in Figure 10.6.

The semi-principal axis lengths of the ellipsoidal hole are

$$L_a = 0.15W, \quad L_b = L_c = 0.1W. \tag{10.19}$$

The hole is initially placed at $(0.35W, 0.35W, 0.35W)$ with the long principal axis (L_a) pointing along the line

$$x = y, \quad z = 0.35W \tag{10.20}$$

The Young's modulus and Poisson's ratio are taken as 1 and 0.3, respectively.

In this problem, the stopping criterion was chosen as

$$\frac{\sqrt{\|I^T\|^2 + \|I^R\|^2}}{\Theta_{\max}} \leq 10^{-7}. \tag{10.21}$$

The optimal location and orientation of the hole were obtained after 44 iterations. The von Mises stress field at four different iterations is shown in Figure 10.11. After

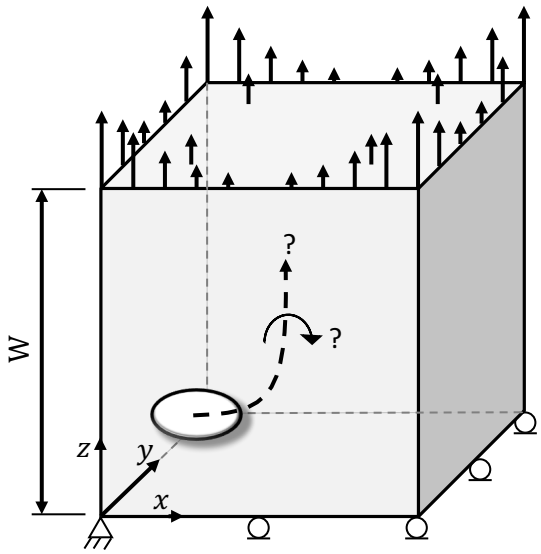
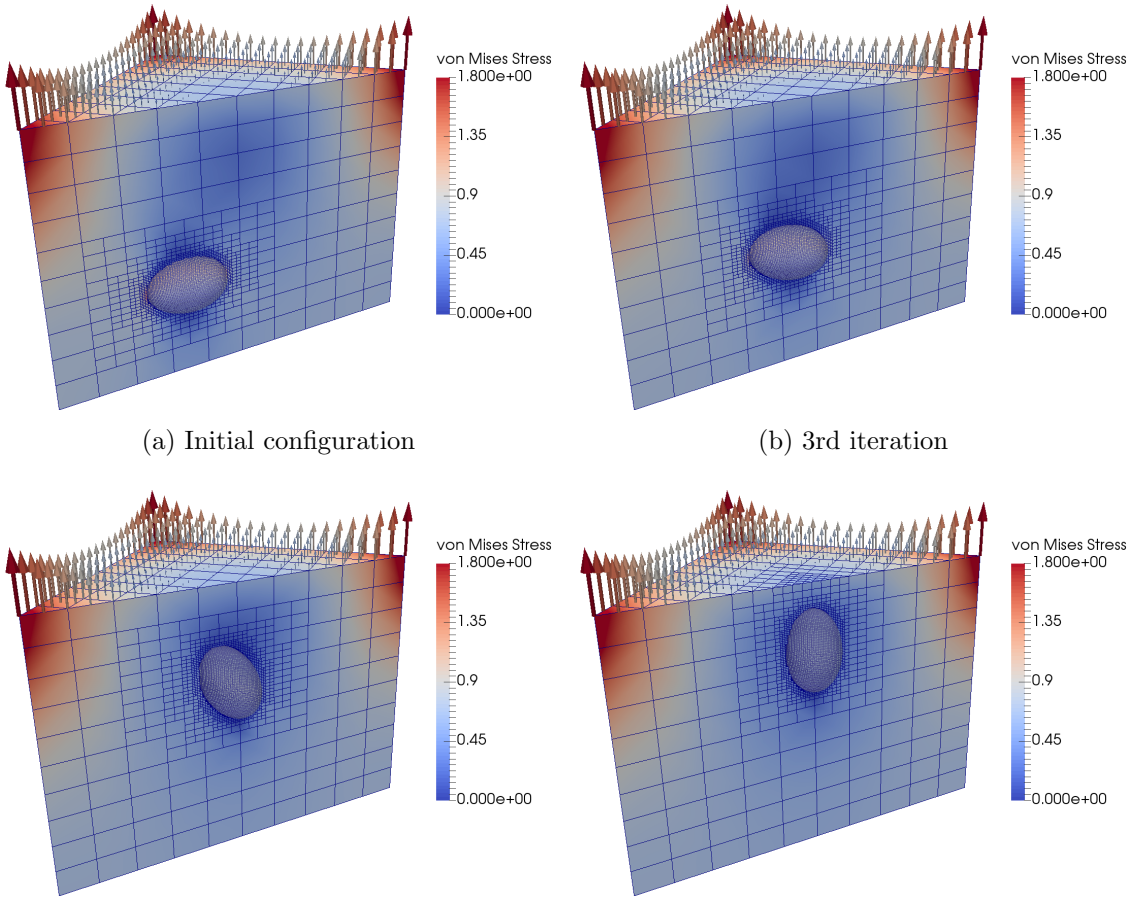


Figure 10.10. Simultaneous optimization of location and orientation of an ellipsoidal hole within a cubic domain. A quadratic load is applied to the top surface of the domain.

the configurational optimization, the hole moves to the central axis of the domain due to the influence of the quadratic loading. Meanwhile, the long principal axis of the ellipsoidal hole becomes aligned with the z-axis to reduce the stress concentration around the hole.



(a) Initial configuration

(b) 3rd iteration

(c) 7th iteration

(d) Final configuration

Figure 10.11. Von Mises stress field and hole configuration at (a) initial configuration, (b) 3rd iteration, (b) 7th iteration, and (d) final configuration (44th iteration). The domain was clipped diagonally for visualization of the hole. The behavioral field was approximated using a three-dimensional, five-level quadratic THB-spline.

11. CLOSURE

11.1 Summary and Novel Contributions

The main theme of this thesis was to develop a sharp interface formalism that enables efficient modeling of moving boundary problems. To begin with, a survey of existing modeling techniques was presented. These techniques were classified according to their meshing need and interface representation. Conforming mesh, explicit interface techniques, such as finite element method, often need a great number of degrees of freedom to model complex geometries and interfaces. Moreover, the analysis mesh needs to be updated at every step of boundary evolution. The generation of a boundary-fitted mesh is, in general, very time-consuming. While non-conforming mesh, implicit interface methods alleviate the need for remeshing, these methods often introduce additional, hard-to solve equations and numerous auxiliary unknowns (e.g., level-set and phase-field variables). Due to the implicit nature of the interface, geometric quantities are not explicitly known and boundary conditions can only be weakly applied on mesh than the interface. Such mesh generation step can be completely eliminated by mesh-free methods. However, in addition to the indirect imposition of boundary conditions, the numerical integration of these methods remains a significant challenge.

A non-conforming mesh, explicit interface method, termed as enriched isogeometric analysis, was proposed to mitigate the cumbersome remeshing step while maintaining the geometric exactness through CAD-standard representation of the domain and interface geometries. The behavioral field is represented by a weighted composition of a continuous domain approximation and enriched boundary approximations where problem-dependent enriching functions and degrees of freedom are defined. The enrichments are constructed according to *a-priori* knowledge of interfacial behavior such

as singularity, weak and strong discontinuities, and interface conditions. The recently developed algebraic distance technique was utilized to construct the weight fields in the composition. Furthermore, the influence of a boundary on a point of interest can be determined by geometrically projecting the point to the boundary. This is realized in a new algebraic point projection algorithm, which was demonstrated to be faster and more robust than classical Newton-Raphson iterations.

In order to effectively integrate over the elements intersected by the immersed boundaries in the non-conforming mesh methods, a novel kd-tree based adaptive quadrature scheme was developed. In contrast to the quad-tree and oct-tree subdivision where all dimensions are bisected simultaneously, the kd-tree subdivision allows dimension-wise splits of quadrature cells and can produce up to 33% and 57% fewer sub-cells than quad-tree and oct-tree, respectively. Furthermore, the truncated hierarchical B-splines were employed to support local refinement in the enriched isogeometric analysis. The efficiency of the adaptive mesh generation is assured through newly proposed sign-based and distance-based refinement algorithms. More importantly, the maximum number of active THB-spline basis functions at a given point, which does not appear to be addressed in prior literature, was studied in this work. An all-at-once algorithm was developed to compute the large number of active THB-spline basis functions efficiently.

The developed techniques are implemented in a 30000-line Fortran 2008 code (OOF-HiDAC). The code is based on hybrid OpenMP/MPI parallelism and is capable for large two- and three-dimensional problems. The methodology was first used to model stationary cracks and crack growth. The Heaviside function and William's tip displacement functions were chosen as crack face and tip enriching functions, respectively. The former function can represent the displacement jump across the crack face whereas the latter functions are used to reproduce the tip singularity. Next, the Stefan problem was modeled with a sharp phase interface that is enriched with hybrid function/derivative values. The function enrichment enables direct imposition of the Gibb-Thomson condition, while the derivative enrichment can naturally cap-

ture the heat flux jump across the interface. Several benchmark problems including isotropic and anisotropic solidification in supercooled liquid were modeled. Finally, the shape optimization through configuration of regular-shaped, finite-sized heterogeneities were studied. The optimization sensitivity is determined by the recently derived configurational derivative. Choosing the THB-spline as domain approximation, a sufficiently smooth behavioral field were obtained near material interfaces for accurate calculation of the configurational derivative. Several numerical examples, including two-dimensional fracture-resistant design and three-dimensional simultaneous optimization of hole location and orientation, were presented.

11.2 Recommendations for Future Research

11.2.1 Technique: T-splines

The T-splines [100, 101] were initially developed for computer-aided geometric design and later applied to engineering analysis [102]. As an alternative local refinement technique, the T-spline framework was compared with THB-splines in Table 6.1. It is worth noting that the main disadvantages of T-splines may be overcome with a good algorithm. In contrast, the drawbacks of THB-splines, such as the need for uniform mesh, are intrinsic limitations.

In this work, the local refinement was enabled by the THB-spline framework due to its ease of implementation. Having already been widely adopted in CAD models, the T-splines will be a powerful analysis tool when a mature algorithm for analysis-suitable T-spline generation becomes available.

11.2.2 Technique: Variational Collocation Method

In the context of computational mechanics, the partial differential equations (PDEs) are solved through weak forms, which are constructed by multiplying the PDEs with a weight function and then integrating over the problem domain. A widely used

approach to solving the weak forms is the Galerkin method which selects the same space for the solution and weight function. Alternatively, the strong form of a PDE can be directly solved by the collocation method which enforces the PDE at a set of collocation points. A significant advantage of the collocation method is that it only requires one point evaluation per unknown whereas the Galerkin method typically needs a full Gauss quadrature over all of the elements. Thus, the collocation method is much faster than the Galerkin method on a per degree of freedom basis. However, without a careful selection of the collocation points, the accuracy of the collocation method per degree of freedom is much lower than that of the Galerkin method.

Recently, Gomez and De Lorenzis [96] proposed a variational collocation method which can produce an exact Galerkin solution by collocation at a special set of points, referred as *Cauchy-Galerkin points*. Montardini et al. [97] further improved the performance of this method for odd-degree splines. The variational collocation method, achieving the efficiency and accuracy simultaneously, is a potential substitute for the Gaussian quadrature. However, this method currently suffers from two limitations:

1. The approximation needs to have $\mathscr{C}^1$ or higher order smoothness, i.e., no singularity, weak and strong discontinuity. This constraint strongly hinters the application of the variational collocation method to multi-phase and moving boundary problems.

2. The analytical solution for the Cauchy-Galerkin points is only known at this time for uniform meshes. Therefore, although the Cauchy-Galerkin points exist for general non-uniform B-splines, THB-splines and T-splines, it is not clear how to obtain them in these cases.

The technique will be a very powerful tool to facilitate the enriched isogeometric analysis if the above two issues can be addressed.

11.2.3 Application: Three-dimensional (3D) Crack Propagation

Several fracture examples were presented in Chapter 8 to demonstrate the proposed enriched isogeometric formulation. A natural extension of this work is the simulation of 3D crack propagation. In the existing literature, majority of techniques for modeling 3D crack propagation rely on an implicit description of the crack surface such as level-sets [160–163]. Recently, an explicit crack representation by means of triangulation was developed to overcome the weakness of the level-set method. However, the crack surface is piece-wise planar ($\mathscr{C}^0$ continuous) and is exact only in the limit of refinement. In the enriched isogeometric analysis, the crack can be represented by a sufficiently smooth NURBS (or T-spline) entity, allowing an accurate calculation of the geometric quantities as the crack propagates. The possible challenges that may be encountered in the extension to 3D crack growth are summarized as follows:

1. Update of the crack surface based on linear elastic fracture mechanics while maintaining a $\mathscr{C}^1$ or higher order continuity.

2. Development of new enriching functions and degrees of freedom associated with 3D crack front. Although the Heaviside function naturally works for the 3D crack surface, the proposed tip enriching functions of Eq. (8.9) only supports two-dimensional cracks.

3. Regeneration of the underlying mesh. It was shown that the generation of the adaptive THB-spline mesh was highly efficient. In fact, the efficiency can be further improved by exploiting the nature of crack propagation by observing that the existing crack does not change in each update. Therefore, it is not necessary to update the whole adaptive mesh but just the region around the crack front. Since the THB-spline mesh is stored in a kd-tree structure, the partial update of the mesh, also referred as *semi-remeshing*, can be carried out by simply expanding the tree.

11.2.4 Application: Diffusion Driven Phase Evolution Problems

The complex physics of diffusion driven phase evolution involves, in addition to bulk diffusion of species, surface phenomena such as surface diffusion motion of voids, grain boundary evolution, and possibly interfacial reactions [164]. The dynamics of the formation, growth and motion of new phases such as voids and intermetalic compounds, are affected by the geometric properties (e.g., interface normal and curvature) and the physical state of stress in the solid as well as the thermal and electrical loads on the solid. As an example, the intermetalic growth in solder joints is illustrated in Figure 11.1.

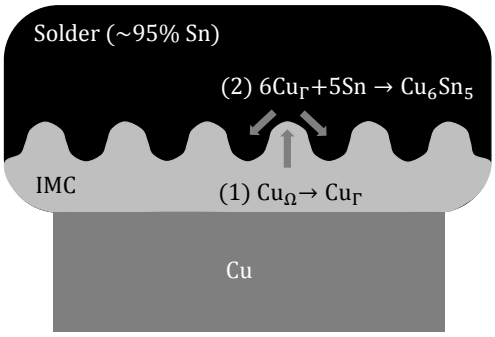


Figure 11.1. Schematic of intermetallic compound (IMC) growth in solder joints. The formation of Cu_6Sn_5 is the consequence of the diffusion of Cu through the Cu_6Sn_5 followed by the reaction with the Sn that constitute over 95% of the modern solder alloys.

The primary challenges to modeling the phase evolution are the tracking of the interface and the application of boundary conditions on the moving interfaces. The classical phase-field solution to the phase evolution problems relies on implicit representation of the geometry, and models the interfaces as being diffuse increasing the mathematical complexity of the resulting equations. The phase-field equations are often non-linear and non-convex, and the phase-field variables require a large number of additional degrees of freedom to represent the geometry relative to a sharp interface model. Alternatively, the enriched isogeometric analysis can be utilized to model

the diffusion driven phase evolution problems. The technique relies on explicit modeling of the phase boundary and so normals and curvatures are explicitly computed at any point on the boundary, which enables a strong imposition of the interfacial conditions. Meanwhile, the high local concentration gradient around the interface can be captured with the THB-spline approximation. Procedures for adaptive time stepping, refinement and coarsening of geometry may also be developed to increase the stability and efficiency of the proposed methodology.

LIST OF REFERENCES

LIST OF REFERENCES

[1] J.M. Melenk. *On generalized finite element methods*. PhD thesis, The University of Maryland, 1995.

[2] T. Strouboulis, I. Babuška, and K. Copps. The design and analysis of the generalized finite element method. *Computer methods in applied mechanics and engineering*, 181(1):43–69, 2000.

[3] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International journal for numerical methods in engineering*, 45(5):601–620, 1999.

[4] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag New York, 2003.

[5] L.-Q. Chen. Phase-field models for microstructure evolution. *Annual review of materials research*, 32(1):113–140, 2002.

[6] H. Ji, D. Chopp, and J.E. Dolbow. A hybrid extended finite element level set method. *International Journal for Numerical Methods in Engineering*, 54(8):1209–1233, July 2002.

[7] F.P. Renken and G. Subbarayan. NURBS-based solutions to inverse boundary problems in droplet shape prediction. *Computer Methods in Applied Mechanics and Engineering*, 190(11-12):1391–1406, December 2000.

[8] D. Natekar, X. Zhang, and G. Subbarayan. Constructive solid analysis: a hierarchical, geometry-based meshless analysis procedure for integrated design and analysis. *Computer-Aided Design*, 36(5):473–486, April 2004.

[9] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, October 2005.

[10] D.J. Benson, Y. Bazilevs, E. De Luycker, M. C. Hsu, M. Scott, T.J.R. Hughes, and T. Belytschko. A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to XFEM. *International Journal for Numerical Methods in Engineering*, 83:765–785, 2010.

[11] E. De Luycker, D.J. Benson, T. Belytschko, Y. Bazilevs, and M.C. Hsu. X-fem in isogeometric analysis for linear fracture mechanics. *International Journal for Numerical Methods in Engineering*, 87(6):541–565, 2011.

[12] S.S. Ghorashi, N. Valizadeh, and S. Mohammadi. Extended isogeometric analysis for simulation of stationary and propagating cracks. *International Journal for Numerical Methods in Engineering*, 89(9):1069–1101, 2012.

[13] A. Tambat and G. Subbarayan. Isogeometric enriched field approximations. *Computer Methods in Applied Mechanics and Engineering*, 245-246:1–21, October 2012.

[14] A. Tambat and G. Subbarayan. Simulations of arbitrary crack path deflection at a material interface in layered structures. *Engineering Fracture Mechanics*, 141:124–139, 2015.

[15] D.R. Lynch and K. O'Neill. Continuously deforming finite elements for the solution of parabolic problems, with and without phase change. *International Journal for Numerical Methods in Engineering*, 17(1):81–96, January 1981.

[16] M.R. Albert and K. O'Neill. Moving boundary-moving mesh analysis of phase change using finite elements with transfinite mappings. *International Journal for Numerical Methods in Engineering*, 23(4):591–607, April 1986.

[17] L. Maréchal. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *Proceedings of the 18th International Meshing Roundtable*, pages 65–84. Springer, 2009.

[18] A. Ural, G. Heber, P.A. Wawrzynek, A.R. Ingraffea, D.G. Lewicki, and J.B.C. Neto. Three-dimensional, parallel, finite element simulation of fatigue crack growth in a spiral bevel pinion gear. *Engineering Fracture Mechanics*, 72(8):1148–1170, 2005.

[19] G. Beckett, J.A. Mackenzie, and M.L. Robertson. A moving mesh finite element method for the solution of two-dimensional stefan problems. *Journal of Computational Physics*, 168(2):500–518, April 2001.

[20] T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method. *Computer methods in applied mechanics and engineering*, 190(32):4081–4193, 2001.

[21] J. Dolbow, N. Moës, and T. Belytschko. Discontinuous enrichment in finite elements with a partition of unity method. *Finite elements in analysis and design*, 36(3):235–260, 2000.

[22] N. Sukumar, D.L. Chopp, and B. Moran. Extended finite element method and fast marching method for three-dimensional fatigue crack propagation. *Engineering Fracture Mechanics*, 70(1):29–48, 2003.

[23] T. Belytschko and R. Gracie. On xfem applications to dislocations and interfaces. *International Journal of Plasticity*, 23(10):1721–1738, 2007.

[24] J. Oswald, R. Gracie, R. Khare, and T. Belytschko. An extended finite element method for dislocations in complex geometries: Thin films and nanotubes. *Computer Methods in Applied Mechanics and Engineering*, 198(21):1872–1886, 2009.

[25] J. Chessa and P. Smolinski. The extended finite element method (XFEM) for solidification problems. *International Journal for Numerical Methods in Engineering*, 53(8):1959–1977, March 2002.

[26] J. Dolbow, E. Fried, and H. Ji. A numerical strategy for investigating the kinetic response of stimulus-responsive hydrogels. *Computer Methods in Applied Mechanics and Engineering*, 194(42):4447–4480, 2005.

[27] R. Duddu, S. Bordas, D. Chopp, and B. Moran. A combined extended finite element and level set method for biofilm growth. *International Journal for Numerical Methods in Engineering*, 74(5):848–870, 2008.

[28] T.J.R. Hughes, L.P. Franca, and G.M. Hulbert. A new finite element formulation for computational fluid dynamics/ VIII. The galerkin least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173–189, 1989.

[29] M.J. Borden, C.V. Verhoosel, M.A. Scott, T.J.R. Hughes, and C.M. Landis. A phase-field description of dynamic brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 217:77–95, 2012.

[30] M.J. Borden, T.J.R. Hughes, C.M. Landis, and C.V. Verhoosel. A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework. *Computer Methods in Applied Mechanics and Engineering*, 273:100–118, 2014.

[31] G. Caginalp. An analysis of a phase field model of a free boundary. *Archive for Rational Mechanics and Analysis*, 92(3):205–245, 1986.

[32] A. Karma and W.-J. Rappel. Phase-field method for computationally efficient modeling of solidification with arbitrary interface kinetics. *Physical Review E*, 53(4):R3017–R3020, April 1996.

[33] A. Karma and W.-J. Rappel. Quantitative phase-field modeling of dendritic growth in two and three dimensions. *Physical Review E*, 57(4):4323–4349, April 1998.

[34] W.J. Boettinger, J.A. Warren, C. Beckermann, and A. Karma. Phase-field simulation of solidification. *Annual Review of Materials Research*, 32(1):163–194, August 2002.

[35] I. Babuška and W.C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[36] M. Ainsworth and J.T. Oden. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1):1–88, 1997.

[37] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[38] W.K. Liu, S. Jun, and Y.F. Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995.

[39] S.N. Atluri and T. Zhu. A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Computational mechanics*, 22(2):117–127, 1998.

[40] T. Belytschko, Y.Y. Lu, and L. Gu. Crack propagation by element-free galerkin methods. *Engineering Fracture Mechanics*, 51(2):295–315, 1995.

[41] T. Belytschko, Y.Y. Lu, L. Gu, and M. Tabbara. Element-free galerkin methods for static and dynamic fracture. *International Journal of Solids and Structures*, 32(17):2547–2570, 1995.

[42] P. Krysl and T. Belytschko. The element free galerkin method for dynamic propagation of arbitrary 3-d cracks. *International Journal for Numerical Methods in Engineering*, 44(6):767–800, 1999.

[43] S.-P. Lin, J.-S. Chen, and S. Liang. A damage analysis for brittle materials using stochastic micro-structural information. *Computational Mechanics*, 57(3):371–385, 2016.

[44] S. Fernández-Méndez and A. Huerta. Imposing essential boundary conditions in mesh-free methods. *Computer methods in applied mechanics and engineering*, 193(12):1257–1275, 2004.

[45] J.-S. Chen, C.-T. Wu, S. Yoon, and Y. You. A stabilized conforming nodal integration for galerkin mesh-free methods. *International journal for numerical methods in engineering*, 50(2):435–466, 2001.

[46] J.-S. Chen, S. Yoon, and C.-T. Wu. Non-linear version of stabilized conforming nodal integration for galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering*, 53(12):2587–2615, 2002.

[47] D. Wang and J.-S. Chen. Locking-free stabilized conforming nodal integration for meshfree mindlin–reissner plate formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(12):1065–1083, 2004.

[48] M.A. Puso, J.S. Chen, E. Zywicz, and W. Elmer. Meshfree and finite element nodal integration methods. *International Journal for Numerical Methods in Engineering*, 74(3):416–446, 2008.

[49] K. Upreti, T. Song, A. Tambat, and G. Subbarayan. Algebraic distance estimations for enriched isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 280:28–56, 2014.

[50] C. Giannelli, B. JüTtler, and H. Speleers. Thb-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[51] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag New York, 2nd edition edition, 1997.

[52] J.M. Melenk and I. Babuška. The partition of unity finite element method: basic theory and applications. *Computer methods in applied mechanics and engineering*, 139(1):289–314, 1996.

[53] M. Rayasam, V. Srinivasan, and G. Subbarayan. CAD inspired hierarchical partition of unity constructions for NURBS-based, meshless design, analysis and optimization. *International Journal for Numerical Methods in Engineering*, 72(12):1452–1489, December 2007.

[54] V. Srinivasan, G. Subbarayan, S. Radhakrishnan, D. Pantuso, and S. Shankar. Hierarchical partition of unity field compositions (hpfc) for optimal design in the presence of cracks. *Mechanics of Advanced Materials and Structures*, 17(7):467–480, 2010.

[55] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[56] M.E. Mortenson. *Geometric Modeling*. John Wiley & Sons, 1985.

[57] Y.L. Ma and W.T. Hewitt. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Computer Aided Geometric Design*, 20(2):79–99, May 2003.

[58] X.-D. Chen, J.-H. Yong, G. Wang, J.-C. Paul, and G. Xu. Computing the minimum distance between a point and a NURBS curve. *Computer-Aided Design*, 40(10-11):1051–1054, October 2008.

[59] M.W. Jones, J.A. Baerentzen, and M. Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on visualization and Computer Graphics*, 12(4):581–599, 2006.

[60] B.A. Payne and A.W. Toga. Distance field manipulation of surface models. *IEEE Computer graphics and applications*, 12(1):65–71, 1992.

[61] A. Guezlec. meshsweeper: dynamic point-to-polygonal mesh distance and applications. *IEEE Transactions on visualization and computer graphics*, 7(1):47–61, 2001.

[62] A. Biswas and V. Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3):133–159, 2004.

[63] V.L. Rvachev. On the analytical description of some geometric objects. *Reports of Ukrainian Academy of Sciences*, 153(4):765–767, 1963.

[64] T.W. Sederberg. *Implicit and parametric curves and surfaces for computer aided geometric design*. PhD thesis, Purdue, 1983.

[65] V. Shapiro. Theory of R-functions and applications: A primer. Technical Report TR91-1219, Department of Computer Science, Cornell University, Ithaca, New York, 1991.

[66] S. Abhyankar. *Algebraic Geometry for Scientists and Engineers*, volume 35 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, Rhode Island, July 1990.

[67] L.A. Piegl and W. Tiller. Parametrization for surface fitting in reverse engineering. *Computer-Aided Design*, 33(8):593–603, 2001.

[68] P. Wriggers. *Computational contact mechanics*. Springer Science & Business Media, 2006.

[69] L. De Lorenzis, I. Temizer, P. Wriggers, and G. Zavarise. A large deformation frictional contact formulation using nurbs-based isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 87(13):1278–1300, 2011.

[70] Y. Bazilevs, V.M. Calo, Y. Zhang, and T.J.R. Hughes. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38(4-5):310–322, 2006.

[71] T. Song, K. Upreti, and G. Subbarayan. A sharp interface isogeometric solution to the stefan problem. *Computer Methods in Applied Mechanics and Engineering*, 284:556–582, 2015.

[72] X.-D. Chen, H. Su, J.-H. Yong, J.-C. Paul, and J.-G. Sun. A counterexample on point inversion and projection for nurbs curve. *Computer Aided Geometric Design*, 24(5):302, 2007.

[73] I. Selimovic. Improved algorithms for the projection of points on nurbs curves and surfaces. *Computer Aided Geometric Design*, 23(5):439–445, 2006.

[74] X.-D. Chen, J.-H. Yong, G. Wang, J.-C. Paul, and G. Xu. Computing the minimum distance between a point and a nurbs curve. *Computer-Aided Design*, 40(10):1051–1054, 2008.

[75] S.-M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 22(3):251–260, 2005.

[76] X.-M. Liu, L. Yang, J.-H. Yong, H.-J. Gu, and J.-G. Sun. A torus patch approximation approach for point projection on surfaces. *Computer Aided Geometric Design*, 26(5):593–598, 2009.

[77] R.N. Goldman, T.W. Sederberg, and D.C. Anderson. Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves. *Computer Aided Geometric Design*, 1(4):327–356, December 1984.

[78] C.S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

[79] R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual review of fluid mechanics*, 31(1):567–603, 1999.

[80] C.S. Peskin. The immersed boundary method. *Acta numerica*, 11:479–517, 2002.

[81] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[82] F.P.T. Baaijens. A fictitious domain/mortar element method for fluid-structure interaction. *International Journal for Numerical Methods in Fluids*, 35(7):743–761, 2001.

[83] L. Zhang, A. Gerstenberger, X. Wang, and W.K. Liu. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 193(21):2051–2067, 2004.

[84] A. Gerstenberger and W.A. Wall. Enhancement of fixed-grid methods towards complex fluid–structure interaction applications. *International Journal for Numerical Methods in Fluids*, 57(9):1227–1248, 2008.

[85] T. Rüberg and F. Cirak. A fixed-grid b-spline finite element technique for fluid–structure interaction. *International Journal for Numerical Methods in Fluids*, 74(9):623–660, 2014.

[86] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 2015.

[87] K.W. Cheng and T.-P. Fries. Higher-order xfem for curved strong and weak discontinuities. *International Journal for Numerical Methods in Engineering*, 82(5):564–590, 2010.

[88] L. Kudela, N. Zander, T. Bog, S. Kollmannsberger, and E. Rank. Efficient and accurate numerical quadrature for immersed boundary methods. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):1, 2015.

[89] X. Zhang, M. Rayasam, and G. Subbarayan. A meshless, compositional approach to shape optimal design. *Computer Methods in Applied Mechanics and Engineering*, 196(17-20):2130–2146, March 2007.

[90] H.-J. Kim, Y.-D. Seo, and S.-K. Youn. Isogeometric analysis for trimmed cad surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(37):2982–2995, 2009.

[91] H.-J. Kim, Y.-D. Seo, and S.-K. Youn. Isogeometric analysis with trimming technique for problems of arbitrary complex topology. *Computer Methods in Applied Mechanics and Engineering*, 199(45):2796–2812, 2010.

[92] A. Düster, J. Parvizian, Z. Yang, and E. Rank. The finite cell method for three-dimensional problems of solid mechanics. *Computer methods in applied mechanics and engineering*, 197(45):3768–3782, 2008.

[93] E. Rank, M. Ruess, S. Kollmannsberger, D. Schillinger, and A. Düster. Geometric modeling, isogeometric analysis and the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 249:104–115, 2012.

[94] D. Schillinger, L. Dede, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, and T.J.R. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of nurbs, immersed boundary methods, and t-spline cad surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249:116–150, 2012.

[95] V. Thiagarajan and V. Shapiro. Adaptively weighted numerical integration in the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 311:250–279, 2016.

[96] H. Gomez and L. De Lorenzis. The variational collocation method. *Computer Methods in Applied Mechanics and Engineering*, 309:152–181, 2016.

[97] M. Montardini, G. Sangalli, and L. Tamellini. Optimal-order isogeometric collocation at galerkin superconvergent points. *Computer Methods in Applied Mechanics and Engineering*, 2016.

[98] M. De Berg, M. Van Kreveld, M. Overmars, and O.C. Schwarzkopf. *Computational geometry*. Springer, 2000.

[99] K. Upreti. *Algebraic level sets for CAD/CAE integration and moving boundary problems*. PhD thesis, Purdue, 2014.

[100] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and t-nurccs. In *ACM transactions on graphics (TOG)*, volume 22, pages 477–484. ACM, 2003.

[101] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *Acm transactions on graphics*, 23(3):276–283, 2004.

[102] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229–263, 2010.

[103] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the t-spline blending functions associated with some particular t-meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(23):1437–1445, 2010.

[104] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, and M.A. Scott. On linear independence of t-spline blending functions. *Computer Aided Geometric Design*, 29(1):63–76, 2012.

[105] M.A. Scott, X. Li, T.W. Sederberg, and T.J.R. Hughes. Local refinement of analysis-suitable t-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206–222, 2012.

[106] P. Morgenstern and D. Peterseim. Analysis-suitable adaptive t-mesh refinement with linear complexity. *Computer Aided Geometric Design*, 34:50–66, 2015.

[107] D.R. Forsey and R.H. Bartels. Hierarchical b-spline refinement. *ACM Siggraph Computer Graphics*, 22(4):205–212, 1988.

[108] A.-V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3554–3567, 2011.

[109] G. Kiss, C. Giannelli, and B. Jüttler. *Algorithms and Data Structures for Truncated Hierarchical B–splines*, pages 304–323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[110] C. Giannelli, B. Jüttler, S.K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Špeh. Thb-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 299:337–365, 2016.

[111] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. Curfman McInnes, K. Rupp, B. Smith, and H. Zhang. Petsc users manual revision 3.5. *Argonne National Laboratory (ANL)*, 2014.

[112] K.A. Johannessen, F. Remonato, and T. Kvamsdal. On the similarities and differences between classical hierarchical, truncated hierarchical and lr b-splines. *Computer Methods in Applied Mechanics and Engineering*, 291:64–101, 2015.

[113] M. Metcalf, J. Reid, and M. Cohen. *Modern Fortran Explained*. Oxford University Press, 2011.

[114] P. Pacheco. *An introduction to parallel programming*. Elsevier, 2011.

[115] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[116] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[117] P.R. Amestoy, L.S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[118] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2):136–156, 2006.

[119] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[120] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' guide*, volume 9. Siam, 1999.

[121] C.V. Verhoosel, M.A. Scott, R. De Borst, and T.J.R. Hughes. An isogeometric approach to cohesive zone modeling. *International Journal for Numerical Methods in Engineering*, 87(1-5):336–360, 2011.

[122] X.Y. Liu, Q.Z. Xiao, and B.L. Karihaloo. Xfem for direct evaluation of mixed mode sifs in homogeneous and bi-materials. *International Journal for Numerical Methods in Engineering*, 59(8):1103–1118, 2004.

[123] M.L. Williams. Stress singularities resulting from various boundary conditions. *Journal of applied mechanics*, 19(4):526–528, 1952.

[124] J.R. Rice. A path independent integral and the approximate analysis of strain concentration by notches and cracks. *Journal of applied mechanics*, 35(2):379–386, 1968.

[125] C.T. Sun and Z.-H. Jin. *Fracture Mechanics*. Academic Press, 2012.

[126] T.L. Anderson. *Fracture mechanics: fundamentals and applications*. CRC Press, 3rd edition edition, 2005.

[127] J. Crank. *Free and Moving Boundary Problems*. Clarendon press Oxford, 1984.

[128] S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving stefan problems. *Journal of Computational Physics*, 135(1):8–29, July 1997.

[129] Y.T. Kim, N. Goldenfeld, and J. Dantzig. Computation of dendritic microstructures using a level set method. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 62(2):2471–2474, August 2000.

[130] N. Zabaras, B. Ganapathysubramanian, and L. Tan. Modelling dendritic solidification with melt convection using the extended finite element method. *Journal of Computational Physics*, 218(1):200–227, 2006.

[131] D. Adalsteinsson and J.A. Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22, January 1999.

[132] H.D. Ceniceros, R.L. Nós, and A.M. Roma. Three-dimensional, fully adaptive simulations of phase-field fluid models. *Journal of Computational Physics*, 229(17):6135 – 6155, 2010.

[133] P. Yue, C. Zhou, J.J. Feng, C.F. Ollivier-Gooch, and H.H. Hu. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219(1):47 – 67, 2006.

[134] C. Zhou, P. Yue, J.J. Feng, C.F. Ollivier-Gooch, and H.H. Hu. 3d phase-field simulations of interfacial dynamics in newtonian and viscoelastic fluids. *Journal of Computational Physics*, 229(2):498 – 511, 2010.

[135] H. Gomez, V. Calo, Y. Bazilevs, and T.J.R. Hughes. Isogeometric analysis of the Cahn-Hilliard phase-field model. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4333–4352, September 2008.

[136] V. Alexiades and A.D. Solomon. *Mathematical Modeling of Melting and Freezing processes*. Hemisphere Publishing Corporation, 1993.

[137] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineerng analysis. *International Journal for Numerical Methods in Engineering*, 24(2):337–357, 1987.

[138] M. Ainsworth and J.T. Oden. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1-2):1–88, March 1997.

[139] E.T.Y. Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21(6):363–370, July 1989.

[140] R. Courant, K. Friedrichs, and H. Lewy. On the Partial Difference Equations of Mathematical Physics. *IBM Journal of Research and Development*, 11(2):215–234, March 1967.

[141] K.A. Rathjen and L.M. Jiji. Heat conduction with melting or freezing in a corner. *Journal of Heat Transfer*, 93(1):101–109, 1971.

[142] J.A. Sethian and J. Straint. Crystal growth and dendritic solidification. *Journal of Computational Physics*, 98(2):231–253, February 1992.

[143] L. Tan. *Multiscale modeling of solidification of multi-component alloys*. PhD thesis, Cornell, 2007.

[144] N. Provatas, N. Goldenfeld, and J. Dantzig. Efficient computation of dendritic microstructures using adaptive mesh refinement. *Physical Review Letters*, 80(15):3308–3311, April 1998.

[145] M.P. Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1(4):193–202, 1989.

[146] M.P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

[147] A. Diaz and O. Sigmund. Checkerboard patterns in layout optimization. *Structural optimization*, 10(1):40–45, 1995.

[148] O. Pironneau. *Optimal shape design for elliptic systems*. Springer Science & Business Media, 2012.

[149] J. Bennett. *The optimum shape: automated structural design*. Springer Science & Business Media, 2012.

[150] C.-Y. Lin and L.-S. Chao. Automated image interpretation for integrated topology and shape optimization. *Structural and Multidisciplinary Optimization*, 20(2):125–137, 2000.

[151] P.-S. Tang and K.-H. Chang. Integration of topology and shape optimization for design of structural components. *Structural and Multidisciplinary Optimization*, 22(1):65–82, 2001.

[152] R. Ansola, J. Canales, J.A. Tárrago, and J. Rasmussen. An integrated approach for shape and topology optimization of shell structures. *Computers & structures*, 80(5):449–458, 2002.

[153] K. Dems and Z. Mróz. On a class of conservation rules associated with sensitivity analysis in linear elasticity. *International Journal of Solids and Structures*, 22(7):737–758, 1986.

[154] J.D. Eshelby. The continuum theory of lattice defects. *Solid state physics*, 3:79–144, 1956.

[155] J.K. Knowles and E. Sternberg. On a class of conservation laws in linearized and finite elastostatics. *Archive for rational mechanics and analysis*, 44(3):187–211, 1972.

[156] B. Budiansky and J.R. Rice. Conservation laws and energy-release rates. *Journal of applied mechanics*, 40(1):201–203, 1973.

[157] H.-Y. Lin and G. Subbarayan. Optimal topological design through insertion and configuration of finite-sized heterogeneities. *International Journal of Solids and Structures*, 50(2):429–446, 2013.

[158] H.-Y. Lin. *Configurational optimization for optimal topological and fracture-resistant designs of solids*. PhD thesis, Purdue, 2014.

[159] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[160] N. Moës, A. Gravouil, and T. Belytschko. Non-planar 3d crack growth by the extended finite element and level setspart i: Mechanical model. *International Journal for Numerical Methods in Engineering*, 53(11):2549–2568, 2002.

[161] A. Gravouil, N. Moës, and T. Belytschko. Non-planar 3d crack growth by the extended finite element and level setspart ii: Level set update. *International Journal for Numerical Methods in Engineering*, 53(11):2569–2586, 2002.

[162] T.C. Gasser and G.A. Holzapfel. 3d crack propagation in unreinforced concrete.: A two-step algorithm for tracking 3d crack paths. *Computer Methods in Applied Mechanics and Engineering*, 195(37):5198–5219, 2006.

[163] T. Rabczuk, S. Bordas, and G. Zi. On three-dimensional modelling of crack growth using partition of unity methods. *Computers & structures*, 88(23):1391–1411, 2010.

[164] A. Udupa, S. Sadasiva, and G. Subbarayan. A framework for studying dynamics and stability of diffusive–reactive interfaces with application to cu6sn5 intermetallic compound growth. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 472(2190), 2016.

VITA

VITA

Tao Song received his B.S. (2012) in Department of Precision Instrument from Tsinghua University, and M.S. (2014) in Mechanical Engineering from Purdue University. His research focuses on development of novel numerical methods for Moving Boundary Problems (MVPs) such as electromigration, crack propagation, phase transition and shape optimization. His interests also include experimental characterization of mechanical properties of electronic components and assemblies. Tao is concerned about reliability issues in IC BEOL/packaging and consumer electronics, and is dedicated to understanding mechanical failure modes based on physical insights as well as numerical studies.

Tao is the recipient of the Tsinghua Outstanding Graduate Award (2012) and the Purdue Adelberg Fellowship (2015). He also won the Best Paper Award (Honorable Mention) and the Best Student Poster Award in the Mechanics Track of IEEE-ITHERM 2016.

During the course of his Ph.D. at Purdue, he also got the opportunity to gain industry experience through summer internships at GLOBALFOUNDRIES Inc. in Summer 2016. He will be joining the hardware reliability group at Microsoft after completing his Ph.D. in 2016.