Justus-Liebig-Universität Gießen,
Mathematisches Institut

Università di Firenze, Dipartimento di
Matematica e Informatica *Ulisse Dini*

Domain discretization and moment-free quadrature for meshless methods

Dissertation zur Erlangung des Grades

Doctor Rerum Naturalium

verliehen von der Justus-Liebig-Universität Gießen,
und des Grades

Dottore di Ricerca in Matematica, Informatica, Statistica

verliehen von der Università di Firenze,
im Rahmen einer Cotutelle-Vereinbarung

Vorgelegt von
Bruno Degli Esposti

Betreuer
Prof. Dr. Oleg Davydov
Prof. Dr. Alessandra Sestini

Promotionszeitraum 2021/2024

UNIVERSITÀ
DEGLI STUDI
FIRENZE

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

[iNdAM
Istituto Nazionale
di Alta Matematica]

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM
Tesi in cotutela con Justus Liebig University Giessen

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN MATEMATICA
CICLO XXXVII

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Matteo Focardi

Domain discretization and moment-free quadrature for meshless methods

Settore Scientifico Disciplinare MATH-05/A

Dottorando
Bruno Degli Esposti

Supervisori
Prof. Oleg Davydov
Prof.ssa Alessandra Sestini

Coordinatore
Prof. Matteo Focardi

Anni 2021/2024

To my supervisors, for your wisdom, your insights, and your never-ending patience. Your trust and respect make me feel worthy of being a doctor more than the title itself ever could.

To my loved ones, near and far, for providing encouragement, love, and support throughout this academic journey. I could not have come this far without you—both in my studies and in life.

The author is a member of the Gruppo Nazionale per il Calcolo Scientifico (GNCS) of the Istituto Nazionale di Alta Matematica (INdAM). The author gratefully acknowledges the support of INdAM, which funded the PhD scholarship, as well as the funding provided through the yearly GNCS projects coordinated by Prof. Francesca Pelosi in 2022 and Prof. Maria Lucia Sampoli in 2024.

Abstract

Meshless methods are a versatile class of numerical techniques that discretize computational domains using scattered, unstructured nodes, avoiding the complexities and limitations of conventional mesh generation. These methods are especially well-suited for problems involving complex geometries and dynamic phenomena such as free boundaries, large deformations, and crack propagation, where traditional meshing approaches often become computational bottlenecks. This thesis addresses two fundamental challenges in the meshless paradigm: robust node generation for real-world 3D domains, and high-order numerical integration on scattered nodes.

First, we propose a fully meshless advancing front node generation algorithm tailored for 3D domains described in the B-rep (boundary representation) format, the standard used in Computer-Aided Design (CAD). Our algorithm efficiently converts trimmed NURBS surfaces into variable-density point clouds that sample both the interior and boundary of the domains. The method operates hierarchically by progressively advancing a front over lower-dimensional entities (edges and faces) before higher-dimensional ones (volumes), and incorporates a novel meshless inclusion test that prevents nodes from being placed outside the domain of interest, even in the case of piecewise smooth boundaries. The algorithm is shown to perform effectively on complex 3D geometries, producing well-distributed node configurations suitable for numerical simulations using meshless finite difference methods.

Second, we introduce a novel approach to numerical integration by simultaneously generating high-order quadrature weights for integrals over Lipschitz domains and their boundaries without relying on meshing or the computation of moments (integrals of basis functions). The weights are obtained on given scattered nodes as a minimum norm solution of a sparse underdetermined linear system arising from a discretization of a suitable boundary value problem. This discretization is achieved using either tensor-product B-spline collocation or meshless finite differences, such as those based on polyharmonic radial basis functions (RBF-FD). The proposed moment-free method is easy to implement, and does not depend on the domain's representation (parametric or implicit), since it only requires as inputs the position of all quadrature nodes and the direction of outward-pointing normals at each node belonging to the boundary. Comprehensive numerical experiments demonstrate the robustness and high accuracy of the method on a number of smooth and piecewise smooth domains in 2D and 3D, including some with reentrant corners and edges.

Together, the contributions presented in this thesis advance the state of the art in meshless methods by addressing critical challenges in node generation and numerical integration, and by providing robust tools for challenging real-world applications involving complex geometries and high-order accuracy requirements.

Sommario

I metodi meshless sono una classe versatile di tecniche numeriche che discretizzano i domini computazionali utilizzando nodi sparsi e non strutturati, evitando le complessità e le limitazioni insite nella generazione convenzionale di mesh. Questi metodi sono particolarmente adatti a problemi che coinvolgono geometrie complesse e fenomeni dinamici come problemi a frontiera libera, grandi deformazioni e propagazione delle fratture, in cui gli approcci tradizionali basati su mesh diventano spesso un collo di bottiglia dal punto di vista computazionale. Questa tesi affronta due sfide fondamentali per il paradigma meshless: la generazione robusta di nodi per domini 3D realistici e l'integrazione numerica di ordine elevato su dati sparsi.

In primo luogo, abbiamo sviluppato un algoritmo completamente meshless di tipo *advancing front* per la generazione di nodi su domini 3D descritti nel formato B-rep (boundary representation), lo standard utilizzato nell'ambito del Computer-Aided Design (CAD). L'algoritmo converte in modo efficiente superfici NURBS con trimming in insiemi di punti a densità variabile che campionano sia l'interno che il bordo dei domini. Il metodo opera in modo gerarchico, avanzando progressivamente il fronte dei nodi su enti geometrici di dimensione inferiore (spigoli e superfici) prima di passare a quelli di dimensione superiore (volumi), e fa ampio uso di un innovativo test di inclusione meshless che impedisce ai nodi di essere posizionati al di fuori del dominio di interesse, anche nel caso di bordi solamente lisci a tratti. L'efficacia dell'algoritmo è stata verificata su geometrie 3D complesse, avendo ottenuto configurazioni di nodi ben distribuite e adatte alle simulazioni numeriche con metodi meshless alle differenze finite.

In secondo luogo, abbiamo introdotto un nuovo approccio per l'integrazione numerica su dati sparsi che genera simultaneamente pesi di quadratura di ordine elevato per integrali su domini lipschitziani e sui loro bordi, senza ricorrere a mesh o al calcolo di momenti (integrali di funzioni base). I pesi sono ottenuti come soluzione di norma minima di un sistema lineare sottodeterminato e sparso, che emerge dalla discretizzazione di un opportuno problema al contorno per equazioni differenziali alle derivate parziali. Tale discretizzazione è ottenuta tramite collocazione di B-spline di tipo tensore prodotto, oppure tramite differenze finite meshless, come quelle basate su funzioni radiali poliarmoniche (RBF-FD). Il metodo proposto, che possiamo definire *mesh-free*, è semplice da implementare e non dipende dalla rappresentazione del dominio (parametrica o implicita), poiché richiede unicamente come input la posizione di tutti i nodi di quadratura e la direzione delle normali uscenti in ogni nodo appartenente al bordo. Esperimenti numerici esaustivi dimostrano la robustezza e l'elevata accuratezza del metodo su domini lisci e lisci a tratti in 2D e 3D, compresi domini con angoli e spigoli rientranti.

Nel complesso, i contributi presentati in questa tesi avanzano lo stato dell'arte nei metodi meshless affrontando delle sfide cruciali nella generazione dei nodi e nell'integrazione numerica, e fornendo strumenti robusti per applicazioni che coinvolgono geometrie complesse e requisiti di accuratezza di ordine elevato.

Zusammenfassung

Netztfreie Methoden sind eine vielseitige Klasse numerischer Techniken, die die Rechengebiete mithilfe verstreuter, unstrukturierter Knoten diskretisieren und so die Komplexitäten und Einschränkungen herkömmlicher Netzgenerierung umgehen. Diese Methoden eignen sich besonders gut für Probleme mit komplexen Geometrien und dynamischen Phänomenen wie freien Oberflächen, großen Deformationen und Rissausbreitungen, bei denen traditionelle Netzgenerierungsansätze häufig zu rechnerischen Engpässen führen. Diese Dissertation befasst sich mit zwei grundlegenden Herausforderungen im Meshless-Paradigma: der robusten Knotengenerierung für anwendungsrelevante 3D-Gebiete und der hochgenauen numerischen Integration auf unregelmässig verstreuten Knoten.

Zunächst wird ein vollständig netztfreier *Advancing Front*-Algorithmus zur Knotengenerierung vorgestellt, der speziell für 3D-Domänen entwickelt wurde, die im B-rep-Format (Boundary Representation) dargestellt sind – dem Standard in der Computer-Aided Design (CAD). Der Algorithmus konvertiert effizient getrimmte NURBS-Flächen in Punktwolken mit variabler Dichte, die sowohl das Innere als auch den Rand des Gebiets diskretisieren. Die Methode arbeitet hierarchisch, indem sie sukzessive eine Front über niedrigdimensionale Entitäten (Kanten und Flächen) vorantreibt, bevor sie sich auf höherdimensionale Entitäten (Volumen) konzentriert. Dabei wird ein neuartiger, netzfreier Einschlusstest integriert, der verhindert, dass Knoten außerhalb des Gebiets platziert werden, selbst bei nur stückweise glattem Rand. Der Algorithmus zeigt seine Effektivität an komplexen 3D-Geometrien und erzeugt gleichmäßig verteilte Knoten-Konfigurationen, die für numerische Simulationen mit netzfreien Finite-Differenzen-Methoden geeignet sind.

Zweitens wird ein neuartiger Ansatz zur numerischen Integration vorgestellt, der gleichzeitig hochgenaue Quadraturgewichte für Integrale über Lipschitz-Gebiete und deren Ränder erzeugt, ohne auf Meshing oder die Berechnung von Momenten (Integrale von Basisfunktionen) zurückzugreifen. Die Gewichte werden auf gegebenen, unregelmässig verstreuten Knoten als Minimum-Norm-Lösung eines dünnbesetzten, unterbestimmten linearen Gleichungssystems berechnet, das sich aus der Diskretisierung eines geeigneten Randwertproblems ergibt. Diese Diskretisierung wird entweder mittels Tensorprodukt-B-Spline-Kollokation oder mit den netzfreien finiten Differenzen wie solchen auf Basis von polyharmonischen Radialbasisfunktionen (RBF-FD) erreicht. Die vorgeschlagene, momentfreie Methode ist leicht zu implementieren und hängt nicht von der Repräsentation des Gebiets (parametrisch oder implizit) ab, da sie lediglich die Position aller Quadraturknoten und die Richtungen der äußeren Normalen an den Randknoten benötigt. Umfassende numerische Experimente belegen die Robustheit und hohe Genauigkeit der Methode auf einer Reihe von glatten und stückweise glatten Gebieten in 2D und 3D, einschließlich solcher mit einspringenden Ecken und Kanten.

Insgesamt leisten die in dieser Dissertation vorgestellten Beiträge einen Fortschritt im Stand der Technik von netzfreien Methoden, indem sie kritische Herausforderungen bei der Knotengenerierung und der numerischen Integration adressieren und robuste Werkzeuge für anspruchsvolle Anwendungsfälle mit komplexen Geometrien und Anforderungen an hohe Genauigkeit bereitstellen.

Contents

CHAPTER 1

Introduction

Meshless methods, also known as meshfree methods, are a broad class of numerical techniques whose defining feature is the ability to discretize the computational domain using only sets of unstructured and unconnected points, eliminating the need for conventional meshing and most of its inherent limitations and inefficiencies [8, 71]. Unlike mesh-based methods, meshless methods do not rely on grids or tessellations as primary discretizations. Instead, they use a set of scattered nodes within the problem domain and along its boundaries to *represent* the domain and its boundaries. While auxiliary structures, such as background grids, may sometimes be used for specific tasks like numerical integration, these structures do not play a role in the discretization process and are not designed to fit the geometry of the computational domain. Overall, this approach makes meshless methods particularly suited for problems involving large deformations, free boundaries, complex fluid-structure interactions and crack propagation, where meshing is especially challenging and often ends up being the bottleneck in numerical simulations [72].

This thesis addresses two key challenges in the context of meshless methods: domain discretization and numerical integration. On the one hand, domain discretization, which involves generating a set of scattered points to represent a computational domain, is a fundamental requirement in all meshless methods. On the other hand, numerical integration is not required by all meshless approaches, but is a core operation when discretizing the weak form of partial differential equations, and when carrying out postprocessing tasks such as error estimation or the computation of physical quantities like fluxes and energies. Both of these challenges, node generation and the design of stable, high-order quadrature formulas, pose significant difficulties in the meshless context, and are the focus of this work.

A common trend in the existing literature is the use of mesh vertices as scattered nodes for meshless methods. This practice, despite being contrary to the spirit of meshless approaches, remains widespread, and can lead to the misconception that node generation is as computationally demanding as mesh generation. We argue that this reliance on mesh-based nodes arises from the relative scarcity of research addressing node generation algorithms specifically tailored for complex, real-world 3D geometries [107]. This gap in the literature is surprising, given the popularity of meshless methods and their aim to surpass the limitations of meshing. To address this shortcoming, this thesis presents a fully meshless node-generation approach for complex geometries, advancing the state of the art in this critical area.

Another significant challenge pertains to numerical integration in the meshless context. Without a mesh, i.e. a computable partition of the integration domain, the classical and efficient approach of composite quadrature cannot be employed, and so the development of stable, high-order quadrature formulas on scattered nodes requires entirely different ideas. Our main contribution in this sense is a novel

moment-free design principle obtained by discretizing the divergence theorem by numerical differentiation formulas, an approach particularly suited for meshless contexts.

Before elaborating on our contributions, we briefly recall the history of meshless methods as a foundation for understanding the broader context of this thesis. The first successful scheme of this kind dates back to 1977, when Smoothed Particle Hydrodynamics (SPH) was introduced for the solution of challenging fluid simulation problems arising in astrophysics [40, 75]. After almost 50 years, the family of meshless methods has grown considerably, often exchanging ideas with alternative discretization techniques, but always keeping its distinctive versatility and geometric flexibility. Today, this area of research remains in a rapid development stage, with new methods and techniques being constantly proposed.

Meshless methods can be roughly categorized into four families. The first one is particle methods, such as SPH, where the emphasis is on movement of the nodes according to the Lagrangian point of view in continuum mechanics. The second is that of Galerkin methods, such as the Discrete Element Method [79], the Element-Free Galerkin Method [9], and the Reproducing Kernel Particle Method [73]. These methods work by discretizing the weak form of the differential problem being solved, with each node usually corresponding to a degree of freedom in the trial space. Another family is collocation methods, such as Kansa's method [59, 60] and the RBF-PU method [4, 115], where the strong form of the differential problem is discretized instead. Finally, the fourth family is that of finite difference methods, such as GFDM [56, 70, 87] and RBF-FD [64, 94, 111, 117], which also work by discretizing the strong form of the differential problem, but, compared to collocation methods, accomplish this goal by discretizing the linear differential operators as linear combinations of pointwise values instead of defining an ansatz space on which the strong form is imposed at every node. Because of the specific techniques being used to define our new quadrature formulas, our focus will be in particular on meshless finite difference methods, although all meshless methods can benefit from the advances in node generation and numerical integration presented in this work.

Node generation, although a simpler task compared to mesh generation, remains challenging for complex domains. Despite its importance, the theory of node generation is relatively underdeveloped, with few works specifically addressing node placement algorithms for complex geometries, especially in 3D and allowing for variable node density [107]. For meshless methods to deliver on their promise of flexibility for domains where meshing is impractical, improvements in node generation techniques are essential.

In this thesis, we contribute to advancing the state of the art by introducing a fully meshless, variable-density advancing front method tailored to 3D domains in boundary representation form (B-rep), the standard in Computer-Aided Design. Our node generation algorithm provides a completely meshless pipeline, enabling the direct conversion of the B-rep, given as a list of trimmed NURBS surface patches, into variable-density scattered nodes that sample both the interior and boundary of the corresponding domain. The algorithm is organized hierarchically, discretizing lower-dimensional entities before higher-dimensional ones, and makes use of a novel meshless inclusion test to prevent the advancing front method from generating nodes outside of the domain of interest.

With regards to numerical integration in a meshless setting, which is a fundamental building block for some meshless methods such as those of Galerkin type, the main achievement in this thesis is the development and validation of a new design principle for quadrature formulas, alternative to the classical notion of *exactness* over a functional space (typically, a polynomial space), also known as *moment-fitting*.

The most distinguishing aspect of our new approach is that stable quadrature weights are found as a minimum-norm solution to a sparse linear system obtained as a discretization of the divergence theorem by numerical differentiation formulas, such as those used in polyharmonic RBF-FD, or by collocation of tensor-product B-splines defined on a bounding box around the integration domain. Our approach simultaneously produces quadrature formulas for a domain Ω and for its boundary, even when $\partial\Omega$ is not smooth. Unlike classical methods based on exactness of quadrature formulas over a finite-dimensional vector space, ours is *moment-free*, in the sense that integrals of monomials or other basis functions (collectively referred to as *moments*) do not have to be (pre)computed. Also, in contrast to composite quadrature, no partition of either Ω or $\partial\Omega$ is needed: our formulas are completely *meshless*, in the sense that only nodes in Ω, nodes on $\partial\Omega$ and outward-pointing normals at the boundary nodes are needed as inputs to the algorithm. Therefore the method is not tied to any specific representation of the integration domain, e.g. parametric or implicit. Numerical experiments and theoretical arguments show that the convergence order of the quadrature formulas matches the consistency order of numerical differentiation formulas or equivalent spline collocation schemes. In either case, the effectiveness of both algorithms is demonstrated by extensive numerical experiments, with the largest tests involving more than 10^5 quadrature nodes, and convergence orders up to 7 for both interior and boundary quadrature on several 2D and 3D domains, including some with reentrant corners and edges.

The structure of this thesis is as follows. Chapter 2 recalls the fundamental concepts of Sobolev spaces, Neumann boundary value problems for Poisson's equation, and other tools from analysis that will be used in the derivation of error bounds for our moment-free quadrature formulas. Moreover, quadrature formulas and numerical differentiation formulas are introduced, with a focus on aspects such as stability, exactness, and consistency order. Following this, Chapter 3 focuses on the development and testing of a meshless advancing front method for node generation on 3D domains in B-Rep format, featuring a novel meshless inclusion test. Chapter 4 presents a new technique for generating high-order quadrature formulas that avoids moment computation, relying instead on numerical differentiation and the divergence theorem. Chapter 5 provides extensive numerical validation of the proposed moment-free quadrature methods, demonstrating their accuracy, stability, and convergence behavior. Lastly, in Chapter 6 we investigate the connection between positive numerical differentiation formulas on scattered data and positive moment-free quadrature formulas.

CHAPTER 2

Preliminaries

2.1 Elliptic boundary value problems

Some results in this thesis, in particular the error estimates for moment-free quadrature formulas, depend crucially on the existence of regular solutions to various boundary value problems. In the elliptic case, the well-posedness of Poisson's equation with Neumann boundary condition on a bounded domain in $\mathbb{R}^d$ or d-dimensional manifold with smooth boundary is a classical result which we recall in this section. We start by fixing notation and introducing concepts commonly used in the theory of partial differential equations (PDEs). For further details, we refer the reader to [33, 47, 110].

Bounded domains and regularity of the boundary

Let Ω be a bounded domain (open and connected set) in $\mathbb{R}^d$ or $\mathcal{M}$, a d-dimensional smooth Riemannian manifold, possibly with boundary. Since the boundary of an arbitrary domain is too rough for our purposes, common assumptions on $\partial\Omega$ are either Lipschitz regularity or C^k regularity. Informally, this means that the domain can be locally represented around each point $x \in \partial\Omega$ as the subgraph of a Lipschitz continuous function, or a k-times differentiable function, respectively.

A domain whose boundary has Lipschitz regularity is called a *Lipschitz domain*. A domains whose boundary has C^∞ regularity is called a *smooth domain*. Global C^k regularity of the boundary can be generalized to piecewise C^k regularity. The latter, unlike global C^k regularity, does not imply Lipschitz regularity (consider e.g. domains with cusps).

For any Lipschitz or piecewise C^1 domain $\Omega \subset \mathbb{R}^d$, let μ be the restriction of the standard Lebesgue measure to Ω, let σ be the standard hypersurface measure on $\partial\Omega$, and let $\nu\colon \partial\Omega \to \mathbb{R}^d$ be the outward-pointing unit normal field along the boundary. The measure σ and the normals ν are meaningfully defined with the help of the local parametrizations of the boundary. For a Lipschitz or piecewise C^1 domain $\Omega \subset \mathcal{M}$, measures μ and σ and normals ν are induced on Ω and $\partial\Omega$ by the ambient Riemannian metric of the manifold $\mathcal{M}$.

Spaces of functions with continuous derivatives

Let us momentarily restrict to the case $\Omega \subset \mathbb{R}^d$. For any integer $k \geq 0$ and for $k = \infty$ we define $C^k(\Omega)$ as the space of functions over Ω that are continuously differentiable up to order k. Partial derivatives of functions $f \in C^k(\Omega)$ are conveniently expressed using multi-index notation: $\partial^\alpha f$ with $\alpha = (\alpha_1, \ldots, \alpha_d)$ correspond to applying each i-th partial derivative operator ∂_{x_i} to f for α_i times. Overall, a total of

$|\alpha| = \alpha_1 + \cdots + \alpha_d$ partial derivatives are taken. The spaces

$$C^k(\overline{\Omega}) := \left\{ f \in C^k(\Omega) \mid \partial^\alpha f \text{ extends continuously to } \overline{\Omega} \text{ for all } |\alpha| \le k \right\}$$

are Banach spaces (complete normed vector spaces) with respect to the supremum norm

$$\|f\|_{C^k(\overline{\Omega})} := \sum_{|\alpha| \le k} \sup_{x \in \overline{\Omega}} |\partial^\alpha f(x)| .$$

Sometimes the order k is omitted, in which case the value $k = 0$ is implied.

The *Hölder spaces* $C^{k,r}(\overline{\Omega})$ with integer $k \ge 0$ and $r \in (0, 1]$ are the spaces of all functions $f \in C^k(\overline{\Omega})$ such that

$$\sup_{x,y \in \Omega,\, x \ne y} \frac{|\partial^\alpha f(x) - \partial^\alpha f(y)|}{\|x - y\|^r} < \infty \quad \text{for all } |\alpha| = k.$$

The Hölder spaces $C^{k,r}(\overline{\Omega})$ are Banach spaces with respect to the norm

$$\|f\|_{C^{k,r}} := \|f\|_{C^k} + \sup_{x,y \in \Omega,\, x \ne y} \frac{|\partial^\alpha f(x) - \partial^\alpha f(y)|}{\|x - y\|^r}.$$

For any vector field $F \in C^1(\overline{\Omega})^d$, the *divergence theorem* states that

$$\int_\Omega \operatorname{div} F(x)\, d\mu(x) = \int_{\partial\Omega} F(x) \cdot \nu(x)\, d\sigma(x). \tag{2.1}$$

The spaces $C^k(\Omega)$, $C^k(\overline{\Omega})$, $C^{k,r}(\overline{\Omega})$ can also be defined in the case $\Omega \subset \mathcal{M}$. Moreover, for a suitable generalization of the divergence operator, identity (2.1) also holds on smooth manifolds. The result is a corollary of Stokes' Theorem for differential forms, see [65] for more details.

Lebesgue and Sobolev spaces

The *Lebesgue space* $L^p(\Omega)$ with exponent $1 \le p < \infty$ consists of measurable functions $f \colon \Omega \to \mathbb{R}$ such that

$$\|f\|_p := \left(\int_\Omega |f(x)|^p \, d\mu(x) \right)^{1/p} < \infty,$$

and for $p = \infty$ consists of measurable functions whose essential supremum is bounded:

$$\|f\|_\infty := \operatorname*{ess\,sup}_{x \in \Omega} |f(x)| < \infty.$$

The p-norm is also relevant when working with finite-dimensional vectors $v \in \mathbb{R}^N$:

$$\|v\|_p := \left(\sum_{i=1}^N |v_i|^p \right)^{1/p} \quad \text{for } p \in [1, \infty), \qquad \|v\|_\infty := \max_{i=1,\dots,N} |v_i| . \tag{2.2}$$

For any pair of conjugated exponents $p, p' \in [1, \infty]$ such that $1/p + 1/p' = 1$, Hölder's inequality states that

$$\|fg\|_1 \le \|f\|_p \, \|g\|_{p'} \quad \text{for all } f \in L^p(\Omega) \text{ and all } g \in L^{p'}(\Omega). \tag{2.3}$$

The same inequality also holds for the discrete norms (2.2).

Historically, the classical spaces $C^k(\Omega)$ proved inadequate for the development of a satisfactory theory of existence and regularity for elliptic partial differential equations. To tackle this problem, *Sobolev spaces* with various orders and exponents have been introduced, and are today a fundamental tool in the modern treatment of partial differential equations.

Taking for granted the notion of *distributional derivative*, the *Sobolev space* $W_p^k(\Omega)$ with integer order $k \geq 0$ and exponent $1 \leq p \leq \infty$ on a domain Ω is the space of all distributions f on Ω such that

$$\partial^\alpha f \in L^p(\Omega) \quad \text{for all } |\alpha| \leq k,$$

and is a Banach space with respect to the *Sobolev norm*

$$\|f\|_{W_p^k} := \sum_{|\alpha| \leq k} \|\partial^\alpha f\|_p .$$

To properly state some error estimates in later chapters, we also need to define the *Sobolev seminorm*

$$|f|_{W_p^k} := \sum_{|\alpha| = k} \|\partial^\alpha f\|_p .$$

For some applications, Sobolev spaces of integer order are not sufficient, and intermediate spaces of fractional order have to be considered. Let $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$ be the floor function. The Sobolev–Slobodeckij space $W_p^s(\Omega)$ with fractional order $s \in \mathbb{R}^+ \setminus \mathbb{Z}$ and exponent $1 \leq p < \infty$ on a domain Ω is the space of all distributions f on Ω such that

$$f \in W_p^k(\Omega) \text{ with } k = \lfloor s \rfloor, \quad \sum_{|\alpha| = k} \int_{\Omega \times \Omega} \frac{|\partial^\alpha f(x) - \partial^\alpha f(y)|^p}{|x - y|^{d + (s-k)p}} \, d\mu(x) d\mu(y) < \infty.$$

Once again, setting $k = \lfloor s \rfloor$, this is a Banach space with respect to the norm

$$\|f\|_{W_p^s} := \|f\|_{W_p^k} + \sum_{|\alpha| = k} \int_{\Omega \times \Omega} \frac{|\partial^\alpha f(x) - \partial^\alpha f(y)|^p}{|x - y|^{d + (s-k)p}} \, d\mu(x) d\mu(y).$$

For $p = 2$, the Sobolev space $W_p^s(\Omega)$ is a Hilbert space and is denoted $H^s(\Omega)$. Standard constructions involving charts and partitions of unity allow Sobolev spaces of order $s \in \mathbb{R}^+$ to be defined also on smooth manifolds (possibly with boundary) and, as a special case, on the boundary of a smooth domain $\Omega \subset \mathbb{R}^d$. We conclude this overview of Sobolev spaces with three fundamental results: the Sobolev embedding theorem, the Stein extension theorem, and the trace theorem. For more information, see [47].

Proposition 2.1.1 (Sobolev embedding)**.** *Let Ω be a bounded Lipschitz domain in $\mathbb{R}^d$. The continuous embedding*

$$W_p^s(\Omega) \subseteq W_{p'}^t(\Omega)$$

holds for all $t \le s$, $p' \ge p$ such that $s - d/p = t - d/p'$, and the continuous embedding

$$W_p^s(\Omega) \subseteq C^{k,r}(\overline{\Omega})$$

holds for all $k < s - d/p < k + 1$ and $r = s - k - d/p$ such that k is a nonnegative integer.

Proposition 2.1.2 (Stein extension)**.** *Let Ω be a bounded Lipschitz domain in $\mathbb{R}^d$. For all orders $s \ge 0$ and exponents $1 \le p \le \infty$, there exists a continuous linear operator E_s from $W_p^s(\Omega)$ into $W_p^k(\mathbb{R}^d)$ such that*

$$(E_s f)_{|\Omega} = f \quad \text{for all } f \in W_p^s(\Omega).$$

Proposition 2.1.3 (Trace theorem)**.** *Let Ω be a bounded domain in $\mathbb{R}^d$ whose boundary has $C^{k,1}$ regularity. The outward-pointing unit normals ν can be extended to an L^∞ vector field in a neighborhood of $\overline{\Omega}$. Let ∂_ν be the directional derivative operator along the extended field ν, and let γ be the operator defined by*

$$\gamma f = f_{|\partial\Omega} \quad \text{for all } f \in C^0(\Omega).$$

Assume that $s - 1/p$ is not an integer, $s \le k + 1$, $s - 1/p = \ell + r$, $r \in (0, 1)$, ℓ is a nonnegative integer. Then the mapping

$$f \to \left\{\gamma f, \gamma\, \partial_\nu f, \ldots, \gamma\, \partial_\nu^\ell f\right\}$$

which is defined for $f \in C^{k,1}(\overline{\Omega})$, has a unique continuous extension as an operator from

$$W_p^s(\Omega) \text{ onto } \prod_{j=0}^{\ell} W_p^{s-j-1/p}(\partial\Omega).$$

This operator has a right continuous inverse which does not depend on p.

The operator γ is called *trace operator*, and its use is implied whenever a function in a Sobolev space is restricted to $\partial\Omega$, or its normal derivatives are considered. For example, the trace theorem allows us to rigorously state the divergence theorem for a vector field $F \in H^1(\Omega)^d$:

$$\int_\Omega \operatorname{div} F \, d\mu = \int_{\partial\Omega} \gamma F \cdot \nu \, d\sigma.$$

The scalar product between the normal field ν and the componentwise trace of F is sometimes denoted $\gamma_\nu F$, and called *normal trace* of F. As a corollary to the divergence theorem, we have

$$\int_\Omega \Delta f \, d\mu = \int_{\partial\Omega} \partial_\nu f \, d\sigma \quad \text{for all } f \in H^2(\Omega). \tag{2.4}$$

This identity also holds on smooth manifolds for a suitable generalization of the Laplace operator known as *Laplace-Beltrami* operator.

Poisson's equation with Neumann boundary condition

We now have all the necessary tools to properly state the following well-posedness result for Poisson's equation with Neumann boundary conditions. A proof can be found in [110].

Proposition 2.1.4 (Well-posedness of Neumann problem)**.** *Let Ω be a smooth bounded domain in $\mathbb{R}^d$ or in a smooth manifold $\mathcal{M}$. For all $f \in H^s(\Omega)$ and $g \in H^{s+1/2}(\partial\Omega)$ with $s \geq 0$, the Neumann boundary value problem*

$$\begin{cases} \Delta u = f & \text{in } \Omega \\ \partial_\nu u = g & \text{on } \partial\Omega \end{cases} \quad \text{with constraint} \quad \int_\Omega u \, d\mu = 0 \tag{2.5}$$

has a unique solution $u \in H^{s+2}(\Omega)$ that depends continuously on f and g, if and only if

$$\int_\Omega f \, d\mu = \int_{\partial\Omega} g \, d\sigma, \tag{2.6}$$

a constraint known as compatibility condition *for the Neumann boundary value problem.*

For domains with nonsmooth boundary, elliptic regularity results such as Proposition 2.1.4 do not hold anymore: no matter how smooth the right-hand side data f, g are, the regularity of u saturates at an order closely related to the global C^k regularity of the boundary.

The boundary value problem (4.29) has a unique solution $u \in H^{3/2}(\Omega)$ if and only if the compatibility condition (2.6) is satisfied, as implied by the results in [57]. However, standard examples in [47] for the corresponding Dirichlet problem on piecewise smooth domains $\Omega \subset \mathbb{R}^2$ with reentrant corners can be modified to apply to the Neumann problem, and generate for any $s > 3/2$ a solution $u \notin H^s(\Omega)$, even when f is infinitely differentiable and $g = 0$. This means that the regularity of u is not only limited by the regularity of f and g, but also by the regularity of $\partial\Omega$.

The following example, adapted from [47], shows that this may happen even on elementary geometries, such as a square.

Example 2.1.5. Let $\Omega = (0,1)^2$, and let $\Gamma_1, \dots, \Gamma_4$ be the left, bottom, right, and top sides of $\partial\Omega$, respectively. For all $i = 1, \dots, 4$ and $s \geq 0$, let γ_i be the trace operator from $H^{s+1/2}(\Omega)$ to $H^s(\Gamma_i)$. Functions in $H^s(\Gamma_i)$ are parametrized by the restrictions of Cartesian coordinates (x, y) to $\partial\Omega$, that is, $x \in [0,1]$ for $i = 2, 4$ and $y \in [0,1]$ for $i = 1, 3$. The following function, defined using polar coordinates (r, θ) centered at the origin, is harmonic in Ω:

$$v(r,\theta) = r^2\big(\log(r)\cos(2\theta) - \theta\sin(2\theta)\big).$$

Let $\varphi\colon [0,1] \to \mathbb{R}$ be a smooth function such that $\varphi(r) \equiv 1$ for all $r \in [0, \frac{1}{3}]$, and $\varphi(r) \equiv 0$ for all $r \in [\frac{2}{3}, 1]$. Then, the function $u(r,\theta) := \varphi(r)\, v(r,\theta)$ is the unique solution (up to a constant) of

$$\begin{cases} \Delta u = \Delta(\varphi v) & \text{in } \Omega \\ \partial_\nu u = 0 & \text{on } \partial\Omega \setminus \Gamma_1 \\ \partial_\nu u = \gamma_1(-\partial_x u) = \pi y \varphi(y) & \text{on } \Gamma_1. \end{cases}$$

The right hand side functions f, g of this Neumann problem are smooth in the sense that they are restrictions to Ω and $\partial\Omega$ of suitable functions that belong to $C^\infty(\mathbb{R}^2)$. Nevertheless, it is clear that $u \notin H^3(\Omega)$, because u and v coincide for $r < 1/3$ and

$$\frac{\partial^3 v(x,y)}{\partial x^3} = \frac{2x}{x^2+y^2} = \frac{2\cos(\theta)}{r} \notin L^2(\Omega).$$

2.2 Quadrature formulas

Numerical integration is the branch of numerical analysis that deals with methods for efficiently and accurately approximating the integral of functions over an integration domain. When the domain is a real interval, the problem of numerical integration is equivalent to computing the (signed) area under the graph of a function, and so for historical reasons it is commonly referred to as the *problem of quadrature*. Some contemporary authors use the term *quadrature* to refer to numerical integration problems in any number of dimensions, while others prefer the term *cubature* for dimensions above one. In this work, we stick to the term *quadrature*, irrespective of the dimension of the integration domain.

The modern theory of integration has been formalized between the 19th and 20th century in the frameworks of real analysis and measure theory, particularly through the developments of mathematicians such as Borel, Baire, and Lebesgue. To properly introduce numerical integration in its most general setting, we make use of basic definitions from measure theory (see [109] for a reference); in later sections, however, we drop the abstractions and focus on the most common settings that arise in practical applications.

Let Ω be an arbitrary set, playing the role of integration domain, let $\mathcal{A}$ be a sigma algebra of measurable sets on Ω, and let μ be a finite measure on Ω, so that the triple $(\Omega, \mathcal{A}, \mu)$ forms a finite measure space. For any measurable function $f\colon \Omega \to \mathbb{R}$, we denote the Lebesgue integral of f over Ω as either

$$\int_\Omega f \, d\mu \quad \text{or} \quad \int_\Omega f(x) \, d\mu(x),$$

depending on whether the integration variable x needs to be stated explicitly or not.

A numerical integration formula, also called a *quadrature formula* or *quadrature rule*, is a combination of a finite set of points $X = \{x_i\}_{i=1}^{N_X} \subset \Omega$ called *quadrature nodes* and a corresponding vector of real numbers $w \in \mathbb{R}^{N_X}$ called *quadrature weights*. The pair (X, w) is designed to approximate the integral of functions f by a weighted sum of pointwise values of f at the nodes in X:

$$\int_\Omega f \, d\mu \approx \sum_{i=1}^{N_X} w_i f(x_i).$$

Given a quadrature formula, its *absolute quadrature error* for an integrand f is

$$e_{abs}(\Omega, \mu, X, w, f) := \left| \int_\Omega f \, d\mu - \sum_{i=1}^{N_X} w_i f(x_i) \right|.$$

The ultimate goal in numerical integration is to minimize this error, but nodes in X and functions f may or may not be fixed in advance, and so three different settings occur in practice.

In the first setting, the function f is fixed, and nodes in X can be chosen along with quadrature weights w to minimize e_{abs}: we seek an approximation of

$$\arg\min_{(X,w)} \; e_{abs}(\Omega, \mu, X, w, f),$$

which can (and should) make use of prior knowledge of the integrand f. This setting typically arises when special integrals are precomputed as part of a larger numerical scheme, or when the value of the integral is an interesting constant in itself.

In the second setting, f can be any element of a given normed functional space $\mathcal{F}$, and the quality of a quadrature formula (X, w) is assessed over all of $\mathcal{F}$: we seek an approximation of

$$\arg\min_{(X,w)} \sup_{f \in \mathcal{F},\, \|f\|=1} e_{abs}(\Omega, \mu, X, w, f).$$

In this setting, the emphasis is usually on the optimal placement of quadrature nodes: it is a well-known fact that, for example, on a bounded non-periodic domain Ω, a uniform distribution of nodes can be sub-optimal, and a denser distribution near the boundary should instead be chosen for smooth functional spaces such as $\mathcal{F} = C^k(\Omega)$.

Finally, in the third setting, the node set X is fixed in advance, and so the only control we have over the worst-case quadrature error e_{abs} is through the choice of weights w: we seek an approximation of

$$\arg\min_{w} \sup_{f \in \mathcal{F},\, \|f\|=1} e_{abs}(\Omega, \mu, X, w, f).$$

In this setting, the emphasis is usually on a robust and stable choice of weights that can compensate for a possibly sub-optimal (or even pathological) placement of quadrature nodes. When nodes in X are not arranged in any regular or geometrically meaningful way, such as on a grid or a mesh, we speak of *scattered nodes*, and (X, w) is called a *meshless quadrature formula*.

Each of the three settings requires its own ideas and algorithms, although some of the most far-reaching techniques in numerical integration are relevant to all of them. In this thesis we mostly focus on the third setting, which is also the most generally applicable.

For a comprehensive introduction to the theory of numerical integration, we refer the reader to [17, 62, 106]. Here we shall only recall some elementary definitions and results that will be relevant in the following chapters.

Consistency and stability of quadrature formulas

As the number of quadrature nodes increases, we expect the absolute quadrature error to decrease, and in the limit $N_X \to \infty$ to eventually converge to zero. For this limit to make sense, however, we need to work not just with a single quadrature formula, but rather with a whole family of formulas.

Definition 2.2.1 (Quadrature scheme)**.** A *quadrature scheme*, or *quadrature process*, is a family of quadrature formulas (X_h, w_h) indexed by the continuous variable $h \in (0, \delta)$, for some $\delta > 0$. A quadrature scheme is said to be *convergent* over a normed functional space $\mathcal{F}$ if, for all $f \in \mathcal{F}$,

$$\lim_{h \to 0^+} e_{abs}(\Omega, \mu, X_h, w_h, f) = 0,$$

and to be *convergent with order* $k > 0$ if, for all $f \in \mathcal{F}$,

$$e_{abs}(\Omega, \mu, X_h, w_h, f) = O(h^k).$$

If a single constant $c(\Omega, \mu, \mathcal{F}) > 0$ can be chosen so that

$$e_{abs}(\Omega, \mu, X_h, w_h, f) \leq c(\Omega, \mu, \mathcal{F}) \, \|f\| \, h^k$$

for all $h \in (0, \delta)$ and $f \in \mathcal{F}$, the scheme is said to be *uniformly convergent with order* k.

As typically happens in numerical analysis (consider e.g. the Lax–Richtmyer theorem), sufficient conditions for convergence can be given in terms of consistency and stability properties. For a quadrature scheme, consistency means that quadrature errors are zero over a set of functions $\mathcal{S}$ that can accurately approximate any function in $\mathcal{F}$, whereas stability means that the discrete 1-norm of w_h remains bounded as $h \to 0$. The next definitions introduce the concepts of consistency and stability in a fully rigorous way.

Definition 2.2.2 (Exactness of a quadrature scheme)**.** A quadrature formula (X, w) is *exact* for a functional space $\mathcal{S}$ if

$$e_{abs}(\Omega, \mu, X, w, s) = 0 \quad \text{for all } s \in \mathcal{S}.$$

Likewise, a quadrature scheme (X_h, w_h) is *exact* for a family of functional spaces $\{\mathcal{S}_h\}_{h \in (0,\delta)}$ if each quadrature formula (X_h, w_h) is exact for the corresponding space $\mathcal{S}_h$.

Definition 2.2.3 (Approximation power of a space)**.** Let $\mathcal{F}$ be a normed subspace of $\mathbb{R}^\Omega$, the vector space of all real functions defined on Ω. A family $\mathcal{S}_\delta = \{\mathcal{S}_h\}_{h \in (0,\delta)}$ of subspaces of $\mathbb{R}^\Omega$ has *pointwise approximation power* $k > 0$ if, for all $f \in \mathcal{F}$,

$$\inf_{s \in \mathcal{S}_h} \sup_{x \in \Omega} |f(x) - s(x)| = O(h^k).$$

If a single constant $a(\Omega, \mathcal{F}, \mathcal{S}_\delta) > 0$ can be chosen so that

$$\inf_{s \in \mathcal{S}_h} \sup_{x \in \Omega} |f(x) - s(x)| \leq a(\Omega, \mathcal{F}, \mathcal{S}_\delta) \, \|f\| \, h^k$$

for all $h \in (0, \delta)$ and $f \in \mathcal{F}$, the family $\mathcal{S}_\delta$ is said to have *uniform approximation power* $k > 0$.

Definition 2.2.4 (Consistency of a quadrature scheme)**.** A quadrature scheme (X_h, w_h) has *consistency order* $k > 0$ over a normed functional space $\mathcal{F}$ if it is exact for a family of spaces $\{\mathcal{S}_h\}_{h\in(0,\delta)}$ with pointwise approximation power k. The concept of *uniform consistency order* is similarly defined in terms of uniform approximation power.

Definition 2.2.5 (Stability of a quadrature scheme)**.** A quadrature scheme (X_h, w_h) is *stable* if there exists a constant $\kappa > 0$ such that

$$\|w_h\|_1 := \sum_{i=1}^{|X_h|} |w_{h,i}| \leq \kappa$$

for all $h \in (0, \delta)$.

We can now prove that a consistent and stable quadrature scheme is convergent, and that uniform consistency implies uniform convergence. This result, although easy to prove, is among the most important in the theory of numerical integration, because exactness is by far the most popular and well-understood design principle for quadrature formulas.

Proposition 2.2.6. *A stable quadrature scheme (X_h, w_h) that has consistency order $k > 0$ over a normed functional space $\mathcal{F}$ is convergent with order k over the same space $\mathcal{F}$. If consistency is uniform, then convergence is also uniform.*

Proof. Let $\mathcal{S}_\delta = \{\mathcal{S}_h\}_{h\in(0,\delta)}$ be the family of spaces with pointwise approximation power k for which the quadrature scheme is exact. Then, by the definition of exactness and by the triangle inequality,

$$\begin{aligned}
\left| \int_\Omega f \, d\mu - \sum_{i=1}^{|X_h|} w_{h,i} f(x_{h,i}) \right| &= \inf_{s\in\mathcal{S}_h} \left| \int_\Omega f \, d\mu - \sum_{i=1}^{|X_h|} w_{h,i} f(x_{h,i}) - \int_\Omega s \, d\mu + \sum_{i=1}^{|X_h|} w_{h,i} s(x_{h,i}) \right| \\
&\leq \inf_{s\in\mathcal{S}_h} \left| \int_\Omega f - s \, d\mu - \sum_{i=1}^{|X_h|} w_{h,i} \big(f(x_{h,i}) - s(x_{h,i}) \big) \right| \\
&\leq \inf_{s\in\mathcal{S}_h} \left\{ \int_\Omega 1 \, d\mu \cdot \sup_{x\in\Omega} |f(x) - s(x)| + \|w_h\|_1 \sup_{x\in\Omega} |f(x) - s(x)| \right\} \\
&\leq (\mu(\Omega) + \kappa) \, O(h^k) = O(h^k).
\end{aligned}$$

The same chain of inequalities proves that, under the assumption of uniform consistency, the definition of uniform convergence is satisfied by the choice

$$c(\Omega, \mu, \mathcal{F}) = (\mu(\Omega) + \kappa) \, a(\Omega, \mathcal{F}, \mathcal{S}_\delta). \qquad \square$$

2.3 Numerical differentiation formulas

A *numerical differentiation formula* for a linear differential operator $\mathcal{L}\colon \mathcal{D} \to \mathbb{R}$ is an approximation of the values of $\mathcal{L}u$ at certain points, based on the values of the

function u at a finite set of surrounding points. More formally, let $\mathcal{D}$ be a subset of $C^0(\overline{D})$, the space of functions over a domain D that are continuous up to its boundary, and let

$$X = \{x_i\}_{i=1}^{N_X}, \quad Y = \{y_i\}_{i=1}^{N_Y}$$

be two finite sets of points in $\overline{D}$, also called *nodes* in this context. A numerical differentiation formula for $\mathcal{L}$ based on X, Y is a collection of weights ℓ_{ij} such that

$$\mathcal{L}u(y_i) \approx \sum_{j=1}^{N_X} \ell_{ij} u(x_j), \quad i = 1, \ldots, N_Y, \tag{2.7}$$

often arranged into a *differentiation matrix* $L = (\ell_{ij}) \in \mathbb{R}^{N_Y \times N_X}$. For all $i = 1, \ldots, N_Y$, the set $X_i = \{x_j \in X \mid \ell_{ij} \neq 0\}$ is called *set of influence* of point y_i, and the set of indices $\{j \mid \ell_{ij} \neq 0\}$ is denoted S_i. To avoid ambiguity when multiple differential operators are involved, these sets may also be denoted $X_{L,i}$ and $S_{L,i}$.

Numerical differentiation formulas (2.7) can be used to discretize the strong form of a partial differential equation, along with its boundary conditions, in a way that generalizes the classical finite difference method from regular grids to arbitrary points X and Y. On scattered nodes, this is an example of a *meshless* method for the numerical solution of a partial differential equation. Given the local character of differentiation operators, the sets of influence S_i in generalized finite difference methods are kept small, relying on refinement of X controlled by a parameter $h > 0$ to increase the accuracy of the approximation.

The standard way to obtain accurate numerical differentiation weights ℓ_{ij} is to impose exactness of (2.7) over a local basis of functions that provides a good approximation of $u \in \mathcal{D}$ in a neighborhood of y_i. When D is a domain in $\mathbb{R}^d$, a natural local approximation tool is given by the spaces Π_q^d of multivariate polynomials of total degree $< q$.

Definition 2.3.1. Let $\mathcal{L}$ be a differential operator of order k. A numerical differentiation formula (2.7) for $\mathcal{L}$ based on $X, Y \subset \mathbb{R}^d$ is said to be *polynomially exact* of order $q \geq 1$ if

$$\mathcal{L}p(y_i) = \sum_{j=1}^{N_X} \ell_{ij} p(x_j) \qquad \text{for all } p \in \Pi_q^d \text{ and } i = 1, \ldots, N_Y, \tag{2.8}$$

and *polynomially consistent* of order $n \geq 1$ if it is exact of order $q = n + k$.

The sets of influence X_i are usually chosen somewhat larger than the minimum needed to admit a numerical differentiation formula exact for all $u \in \Pi_q^d$, and the extra degrees of freedom are used either to minimize a (semi-)norm of the weight vectors $(\ell_{ij})_{j \in S_i}$, or to enhance the local approximation space by adding radial basis functions. We refer to [26, 27, 21] for the theory and error bounds for these methods.

Polynomial type methods are also available on certain manifolds, such as the d-dimensional sphere, where spherical harmonics may be employed. Otherwise, numerical differentiation formulas on arbitrary manifolds may be generated with the help of positive definite kernels. Suitable local error bounds for kernel-based

numerical differentiation may be found in [25] for domains in $\mathbb{R}^d$ and in [22, Section 4.4] for reproducing kernels of Sobolev spaces on manifolds.

We now provide additional information on the kinds of numerical differentiation formulas that are used in this thesis.

Formulas based on exactness for polyharmonic splines

This family of formulas is generated by the *polyharmonic radial basis kernel*

$$K(x,y) = \|x-y\|^{2m-1}, \quad x,y \in \mathbb{R}^d,$$

with a polynomial augmentation term in Π_q^d, $q \geq m$, which is a standard approach in RBF-FD since [6]. We use $m = q$. Thus, the weights ℓ_{ij} are obtained by requiring the exactness of (2.7) for all functions of the form

$$u(x) = \sum_{j \in S_i} c_j \|x - x_j\|^{2q-1} + \tilde{p}(x), \quad c_j \in \mathbb{R},\ \tilde{p} \in \Pi_q^d,$$

with

$$\sum_{j \in S_i} c_j p(x_j) = 0 \text{ for all } p \in \Pi_q^d.$$

In most cases the square linear system that arises from these conditions is regular. However, as shown in [21], numerical differentiation weights are uniquely determined and can be computed by a null space method also for certain *deficient* sets of influence, such as the five point stencil for the Laplacian on gridded nodes.

Under appropriate assumptions on the sets of influence X_i, such as quasi-uniformity and boundedness of the polynomial Lebesgue constants, the errors of polyharmonic numerical differentiation can be estimated as

$$\max_{1 \leq i \leq N_Y} \left| \mathcal{L}u(y_i) - \sum_{j \in S_i} \ell_{ij} u(x_j) \right| \leq C h^{q-k} \, |u|_{W^q_\infty(D)},$$

where variable h is the maximum diameter of the sets $\{y_i\} \cup X_i$ for $i = 1, \ldots, N_Y$, and variable C is a positive constant independent of h and u.

Positive numerical differentiation formulas

Suppose that $Y \subset X$, and that the sets X, Y are ordered so that $x_i = y_i$ for $i = 1, \ldots, N_Y$. A numerical differentiation formula is said to be *positive* if

$$\ell_{ii} > 0 \text{ for all } i = 1, \ldots, N_Y \quad \text{and} \quad \ell_{ij} \leq 0 \text{ for all } j \in S_i \setminus \{i\},$$

in which case the differentiation matrix L is also said to have *positive type*. Although some authors define positive formulas with reversed inequalities in the case of elliptic differential operators [91, 26], this definition ensures that positive-definite operators such as $\mathcal{L} = -\Delta$ can be approximated with positive formulas, such as the classical five-point stencil in 2D.

The numerical solution of a boundary value problem

$$\begin{cases} \mathcal{L}u = f & \text{in } D, \\ \mathcal{B}u = g & \text{on } \partial D \end{cases} \tag{2.9}$$

using a meshless finite difference method typically proceeds as follows. First, two disjoint sets of nodes Y and Z are placed in the interior and on the boundary of D, respectively, and X is defined as their union. Then, numerical differentiation formulas for $\mathcal{L}$ based on X, Y and for $\mathcal{B}$ based on X, Z are computed, leading to differentiation matrices

$$L = (\ell_{ij}) \in \mathbb{R}^{N_Y \times N_X} \quad \text{and} \quad B = (b_{ij}) \in \mathbb{R}^{N_Z \times N_X}.$$

Let $A \in \mathbb{R}^{N_X \times N_X}$ be the square matrix obtained by vertically stacking L and B. An approximation of u in (2.9) is found by solving the linear system with matrix A and right-hand side obtained by vertically stacking $f_{|Y}$ and $g_{|Z}$. If L and B have positive type, then A has the L-matrix property.

Definition 2.3.2 (L-matrix)**.** A square matrix $A \in \mathbb{R}^{N \times N}$ is an *L-matrix* if $a_{ii} > 0$ for all $i = 1, \ldots, N$, and $a_{ij} \leq 0$ otherwise.

The L-matrix property is important in the theory of numerical schemes of finite difference type, because it is often a precursor to the M-matrix property, which guarantees that A is not singular, that all coefficient of A^{-1} are nonnegative, and that a counterpart of the maximum principle for elliptic operators holds in the discrete setting [16].

Definition 2.3.3 (M-matrix)**.** A square matrix $A \in \mathbb{R}^{N \times N}$ is an *M-matrix* if it can be written in the form $sI - P$, with P being a square matrix with nonnegative coefficients, I being the identity matrix, and $s \in \mathbb{R}^+$ being strictly larger than the spectral radius of P.

A sufficient condition for an L-matrix to be an M-matrix is that all of its rows are *strictly diagonally dominant*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad i = 1, \ldots, N,$$

but this property is quite strong, and so it is rarely satisfied in practice. For example, when the i-th row of A is given by a positive differentiation formula for $\mathcal{L} = -\Delta$ that is exact for constants (polynomial exactness of order $q \geq 1$), the elements of the i-th row must sum to zero, and so

$$|a_{ii}| = a_{ii} = \sum_{j \neq i} -a_{ij} = \sum_{j \neq i} |a_{ij}|.$$

The row is said to be only *weakly diagonally dominant*, in the sense that

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|.$$

If all rows of an L-matrix are weakly diagonally dominant and at least one row is strictly diagonally dominant, sufficient conditions for A to be an M-matrix can be given in terms of the nonzero pattern of A. A proof can be found in [50].

Definition 2.3.4. The *matrix graph* of a square matrix $A \in \mathbb{R}^{N\times N}$ is the directed graph with vertices $V = \{1, \dots, N\}$ and edges $E = \{i \to j \mid a_{ij} \neq 0\}$. Given two vertices $i, j \in V$, we say that i is *connected* to j if there exists a path in the graph that starts at i and ends at j, in which case we also say that the i-th row of A is connected to the j-th row.

Proposition 2.3.5. *Let $A \in \mathbb{R}^{N\times N}$ be a square matrix such that*

1. *A is an L-matrix.*
2. *Every row of A is weakly diagonally dominant.*
3. *Every row of A is connected to at least one strictly diagonally dominant row.*

Then A is an M-matrix.

Consider, as an illustrative example, the case where $\mathcal{L} = -\Delta$ and $\mathcal{B} = I$, for which the boundary value problem (2.9) corresponds to Poisson's equation with Dirichlet boundary conditions. If $\mathcal{L}$ and $\mathcal{B}$ are discretized by a differentiation matrix $L \in \mathbb{R}^{N_Y\times N_X}$ of positive type and a block matrix

$$B = \begin{pmatrix} 0_{N_Z\times N_Y} & I_{N_Z\times N_Z} \end{pmatrix} \in \mathbb{R}^{N_Z\times N_X},$$

with I being the identity matrix, the square matrix A obtained by vertical concatenation of L and B is an L-matrix whose rows are all weakly diagonally dominant. If the stencils $S_{L,i}$ are chosen so that every internal node is connected to a boundary node in the sense of Definition 2.3.4, a condition which naturally occurs in every reasonable finite difference scheme, then Proposition 2.3.5 guarantees that A is an M-matrix.

The same reasoning can be applied to the setting of *mixed* boundary conditions, where the boundary of D is partitioned into Γ_D and Γ_N, and Dirichlet boundary conditions are imposed on Γ_D, whereas Neumann boundary conditions are imposed on Γ_N. In this case, the differentiation matrix B_N for the normal derivative operator ∂_ν must be of positive type, too. Assuming that the differentiation formulas in B_N are exact for constants, all rows of B_N must sum to zero, and so cannot be strongly diagonally dominant. To apply Proposition 2.3.5, therefore, not only every node in the interior of D must be connected to a node on the boundary Γ_D, but also every node on the boundary Γ_N.

If $\Gamma_D = \emptyset$, A is not an M-matrix anymore, but this is to be expected, because the pure Neumann problem for Poisson's equation does not have a unique solution, see Proposition 2.1.4. The solution is only determined up to a constant, and one way to fix it numerically is to enforce the discrete solution at the i-th node x_i to be equal to a constant $c \in \mathbb{R}$ by replacing the i-th row of A with the unit vector e_i with 1 on the diagonal, and the i-th element of the right-hand side $(f_{|Y}^T, g_{|Z}^T)^T$ with c. The following proposition shows that this approach leads once again to an M-matrix.

Proposition 2.3.6. *Let $A \in \mathbb{R}^{N\times N}$ be a square matrix such that*

1. *A is an L-matrix.*
2. *Every row of A is weakly diagonally dominant.*

3. Every two rows of A are connected.

Then, for all $i = 1, \ldots, N$, the matrix $\tilde{A}_i$ obtained by replacing the i-th row of A with e_i is an M-matrix.

CHAPTER 3

Meshless domain discretizations

Despite great advances in mesh generation algorithms, one of the most challenging and time consuming parts of mesh-based numerical simulation on geometrically complex 3D domains remains to this day the generation and management of meshes [72]. An increasingly popular alternative approach to numerical simulation that avoids the issue of mesh generation entirely is that of *meshless* (or *meshfree*) methods, in which approximations of unknown functions are determined exclusively from pointwise values at a scattered set of nodes. The step of mesh generation is replaced by the generation of a point cloud, and this provides strong motivation for studying robust and efficient ways of generating scattered nodes with prescribed density over arbitrary domains in $\mathbb{R}^3$ [107].

Although the topic of point cloud generation has been an active area of research for more than 20 years [74], interest has recently grown, as demonstrated by several different algorithms being published in the last few years, along with corresponding open source implementations [54, 80, 96, 113]. All four algorithms can handle 3D geometries and are based on either rejection sampling of nodes from an ambient Cartesian grid, or on advancing front methods (see [107] for a detailed description of the two approaches).

Traditional Computer-Aided Engineering workflows rely on simulations on meshes generated from CAD geometries, and the standard format in CAD is boundary representation (B-Rep). From a mathematical point of view, a B-Rep is the description of the boundary of a 3D domain as a finite union of parametric surfaces called *patches*, along with connectivity information and trimming curves in the parametric domain of each patch. Trimming curves are closed simple curves that delimit regions in parameter space whose image in model space is excluded from the patch, and, hence, from the boundary of the 3D domain. Trimming is treated by all CAD kernels as one of the most fundamental operations that allows the construction of complex geometries, although its use comes at a price, such as the introduction of small gaps and overlaps between patches caused by the inevitable approximation of trimming curves [76].

To this day, very few studies have been published on the subject of node generation for domains in B-rep form, see for example [32, 54], and mature software for this task is limited to commercial packages for meshless simulations such as [38, 83]. On the contrary, the problem of mesh generation from CAD geometries has been studied extensively, and well-established open source software exists for this task, such as `gmsh` [39]. This is the reason why, even today, a large share of meshless literature paradoxically uses the vertices of a mesh as the point cloud for meshless methods, and this can lead to the misconception that node generation is as computationally intensive and algorithmically complex as mesh generation, even though meshless advancing front algorithms can be one order of magnitude faster than comparable

advancing front mesh generation algorithms [74].

In this chapter, we contribute to ongoing efforts to extend existing node generation algorithms to 3D domains in B-rep form by describing a variable density advancing front method specifically tailored to such domains. Although similar ideas were recently presented in [32], we developed our approach independently and became aware of that work only in the later stages of our research. Our method, much like the one in [32], overcomes two key issues.

First, nodes must be placed on the boundary of a 3D domain taking trimming into account. Rather than generating nodes over the whole parametric domain of each patch and discarding nodes in the regions excluded by trimming curves, we generate nodes directly in the included regions with another advancing front algorithm (this time in 2D), where equispaced nodes on trimming curves play the role of boundary nodes. The image of interior nodes produced by the 2D advancing front algorithm on each patch will then define the set of boundary nodes for the 3D advancing front algorithm.

Second, generation of interior nodes in an advancing front method requires an inclusion test to check whether new candidates are inside the domain during the expansion of the front. Although inclusion tests on B-rep geometries can be answered exactly (up to machine precision) using ray casting [34] or generalized winding number algorithms [105], such methods are impractical in this setting because of their high computational cost (although it could be reduced, see [103] for the case of 2D domains). Instead, we propose a fast and completely *meshless* inclusion test that only takes as inputs a set of nodes on the boundary of the domain (which, as explained, can itself be generated by an advancing front method), and corresponding normal vectors. Our algorithm is a generalization of the simple meshless inclusion test based on nearest-neighbor search already in use by several authors [32, 81, 93, 95], which is however unreliable in the case of non-smooth boundaries, and can lead to the front being incorrectly advanced in the exterior of the domain during node generation. In [32], the authors propose a solution to this issue based on supersampling of boundary nodes. In this work, we do not increase the density of boundary nodes; instead, we rely on a more robust meshless inclusion test based on K nearest neighbors, with $K > 1$. Besides its applications to node generation, this test is a meshless algorithm of independent interest, and its robustness has been checked on several 2D and 3D domains.

3.1 Inclusion tests

Every algorithm for node generation on a bounded domain $\Omega \subset \mathbb{R}^d$ with nontrivial boundary needs to run *inclusion tests* on arbitrary points $y \in \mathbb{R}^d$, that is, it needs a fast and reliable way to detect whether $y \in \Omega$. If Ω is described as the strict sublevel set of a continuous function $\phi\colon \mathbb{R}^d \to \mathbb{R}$, then it is enough to evaluate $\phi(y)$ to know if y is inside the domain. If Ω is given instead in B-rep format, the standard representation in Computer-Aided Design, inclusion tests require drastically more complex algorithms.

For some applications, inclusion has to be checked *exactly*, usually up to rounding errors introduced by floating point arithmetic. The most popular approach in this

case is based on *ray-casting*, and it works by choosing one half-line in $\mathbb{R}^d$ with initial point y and counting the number of its intersections with $\partial\Omega$. If the half-line is tangent to $\partial\Omega$ at any point of intersection, or if a normal vector is not well-defined on $\partial\Omega$ at those points, the test is inconclusive, and a different ray must be considered. Otherwise, unless y itself belongs to $\partial\Omega$, an odd number of intersections implies that $y \in \Omega$, whereas an even number of intersections implies that $y \notin \overline{\Omega}$. In the case of bicubic Bezier patches, each patch can intersect a given ray at up to 18 different points [58]. This means that, for each patch whose bounding box has non-empty intersection with the ray, an expensive root-finding algorithm has to be used to compute the complete set of intersections. Moreover, ray casting has to take trimming curves into account, see [34] for more details.

For applications where checking *approximate* inclusion is sufficient, much faster tests can be devised. For example, a triangulation of $\partial\Omega$ with maximum diameter $h > 0$ can be used instead of the exact boundary during ray-casting, which dramatically simplifies the inclusion test and reduces its computational cost, especially if a bounding volume hierarchy is used to skip unnecessary ray-triangle intersections. For example, the state-of-the-art computational geometry library CGAL uses Axis-Aligned Bounding Box trees (AABB-trees) for this purpose [1]. One way to asymptotically characterize the accuracy of an approximate inclusion test is through the following definition.

Definition 3.1.1. Let h be a generic discretization parameter for Ω and its boundary. We say that an approximate inclusion test has *order* $k \geq 0$ if there exists a constant $c \in \mathbb{R}$ depending only on the shape of Ω such that any point with distance from the boundary greater than ch^k is classified correctly as being either inside or outside of Ω.

Under the right assumptions on the boundary triangulation, ray casting on a piecewise linear approximation of Ω is an approximate inclusion test of order 2. Unlike mesh generation, where a low order approximation of the boundary places a fundamental limit to the maximum order of convergence attainable with e.g. finite element methods, the task of node generation can be carried out with approximate inclusion tests of low order, such as two, without necessarily saturating the convergence order of meshless numerical schemes (see for example Figure 3.12 and the convergence tests in Section 5.2).

3.1.1 The INN1 inclusion test

Ray casting using a triangulation of $\partial\Omega$ is a practical choice for the kind of inclusion tests required by advancing front algorithms for node generation, as first shown in [74]. In recent years, however, several authors working on node generation have moved away from mesh-based methods in favor of the meshless inclusion test described in Algorithm 3.1. Instead of a triangulation, the algorithm takes as input a set of scattered nodes

$$Z = \{z_i\}_{i=1}^{N_Z} \subset \partial\Omega,$$

along with corresponding unit vectors ν_i normal to the boundary and pointing outwards with respect to Ω. In this context, the discretization parameter h can be

taken as the supremum of the diameter of geodesic disks on $\partial\Omega$ that are disjoint from Z.

Algorithm 3.1 – Approximate inclusion test INN1

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in Ω.
Input: Set of boundary nodes $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.

Output: Boolean result of the inclusion test.

1: Find the index i_1 of the nearest neighbor in Z to the query point y.
2: **if** $(z_{i_1} - y) \cdot \nu_{i_1} > 0$ **then**
3: **return** true
4: **else**
5: **return** false
6: **end if**

The intuition behind Algorithm 3.1 is that, under the assumption of smooth boundary, any point $z^* \in \partial\Omega$ that minimizes the distance to y satisfies

$$(z^* - y) \cdot \nu(z^*) = \begin{cases} 0 & \text{if } y \in \partial\Omega, \\ +\|z^* - y\| & \text{if } y \in \Omega, \\ -\|z^* - y\| & \text{if } y \notin \overline{\Omega}, \end{cases}$$

which means that the vector $z^* - y$ is parallel to $\nu(z^*)$, and has the same orientation if and only if $y \in \Omega$. Therefore, when we approximate z^* with $z_{i_1} \in Z$, we can expect that

$$\operatorname{sign}\big((z_{i_1} - y) \cdot \nu_{i_1}\big) = \operatorname{sign}\big((z^* - y) \cdot \nu(z^*)\big),$$

as long as y is sufficiently distant from the boundary and the discretization parameter h is sufficiently small. Algorithm 3.1 effectively works by reconstructing an approximate signed distance field to the boundary

$$\phi_Z(y) := \operatorname{sign}\big((z_{i_1} - y) \cdot \nu_{i_1}\big) \, \|z_{i_1} - y\| \,,$$

just like similar algorithms for closed triangle meshes [5].

Although Algorithm 3.1 is used by several authors working on the topic of node generation, see e.g. [32, 81, 93, 95], it has not been given a name yet, and its order has not been analyzed. We call it INN1 based on the fact that it is an Inclusion test using one Nearest Neighbor (later generalized to INNK using K nearest neighbors). We conjecture that INN1 is an approximate inclusion test of order 2 on smooth domains: to see why, consider the Voronoi diagram in $\mathbb{R}^d$ built from the set Z, and let $V_1, \ldots, V_{N_Z}$ be its cells. Moreover, for each $z_i \in Z$, let H_i be the half-space

$$H_i = \big\{ y \in \mathbb{R}^d \mid (z_i - y) \cdot \nu_i \geq 0 \big\} \,.$$

Then, the set of points for which Algorithm 3.1 returns true can be characterized as the interior of the polytope

$$\Omega_{\text{NN}} = \bigcup_{i=1}^{N_Z} V_i \cap H_i,$$

and so INN1 is equivalent to ray casting using $\partial\Omega_{\text{NN}}$ as a piecewise linear approximation of $\partial\Omega$. Since $\partial\Omega_{\text{NN}}$ is tangent to $\partial\Omega$ at every point in Z, we expect the piecewise linear approximation to have order two.

There are several reasons why, assuming comparable accuracy for the same discretization parameter h, the test INN1 is more suitable for the task of node generation compared to ray casting using a triangulation of the boundary:

- The test INN1 allows the definition of a dimension-independent meshless node generation algorithm, which can be run in dimension $d-1$ to place nodes Z on each boundary patch of Ω, and then in dimension d to place nodes inside the same domain. This is possible because the dependency on mesh generation in dimension $d-1$ has been removed: a purely meshless approach makes it possible to reuse the same code and data structures for all kinds of node generation tasks, which greatly benefits simplicity and performance.

- The test INN1 is faster than ray casting. If a space-partitioning data structure such as a k-d tree is used to speed up the nearest neighbor queries, INN1 has an average time complexity of $O(\log(N_Z))$. If a bounding volume hierarchy is used to skip unnecessary ray-triangle intersections, ray casting performance also behaves logarithmically with respect to the number of faces in the triangulation of $\partial\Omega$ for practical problems. However, benchmarks suggest that a nearest-neighbor query on a k-d tree is about one order of magnitude faster than ray casting using AABB-trees, as can be seen by comparing the performance reported by the `nanoflann` library for k-d trees [11] with the performance reported by CGAL [1]. Faster mesh-based alternatives to ray casting such as the algorithm described in [5], which is similar in spirit to INN1, still require more computation than a single nearest-neighbor query.

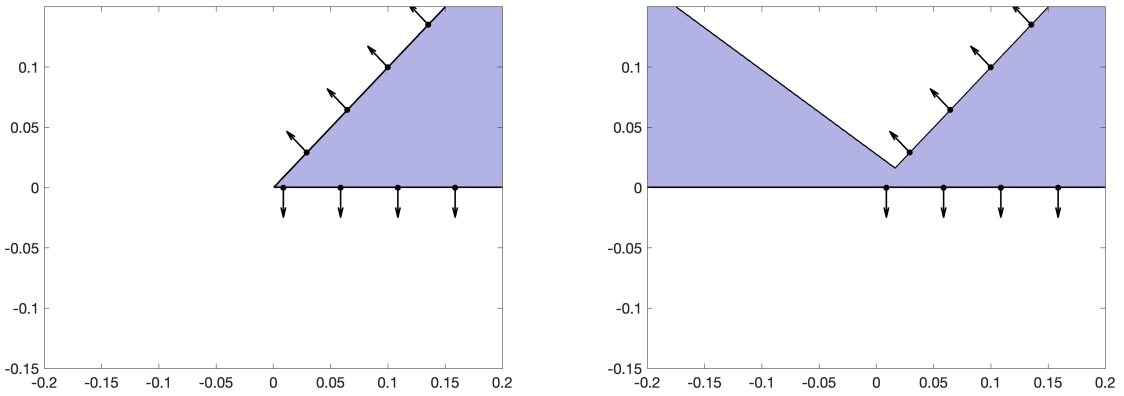
- If the boundary representation of Ω is not watertight, INN1 can tolerate gaps of size up to h between adjacent patches with no additional effort, whereas mesh-based methods require a closed watertight boundary mesh.

- Algorithm 3.1 can be trivially modified to return the value of the approximate signed distance field $\phi_Z(y)$ instead of a simple Boolean result. The additional information about the distance of y to $\partial\Omega$ (with error at most h) can be used by e.g. node generation algorithms to locally increase the density of interior nodes near the boundary. Ray casting cannot provide such information.

Mesh-based inclusion tests are however more reliable than INN1 if the boundary of Ω is not smooth, and the reason is that triangulations can accurately capture the edges and corners of $\partial\Omega$, whereas the clipped Voronoi cells $V_i \cap H_i$ can potentially extend infinitely out of the domain, as shown in Figure 3.1. In this example, nodes Z have been placed with approximately constant spacing $h = 0.05$ along the boundary of the disk sector with internal angle $\omega = \pi/4$ defined in the polar coordinates (r, θ) centered at the origin by

$$\Omega_\omega = \left\{ (r,\theta) \in \mathbb{R}^2 \;\middle|\; 0 < r < 1 \text{ and } 0 < \theta < \omega \right\}.$$

Looking at the plots, it is clear that the test INN1 returns false positives in an unbounded triangle-shaped region that extends beyond the interior of Ω, and that



(a) Exact domain, for reference. **(b)** Region on which INN1 returns true.

Figure 3.1: Behavior of test INN1 on the disk sector with $\omega = \pi/4$. Boundary nodes Z and corresponding normals generated with $h = 0.05$ are drawn on top.

this problem cannot be fixed by supersampling, i.e. by choosing a smaller value of h, because of scale invariance. This means that INN1 is not an inclusion test of any order $k \geq 0$ on non-smooth domains.

This shortcoming can be addressed in two ways. One is to only consider boundary nodes Z that satisfy some additional properties, besides the spacing assumptions controlled by h. For example, when $d = 2$, the test INN1 can be proved to be reliable if the last node before each corner and the first node after each corner are placed symmetrically with respect to the line that bisects the corner. Similar sufficient conditions for 3D edges and corners are not as easy to formulate, however, and more research is required on this topic. Moreover, it would be useful to have a meshless inclusion test that works reliably for a generic set of quasi-uniform boundary nodes Z, without assuming to be in control of their exact placement. The other way to address the shortcoming is to modify Algorithm 3.1, for example by considering more than one nearest neighbor. This is the approach that has been followed in this work.

3.1.2 The INNK inclusion tests

Algorithm 3.1 can be generalized in several ways to the case where $K \geq 2$ nearest neighbors of point y in the set Z are considered. Let $i_1, \ldots, i_K$ be the indices of these nodes, and for each $j = 1, \ldots, K$ define a Boolean variable b_j that is true if and only if $(z_{i_j} - y) \cdot \nu_{i_j} > 0$, that is, if and only if y passes the half-space inclusion test with respect to the K nearest boundary nodes $z_{i_1}, \ldots, z_{i_K}$.

A first generalization of Algorithm 3.1 is to answer the inclusion query positively if and only if all Boolean variables b_i are true. A second generalization is answer positively if any Boolean variable b_i is true, and a third generalization is to answer positively if the majority of the Boolean variables are true (in this case, suppose that K is odd to avoid draws). In the following, we refer to these three generalizations with the names INNK-all, INNK-any, and INNK-majority, respectively. A reference implementation of the three tests is given in Algorithms 3.2 to 3.4.

Figure 3.2 shows the behavior of tests INN2-all, INN2-any, and INN3-majority on the disk sector with $\omega = \pi/4$ for a uniform distribution of boundary nodes with

Algorithm 3.2 – Approximate inclusion test INNK-all

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in Ω.
Input: Set of boundary nodes $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest neighbors to search for.

Output: Boolean result of the inclusion test.

1: Find the indices $i_1, \ldots, i_K$ of the K points in set Z that are closest to query point y.
2: **for** $i \in \{i_1, \ldots, i_K\}$ **do**
3: **if** $(z_i - y) \cdot \nu_i \leq 0$ **then**
4: **return** false
5: **end if**
6: **end for**
7: **return** true

Algorithm 3.3 – Approximate inclusion test INNK-any

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in Ω.
Input: Set of boundary nodes $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest neighbors to search for.

Output: Boolean result of the inclusion test.

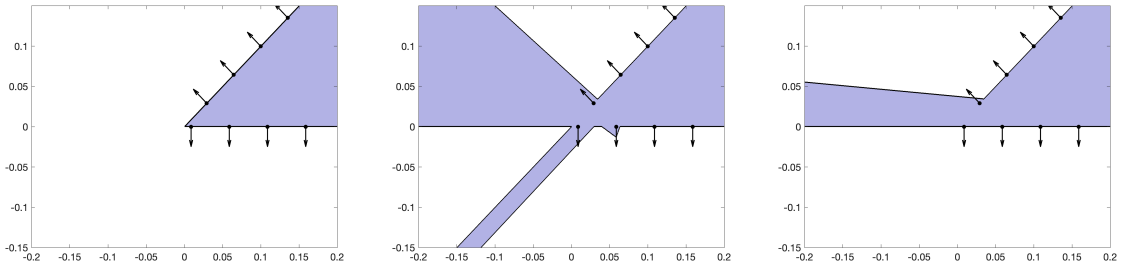
1: Find the indices $i_1, \ldots, i_K$ of the K points in set Z that are closest to query point y.
2: **for** $i \in \{i_1, \ldots, i_K\}$ **do**
3: **if** $(z_i - y) \cdot \nu_i > 0$ **then**
4: **return** true
5: **end if**
6: **end for**
7: **return** false

Algorithm 3.4 – Approximate inclusion test INNK-majority

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in Ω.
Input: Set of boundary nodes $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest neighbors to search for. K must be odd.

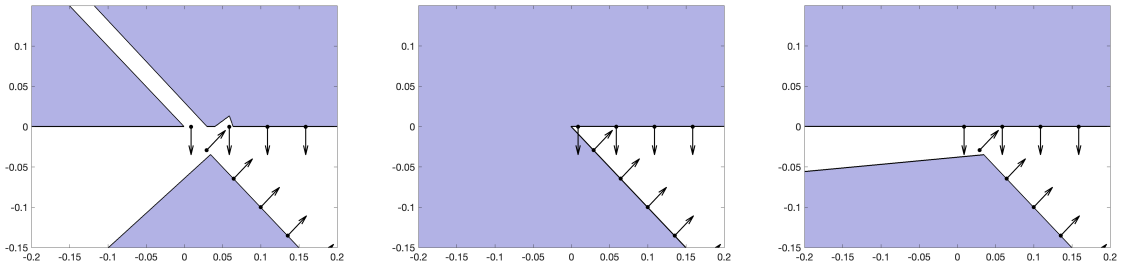
Output: Boolean result of the inclusion test.

1: Find the indices $i_1, \ldots, i_K$ of the K points in set Z that are closest to query point y.
2: Initialize n to 0.
3: **for** $i \in \{i_1, \ldots, i_K\}$ **do**
4: **if** $(z_i - y) \cdot \nu_i > 0$ **then**
5: Set $n = n + 1$.
6: **end if**
7: **if** $n > K/2$ **then**
8: **return** true
9: **end if**
10: **end for**
11: **return** false



(a) Test INN2-all. **(b)** Test INN2-any. **(c)** Test INN3-majority.

Figure 3.2: Behavior of different inclusion tests on the disk sector with $\omega = \pi/4$. Boundary nodes Z and corresponding normals generated with $h = 0.05$ are drawn on top.



(a) Test INN2-all. **(b)** Test INN2-any. **(c)** Test INN3-majority.

Figure 3.3: Behavior of different inclusion tests on the disk sector with $\omega = 7\pi/4$. Boundary nodes Z and corresponding normals generated with $h = 0.05$ are drawn on top.

$h = 0.05$. Compared to Figure 3.1, it is clear that INN2-all performs much better than INN1, being able to accurately reproduce the corner without any false positives or negatives. On the contrary, the tests INN2-any and INN3-majority return true over an unbounded region that extends outside of Ω just like INN1, and are therefore unsuitable for inclusion tests on non-smooth domains.

If we consider reentrant corners with $\omega > \pi$, however, the situation is reversed, as shown in Figure 3.3. This time, it is INN2-any that perfectly reproduces the corner, whereas INN2-all returns false negatives on two large regions that extend all the way to $r = 1/2$, beyond which the closest points to y will be those on the outer arc of Ω_ω, and not the ones near the origin anymore (not shown here in the picture; recall that the disk sector has radius $r = 1$). Test INN3-majority does not perform well either, and we must come to the conclusion that none of the three generalizations considered so far are suitable for a generic non-smooth domain: test INN2-all is only adequate around non-reentrant corners, whereas INN2-any is only adequate around reentrant corners.

In general, we advise against any inclusion test whose outcome is simply a Boolean formula in the variables $b_1, \dots, b_K$: the formula should rather depend explicitly on the nodes $z_{i_1}, \dots, z_{i_K}$ and their normals. As an example, consider the test with $K = 2$ that returns b_1 whenever $b_1 = b_2$, and that otherwise uses the points z_{i_1}, z_{i_2} and the corresponding normals ν_{i_1}, ν_{i_2} to estimate the size of the internal angle ω. If $\omega \leq \pi$, the value $b_1 \wedge b_2$ is returned (equal to false), whereas if $\omega > \pi$, the value $b_1 \vee b_2$ is returned (equal to true).

This inclusion test that switches between INN2-all and INN2-any depending on

the internal angle ω can be shown to be effective in two dimensions, but we are not aware of any way to generalize it to higher dimensions. Since our ultimate goal in this chapter is to construct a node generation algorithm for 3D domains in B-rep format, we have opted for the approach described in Algorithm 3.5, which is less geometrical in nature and independent of $d \geq 1$.

Let us call a query point $y \in \mathbb{R}^d$ *non-ambiguous* if the Boolean variables $b_1, \ldots, b_K$ are either all true or all false. Algorithm 3.5 is based on the empirical observation that, for a sufficiently large K (depending on d), tests from the INNK family are quite reliable for non-ambiguous query points, but otherwise struggle to simultaneously avoid false positives and false negatives, depending on the shape of the boundary near $z_{i_1}, \ldots, z_{i_K}$. Instead of improving the decision heuristic, the new algorithm tries to relocate y to better positions (hence the name INNK-relocate), and then recursively restarts the test, hoping to find non-ambiguous points in the process.

Let us now describe Algorithm 3.5 in more detail. In typical applications, most query points y are non-ambiguous, so the inclusion test will return quickly after computing n. If y is ambiguous, however, a first check is done to see if y is too close to the boundary, in which case we simply return false to minimize the chance of false positives. If we had access to the discretization parameter h used to place points Z on the boundary, this proximity check could be simplified to

$$\|y - z_{i_1}\| \leq 2h,$$

where the constant 2 acts as a safeguard (the exact value is not critical). To avoid including h as an input, however, we use the diameter of the set $\{z_{i_1}, \ldots, z_{i_K}\}$ instead.

Suppose now that y is not too close to the boundary. Then a displacement Δy of size

$$\|\Delta y\| = \alpha \|y - z_{i_1}\|, \quad \text{with } \alpha < 1,$$

is not supposed to cross the boundary of Ω, and we can expect $y + \Delta y \in \Omega$ to hold if and only if $y \in \Omega$. A reasonable value for α that we have used in our numerical experiments is 0.8, but the exact value is not critical.

What remains to be determined is an appropriate direction for the displacement Δy. Looking at Figures 3.2 and 3.3 again, we see that ambiguous points occur along stripes or cones emanating from the tip of the corner, and so the direction parallel to $y - z_{i_1}$ must be avoided. One approach is to choose Δy orthogonal to $y - z_{i_1}$, but this identifies the direction Δy uniquely (up to its sign) only for $d = 2$. We have chosen instead Δy along the positive Cartesian direction that minimizes the scalar product with $y - z_{i_1}$, i.e. the one closest to being orthogonal to $y - z_{i_1}$. Once the direction and magnitude of Δy have been determined, we recursively call the algorithm with the new query points $y + \Delta y$ and $y - \Delta y$, hoping to find non-ambiguous points before the maximum recursion depth $m = 3$ is reached. Since the test INNK-relocate is assumed to be reliable for non-ambiguous points, the test returns true if *any* of the two recursive calls returns true.

The behavior of test INN2-relocate on disk sectors with internal angles ranging from $\pi/4$ to $7\pi/4$ can be seen in Figure 3.4. The results are free from false positives, which makes INNK-relocate particularly suitable for advancing front methods, where false positives can lead to nodes being placed in the exterior of Ω. Moreover, false negatives are kept to a minimum: experimental evidence suggests that INNK-relocate

Algorithm 3.5 – Approximate inclusion test INNK-relocate

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in Ω.
Input: Set of boundary nodes $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest neighbors to search for.
Input: Maximum recursion depth m. If not specified, $m = 3$.

Output: Boolean result of the inclusion test.

Find the indices $i_1, \ldots, i_K$ of the K points in set Z that are closest to query point y.
Initialize n to 0.
for $i \in \{i_1, \ldots, i_K\}$ **do**
    if $(z_i - y) \cdot \nu_i > 0$ **then**
        Set $n = n + 1$
    end if
end for
 # If $n = K$ or $n = 0$, then y is non-ambiguous
if $n = K$ **then**
    return true
else if $n = 0$ **then**
    return false
end if
 # Return false if y is too close to the boundary
Set δ equal to the diameter of the set $\{z_{i_1}, \ldots, z_{i_K}\}$.
if $\|y - z_{i_1}\| \leq 2\delta$ **then**
    return false
end if
 # Try a recursive relocation strategy to find a non-ambiguous y
if $m \leq 0$ **then**
    return false
end if
Set $v = z_{i_1} - y$.
Find $j \in \{1, \ldots, d\}$ that minimizes $|v_j|$.
Initialize $\Delta y \in \mathbb{R}^d$ to the all-zero vector whose j-th element is equal to $0.8\,\|v\|$.
if INNK-RELOCATE($y + \Delta y$, Z, K, $m - 1$) or INNK-RELOCATE($y - \Delta y$, Z, K, $m - 1$) **then**
    return true
end if
return false

behaves like an inclusion test of order 1 around reentrant corners (or edges, in 3D), which is good enough for plenty of meshless numerical schemes, and even high-order ones, as demonstrated in this thesis (see for example Figure 3.12 and the convergence tests in Section 5.2). The behavior of INNK-relocate around 3D corners and edges has not been validated by a systematic approach as in the 2D case, where corners can be conveniently classified by one parameter $\omega \in (0, 2\pi)$, but rather by inspecting the quality of nodes generated by an advancing front algorithm that relies on INNK-relocate for inclusion queries on complex CAD domains. The results of such tests are shown in the next section; we conclude this section with two remarks.

First, meshless inclusion tests relying on $K > 1$ nearest neighbors cannot perform better than $K = 1$ if all points $z_{i_1}, \ldots, z_{i_K}$ belong to a cluster whose radius is significantly smaller than h, because there is no guarantee around corners that the cluster will contain at least one point from each incident hypersurface of dimension $d - 1$ (e.g. from each incident face, in 3D). To avoid this issue, boundary nodes in Z should either be spaced in a quasi-uniform way by enforcing minimum spacing between nodes, or at least a bound on the maximum cardinality of such clusters should be established. When the set of boundary nodes Z has been generated with our advancing front scheme (see Section 3.2), we have found $K = 2$ to be sufficient for 2D domains, and $K = 5$ to be sufficient for 3D domains.

Second, the choices $\alpha = 0.8$ and $m = 3$ proved effective in all of our numerical experiments. However, for more complex geometries, it may be necessary to use a smaller value for α and a larger value for m, based on the idea that moving y in many small steps is a more reliable approach than moving it in fewer, larger steps.

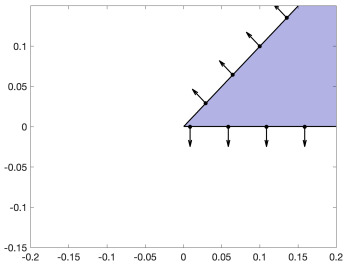
3.2 Node generation

3.2.1 Advancing front algorithms

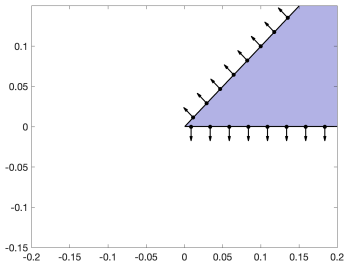
Before introducing our variable-density point cloud generation algorithm for domains $\Omega \subset \mathbb{R}^3$ in B-rep form, we first need, as prerequisites, node placement routines on curves, on trimmed surface patches, and on solids enclosed by a set of boundary points. An elegant way to uniformly handle all of these cases is to define a single variable-density node generation method that can perform all the tasks above for any dimension d of the object to be discretized, and any dimension $n \geq d$ of the space where the object may be immersed. For example, $(d, n) = (1, 3)$ for a curve in $\mathbb{R}^3$, $(d, n) = (2, 3)$ for a trimmed surface patch, and $(d, n) = (3, 3)$ for a solid. Advancing front node placement methods are well suited for this purpose because they are naturally dimension-independent [31, 114], although the special case $d = n$ may be treated differently in implementations for simplicity or performance reasons [95].

The typical setting for a variable-density advancing front node placement method is the following. Let H be an hyperrectangle in $\mathbb{R}^d$, i.e. the product of d compact intervals, and let $G\colon H \to \mathbb{R}^n$ be a smooth function whose Jacobian J_G has maximum rank at every point $y \in H$. In other words, $G(H)$ is a d-dimensional manifold immersed in $\mathbb{R}^n$ and parametrized by G. The goal of the method is to iteratively fill H with a set of parameters

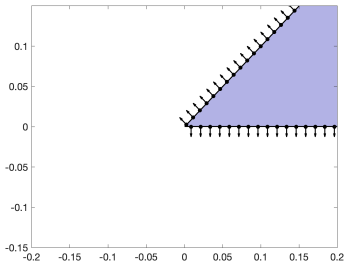
$$Y = \{y_i\}_{i=1}^{N_Y} \subset H$$



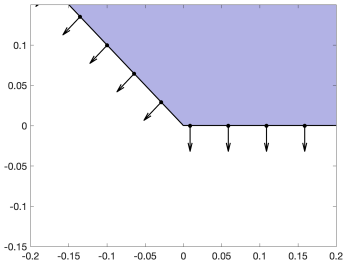
(a) Test with $\omega = \pi/4$ and $h = 0.05$.



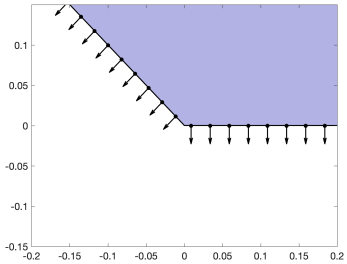
(b) Test with $\omega = \pi/4$ and $h = 0.025$.



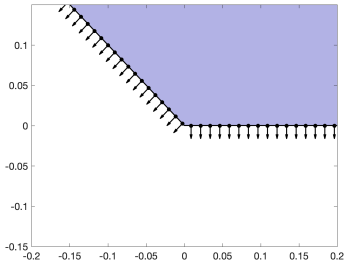
(c) Test with $\omega = \pi/4$ and $h = 0.0125$.



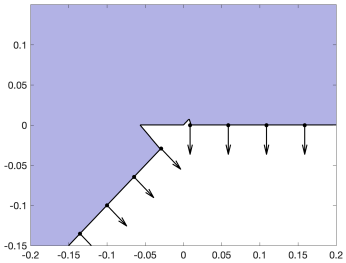
(d) Test with $\omega = 3\pi/4$ and $h = 0.05$.



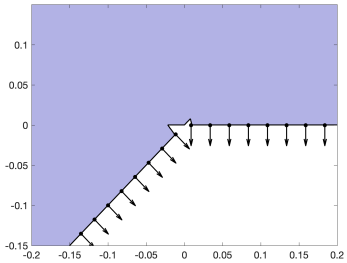
(e) Test with $\omega = 3\pi/4$ and $h = 0.025$.



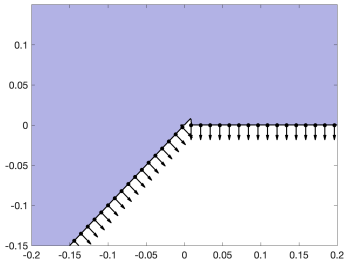
(f) Test with $\omega = 3\pi/4$ and $h = 0.0125$.



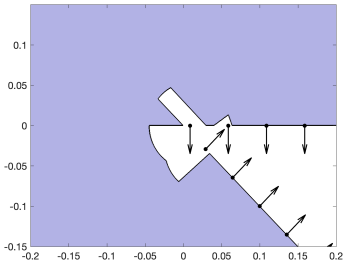
(g) Test with $\omega = 5\pi/4$ and $h = 0.05$.



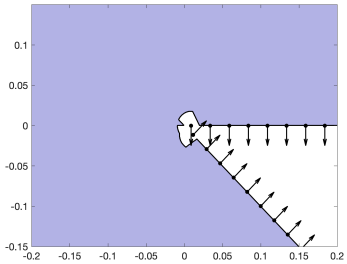
(h) Test with $\omega = 5\pi/4$ and $h = 0.025$.



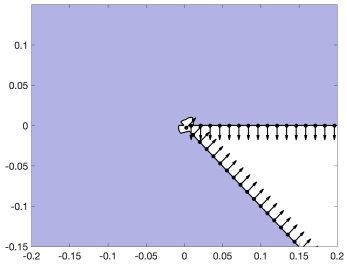
(i) Test with $\omega = 5\pi/4$ and $h = 0.0125$.



(j) Test with $\omega = 7\pi/4$ and $h = 0.05$.



(k) Test with $\omega = 7\pi/4$ and $h = 0.025$.



(l) Test with $\omega = 7\pi/4$ and $h = 0.0125$.

Figure 3.4: Behavior of test INN2-relocate on disk sectors for several values of ω and h.

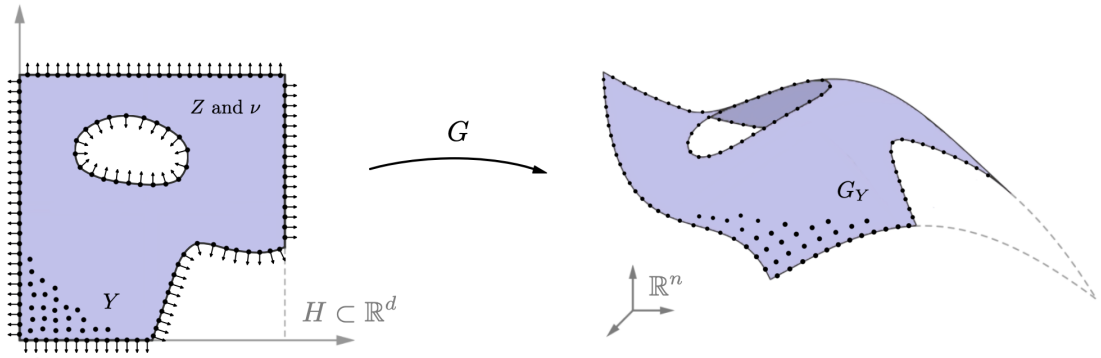


Figure 3.5: Generation of nodes Y in parameter space and nodes G_Y in model space for a parametric surface $G\colon H \to \mathbb{R}^3$ using an advancing front method. Nodes in Y must pass a meshless inclusion test defined by boundary nodes Z and outward-pointing normals ν.

such that every node $G(y_i) \in G_Y := G(Y)$ is approximately at a distance $h(G(y_i))$ from its neighbors on the manifold $G(H)$. The function $h\colon \mathbb{R}^n \to \mathbb{R}^+$ is known as *spacing parameter*, and controls the distance of nodes generated around every point in $G(H)$.

An example with $(d, n) = (2, 3)$ is shown in Figure 3.5. When objects from CAD are considered, G is a NURBS parametric surface, $H \subset \mathbb{R}^d$ is called *untrimmed* parameter space, and $\mathbb{R}^n$ is called *model* space. In this context, it is common to describe a subdomain of interest in H through trimming [76]. For example, in the figure, the immersed surface $G(H)$ is modified by removing one disk and one corner from the untrimmed parameter space H. The resulting subdomain, shown in purple, is called *trimmed* parameter space.

Similarly, when $(d, n) = (3, 3)$ and advancing front methods are used to place nodes in the interior of a bounded domain Ω, we may take G to be the identity mapping from a bounding box H around Ω to itself: once again, we need to restrict H so that $Y \subset \Omega$. To stay true to the meshless approach, we do not use a mesh to represent the subdomain of interest in H. This is where the meshless inclusion tests of Section 3.1 come into play: the advancing front algorithm takes as inputs a set of boundary nodes $Z = \{z_i\}_{i=1}^{N_Z} \subset H$, and a corresponding set of unit vectors $\{\nu_i\}_{i=1}^{N_Z} \subset \mathbb{R}^d$ that play the role of normals pointing out of the subdomain of interest, so that only nodes $y \in H$ passing an approximate inclusion test such as INN1 or INNK-relocate with respect to Z and ν are considered.

We can now describe our advancing front method, as implemented in Algorithm 3.6. A set of starting nodes $Y \subset H$ is either prescribed, or chosen equal to Z, or generated at random in H. In either case, the nodes in Y are inserted into a queue, G_Y is initialized as $G(Y)$, and then the iterative part of the algorithm begins. Let $C = \{c_j\}_{j=1}^{N_C}$ be a set of unit vectors distributed in an approximately uniform way on the surface of the unit sphere in $\mathbb{R}^d$. While the queue is not empty, the first node in the queue y_i is pulled out and *expanded*, and a random rotation matrix R is computed. Expansion of a node means that, for all directions $c_j \in C$, we compute $\hat{y} = y_i + \alpha \Delta y$ with

$$\Delta y = R c_j, \quad \alpha = h(G(y_i)) \,/\, \big(\|J_G(y_i)\Delta y\| + 10^{-15}\big).$$

With this choice, each $G(\hat{y})$ is approximately at a distance $h(G(y_i))$ from $G(y_i)$, and so is a good candidate for expanding the set of nodes G_Y. However, before $G(\hat{y})$ can be accepted, three requirements need to be checked:

- The parameter $\hat{y} \in \mathbb{R}^d$ must belong to H.
- The parameter $\hat{y}$ must pass the inclusion test INNK-relocate defined by boundary nodes Z and outward-pointing normals ν.
- No point $x \in G_Y$ can be closer to $G(\hat{y})$ than $G(y_i)$.

The third check not only enforces a minimum distance between elements in G_Y, but also causes the advancing front algorithm to terminate when H has been filled. If all three requirements are met, the new node $G(\hat{y})$ is appended to G_Y, and the new parameter $\hat{y}$ is appended to Y and to the queue.

Algorithm 3.6 unifies the advancing front methods described in [31, 95] into a single dimension-independent algorithm, with two changes. First, we use INNK-relocate instead of INN1 for the inclusion tests with respect to Z and ν. In this way, supersampling of the boundary nodes Z, as described in e.g. [32], is not required. Second, the directions in the set C are transformed not only by applying a random rotation matrix R, but also by applying the matrix $M = L^{-1}$, with $L \in \mathbb{R}^{d\times d}$ being the lower triangular factor of the Cholesky decomposition of $J_G(y_i)^T J_G(y_i)$. In this way, if the angle between two directions c_1 and c_2 is $\theta \in [0, \pi]$, the angle between $G(\hat{y}_1) - G(y_i)$ and $G(\hat{y}_2) - G(y_i)$ remains θ, with no distortions caused by the mapping G.

Without this correction, sets of nodes G_Y of low quality may be generated for highly anisotropic or badly scaled parametrizations G, i.e. parametrizations for which the ratio $\sigma_{\max}/\sigma_{\min}$ of the largest to smallest singular value of $J_G(y)$ is much greater than 1, for example 10^2 or 10^3. Such parametrizations can unfortunately appear in the boundary representation produced by CAD software; as an example, consider the turbine blade model [45]. Figure 3.6 compares the node distributions produced by Algorithm 3.6 on some boundary patches of the blade with $h \equiv 8$ and

$$\Delta y = Rc_j \quad \text{and} \quad \Delta y = MRc_j,$$

respectively. In both cases, the set C contains $N_C = 15$ vectors uniformly spaced along the unit circle, the same value as in [31, 95]. The flat surface of the turbine blade facing the camera, despite being shaped like a square, has been parametrized very poorly by the CAD software, with $\sigma_{\max}/\sigma_{\min} \approx 10^3$. Figure 3.6 clearly shows what kind of artifacts are present in the node distribution G_Y generated by an advancing front method without the correction provided by the matrix M. We remark that the cost of computing M is negligible compared to the innermost while loop in Algorithm 3.6, because $d \leq 3$ in typical applications. For more details about Algorithm 3.6, its computational complexity, the quality of its output, and the choice of directions C, we refer the reader to the original papers [31, 95].

3.2.2 Advancing front on domains in B-rep form

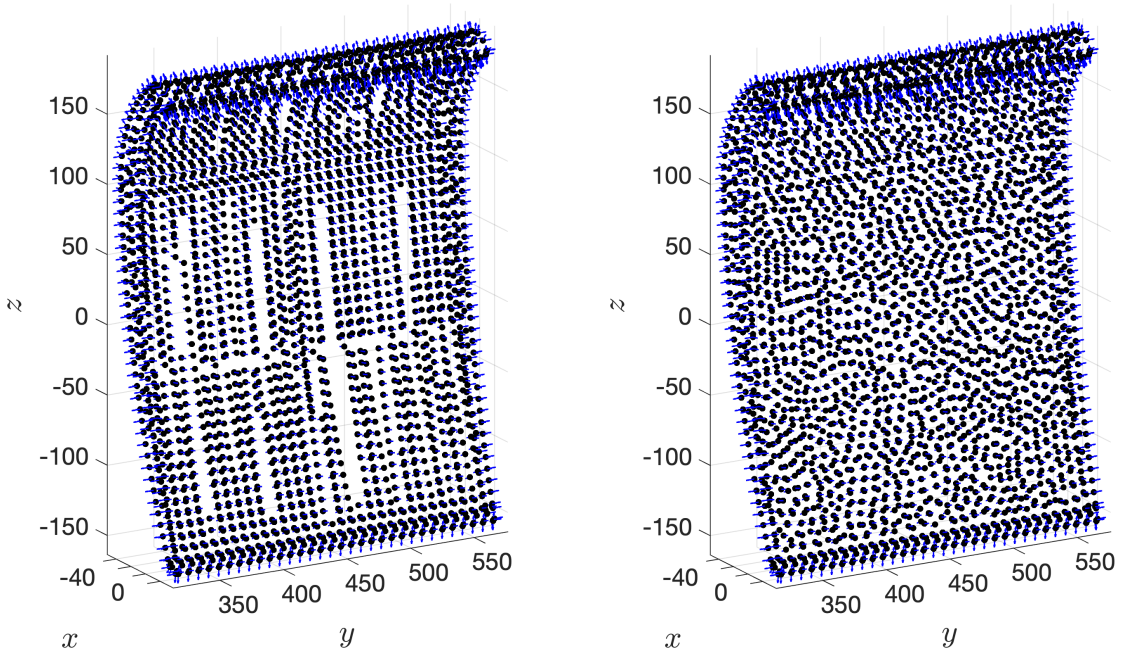
The standard way to represent a domain $\Omega \subset \mathbb{R}^3$ in Computer-Aided Design (CAD) is called *boundary representation* (B-rep for short), and consists of a description of

Algorithm 3.6 – Advancing front method for node generation with variable spacing

Input: Dimension d of parameter space.
Input: Dimension $n \geq d$ of model space.
Input: Axis-aligned box $H \subset \mathbb{R}^d$, i.e. a product of d compact intervals.
Input: Set of nodes $Z = \{z_i\}_{i=1}^{N_Z} \subset H$.
Input: Set of unit vectors $\nu = \{\nu_i\}_{i=1}^{N_Z} \subset \mathbb{R}^d$.
Input: Set of starting nodes $Y_0 \subset H$.
Input: Function $G\colon H \to \mathbb{R}^n$ and its Jacobian $J_G\colon H \to \mathbb{R}^{n\times d}$.
Input: Node spacing function $h\colon \mathbb{R}^n \to \mathbb{R}^+$.
Input: Upper bound $N_{\max}$ to the size of the output G_Y.

Output: Set of nodes $Y = \{y_i\}_{i=1}^{N_Y} \subset \mathbb{R}^d$ placed in parameter space.
Output: Set of nodes $G_Y \subset \mathbb{R}^n$ placed in model space.

if Y_0 is empty **then**
  if Z is empty **then**
    Initialize Y with a random node in H.
  else
    Initialize Y as Z.
  end if
else
  Initialize Y as Y_0.
end if
Initialize G_Y as $G(Y)$, or its first $N_{\max}$ elements if $|Y| > N_{\max}$.
Initialize a dynamic k-d tree T_{G_Y} with the nodes in G_Y.
Initialize a set of directions $C = \{c_j\}_{j=1}^{N_C} \subset \mathbb{R}^d$ as in [95].
Set $i = 1$
while $i \leq |Y|$ and $|Y| \leq N_{\max}$ **do**
  Initialize $R \in \mathbb{R}^{d\times d}$ as a random rotation matrix.
  Compute the Cholesky decomposition $J_G(y_i)^T J_G(y_i) = LL^T \in \mathbb{R}^{d\times d}$.
  Compute $M = L^{-T}$.
  Set $j = 1$
  while $j \leq N_C$ and $|Y| < N_{\max}$ **do**
    Set $\Delta y = MRc_j$
    Set $\alpha = h(G(y_i)) \,/\, \left(\|J_G(y_i)\Delta y\| + 10^{-15}\right)$
    Set $\hat{y} = y_i + \alpha\Delta y$
    if $\hat{y}$ is not in H **then**
      Set $j = j + 1$ and continue from line 19.
    end if
    if Z is not empty and INNK-RELOCATE($\hat{y}$, Z, ν) returns false **then**
      Set $j = j + 1$ and continue from line 19.
    end if
    Use T_{G_Y} to find the point x in set G_Y that is closest to $G(\hat{y})$.
    if $\|G(\hat{y}) - x\| < (1 - 10^{-10})\,\|G(\hat{y}) - G(y)\|$ **then**
      Set $j = j + 1$ and continue from line 19.
    end if
    Append $\hat{y}$ to the end of Y.
    Append $G(\hat{y})$ to the end of G_Y.
    Insert $G(\hat{y})$ into T_{G_Y}.
    Set $j = j + 1$
  end while
  Set $i = i + 1$
end while



(a) Matrix M not applied to Δy when generating new candidates in the advancing front method.

(b) Matrix M is applied to Δy when generating new candidates in the advancing front method.

Figure 3.6: Comparison of nodes generated on the surface of a turbine blade CAD geometry with spacing $h \equiv 8$. On the left: classical advancing front method. On the right: Algorithm 3.6.

$\partial\Omega$ as a finite union of parametric surfaces, known as *patches*. An advancing front node generation algorithm such as Algorithm 3.6, however, is defined only for a single parametrization G, so a natural idea is to invoke the algorithm for each patch of the B-rep. In this section we explain how this idea can be realized in practice.

Before describing our node generation algorithm, let us first define boundary representations more precisely and introduce important notation.

Definition 3.2.1. The *boundary representation* of a bounded domain $\Omega \subset \mathbb{R}^3$ is formally defined as the 5-tuple

$$\left(\{\Gamma_i\}_{i=1}^{N_F}, \{H_i\}_{i=1}^{N_F}, \{G_i\}_{i=1}^{N_F}, \{T_i\}_{i=1}^{N_F}, \{\gamma_{ij}\}_{i=1,j=1}^{N_F,N_E(i)}\right) \tag{3.1}$$

where

1. The patches $\Gamma_i \subset \partial\Omega$ are a *partition* of $\partial\Omega$, in the sense that they have nonempty, mutually disjoint interiors and their union is equal to $\partial\Omega$.

2. The *untrimmed parameter spaces* H_i are nonempty rectangles in $\mathbb{R}^2$.

3. The *parametric mappings* $G_i \colon H_i \to \mathbb{R}^3$ are tensor-product NURBS surfaces such that $\Gamma_i \subseteq G_i(H_i)$. $\mathbb{R}^3$ is called *model space*.

4. The *trimmed parameter spaces* T_i are subsets of H_i such that $G_i(T_i) = \Gamma_i$. The boundary of each T_i must be parametrizable by $N_E(i)$ NURBS curves.

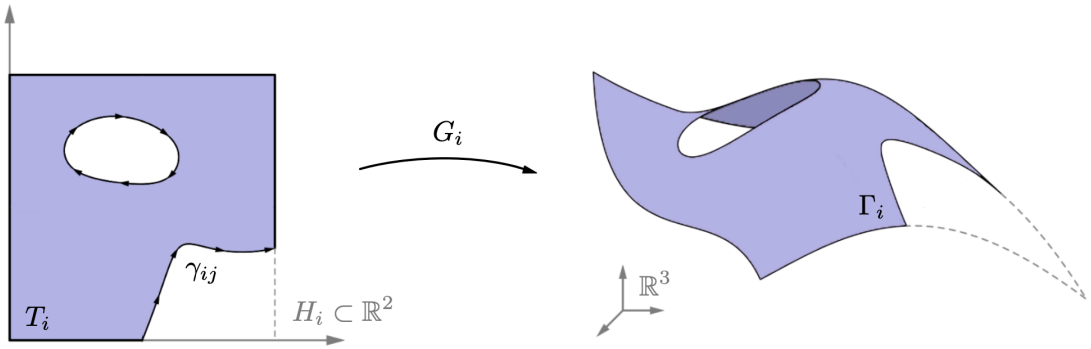


Figure 3.7: Parametric mapping G_i from the trimmed parameter space T_i to the trimmed surface patch Γ_i in model space. Trimming curves γ_{ij} define T_i as a subset of the rectangle H_i.

5. The *trimming curves* $\gamma_{ij} \colon [0,1] \to \partial T_i \subset H_i$ are simple NURBS curves whose images for each $i = 1, \ldots, N_F$ have mutually disjoint interiors and union equal to ∂T_i. The orientation is chosen so that T_i stays on the left when traveling along the curves.

To help the reader get accustomed to the notation, an example for each of the variables $\Gamma_i, H_i, G_i, T_i, \gamma_{ij}$ is shown in Figure 3.7. In some contexts, it may be more natural to only call *trimming curves* the curves whose image is disjoint from ∂H_i. However, in our setting, all curves are called *trimming curves* for the sake of consistent notation and terminology.

We can now describe our variable-density node generation algorithm for domains in B-rep form. The algorithm is structured in a hierarchical way, with lower-dimensional objects being discretized before higher-dimensional ones. First, auxiliary nodes Z_{ij} are placed along each trimming curve γ_{ij} of each trimmed parameter space T_i using Algorithm 3.6 with $(d, n) = (1, 2)$. Then, the same advancing front algorithm with $(d, n) = (2, 3)$ is used to place nodes Y_i in the interior of each trimmed parameter space T_i using the auxiliary nodes

$$Z_i := \bigcup_{j=1}^{N_E(i)} Z_{ij}$$

for the inclusion tests. Finally, nodes X are placed in the interior of Ω by Algorithm 3.6 with $(d, n) = (3, 3)$ using

$$G_Y := \bigcup_{i=1}^{N_F} G_i(Y_i)$$

as the boundary nodes for the inclusion tests.

We remark that the auxiliary nodes in Z_i are only used to generate Y_i, and can be discarded afterwards (hence the name *auxiliary*): their image through G_i is not included in the set G_Y, or else some nodes would be placed along the interface between neighboring patches, where the normal vectors may not be well-defined (consider for example the edges of a cube). Algorithm 3.6 relies on the inclusion test

INNK-relocate to generate Y_i, which, as explained at the end of Section 3.1, can only work reliably for a set of auxiliary nodes Z_i with smoothly varying density in T_i. Although it would be natural to place auxiliary nodes in Z_{ij} along the corresponding trimming curve γ_{ij} so that their image through G_i is spaced according to h (to match the density of Y_i that will be generated afterwards), this naive approach has two serious flaws.

First, the density of nodes in Z_i across two consecutive trimming curves γ_{i1} and γ_{i2} is continuous if and only if

$$\|J_G(\gamma_{i1}(1))\gamma'_{i1}(1)\| = \|J_G(\gamma_{i2}(0))\gamma'_{i2}(0)\|,$$

an identity over which we have no control, and that does not hold for the vast majority of trimming curves in a typical boundary representation. Second, the density of nodes in Z_i will be too low near points $z \in \partial T_i$ where $J_G(z)$ is singular or nearly singular, in the sense that the product of its singular values is very close to zero. This is the case, for example, whenever a quadrilateral patch is used to parametrize a spherical triangle by collapsing one edge to a single point.

To solve the two issues, we suggest to place all auxiliary nodes in Z_i using a constant spacing function equal to

$$\delta_i \approx \min_{z \in \partial T_i} \frac{h(G_i(z))}{\|\nabla(h \circ G_i)(z)\|}.$$

The value δ_i can be understood as the smallest spacing that would be used at some point when generating Z_i using the naive approach. In this way we enforce a minimum density near points $z \in \partial T_i$ where $J_G(z)$ is singular or nearly singular (as long as $J_G(z)$ is not singular for all $z \in \partial T_i$), and we make sure by construction that the density of Z_i is constant across different trimming curves.

As for the density of G_Y on $\partial\Omega$, the idea to generate parameters Y_i in T_i so that their image through G_i is spaced according to h works well in practice, and requires no further discussion. Finally, nodes X in Ω are also spaced according to h.

A complete description of our variable-density node generation algorithm is given in Algorithm 3.7. Since the main structure of the algorithm has already been explained, we add a few comments to clarify some implementation details.

- Each trimming curve γ_{ij} is discretized twice: once at line 7 to approximate δ_i, and once at line 25 to generate Z_{ij}.

- The choice of starting parameters s_L and s_R for the discretization of each curve γ_{ij} is made so that the auxiliary nodes Z_i, in addition to being uniformly spaced in T_i according to δ_i, are also symmetrical around corners of T_i. As explained in Section 3.1, this further improves the reliability of 2D inclusion tests based on nearest neighbor search, including INNK-relocate.

- Nodes are placed along γ_{ij} by an advancing front method with starting parameters s_L and s_R. When the two fronts meet, a gap of size up to 2δ may be left in the middle of Z_{ij}. This in turn may affect negatively the subsequent generation of nodes $Y_i \subset T_i$. The code from line 34 to 38 checks the size of the gap and adds a midpoint, if necessary.

Algorithm 3.7 – Variable-density node generation algorithm for a B-rep domain

Input: Boundary representation of a domain $\Omega \subset \mathbb{R}^3$, as in Definition 3.2.1
Input: Node spacing function $h\colon \mathbb{R}^3 \to \mathbb{R}^+$.
Input: Upper bound $N_{\max}$ to the number of generated points.

Output: Set of nodes $G_Y \subset \partial\Omega$ placed on the boundary of the domain and spaced according to h.
Output: Set of unit vectors $\nu_Y \subset \mathbb{R}^3$, the outward-pointing normals to $\partial\Omega$ at G_Y.
Output: Set of nodes $X \subset \Omega$ placed in the interior of the domain and spaced according to h.

1: Initialize $G_Y = \emptyset$ and $Z_{\max} = \max(10^6, N_{\max})$.
2: Set $i = 1$
3: **while** $i \leq N_F$ and $N_{\max} > 0$ **do**
4: Initialize δ_i as $+\infty$.
5: Set $j = 1$
6: **while** $j \leq N_E(i)$ and $N_{\max} > 0$ **do**
7: Call Algorithm 3.6 with $d = 1$, $n = 3$, $H = [0, 1]$, no boundary nodes, no normals, no starting nodes, function $G_i \circ \gamma_{ij}$ with Jacobian $(J_{G_i} \circ \gamma_{ij})\gamma'_{ij}$, node spacing h and upper bound $N_{\max}$. Store the output in variables S and G_S.
8: **for** $k = 1, \ldots, |S| - 1$ **do**
9: $\delta_i = \min(\delta_i, \|\gamma_{ij}(s_{k+1}) - \gamma_{ij}(s_k)\|)$
10: **end for**
11: Set $j = j + 1$
12: **end while**
13: **if** $\delta_i = +\infty$ **then**
14: *# Patch is too small, skip it*
15: Set $i = i + 1$ and continue from line 3.
16: **end if**
17: Initialize $Z_i = \emptyset$ and $\nu_Z = \emptyset$.
18: Set $j = 1$
19: **while** $j \leq N_E(i)$ and $N_{\max} > 0$ **do**
20: Set $s_L = (\delta_i/2) \,/\, (\|\gamma'_{ij}(0)\| + 10^{-15})$ and $s_R = 1 - (\delta_i/2) \,/\, (\|\gamma'_{ij}(1)\| + 10^{-15})$.
21: **if** $s_L \geq s_R$ **then**
22: *# Trimming curve is too short, skip it*
23: Set $j = j + 1$ and continue from line 19.
24: **end if**
25: Call Algorithm 3.6 with $d = 1$, $n = 2$, $H = [0, 1]$, no boundary nodes, no normals, starting nodes s_L and s_R, function γ_{ij} with Jacobian γ'_{ij}, node spacing δ_i and upper bound $Z_{\max}$. Store the output in variables S and Z_{ij}. Include s_L and s_R at the start of S. Set $N = |Z_{ij}|$.
26: **if** $N = Z_{\max}$ **then**
27: Raise an error, because Z_{ij} is too large.
28: **end if**
29: **for** $k = 1, \ldots, N$ **do**
30: Append the k-th element of Z_{ij} to the end of Z_i.
31: Append to the end of ν_Z the unit vector orthogonal to $\gamma'_{ij}(s_k)$ pointing out of T_i.
32: **end for**
33: *# Fix a possible gap in the distribution of Z_{ij} where the 1D advancing fronts meet*
34: **if** $\|\gamma_{ij}(s_N) - \gamma_{ij}(s_{N-1})\| > 1.2\,\delta_i$ **then**
35: Set $s = (s_N + s_{N-1})/2$
36: Append $\gamma_{ij}(s)$ to the end of Z_i.
37: Append to the end of ν_Z the unit vector orthogonal to $\gamma'_{ij}(s)$ pointing out of T_i.
38: **end if**
39: Set $j = j + 1$
40: **end while**
41: Initialize $Y_i = \emptyset$
42: Call Algorithm 3.6 with $d = 2$, $n = 3$, $H = H_i$, boundary nodes Z_i, boundary normals ν_Z, no starting nodes, function G_i with Jacobian J_{G_i}, node spacing h and upper bound $N_{\max}$. Store the output in variables Y_i and G_{Y_i}.
43: **for** $k = 1, \ldots, |Y_i|$ **do**
44: Append the k-th element of G_{Y_i} to the end of G_Y.
45: Append to the end of ν_Y the unit vector orthogonal to the columns of $J_{G_i}(y)$ pointing out of Ω, with y being the k-th element of Y_i.
46: $N_{\max} = N_{\max} - 1$
47: **end for**
48: Set $i = i + 1$
49: **end while**
50: Call Algorithm 3.6 with $d = 3$, $n = 3$, H being a bounding box around Ω, boundary nodes G_Y, boundary normals ν_Y, no starting nodes, identity function with identity Jacobian, node spacing h and upper bound $N_{\max}$. The two outputs are identical, store one of them in X.

- To prevent the size of Z_i from being excessively large if $\delta_i \approx 0$, we set an upper bound to $|Z_i|$, above which the algorithm terminates unsuccessfully.

- The set G_Y does not contain the points $G_i(Z_i)$, because the normal vector to $\partial\Omega$ may not be well-defined at those points. However, to prevent gaps in the node distribution of size up to $2h$ between neighboring patches, the value of h is halved around the interface between different patches. The details are not given in Algorithm 3.6 to keep the code as simple as possible.

In computer-aided design, trimming curves γ_{ij} and parametrizations G_i are NURBS curves and surfaces, respectively. Although Algorithm 3.7 does not mention this aspect explicitly, NURBS parametrizations and their derivatives are evaluated in a stable and efficient way using de Boor's algorithm.

All the algorithms in this chapter have been implemented in C++ and published under version 3 of the LGPL license as a library called *NodeGenLib*, available at

`https://github.com/BrunoDegliEsposti/NodeGenLib`

The library also contains MATLAB MEX files that provide an interface to C++ code, so that all the algorithms can be conveniently accessed in a MATLAB environment.

NodeGenLib can load boundary representations from industry-standard STEP files, as defined in the ISO 10303 series [53], by using the open-source platform Open CASCADE Technology [84]. Although this platform provides a complete CAD kernel, we only make use of its STEP files reader and its implementation of de Boor's algorithm, and do not rely on any of its algorithms related to meshing or computational geometry. The combination of the inclusion test INNK-relocate, the advancing front method in Algorithm 3.6 and the node generation code in Algorithm 3.7 provides a completely meshless pipeline from STEP files to variable-density point clouds in the interior and on the boundary of a domain in B-rep form.

Numerical results

To validate Algorithm 3.7, we generate nodes over four different domains: a solid that spells out the word CAD [85], a gear shaft [43], the body of a winged corkscrew [46], and the well-known Stanford bunny test geometry [44]. In this way, we showcase the behavior of our algorithm on domains with very different geometrical features: the first two are mechanical parts with sharp edges, the third has smoother transitions between patches and consistently low thickness, and the fourth is an organic shape made of 700 small quadrilateral NURBS surfaces. Each domain has been rescaled to have unit volume, and translated so that its center of mass is (approximately) at the origin. Figures 3.8 to 3.11 show the distribution of boundary nodes G_Y and interior nodes X produced by Algorithm 3.7 with constant spacing function $h \equiv 0.02$. At the time of writing, NodeGenLib can generate about 5000 nodes per second on an Apple M1 processor on the domains that we have considered. Our performance is on par with the results in [32].

Visual inspection of the generated nodes shows a uniform distribution in the interior and on the surface of each domain, even around edges, corners, and regions of

large curvature. To demonstrate that the nodes are suitable as a domain discretization for meshless methods, we solve the elliptic boundary value problem

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega \end{cases} \tag{3.2}$$

on all four domains using the RBF-FD method based on polyharmonic radial basis kernel

$$K(x, y) = \|x - y\|^{2q-1}$$

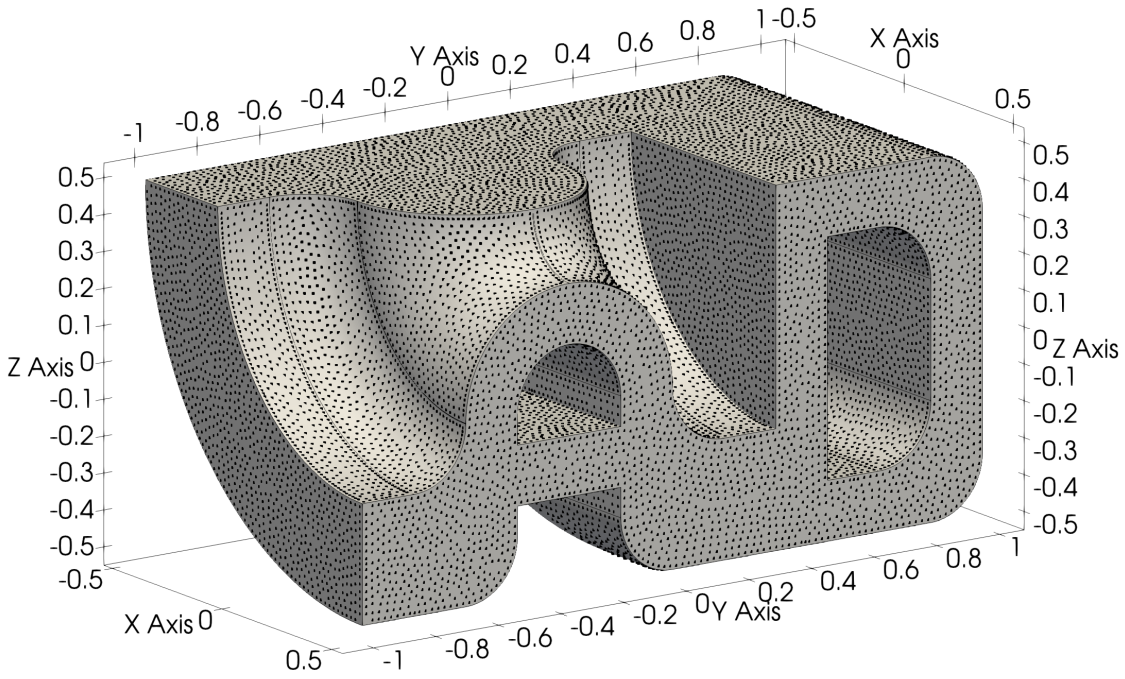
and a polynomial augmentation term in Π_q^d, see Section 2.3 for more information on numerical differentiation formulas. Stencil selection is done by choosing the $2\dim(\Pi_q^d)$ closest nodes, as recommended in [6]. The right-hand side data f, g are chosen as

$$f = -\Delta u_F, \quad g = u_F|_{\partial\Omega}$$

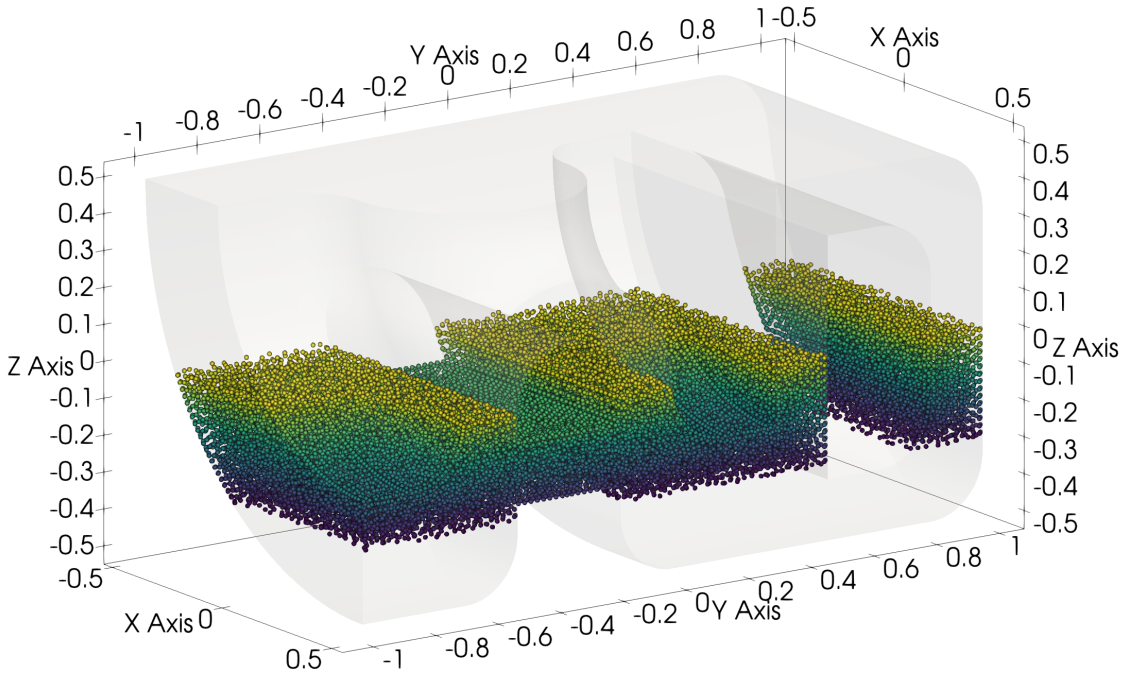
with u_F equal to the 3D generalization of Franke's function described in [89], suitably rescaled from $[0, 1]^3$ to $[-1, 1]^3$. Let $\hat{u}$ be the numerical solution obtained with the RBF-FD method on the interior nodes X generated by Algorithm 3.7. The exact solution $u = u_F$ is known a priori, and so the absolute RMS numerical errors

$$e_2 := \sqrt{\frac{1}{N_X} \sum_{i=1}^{N_X} (u_F(x_i) - \hat{u}_i)^2}$$

can be computed. Figure 3.12 shows the errors as a function of h on our four test domains for different values of the polynomial augmentation order q. We observe $\mathcal{O}(h^2)$ convergence rate for $q = 3, 4$, and $\mathcal{O}(h^4)$ convergence rate for $q = 5$, which is line with the observation that RBF-FD with odd augmentation order q has superconvergent behavior for Poisson's equation with Dirichlet boundary conditions (usually, one would expect the convergence order to match the polynomial consistency order $q - 2$), see for example [23, 61]. The high errors for $q = 5$ and $h \geq 0.02$ in the case of the corkscrew body can be attributed to the small thickness of the domain, which is about 0.05 in its thinnest regions (see points with $y \approx -2$ in Figure 3.10). Indeed, when the ratio $0.05/h$ grows and reaches q, the errors drop dramatically and $\mathcal{O}(h^4)$ convergence rate is observed like in the other tests.

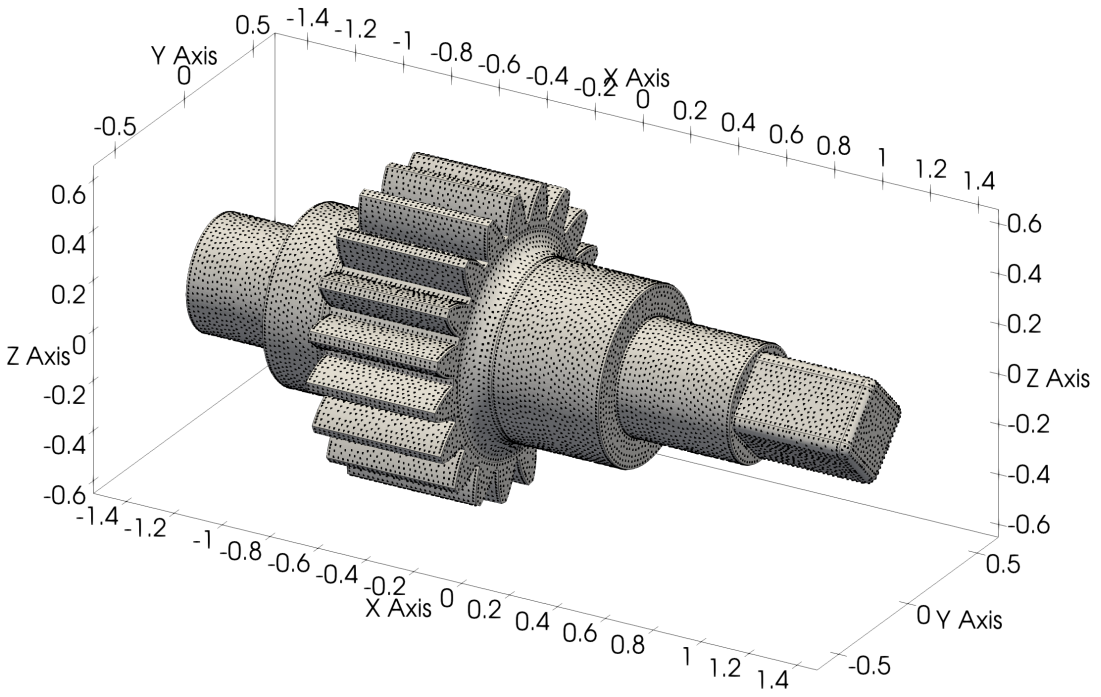


(a) Boundary nodes G_Y drawn on top of $\partial\Omega$. Edges between patches are shown using solid lines.

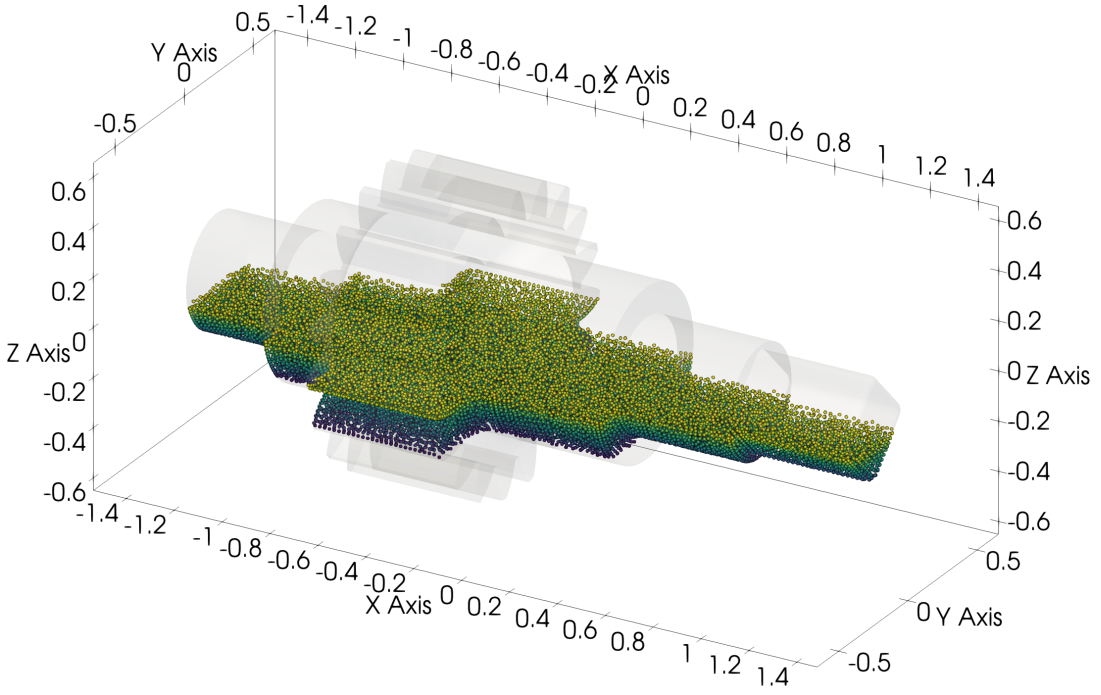


(b) Horizontal slice of interior nodes X. Color map is used solely for visualization and does not convey any additional information.

Figure 3.8: Node generation on a domain that spells out the word CAD using Algorithm 3.7 with constant node spacing $h = 0.02$.

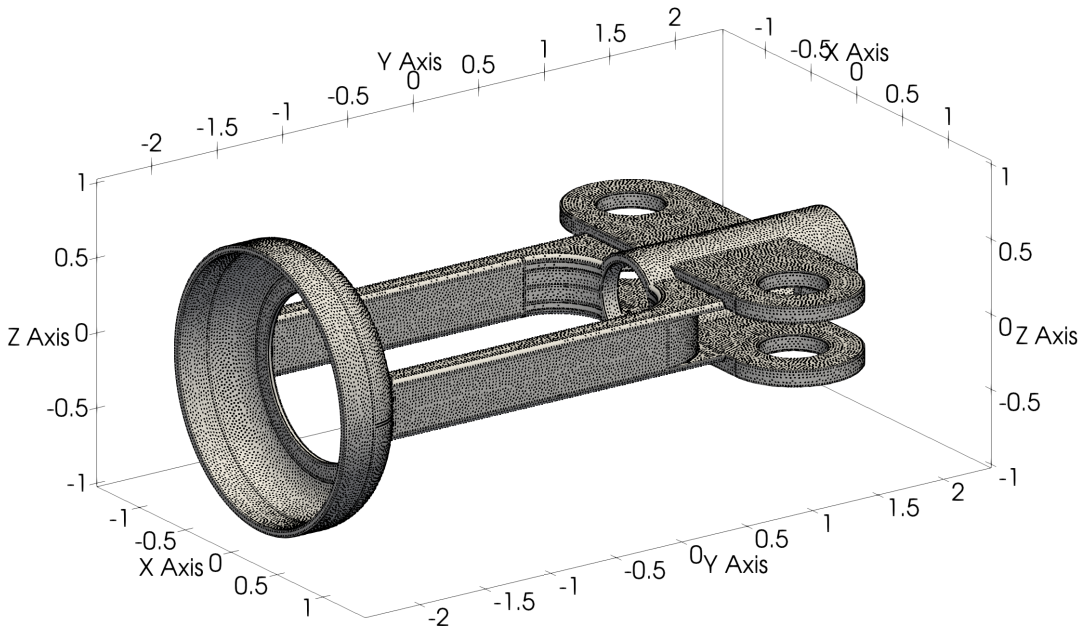


(a) Boundary nodes G_Y drawn on top of $\partial\Omega$. Edges between patches are shown using solid lines.

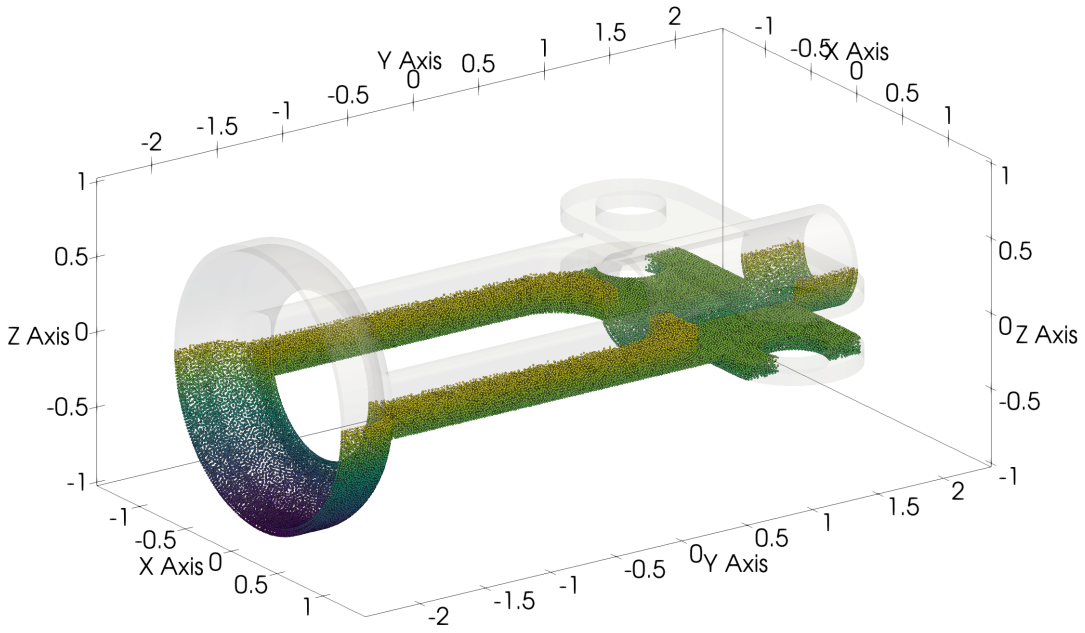


(b) Horizontal slice of interior nodes X. Color map is used solely for visualization and does not convey any additional information.

Figure 3.9: Node generation on a gear shaft using Algorithm 3.7 with constant node spacing $h = 0.02$.

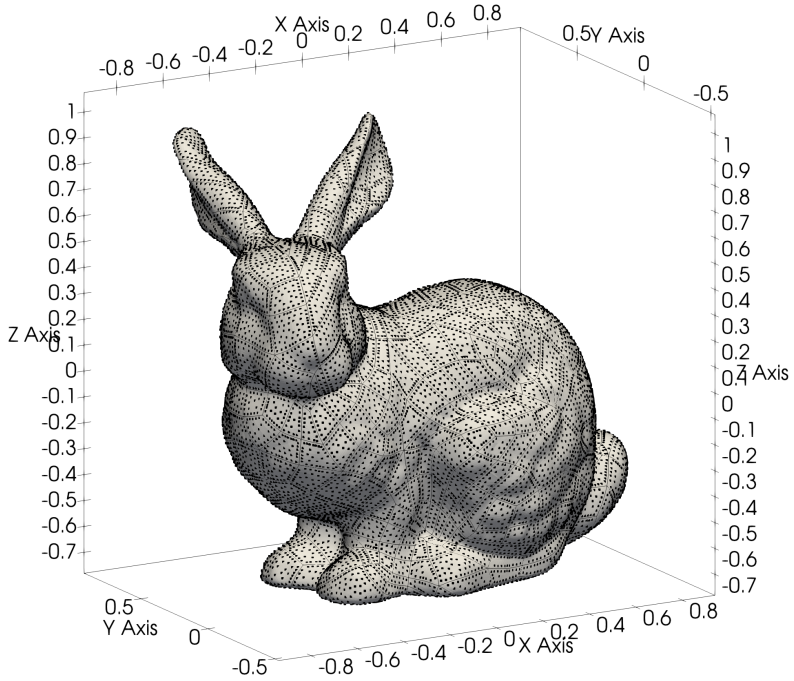


(a) Boundary nodes G_Y drawn on top of $\partial\Omega$. Edges between patches are shown using solid lines.

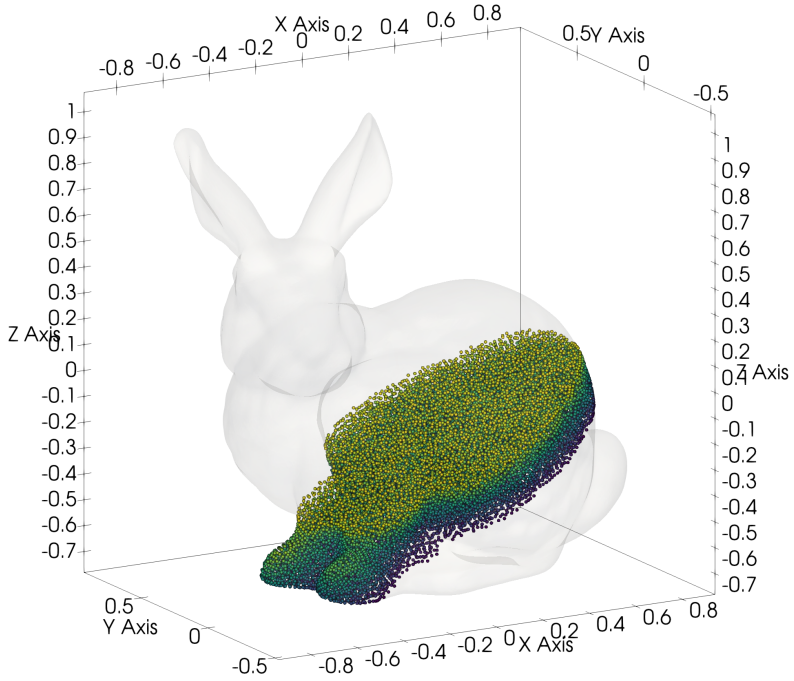


(b) Horizontal slice of interior nodes X. Color map is used solely for visualization and does not convey any additional information.

Figure 3.10: Node generation on the body of a corkscrew using Algorithm 3.7 with constant node spacing $h = 0.02$.

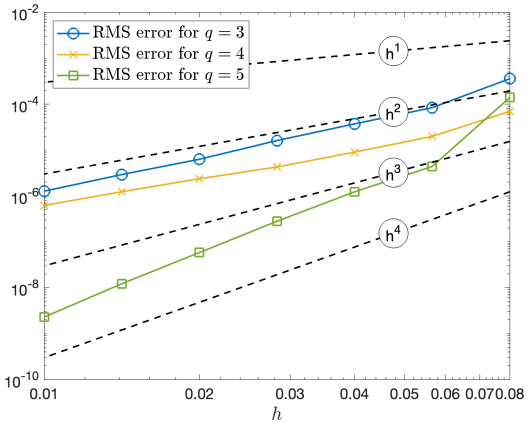


(a) Boundary nodes G_Y drawn on top of $\partial\Omega$. Edges between patches are shown using solid lines.

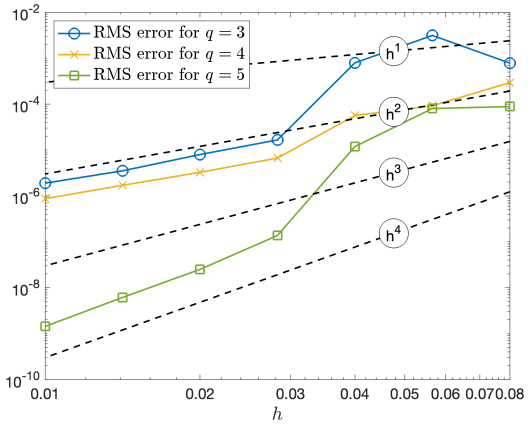


(b) Diagonal slice of interior nodes X. Color map is used solely for visualization and does not convey any additional information.

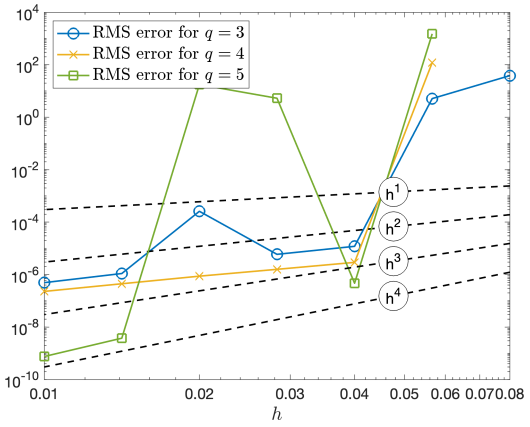
Figure 3.11: Node generation on the Stanford bunny test geometry using Algorithm 3.7 with constant node spacing $h = 0.02$.



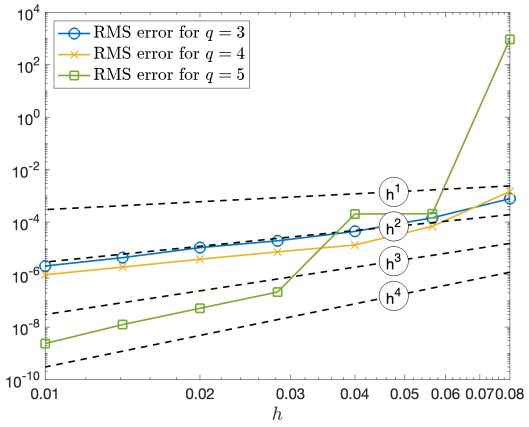
(a) RMS errors for the CAD text domain.



(b) RMS errors for the gear shaft domain.



(c) RMS errors for the corkscrew body domain.



(d) RMS errors for the Stanford bunny domain.

Figure 3.12: Solution of the elliptic boundary value problem (3.2) on four test domains. Absolute RMS errors plotted as functions of h for polynomial augmentation orders $q = 3, 4, 5$. Manufactured solution based on a rescaled 3D generalization of Franke's function.

CHAPTER 4

Meshless moment-free quadrature

Efficient high order numerical integration over piecewise smooth domains Ω in $\mathbb{R}^2$ and $\mathbb{R}^3$ and on surfaces is a fundamental tool in numerical analysis and engineering applications, see for example the recent literature reviews in [30, 48]. In this chapter we present a new technique for generating quadrature formulas on scattered nodes that only requires as inputs the outward-pointing normals along $\partial\Omega$ at the boundary nodes, in addition to the positions of the nodes themselves, and that relies on the discretization of a boundary value problem for the divergence operator by numerical differentiation and the least squares solution of a sparse underdetermined linear system.

Recall from Chapter 2 that, for a given set of quadrature nodes $Y = \{y_1, \dots, y_N\} \subset \overline{\Omega}$, the classical way to obtain a weight vector $w = (w_i)_{i=1}^N \in \mathbb{R}^N$ of a quadrature formula

$$\int_\Omega f(x)\, d\mu(x) \approx \sum_{i=1}^N w_i f(y_i) \tag{4.1}$$

for the integral with respect to some measure μ over Ω is to choose a finite dimensional functional space

$$\mathcal{S} = \operatorname{span}\{s_1, \dots, s_M\} \subset C(\overline{\Omega})$$

with good approximation properties, such as (piecewise) polynomials or radial basis functions, and require the exactness of the formula for all $f \in \mathcal{S}$, which is equivalent to the exact reproduction of the *moments* $\int_\Omega s_j\, d\mu$,

$$\int_\Omega s_j(x)\, d\mu(x) = \sum_{i=1}^N w_i s_j(y_i), \quad j = 1, \dots, M. \tag{4.2}$$

This design principle based on exactness (called *moment fitting* by some authors) leads to a linear system for the weight vector w as soon as all the moments are known.

On complex geometries, however, moments cannot be found with analytical methods, and so the task of computing a quadrature formula with the moment fitting approach is as difficult as integrating a generic function f with respect to μ, the task which motivates the construction of a quadrature formula over Ω in the first place. For this reason, the moment fitting approach is typically restricted to elementary regions such as cubes or simplices, where analytic methods are effective, and then composite quadrature formulas are obtained by meshing the domain into smooth images of such elementary regions. This approach is used successfully in e.g. the finite element method, but has the downside of making high order integration depend on high order mesh generation, which can be prohibitively costly and non-robust in applications

[112, 2, 48]. Instead of meshing into images of elementary regions, different types of partitioning may be used, followed by the dimension reduction approach that replaces volume or surface integrals by repeated univariate integration, such as the standard iterated integration on generalized rectangles or sectors [106, 92]. Recent versions of the dimension reduction method [101, 48, 49] rely on the divergence theorem over arbitrary subregions with known parametrizations of their piecewise smooth boundaries. However, a significant drawback of these methods is that many quadrature nodes must be placed outside of $\overline{\Omega}$ in its bounding box, unless the shape of the integration domain satisfies some restrictive assumptions (see e.g. the concept of base-line in [101]).

When the quadrature nodes in Y are fixed in advance (for example, supplied by the user) and the task is to generate the weight vector w, the problem setting is related to scattered data fitting, because the integral $\int_\Omega f\,d\mu$ can be approximated by $\int_\Omega s\,d\mu$ if s is an approximation of f, and this leads to a quadrature formula as soon as s is obtained for each fixed Y by a linear scattered data fitting operator $\mathcal{Q}_Y : \mathbb{R}^N \to \mathcal{S}$ as

$$s = \mathcal{Q}_Y f_{|Y}, \quad \text{with } f_{|Y} := \big(f(y_i)\big)_{i=1}^N.$$

The most established tool for scattered data fitting is kernel-based interpolation, in particular with radial basis functions [116]. The meshless quadrature formulas obtained in this way provide optimal recovery of the integration functional [77], in particular polyharmonic and thin plate spline RBF give rise to optimal quadrature formulas for multivariate Sobolev spaces [97]. However, numerical implementation of these methods for general domains is problematic because the corresponding linear system (4.2) is not sparse, and the moments of the radial basis functions have to be computed. The first problem may be tackled by either partitioning Ω or using spaces $\mathcal{S}$ with locally supported bases such as B-splines, but the computation of the moments remains a major obstacle unless Ω is a polygonal domain [102]. A dimension reduction approach for the moments over curved triangles has been suggested in [88]. When using polynomial spaces $\mathcal{S}$ on scattered nodes, a feasible approach is to choose the polynomial degree such that the dimension M of $\mathcal{S}$ is significantly smaller than the number of nodes N, which typically makes the system (4.2) underdetermined and solvable. Then a weight vector w satisfying (4.2) may be found as a minimum norm solution [67, 36, 41, 42, 104] or by subset selection [100]. These methods require the moments of the polynomials, in particular [104] suggests a sophisticated algorithm for computing them on bivariate domains bounded by B-spline curves.

The method that we suggest does not require any meshing, and avoids the moment computation problem entirely. Instead of exact reproduction of moments, it relies on the discretization of two linear differential operators $\mathcal{L}$ and $\mathcal{B}$ satisfying the identity

$$\int_\Omega \mathcal{L}u\,d\mu = \int_{\partial\Omega} \mathcal{B}u\,d\sigma. \tag{4.3}$$

Many identities of this form can be obtained from the generalized Stokes theorem. For example, a natural choice is $\mathcal{L} = \Delta$, the Laplace operator (or Laplace-Beltrami operator when Ω is a Riemannian manifold), and $\mathcal{B} = \partial_\nu$, the outward normal derivative at the boundary. This pair, however, may lead to inaccurate quadrature formulas on domains with nonsmooth boundary, as will be explained in Sections 4.6

and 5.1. Instead, we recommend $\mathcal{L} = \mathrm{div}$, the divergence operator, and $\mathcal{B} = \gamma_\nu$, the normal trace operator, as a general purpose choice, which works well in all of our numerical tests on Lipschitz domains in 2D and 3D with either smooth or piecewise smooth boundary.

Given two functions $f : \Omega \to \mathbb{R}$ and $g : \partial\Omega \to \mathbb{R}$, we look for a *combined quadrature* to approximate the difference of two integrals

$$\int_\Omega f(x)\, d\mu(x) - \int_{\partial\Omega} g(x)\, d\sigma(x) \approx \sum_{i=1}^{N_Y} w_i f(y_i) - \sum_{i=1}^{N_Z} v_i g(z_i), \tag{4.4}$$

with knots $Y = \{y_1, \dots, y_{N_Y}\} \subset \overline{\Omega}$ and $Z = \{z_1, \dots, z_{N_Z}\} \subset \partial\Omega$ and weight vectors $w = (w_i)_{i=1}^{N_Y} \in \mathbb{R}^{N_Y}$ and $v = (v_i)_{i=1}^{N_Z} \in \mathbb{R}^{N_Z}$. Clearly, (4.4) provides in particular a quadrature formula with knots in Y and weight vector w for $\int_\Omega f\, d\mu$ if we set $g \equiv 0$, and a quadrature formula with knots in Z and weight vector v for $\int_{\partial\Omega} g\, d\sigma$ if we set $f \equiv 0$. Assuming that the nodes in Y and Z are provided by the user, or placed by a node generation algorithm [107], we propose a method for the computation of the weight vectors w and v.

As in the classical approach based on the reproduction of moments (4.2), we may employ a finite dimensional space of functions $\mathcal{S} = \mathrm{span}\{s_1, \dots, s_M\}$. However, we rely on the simultaneous approximation of f by $\mathcal{L}s$ and g by $\mathcal{B}s$, rather than on the direct approximation of f by $s \in \mathcal{S}$. Therefore, we require the exactness of the combined formula (4.4) for all pairs f, g obtained as $f = \mathcal{L}s$ and $g = \mathcal{B}s$ from the same $s \in \mathcal{S}$. According to (4.3), the left hand side of (4.4) is zero in this case, so that the exactness condition is equivalent to

$$\sum_{i=1}^{N_Y} w_i \sum_{j=1}^{M} c_j \mathcal{L}s_j(y_i) - \sum_{i=1}^{N_Z} v_i \sum_{j=1}^{M} c_j \mathcal{B}s_j(z_i) = 0 \quad \text{for all } c_j \in \mathbb{R}.$$

In addition, we require the exactness of (4.4) for a single pair of functions $\hat{f}, \hat{g}$ with known integrals $\int_\Omega \hat{f}\, d\mu$ and $\int_{\partial\Omega} \hat{g}\, d\sigma$, such that

$$\int_\Omega \hat{f}(x)\, d\mu(x) - \int_{\partial\Omega} \hat{g}(x)\, d\sigma(x) \neq 0. \tag{4.5}$$

In particular, we may use $\hat{f} \equiv 1$, $\hat{g} \equiv 0$ if we know the measure of Ω, or $\hat{f} \equiv 0$, $\hat{g} \equiv 1$ if we know the boundary measure of $\partial\Omega$. If neither is known, then we may employ $\hat{f} \equiv 0$, $\hat{g} = \partial_\nu \Phi$ with Φ being a fundamental solution of the Laplace(-Beltrami) equation centered in Ω, see Section 4.4. In any case, we get the following linear system for the weight vectors w and v,

$$\begin{cases} \displaystyle\sum_{i=1}^{N_Y} w_i \ell_{ij} - \sum_{i=1}^{N_Z} v_i b_{ij} = 0, \quad j = 1, \dots, M, \\ \displaystyle\sum_{i=1}^{N_Y} w_i \hat{f}(y_i) - \sum_{i=1}^{N_Z} v_i \hat{g}(z_i) = \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma, \end{cases} \tag{4.6}$$

where ℓ_{ij} and b_{ij} are the entries of the collocation matrices

$$L = \big(\mathcal{L}s_j(y_i)\big)_{i=1,j=1}^{N_Y,M}, \quad B = \big(\mathcal{B}s_j(y_i)\big)_{i=1,j=1}^{N_Z,M} \tag{4.7}$$

for the operators $\mathcal{L}, \mathcal{B}$ w.r.t. the sets Y, Z and the space $\mathcal{S}$. If Ω is a closed manifold, then $\partial\Omega = \emptyset$ and (4.6) simplifies in that everything related to the integral over $\partial\Omega$ disappears.

Moreover, we may move away from using any function space $\mathcal{S}$ and replace L and B by numerical differentiation matrices on irregular nodes, as those used in the meshless finite difference method, see Section 2.3, which leads to a purely meshless approach with full flexibility of arbitrarily distributed degrees of freedom associated with an additional finite set $X = \{x_1, \dots, x_{N_X}\}$ discretizing either $\overline{\Omega}$ or some larger set $D \supset \overline{\Omega}$. In the case of numerical differentiation we again compute (w, v) by solving the linear system (4.6), with the coefficients ℓ_{ij}, b_{ij} of matrices L, B obtained as the weights of suitable numerical differentiation formulas

$$\mathcal{L}u(y_i) \approx \sum_{k=1}^{N_X} \lambda_{ik}^T u(x_k), \quad i = 1, \dots, N_Y, \qquad \mathcal{B}u(z_i) \approx \sum_{k=1}^{N_X} \beta_{ik}^T u(x_k), \quad i = 1, \dots, N_Z, \tag{4.8}$$

where

$$L = \big(\ell_{ij}\big)_{i=1,j=1}^{N_Y,M} = \big(\lambda_{ik}^T\big)_{i=1,k=1}^{N_Y,N_X}, \quad B = \big(b_{ij}\big)_{i=1,j=1}^{N_Z,M} = \big(\beta_{ik}^T\big)_{i=1,k=1}^{N_Z,N_X}. \tag{4.9}$$

Note that λ_{ij} are scalars if u is scalar-valued, but must be vectors in more general settings, for example when $\mathcal{L}$ is the divergence operator on a manifold and u is a vector field.

We choose the functional space $\mathcal{S}$ or the set of nodes X such that the system (4.6) is underdetermined, and compute the combined quadrature weight vector (w, v) as its solution with the smallest 2-norm,

$$\begin{gathered} \|w\|_2^2 + \|v\|_2^2 \to \min \\ \text{subject to} \quad \begin{pmatrix} L^T & -B^T \\ \hat{f}|_Y^T & -\hat{g}|_Z^T \end{pmatrix} \begin{pmatrix} w \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \end{pmatrix}, \end{gathered} \tag{4.10}$$

where

$$\hat{f}_{|Y} = \big(\hat{f}(y_i)\big)_{i=1}^{N_Y}, \quad \hat{g}_{|Z} = \big(\hat{g}(z_i)\big)_{i=1}^{N_Z},$$

which is a linearly constrained least squares problem. Efficient algorithms for this problem are available and can handle very large node sets Y, Z if the constraints are given by a sparse matrix [10]. The sparsity of the matrices L and B is achieved by using a locally supported basis of $\mathcal{S}$, such as tensor-product B-splines, or by choosing small sets of influence in the numerical differentiation formulas (4.8).

The rest of the chapter is organized as follows. In Sections 4.1 and 4.2 we rigorously state the mathematical setting of our approach, which, although abstract, allow us to perform a very general error analysis of the new quadrature formulas. In Section 4.3 we describe a general framework for the development of moment-free quadrature formulas, followed by the details of the options available for each of its aspects, namely the choice of the functions $\hat{f}$ and $\hat{g}$ satisfying (4.5) in Section 4.4, the choice of differential operators $\mathcal{L}$ and $\mathcal{B}$ in Section 4.5, the choice of methods for numerical differentiation in Section 4.6 (either collocation or meshless finite differences), and the choice of norm to minimize when solving the underdetermined linear system (4.6) in Section 4.7. Finally, two algorithms are given in Section 4.8 with all the necessary details for practical implementation.

4.1 Quadrature arising from numerical differentiation

Let $\mathcal{M}$ be a C^0 manifold of dimension d, and $\Omega \subset \mathcal{M}$ a domain (that is, a connected and open subset) with compact closure $\overline{\Omega}$, and let μ, σ be finite, strictly positive Borel measures defined on $\overline{\Omega}$ and $\partial\Omega$, respectively. We suppose that $\mu(\partial\Omega) = 0$, so that the space $L^1(\Omega)$ of absolutely integrable functions can be identified with $L^1(\overline{\Omega})$.

For any given set Y of N_Y scattered nodes in $\overline{\Omega}$, and, in the case of nonempty boundary $\partial\Omega \neq \emptyset$, any given set Z of N_Z scattered nodes in $\partial\Omega$, we seek quadrature weights $w \in \mathbb{R}^{N_Y}$ and $v \in \mathbb{R}^{N_Z}$ such that

$$\int_\Omega f(x)\, d\mu(x) \approx \sum_{i=1}^{N_Y} w_i f(y_i) \quad \text{and} \quad \int_{\partial\Omega} g(x)\, d\sigma(x) \approx \sum_{i=1}^{N_Z} v_i g(z_i) \tag{4.11}$$

for all sufficiently regular integrands $f\colon \overline{\Omega} \to \mathbb{R}$ and $g\colon \partial\Omega \to \mathbb{R}$. To make the pointwise evaluation $f(y_i)$ and $g(z_i)$ a meaningful operation we assume that $f \in L^1_Y(\Omega)$ and $g \in L^1_Z(\partial\Omega)$, where $L^1_Y(\Omega)$ and $L^1_Z(\partial\Omega)$ are the subsets of $L^1(\Omega)$ and $L^1(\partial\Omega)$ whose elements (equivalence classes of functions) have at least one representative that is continuous at each point of Y and Z, respectively.

We denote the two quadrature formulas by (Y, w) and (Z, v), and compute them *simultaneously*. Moreover, they arise as a *combined quadrature*

$$Q(f, g) := \sum_{i=1}^{N_Y} w_i f(y_i) - \sum_{i=1}^{N_Z} v_i g(z_i)$$

for the difference[1] of the two integrals

$$I(f, g) := \int_\Omega f(x)\, d\mu(x) - \int_{\partial\Omega} g(x)\, d\sigma(x).$$

Let $\mathcal{L}\colon \mathcal{D} \to L^1(\Omega)$ and $\mathcal{B}\colon \mathcal{D} \to L^1(\partial\Omega)$ be operators such that

$$\int_\Omega \mathcal{L}u\, d\mu = \int_{\partial\Omega} \mathcal{B}u\, d\sigma \quad \text{for all } u \in \mathcal{D}, \tag{4.12}$$

with $\mathcal{D}$ being a common domain of definition for the two operators. In particular, we have in mind the situation where $\mathcal{D}$ is a linear space of functions on $\overline{\Omega}$ or on a set containing it, both $\mathcal{L}$ and $\mathcal{B}$ are linear differential operators, measures μ, σ are induced by a Riemannian metric in $\mathcal{M}$, and identity (4.12) holds by the divergence theorem or another corollary of the generalized Stokes theorem. When $\Omega = \mathcal{M}$ is a closed manifold with $\partial\Omega = \emptyset$, the measure σ and the operator $\mathcal{B}$ are clearly irrelevant, and the right-hand side in (4.12) is understood to be identically zero.

[1] Note that we could use the sum of the two integrals instead of their difference, and respectively the sum of the two quadrature formulas in the definition of the combined quadrature, which would look more natural in a sense. The difference, however, matches the usual form of the compatibility condition (4.12) and emphasizes the "cancellation" nature of the equation $I(f, g) = 0$ that plays a crucial role in the method.

The underlying idea of the method may be roughly explained as follows. The identity (4.12) implies that

$$I(f,g) = I(\mathcal{L}u, \mathcal{B}u) = 0$$

as soon as $f = \mathcal{L}u$ and $g = \mathcal{B}u$ for the same function $u \in \mathcal{D}$. If u is discretized by a vector $c \in \mathbb{R}^M$, and matrices $L \in \mathbb{R}^{N_Y \times M}$ and $B \in \mathbb{R}^{N_Z \times M}$ are such that

$$Lc \approx \mathcal{L}u|_Y = f|_Y, \quad Bc \approx \mathcal{B}u|_Z = g|_Z,$$

then

$$Q(f,g) \approx w^T Lc - v^T Bc = (w^T L - v^T B)\, c,$$

and hence requiring that the weight vectors w, v satisfy $w^T L - v^T B = 0$ implies

$$Q(f,g) \approx 0 = I(f,g) \quad \text{for all } (f,g) \text{ such that } f = \mathcal{L}u,\ g = \mathcal{B}u \text{ for some } u \in \mathcal{D}.$$

In addition, we require from w, v that

$$Q(\hat{f},\hat{g}) = I(\hat{f},\hat{g}) \quad \text{for a single pair } (\hat{f},\hat{g}) \text{ with } I(\hat{f},\hat{g}) \neq 0.$$

As we will demonstrate, this combination of conditions produces accurate quadrature formulas (Y, w) and (Z, v) for appropriate choices of the operators $\mathcal{L}, \mathcal{B}$ and numerical differentiation matrices L, B.

4.2 Error analysis in an abstract setting

For any given operators $\mathcal{L}$ and $\mathcal{B}$ and finite sets $Y \subset \overline{\Omega}$ and $Z \subset \partial\Omega$, we say that a triple (Ψ, L, B), where $\Psi\colon \mathcal{D} \to \mathbb{R}^M$, $L \in \mathbb{R}^{N_Y \times M}$ and $B \in \mathbb{R}^{N_Z \times M}$, is a *numerical differentiation scheme* with discretization operator Ψ and differentiation matrices L, B. We denote by $\varepsilon_L(u)$ and $\varepsilon_B(u)$ the recovery error with which the matrices L and B approximate the pointwise values of $\mathcal{L}u$ and $\mathcal{B}u$ using the information about $u \in \mathcal{D}$ contained in $\Psi(u)$:

$$\varepsilon_L(u) := \left\| \mathcal{L}u|_Y - L\Psi(u) \right\|_\infty = \max_{i=1,\dots,N_Y} \left| \mathcal{L}u(y_i) - \big(L\Psi(u)\big)_i \right|, \tag{4.13}$$

$$\varepsilon_B(u) := \left\| \mathcal{B}u|_Z - B\Psi(u) \right\|_\infty = \max_{i=1,\dots,N_Z} \left| \mathcal{B}u(z_i) - \big(B\Psi(u)\big)_i \right|. \tag{4.14}$$

The vector $\Psi(u) \in \mathbb{R}^M$ may correspond to pointwise values of u over a scattered set of nodes, coefficients of splines or radial basis functions that approximate u, Fourier coefficients, or other kinds of discrete information about u. Tools of approximation theory may then be used to define matrices L and B that accurately reconstruct the pointwise values of $\mathcal{L}u$ and $\mathcal{B}u$ from a given vector $\Psi(u)$.

Our approach is motivated by the following proposition that estimates the combined quadrature error

$$\delta(f,g) := I(f,g) - Q(f,g).$$

Proposition 4.2.1 (Error bound for combined quadrature)**.** *Let $(\hat{f}, \hat{g}) \in L^1_Y(\Omega) \times L^1_Z(\partial\Omega)$ be a fixed pair of functions such that*

$$I(\hat{f}, \hat{g}) \neq 0.$$

Given any pair $(f, g) \in L^1_Y(\Omega) \times L^1_Z(\partial\Omega)$, let $\mathcal{U}_{f,g} \subset \mathcal{D}$ be the set of solutions u to the boundary value problem

$$\begin{cases} \mathcal{L}u = f - \alpha\hat{f} & \text{in } \Omega \\ \mathcal{B}u = g - \alpha\hat{g} & \text{on } \partial\Omega, \end{cases} \tag{4.15}$$

where the coefficient $\alpha \in \mathbb{R}$ is chosen so that (4.15) *satisfies the compatibility condition* (4.12)*:*

$$\alpha = I(f, g)/I(\hat{f}, \hat{g}).$$

Then for any pair of quadrature formulas (4.11)*, the following estimate of the combined error holds:*

$$|\delta(f,g)| \leq |\alpha|\,\hat{\varepsilon} + \inf_{u \in \mathcal{U}_{f,g}} \Big\{ \|w\|_1\, \varepsilon_L(u) + \|v\|_1\, \varepsilon_B(u) + \|\Psi(u)\|_\infty \left\| L^T w - B^T v \right\|_1 \Big\}, \tag{4.16}$$

where $\hat{\varepsilon} := |\delta(\hat{f}, \hat{g})|$.

Proof. For all $u \in \mathcal{U}_{f,g}$, by the linearity of δ,

$$|\delta(f,g)| = \Big|\delta(f - \alpha\hat{f}, g - \alpha\hat{g}) + \delta(\alpha\hat{f}, \alpha\hat{g})\Big| = \Big|\delta(\mathcal{L}u, \mathcal{B}u) + \alpha\delta(\hat{f}, \hat{g})\Big| \leq |\delta(\mathcal{L}u, \mathcal{B}u)| + |\alpha|\,\hat{\varepsilon}.$$

Moreover, by the compatibility condition (4.12) and by the definitions of $\varepsilon_L(u)$ and $\varepsilon_B(u)$,

$$\begin{aligned} |\delta(\mathcal{L}u, \mathcal{B}u)| &= \Big| \sum_{i=1}^{N_Y} w_i(\mathcal{L}u)(y_i) - \sum_{i=1}^{N_Z} v_i(\mathcal{B}u)(z_i) \Big| \\ &= \Big| \sum_{i=1}^{N_Y} w_i(\mathcal{L}u)(y_i) \pm \sum_{i=1}^{N_Y} w_i\big(L\Psi(u)\big)_i \pm \sum_{i=1}^{N_Z} v_i\big(B\Psi(u)\big)_i - \sum_{i=1}^{N_Z} v_i(\mathcal{B}u)(z_i) \Big| \\ &\leq \|w\|_1\, \varepsilon_L(u) + \big| w^T L\Psi(u) - v^T B\Psi(u) \big| + \|v\|_1\, \varepsilon_B(u) \\ &\leq \|w\|_1\, \varepsilon_L(u) + \|v\|_1\, \varepsilon_B(u) + \|\Psi(u)\|_\infty \left\| L^T w - B^T v \right\|_1. \end{aligned}$$

Taking the infimum over $u \in \mathcal{U}_{f,g}$ completes the proof. □

By choosing either $f \equiv 0$ or $g \equiv 0$, we obtain separate error bounds for both quadrature formulas (4.11).

Corollary 4.2.2 (Separate error bounds)**.** *By the previous proposition, the special cases*

$$\delta(f, 0) = \int_\Omega f\, d\mu - \sum_{i=1}^{N_Y} w_i f(y_i), \qquad \delta(0, g) = -\int_{\partial\Omega} g\, d\sigma + \sum_{i=1}^{N_Z} v_i g(z_i),$$

immediately imply the following error bounds for quadrature formulas (Y, w) *and* (Z, v)*,*

$$\Big| \int_\Omega f \, d\mu - \sum_{i=1}^{N_Y} w_i f(y_i) \Big| \le |\alpha| \, \hat{\varepsilon} + \inf_{u \in \mathcal{U}_f} \Big\{ \|w\|_1 \, \varepsilon_L(u) + \|v\|_1 \, \varepsilon_B(u) + \|\Psi(u)\|_\infty \, \big\| L^T w - B^T v \big\|_1 \Big\},$$

$$\Big| \int_{\partial\Omega} g \, d\sigma - \sum_{i=1}^{N_Z} v_i g(z_i) \Big| \le |\beta| \, \hat{\varepsilon} + \inf_{u \in \mathcal{U}_g} \Big\{ \|w\|_1 \, \varepsilon_L(u) + \|v\|_1 \, \varepsilon_B(u) + \|\Psi(u)\|_\infty \, \big\| L^T w - B^T v \big\|_1 \Big\},$$

with $\mathcal{U}_f$ *and* $\mathcal{U}_g$ *being the sets of solutions to the boundary value problems*

$$\begin{cases} \mathcal{L}u = f - \alpha \hat{f} & \text{in } \Omega \\ \mathcal{B}u = -\alpha \hat{g} & \text{on } \partial\Omega, \end{cases} \qquad \text{and} \qquad \begin{cases} \mathcal{L}u = -\beta \hat{f} & \text{in } \Omega \\ \mathcal{B}u = g - \beta \hat{g} & \text{on } \partial\Omega, \end{cases} \tag{4.17}$$

respectively, where $\alpha = I(f, 0)/I(\hat{f}, \hat{g})$ *and* $\beta = I(0, g)/I(\hat{f}, \hat{g})$ *are again determined by the compatibility condition* (4.12)*.*

The case of a closed manifold simplifies to just one quadrature formula.

Corollary 4.2.3 (Manifold without boundary)**.** *If* $\partial\Omega = \emptyset$*, then there is only one quadrature formula* (Y, w)*, and the argumentation of Proposition 4.2.1 leads to the error estimate*

$$\Big| \int_\Omega f \, d\mu - \sum_{i=1}^{N_Y} w_i f(y_i) \Big| \le |\alpha| \, \hat{\varepsilon} + \inf_{u \in \mathcal{U}_f} \Big\{ \|w\|_1 \, \varepsilon_L(u) + \|\Psi(u)\|_\infty \, \big\| L^T w \big\|_1 \Big\},$$

where $\mathcal{U}_f$ *is the set of solutions to the equation* $\mathcal{L}u = f - \alpha \hat{f}$ *in* Ω*, and* $\alpha = \int_\Omega f \, d\mu / \int_\Omega \hat{f} \, d\mu$*, with* $\hat{f} \in L^1_Y(\Omega)$ *such that* $\int_\Omega \hat{f} \, d\mu \neq 0$*.*

4.3 A general framework

Motivated by Proposition 4.2.1, we compute quadrature weights w, v by minimizing the error $|\delta(f, g)|$ of the combined quadrature. It is clear from (4.16) that $|\delta(f, g)|$ is small if

$$|\alpha| \, \hat{\varepsilon}, \ \varepsilon_L(u), \ \varepsilon_B(u) \text{ and } \big\| L^T w - B^T v \big\|_1 \text{ are small,}$$

and

$$\|w\|_1 \, , \ \|v\|_1 \text{ and } \|\Psi(u)\|_\infty \text{ are bounded for } N_Y, N_Z \to \infty.$$

Moreover, if we choose the weight vectors w and v such that $L^T w - B^T v = 0$, then the size of $\|\Psi(u)\|_\infty$ is irrelevant. We may also require that $\hat{\varepsilon} = |\delta(\hat{f}, \hat{g})| = 0$, which is just another linear equation for w and v to satisfy. Assuming that $\varepsilon_L(u)$ and $\varepsilon_B(u)$ are small thanks to (a) the smoothness of u enforced for smooth $f, g, \hat{f}, \hat{g}$ by the appropriate choice of the operators $\mathcal{L}, \mathcal{B}$, and (b) the choice of the numerical differentiation scheme, we may use any remaining degrees of freedom in w and v in order to minimize the stability constants $\|w\|_1$ and $\|v\|_1$. We can minimize the joint 1-norm $\|w\|_1 + \|v\|_1$, or some other norm $\|(w, v)\|_\sharp$ that may be preferable for computational or other reasons. All these ideas lead us to the following general framework for quadrature formulas arising from differentiation.

Framework 4.3.1. Given Ω, μ, σ, Y, Z as introduced in Section 4.1, compute quadrature weights $w^* \in \mathbb{R}^{N_Y}$ and $v^* \in \mathbb{R}^{N_Z}$ as follows:

1. Choose auxiliary functions $\hat{f} \in L^1_Y(\Omega)$ and $\hat{g} \in L^1_Z(\partial\Omega)$ such that
$$\int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \neq 0.$$

2. Choose operators $\mathcal{L}$ and $\mathcal{B}$ such that
$$\int_\Omega \mathcal{L}u\, d\mu = \int_{\partial\Omega} \mathcal{B}u\, d\sigma \quad \text{for all } u \in \mathcal{D},$$
where the set $\mathcal{D}$ contains at least one solution of the boundary value problem (4.15) for each pair of functions $f \in L^1_Y(\Omega)$ and $g \in L^1_Z(\partial\Omega)$ whose integrals we wish to approximate using formulas (Y, w^*) and (Z, v^*).

3. Choose a numerical differentiation scheme (Ψ, L, B) for the linear reconstruction of the pointwise values $\mathcal{L}u_{|Y}$ and $\mathcal{B}u_{|Z}$ from the vector $\Psi(u)$ in the form of matrix-vector products $L\Psi(u)$ and $B\Psi(u)$.

4. Assemble the non-homogeneous linear system with unknowns (w, v)
$$L^T w - B^T v = 0, \qquad \sum_{i=1}^{N_Y} w_i \hat{f}(y_i) - \sum_{i=1}^{N_Z} v_i \hat{g}(z_i) = \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma, \quad (4.18)$$
which can be written more compactly as $Ax = b$, with
$$A = \begin{pmatrix} L^T & -B^T \\ \hat{f}|_Y^T & -\hat{g}|_Z^T \end{pmatrix}, \quad x = \begin{pmatrix} w \\ v \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \end{pmatrix}. \quad (4.19)$$

5. Compute a solution (w^*, v^*) of (4.18) that minimizes some norm
$$\|(w, v)\|_\sharp : \mathbb{R}^{N_Y} \times \mathbb{R}^{N_Z} \to \mathbb{R}^+$$
defined on the space of all possible weights (w, v).

The process described in the framework terminates successfully as long as the linear system (4.18) is consistent, in which case Corollary 4.2.2 implies that the quadrature formulas (Y, w^*) and (Z, v^*) satisfy the error bounds
$$\Big| \int_\Omega f\, d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i) \Big| \leq \inf_{u \in \mathcal{U}_f} \Big\{ \|w^*\|_1\, \varepsilon_L(u) + \|v^*\|_1\, \varepsilon_B(u) \Big\}, \quad (4.20)$$
$$\Big| \int_{\partial\Omega} g\, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \leq \inf_{u \in \mathcal{U}_g} \Big\{ \|w^*\|_1\, \varepsilon_L(u) + \|v^*\|_1\, \varepsilon_B(u) \Big\}. \quad (4.21)$$

Compare the error estimate (4.20) with the one obtained in the proof of Proposition 2.2.6: the former can be understood as the counterpart of the latter when exactness over a functional space $\mathcal{S}$ is replaced by (4.18).

The consistency of (4.18) may be characterized in terms of a certain discrete incompatibility condition as follows. Recall that $\hat{f}$ and $\hat{g}$ satisfy $I(\hat{f},\hat{g}) \neq 0$ and hence in view of (4.12) are incompatible as the right hand sides of the boundary value problem for the operators $\mathcal{L}$ and $\mathcal{B}$, that is, the problem

$$\begin{cases} \mathcal{L}u = \hat{f}, \\ \mathcal{B}u = \hat{g}, \end{cases} \tag{4.22}$$

is not solvable for any $u \in \mathcal{D}$.

Proposition 4.3.2. *The linear system $Ax = b$ given by* (4.19) *is consistent if and only if the following* discrete incompatibility condition *holds: the linear system*

$$\begin{cases} Lc = \hat{f}|_Y, \\ Bc = \hat{g}|_Z, \end{cases} \tag{4.23}$$

is inconsistent, i.e. has no solution $c \in \mathbb{R}^M$. Moreover, if the linear system (4.19) *is consistent, then its solution (w^*, v^*) that minimizes the norm $\|(w,v)\|_\sharp$ satisfies*

$$\|(w^*,v^*)\|_\sharp = |I(\hat{f},\hat{g})| \Big(\inf_{c\in\mathbb{R}^M} \left\| \begin{pmatrix} Lc - \hat{f}|_Y \\ Bc - \hat{g}|_Z \end{pmatrix} \right\|'_\sharp \Big)^{-1}, \tag{4.24}$$

where $\|\cdot\|'_\sharp$ denotes the norm on $\mathbb{R}^{N_Y+N_Z}$ dual to $\|\cdot\|_\sharp$.

Proof. The characterization essentially follows from the identity $\mathrm{Im}(A)^\perp = \mathrm{Ker}(A^T)$ and the usual assumptions on $(\hat{f},\hat{g})$. We can reason as follows:

$$\begin{aligned} b \in \mathrm{Im}(A) \quad &\Leftrightarrow \quad \begin{pmatrix} 0 \\ I(\hat{f},\hat{g}) \end{pmatrix} \in \mathrm{Ker}(A^T)^\perp \\ &\Leftrightarrow \quad \begin{pmatrix} 0 & I(\hat{f},\hat{g}) \end{pmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} = 0 \quad \text{for all } \begin{pmatrix} c \\ \lambda \end{pmatrix} \in \mathrm{Ker}(A^T) \\ &\Leftrightarrow \quad \lambda = 0 \quad \text{for all } \begin{pmatrix} c \\ \lambda \end{pmatrix} \in \mathrm{Ker}(A^T). \end{aligned}$$

The last statement in the chain is equivalent to saying that $\lambda = 0$ for all $c \in \mathbb{R}^M$ and $\lambda \in \mathbb{R}$ that satisfy

$$\begin{cases} Lc + \lambda \hat{f}|_Y = 0, \\ -Bc - \lambda \hat{g}|_Z = 0, \end{cases}$$

which is in turn equivalent to (4.23) being inconsistent.

To show (4.24) we assume that (4.19) is consistent. Then it follows from [26, Lemma 8] that

$$\begin{aligned} \inf\left\{ \|x\|_\sharp \mid Ax = b \right\} &= \sup\left\{ b^T \begin{pmatrix} c \\ \lambda \end{pmatrix} \mid c \in \mathbb{R}^M,\ \lambda \in \mathbb{R} \ \text{ with } \left\| A^T \begin{pmatrix} c \\ \lambda \end{pmatrix} \right\|'_\sharp \leq 1 \right\}, \\ &= \sup\left\{ \lambda I(\hat{f},\hat{g}) \mid c \in \mathbb{R}^M,\ \lambda \in \mathbb{R} \ \text{ with } \left\| \begin{pmatrix} Lc + \lambda \hat{f}|_Y \\ Bc + \lambda \hat{g}|_Z \end{pmatrix} \right\|'_\sharp \leq 1 \right\} \end{aligned}$$

$$= |I(\hat{f},\hat{g})| \sup\Big\{\lambda > 0 \mid \exists\, \tilde{c} \in \mathbb{R}^M \ \text{ with } \left\| \begin{pmatrix} L\tilde{c} - \hat{f}|_Y \\ B\tilde{c} - \hat{g}|_Z \end{pmatrix} \right\|_\sharp' \leq \frac{1}{\lambda}\Big\},$$

and (4.24) follows. □

Since (4.23) is a discretization of (4.22), we expect that the discrete incompatibility condition holds whenever the discretization obtained via the choice of Y, Z and the numerical differentiation scheme is sufficiently accurate for the linear system to inherit this important feature of the continuous problem.

Even though we cannot guarantee the consistency of (4.18) in general, in particular for arbitrary user-supplied quadrature nodes Y, Z, a simple strategy of choosing the size M of the discretization vector $\Psi(u)$ such that the linear system (4.18) is underdetermined with significantly fewer rows than columns makes the system solvable in all of our numerical experiments, leaving enough room for enforcing reasonable size of the stability constants $\|w\|_1$ and $\|v\|_1$ through minimization, see Chapter 5 for more details.

For completeness, we note that Framework 4.3.1 is not the only way to derive numerical integration schemes based on the error analysis in Proposition 4.2.1. For example, if a sufficiently accurate quadrature formula (Z, v) for the integral over $\partial\Omega \neq \emptyset$ is known and fixed in advance, then one could seek weights w^* by minimizing a norm $\|\cdot\|_\sharp : \mathbb{R}^{N_Y} \to \mathbb{R}_+$ over the solution space of the non-homogeneous equation $L^T w = B^T v$. Alternatively, without fixing v, one might relax the constraint $L^T w - B^T v = 0$ by instead minimizing the expression

$$\lambda_1 \|w\|_2^2 + \lambda_2 \|v\|_2^2 + \lambda_3 \left\|L^T w - B^T v\right\|_2^2,$$

where the coefficients $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+$ are chosen to balance the terms appropriately. The accuracy and stability analysis of quadrature formulas generated by these alternative methods lies beyond the scope of this thesis. Among the variants explored in our preliminary numerical experiments, however, Framework 4.3.1 appeared to achieve the best trade-off between accuracy, generality, and computational efficiency.

In the following sections we illustrate and compare several options for the choices to be made at each step of Framework 4.3.1.

4.4 Auxiliary functions $\hat{f}$ and $\hat{g}$

The non-homogeneous constraint

$$\sum_{i=1}^{N_Y} w_i \hat{f}(y_i) - \sum_{i=1}^{N_Z} v_i \hat{g}(z_i) = \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \neq 0 \tag{4.25}$$

plays a fundamental role in the definition of linear system (4.18), because it rules out the trivial solution $(w^*, v^*) = (0, 0)$. In practice, choices for $\hat{f}$ and $\hat{g}$ need not be complicated:

$$(\hat{f},\hat{g}) \equiv (1,0) \quad \text{or} \quad (\hat{f},\hat{g}) \equiv (0,1) \tag{4.26}$$

are usually effective, as long as at least one of the moments

$$|\Omega| := \int_\Omega 1 \, d\mu, \qquad |\partial\Omega| := \int_{\partial\Omega} 1 \, d\sigma$$

is either known in advance, or can be numerically approximated in a straightforward way. In particular, this can be easily done using standard quadrature formulas when the boundary $\partial\Omega$ is defined by explicit parametric patches over simple regions in the parameter domain.

As a fully *moment-free* alternative, however, we suggest the following approach that only requires knowing sufficiently many points on the boundary $\partial\Omega$ to generate the set Z of quadrature nodes, as well as the outward-pointing normals on $\partial\Omega$ at Z. This applies for instance to implicit surfaces as used in the level-set method, and to trimmed multipatch surfaces as used in Computer Aided Design.

Assume that Ω is a domain in $\mathbb{R}^d$ and μ and σ are the standard Lebesgue and hypersurface measures. Let $\Phi(x, x_0)$ be the fundamental solution of the Laplace equation centered at a point $x_0 \in \Omega$:

$$\Phi(x,x_0) = \frac{1}{2\pi}\log\big(\|x-x_0\|\big) \quad \text{if } d=2, \qquad \Phi(x,x_0) = -\frac{\|x-x_0\|^{2-d}}{d(d-2)\omega_d} \quad \text{if } d=1 \text{ or } d>2,$$

with ω_d being the measure of the d-dimensional unit ball. Let ν be the outward-pointing unit normal field along $\partial\Omega$, and let ∂_ν be the normal derivative operator on $\partial\Omega$. By Green's third identity,

$$\int_{\partial\Omega} \partial_\nu \Phi(x,x_0)\, d\sigma(x) = 1,$$

and so the choice

$$\hat f(x) = 0 \quad \text{and} \quad \hat g(x) = \partial_\nu \Phi(x,x_0) = \frac{\nu(x)^T(x-x_0)}{d\omega_d \, \|x-x_0\|^d} \tag{4.27}$$

satisfies

$$\int_\Omega \hat f \, d\mu - \int_{\partial\Omega} \hat g \, d\sigma = -1$$

independently of the shape of the domain and the position of $x_0 \in \Omega$. Observe that the singularity of $\Phi(x, x_0)$ in Ω plays no role in the solution of the boundary value problems (4.17), because $\hat f \equiv 0$. In practice, care must be taken to avoid points x_0 too close to the boundary, or else the smoothness of the solution u of (4.15) may be affected by the near-singularity of $\hat g$, and as a consequence the quantities $\varepsilon_L(u)$, $\varepsilon_B(u)$ in the quadrature errors estimates may blow up.

Closed-form expressions for the fundamental solutions of the Laplace equation are known on some manifolds, too, see for example the list in [7]. This approach can be useful, for example, to compute a quadrature formula on a subdomain of the d-dimensional sphere with unknown $|\Omega|$ and $|\partial\Omega|$.

4.5 Operators $\mathcal{L}$ and $\mathcal{B}$

The error bounds in Section 4.2 depend on the accurate reconstruction of pointwise values of $\mathcal{L}u$ and $\mathcal{B}u$ from a given vector $\Psi(u)$, with u being a solution to a boundary value problem of the form

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega, \\ \mathcal{B}u = g & \text{on } \partial\Omega, \end{cases} \qquad \text{with} \quad \int_\Omega f \, d\mu = \int_{\partial\Omega} g \, d\sigma. \tag{4.28}$$

No matter which numerical differentiation method is used to define matrices L and B, the accuracy of the reconstruction, and thus the quantities $\varepsilon_L(u)$ and $\varepsilon_B(u)$ in (4.16), depend crucially on the regularity of u as measured by e.g. Sobolev norms. On the contrary, the accuracy of the quadrature formulas should rather depend on the regularity of the integrands f and g, as is natural in numerical integration. Therefore, we are led to consider operators $\mathcal{L}$ and $\mathcal{B}$ for which the boundary value problem (4.28) admits solutions of sufficiently high regularity, depending on the smoothness of f and g.

The elliptic approach

Consider the boundary value problem for Poisson's equation with Neumann boundary conditions posed on a bounded domain Ω in $\mathbb{R}^d$ or in a smooth manifold $\mathcal{M}$:

$$\begin{cases} \Delta u = f & \text{in } \Omega, \\ \partial_\nu u = g & \text{on } \partial\Omega, \end{cases} \tag{4.29}$$

with Δ and ∂_ν being the Laplace-Beltrami and normal derivative trace operators defined in Section 2.1. The result on existence and regularity of solutions to (4.29) stated in Proposition 2.1.4 makes the choice

$$\mathcal{L} = \Delta, \quad \mathcal{B} = \partial_\nu \tag{4.30}$$

fully satisfactory for a smooth domain Ω: solutions to the problem are guaranteed to exist, and to be more regular than the corresponding data f, g in the right-hand side.

However, when $\partial\Omega$ is not smooth, as is often the case in practical applications, comparable elliptic regularity results do not hold anymore, as clearly shown by Example 2.1.5. In this case, the solution u to the boundary value problem may not be regular enough for the numerical differentiation scheme

$$\Delta u_{|Y} \approx L\Psi(u), \quad \partial_\nu u_{|Z} \approx B\Psi(u)$$

to be sufficiently accurate even for highly smooth f and g, which indicates that the choice (4.30) may be problematic because the numerical differentiation errors $\varepsilon_L(u)$ and $\varepsilon_B(u)$ featuring in the estimate (4.16) are presumably affected by the loss of smoothness of u.

In principle, the estimate (4.16) that relates the error of our quadrature to the numerical differentiation errors $\varepsilon_L(u)$ and $\varepsilon_B(u)$ is just an upper bound, with no guarantee of being tight. Moreover, on Lipschitz domains in $\mathbb{R}^2$ with piecewise

smooth boundary, it can be proved that u still has regularity H^{s+2} away from corners, so one could hope that a few inaccurate approximations of $\Delta u(y_i)$ and $\partial_\nu u(z_i)$ around corners would not affect the overall accuracy of the quadrature formulas. However, our numerical experiments in Chapter 5 demonstrate that a catastrophic loss of accuracy in the quadrature formulas may occur on a Lipschitz domain with piecewise smooth boundary, whereas we do not register this behavior for the alternative divergence approach suggested below.

The divergence approach

The most basic instance of the identity (4.12) is clearly the divergence theorem, which suggests the choice

$$\mathcal{L} = \operatorname{div}, \quad \mathcal{B} = \gamma_\nu, \tag{4.31}$$

where div is the divergence operator on the manifold and γ_ν is the normal trace operator defined in Section 2.1. This leads us to the boundary value problem

$$\begin{cases} \operatorname{div} F = f & \text{in } \Omega, \\ \gamma_\nu(F) = g & \text{on } \partial\Omega, \end{cases} \quad \text{with } \int_\Omega f \, d\mu = \int_{\partial\Omega} g \, d\sigma, \tag{4.32}$$

where the compatibility condition on the right must be satisfied due to the divergence theorem, and the solution F is sought in an appropriate space $\mathcal{D}$ of vector fields on Ω.

Any solution u of the Neumann problem (4.29) gives rise to a solution $F = \nabla u$ of (4.32). Hence, no smooth solutions are ever lost by considering operators div and γ_ν instead of Δ and ∂_ν. However, unlike (4.29), the boundary value problem (4.32) is underdetermined, with highly non-unique solutions F. Recall that the error bound (4.16) involves the infimum over all solutions of (4.15), so the non-uniqueness is an advantage as (4.16) may rely on the solution F with the smallest numerical differentiation errors $\varepsilon_L(F)$ and $\varepsilon_B(F)$. In particular, one may hope that on Lipschitz domains with piecewise smooth boundary there still exists a solution F of (4.32) that inherits a high order of smoothness from smooth functions f, g in the right hand side, even when the same is not true about (4.29).

Although we have not found results of this type in the literature, considerable attention has been paid to the corresponding Dirichlet problem, where the full trace of F on the boundary is prescribed instead of the normal trace $\nu^T F$ in (4.32). The Dirichlet problem, however, requires additional compatibility conditions between f and the boundary data when Ω is not sufficiently smooth. In particular, if F vanishes on the boundary of a polygon $\Omega \subset \mathbb{R}^2$, then its divergence f must vanish at the vertices of Ω, which is a quite unnatural condition to impose on integrands f. Nevertheless, several results on the Dirichlet problem for domains in $\mathbb{R}^d$ imply that a solution F of (4.32) belongs to $H^{s+1}(\Omega)^d$ when $f \in H^s(\Omega)$ and g satisfies certain smoothness and compatibility assumptions. In particular, this follows from [99, Lemma II.2.3.1] for Lipschitz domains in $\mathbb{R}^d$, with $s \in \mathbb{N}$, when $g = 0$ and f vanishes on the boundary $\partial\Omega$ to order $s - 1$. Furthermore, [3] gives the same regularity result for polygons in $\mathbb{R}^2$ when $g = 0$ and f vanishes at all vertices of the polygon, and also for non-homogeneous g, with smoothness and compatibility conditions that are

difficult to recast to our setting, in which g is the normal projection of the Dirichlet boundary data studied in [3]. These results are generalized in [78] to arbitrary bounded Lipschitz domains in smooth manifolds, see in particular [78, Corollary 1.4] for the case $\Omega \subset \mathbb{R}^d$.

Our numerical results in Chapter 5 for the divergence-based quadrature do not indicate any loss of convergence order on non-smooth domains, which suggests that the smoothness of F is not lost. Moreover, the following elementary construction produces a smooth solution F of (4.32) on $\Omega = (0,1)^2$, whenever f is smooth in Ω and g is smooth on each side of the square, in contrast to Example 2.1.5 for the elliptic Neumann problem.

Smooth solution of divergence boundary problem on the square

As before, let $\Gamma_1, \dots, \Gamma_4$ denote the left, bottom, right, and top sides of $\partial\Omega$, and let g_i for $i = 1, \dots, 4$ be the univariate functions representing $g_{|\Gamma_i}$ in the natural Cartesian parametrizations. We set $\tilde{g}_i := g_i - \alpha_i$, with $\alpha_i = \int_0^1 g_i(t)\,dt$, and consider the vector fields

$$G_1(x,y) = \begin{pmatrix} (x-1)\tilde{g}_1(y) \\ -\int_0^y \tilde{g}_1(t)\,dt \end{pmatrix}, \quad G_2(x,y) = \begin{pmatrix} -\int_0^x \tilde{g}_2(t)\,dt \\ (y-1)\tilde{g}_2(x) \end{pmatrix},$$

$$G_3(x,y) = \begin{pmatrix} x\tilde{g}_3(y) \\ -\int_0^y \tilde{g}_3(t)\,dt \end{pmatrix}, \quad G_4(x,y) = \begin{pmatrix} -\int_0^x \tilde{g}_4(t)\,dt \\ y\tilde{g}_4(x) \end{pmatrix}$$

that satisfy

$$\operatorname{div} G_i = 0, \qquad \nu^T G_i|_{\partial\Omega} = \begin{cases} \tilde{g}_i & \text{in } \Gamma_i, \\ 0 & \text{otherwise.} \end{cases}$$

By construction, it follows that

$$\tilde{G}(x,y) := G_1(x,y) + G_2(x,y) + G_3(x,y) + G_4(x,y) + \begin{pmatrix} \alpha_1(x-1) + \alpha_3 x \\ \alpha_2(y-1) + \alpha_4 y \end{pmatrix}$$

is a solution of the problem

$$\operatorname{div} \tilde{G} = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4, \quad \nu^T \tilde{G}|_{\partial\Omega} = g.$$

Moreover, let $\alpha = \int_\Omega f(x,y)\,dx\,dy$, $h(y) = \int_0^1 f(\xi, y)\,d\xi - \alpha$ and $\tilde{f}(x,y) = f(x,y) - \alpha - h(y)$. Then the vector field

$$\tilde{F}(x,y) = \begin{pmatrix} \int_0^x \tilde{f}(\xi, y)\,d\xi \\ \int_0^y h(\zeta)\,d\zeta \end{pmatrix}$$

satisfies

$$\operatorname{div} \tilde{F} = f - \alpha, \qquad \nu^T \tilde{F}|_{\partial\Omega} = 0.$$

Since $\alpha = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ by the compatibility condition, we conclude that $F = \tilde{F} + \tilde{G}$ solves (4.32). It is easy to see that for any $k \geq 0$ we have $F \in C^k(\overline{\Omega})^2$ as soon as $f \in C^k(\overline{\Omega})$ and $g_i \in C^k([0,1])$ for all $i = 1, \dots, 4$, which even allows g to be discontinuous at the corners of $\partial\Omega$.

4.6 Numerical differentiation schemes

The choice of the numerical differentiation scheme (Ψ, L, B) is crucial for the performance of the method, because the factors $\varepsilon_L(u)$ and $\varepsilon_B(u)$ in the error bounds (4.20) and (4.21), as well as the solvability of the linear system (4.19) with reasonable stability constants $\|w\|_1$ and $\|v\|_1$ depend on this choice. In particular, the discrete incompatibility condition (4.23) must be satisfied. Moreover, for an efficient numerical implementation it is important that the differentiation matrices

$$L = \big(\ell_{ij}\big)_{i=1,j=1}^{N_Y,M} \quad \text{and} \quad B = \big(b_{ij}\big)_{i=1,j=1}^{N_Z,M}$$

are sparse.

We describe two approaches that work well in our numerical tests. The first is based on meshless numerical differentiation formulas, as used in the generalized finite difference methods, and the second employs tensor-product splines.

In both cases we assume that $\mathcal{L}$ and $\mathcal{B}$ are linear differential operators of integer order, well-defined for functions in a space $\mathcal{D}$ of functions on the closure of a domain D such that $\Omega \subset D$. The domains Ω and D need not have the same dimension; indeed, when $\mathcal{M}$ is embedded into an ambient space $\mathcal{A}$ such as a surface $\mathcal{M}$ in $\mathcal{A} = \mathbb{R}^3$, one may prefer to choose a domain D in $\mathcal{A}$ rather than in $\mathcal{M}$, leading to an *immersed* approach.

For the sake of notational simplicity, but without loss of generality, we assume in this chapter that the functions in $\mathcal{D}$ are scalar-valued.

4.6.1 Meshless finite difference formulas

Assuming that $\mathcal{D} \subset C^0(\overline{D})$, we choose a finite set $X = \{x_1, \dots, x_M\} \subset D$, and define the discretization operator $\Psi\colon \mathcal{D} \to \mathbb{R}^M$ by pointwise evaluation over X,

$$\Psi(u) = \big(u(x_1), \dots, u(x_M)\big)^T, \tag{4.33}$$

and approximate $\mathcal{L}u(y_i)$ and $\mathcal{B}u(z_i)$ at the quadrature nodes $y_i \in Y$ and $z_i \in Z$ by appropriate numerical differentiation formulas, as introduced in Section 2.3,

$$\mathcal{L}u(y_i) \approx \sum_{j=1}^{M} \ell_{ij}\, u(x_j), \qquad \mathcal{B}u(z_i) \approx \sum_{j=1}^{M} b_{ij}\, u(x_j). \tag{4.34}$$

Since integer order differentiation is a local operation, sufficiently accurate formulas can be found with the sums in (4.34) restricted to small sets of indices $S_{L,i}$ and $S_{B,i}$ corresponding to the subset of nodes in X close to y_i and z_i, known as *sets of influence*. This in turn implies that $\ell_{ij} = 0$ for $j \notin S_{L,i}$, $b_{ij} = 0$ for $j \notin S_{B,i}$, and so the matrices L and B are sparse.

When X is a gridded set and $Y \cup Z \subset X$, formulas of this type can be obtained by classical finite differences as those used in the finite difference method for partial differential equations. For irregular sets X, Y, Z, as typically needed on more complicated domains Ω, numerical differentiation formulas are in the core of the meshless finite difference methods, such as RBF-FD or GFDM, see for example

[35, 6, 55, 108, 22] and references therein. Therefore, we generally refer to (4.34) as *meshless finite difference formulas*.

Let Ω be a domain in $\mathbb{R}^d$, and $\Omega \subset D \subset \mathbb{R}^d$. Denote by k_L and k_B the orders of the operators $\mathcal{L}$ and $\mathcal{B}$, respectively. Suppose we use polyharmonic numerical differentiation formulas with $q = q_L > k_L$ for the operator $\mathcal{L}$, and $q = q_B > k_B$ for $\mathcal{B}$, as defined in Section 2.3. Under appropriate assumptions on the sets of influence

$$X_{L,i} := \{x_j \mid j \in S_{L,i}\} \quad \text{and} \quad X_{B,i} := \{x_j \mid j \in S_{B,i}\},$$

such as quasi-uniformity and boundedness of the polynomial norming constants, the errors (4.13) and (4.14) of the polyharmonic numerical differentiation can be estimated as

$$\varepsilon_L(u) = \big\|\mathcal{L}u_{|Y} - L\Psi(u)\big\|_\infty = \max_{1\le i\le N_Y} \Big|\mathcal{L}u(y_i) - \sum_{j\in S_{L,i}} \ell_{ij}\, u(x_j)\Big| \le C_L h_L^{q_L-k_L} |u|_{W_\infty^{q_L}(D)},$$

$$\varepsilon_B(u) = \big\|\mathcal{B}u_{|Z} - B\Psi(u)\big\|_\infty = \max_{1\le i\le N_Z} \Big|\mathcal{B}u(z_i) - \sum_{j\in S_{B,i}} b_{ij}\, u(x_j)\Big| \le C_B h_B^{q_B-k_B} |u|_{W_\infty^{q_B}(D)},$$

where variable h_L is the maximum diameter of the sets $\{y_i\} \cup X_{L,i}$ for $i = 1, \dots, N_Y$, variable h_B is the maximum diameter of the sets $\{z_i\} \cup X_{B,i}$ for $i = 1, \dots, N_Z$, and variables C_L, C_B are positive constants independent of h_L, h_B, and u. Note that a simple method for the selection of the sets of influence that ensures in practice that the errors behave in accordance with these estimates is to compose $X_{L,i}$ of $2 \dim \Pi_{q_L}^d$ closest nodes of y_i in X, and $X_{B,i}$ of $2 \dim \Pi_{q_B}^d$ closest nodes of z_i, as recommended in [6] for RBF-FD. Further methods that try to optimize the selection can be found in [23] and references therein. In any case, the number of nodes in $X_{L,i}$ and $X_{B,i}$ remains bounded if q_L and q_B are fixed, which in turn implies that the matrices L and B are sparse and that the diameters h_L and h_B shrink as the density of the nodes increases.

Assuming that $k_B = k_L - 1$, as in both the elliptic and the divergence settings of Section 4.5, we get the same approximation order for $\varepsilon_L(u)$ and $\varepsilon_B(u)$ by choosing $q_B = q_L - 1$. Hence, by (4.20) and (4.21), the quadrature formulas computed by Framework 4.3.1 satisfy the estimates

$$\Big| \int_\Omega f\, d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i) \Big| \le C \big(\|w^*\|_1 + \|v^*\|_1 \big) h^{q-k} \inf_{u\in\mathcal{U}_f} \max \big\{ |u|_{W_\infty^{q-1}(D)}, |u|_{W_\infty^q(D)} \big\}, \tag{4.35}$$

$$\Big| \int_{\partial\Omega} g\, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \le C \big(\|w^*\|_1 + \|v^*\|_1 \big) h^{q-k} \inf_{u\in\mathcal{U}_g} \max \big\{ |u|_{W_\infty^{q-1}(D)}, |u|_{W_\infty^q(D)} \big\}, \tag{4.36}$$

where

$$C = \max\{C_L, C_B\}, \quad h = \max\{h_L, h_B\}, \quad q = q_L = q_B + 1, \quad k = k_L = k_B + 1.$$

These estimates show that, as soon as the stability constants $\|w^*\|_1$ and $\|v^*\|_1$ remain bounded for $h \to 0$ and the sets $\mathcal{U}_f$ and $\mathcal{U}_g$ contain functions $u \in W_\infty^q(D)$,

the errors of the two quadrature formulas behave as $\mathcal{O}(h^{q-k})$. In order to relate the convergence order to the smoothness of the integrands f and g, we need regularity results for the boundary value problems (4.17), such as those discussed in Section 4.5.

For example, in the setting of a smooth domain $\Omega \subset \mathbb{R}^d$ and operators $\mathcal{L} = \Delta$, $\mathcal{B} = \partial_\nu$ (with $k = 2$), assuming that $\hat{f}$ and $\hat{g}$ are infinitely differentiable, we may use Proposition 2.1.4 to infer that there exists $u_f \in U_f$ that belongs to $H^{s+2}(\Omega)$ as soon as $f \in H^s(\Omega)$. By the Sobolev embedding theorem, $H^{s+2}(\Omega) \subset W^q_\infty(\Omega)$ if $s + 2 > q + d/2$, and by the Stein extension theorem u_f may be extended to a function in $W^q_\infty(D)$. Therefore, the estimate

$$\Big| \int_\Omega f \, d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i) \Big| \leq C_1 \big(\|w^*\|_1 + \|v^*\|_1 \big) h^{q-2} \|f\|_{H^s(\Omega)} \tag{4.37}$$

holds for all functions $f \in H^s(\Omega)$ with $s > q - 2 + d/2$, where C_1 is independent of f. Similarly, we derive from Proposition 2.1.4 that there exists $u_g \in U_g$ that belongs to $H^{s+3/2}(\Omega)$ as soon as $g \in H^s(\partial\Omega)$. By the same arguments with the Sobolev and Stein theorems, u_g may be extended to a function in $W^q_\infty(D)$ if $s + 3/2 > q + d/2$, and we obtain the estimate

$$\Big| \int_{\partial\Omega} g \, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \leq C_2 \big(\|w^*\|_1 + \|v^*\|_1 \big) h^{q-2} \|g\|_{H^s(\partial\Omega)} \tag{4.38}$$

for all functions $g \in H^s(\partial\Omega)$ with $s > q - 2 + (d+1)/2$, where C_2 is independent of g. Note that both estimates (4.37) and (4.38) are supposed to be suboptimal as they possibly require a higher smoothness of f and g than strictly needed for the $\mathcal{O}(h^{q-2})$ convergence order of the quadrature, since the natural assumption for this should be $f \in H^{q-2}(\Omega)$ and $g \in H^{q-2}(\partial\Omega)$, respectively, see [98, Chapter 4].

Choosing discretization nodes

In practical applications, quadrature formulas often need to be defined on fixed, user-supplied nodes, so in Framework 4.3.1 we do not assume to be in control of the placement of quadrature nodes Y and Z. The set $X \subset \overline{D}$ of discretization nodes that determines the operator Ψ, however, can be chosen freely, and so it must be computed as part of the algorithm.

First of all, we have to choose D, which may coincide with Ω, but it may also be a larger domain in $\mathcal{M}$ or in the ambient space $\mathcal{A}$, for example a bounding box around Ω. While exploring these options numerically, we have seen that using $D = \Omega$ and irregular X was consistently a better choice than a bounding box D with a Cartesian grid X, see the numerical results in Section 5.1.1.

The generation of X should take into account multiple criteria that in part contradict one another. On the one hand, for any fixed quadrature nodes Y and Z, we want the set X to be as large as possible, in order to improve the errors $\varepsilon_L(u)$ and $\varepsilon_B(u)$ of the numerical differentiation thanks to a higher density of X in $\overline{\Omega}$. On the other hand, a smaller set X produces more stable quadrature formulas. Indeed, let us consider two nested sets of nodes $X \subset \tilde{X}$, and corresponding matrices $L, B, \tilde{L}, \tilde{B}$

so that L, B are submatrices of $\tilde{L}, \tilde{B}$. Then, the solution $(\tilde{w}^*, \tilde{v}^*)$ to (4.18) with $\tilde{L}, \tilde{B}$ that minimizes the 1-norm $\|w\|_1 + \|v\|_1$ satisfies not just

$$\tilde{L}^T\tilde{w}^* - \tilde{B}^T\tilde{v}^* = 0, \quad \text{but also} \quad L^T\tilde{w}^* - B^T\tilde{v}^* = 0,$$

and so $\|w^*\|_1 + \|v^*\|_1 \le \|\tilde{w}^*\|_1 + \|\tilde{v}^*\|_1$. Moreover, as soon as X is large enough so that $M + 1 > N_Y + N_Z$, the linear system (4.18) is overdetermined, i.e. it has more equations than unknowns, and so it may not admit solutions anymore. In general, X should not be unnecessarily irregular, although a higher density of the nodes may be advantageous near the boundary of Ω, especially in the vicinity of its corners or fine features. In the numerical tests we only consider quasi-uniform nodes. For more details on the distribution of discretization nodes X and how the size N_X should be chosen, see Section 5.1.1.

4.6.2 Numerical differentiation by tensor-product splines

Let $\mathcal{S}$ be an M-dimensional linear subspace of $\mathcal{D}$ spanned by a basis $\{s_1, \dots, s_M\} \subset \mathcal{D}$. Any map

$$\Psi\colon \mathcal{D} \to \mathbb{R}^M, \quad \Psi(u) = \big(c_1(u), \dots, c_M(u)\big)^T,$$

can be seen as a discretization operator that gives rise to the numerical differentiation scheme

$$\mathcal{L}u(y_i) \approx \mathcal{L}\Big(\sum_{j=1}^{M} c_j(u)s_j\Big)(y_i) = \sum_{j=1}^{M} c_j(u)\mathcal{L}s_j(y_i) = (L\Psi(u))_i, \quad i = 1, \dots, N_Y$$

$$\mathcal{B}u(z_i) \approx \mathcal{B}\Big(\sum_{j=1}^{M} c_j(u)s_j\Big)(z_i) = \sum_{j=1}^{M} c_j(u)\mathcal{B}s_j(z_i) = (B\Psi(u))_i, \quad i = 1, \dots, N_Z,$$

given by the matrices $L \in \mathbb{R}^{N_Y \times M}$ and $B \in \mathbb{R}^{N_Z \times M}$ with entries

$$\ell_{ij} = \mathcal{L}s_j(y_i) \quad \text{and} \quad b_{ij} = \mathcal{B}s_j(z_i). \tag{4.39}$$

Since these matrices do not depend on Ψ, the weights w^*, v^* computed in Framework 4.3.1 do not depend on it either, and hence the choice of Ψ is irrelevant for the implementation of our quadrature in this setting. However, numerical differentiation errors

$$\varepsilon_L(u) = \max_{i=1,\dots,N_Y} \Big|\mathcal{L}u(y_i) - \mathcal{L}\Big(\sum_{j=1}^{M} c_j(u)s_j\Big)(y_i)\Big|$$

$$\varepsilon_B(u) = \max_{i=1,\dots,N_Z} \Big|\mathcal{B}u(z_i) - \mathcal{B}\Big(\sum_{j=1}^{M} c_j(u)s_j\Big)(z_i)\Big|$$

of (4.13)–(4.14) do depend on Ψ. Therefore we may take the infimum of (4.20) and (4.21) over all possible Ψ, which is equivalent to the infimum over all $s = \sum_{j=1}^{M} c_j(u)s_j \in \mathcal{S}$, and implies the estimates

$$\Big|\int_\Omega f\,d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i)\Big| \le \inf_{u \in \mathcal{U}_f} \inf_{s \in \mathcal{S}} \Big\{\|w^*\|_1 \,\|\mathcal{L}(u-s)|_Y\|_\infty + \|v^*\|_1 \,\|\mathcal{B}(u-s)|_Z\|_\infty\Big\}, \tag{4.40}$$

$$\Big| \int_{\partial\Omega} g \, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \leq \inf_{u \in \mathcal{U}_g} \inf_{s \in \mathcal{S}} \Big\{ \|w^*\|_1 \, \|\mathcal{L}(u-s)|_Y\|_\infty + \|v^*\|_1 \, \|\mathcal{B}(u-s)|_Z\|_\infty \Big\}. \tag{4.41}$$

Note that the weights w^* and v^* do not depend on the choice of the basis $\{s_1, \dots, s_M\}$ of a given space $\mathcal{S}$. Indeed, if we choose another basis and let V be the change-of-basis matrix, then the new differentiation matrices are given by $\tilde{L} = LV$ and $\tilde{B} = BV$, hence the condition $\tilde{L}^T w - \tilde{B}^T v = V^T(L^T w - B^T v) = 0$ is equivalent to $L^T w - B^T v = 0$ because V is invertible. Nevertheless, the matrices L and B still depend on the choice of the basis, and their properties strongly influence the efficiency and stability of the computation of the weights. In particular, in order to obtain sparse matrices L and B, we need locally supported basis functions s_j, such that only a few of them do not vanish in the vicinity of each point y_i or z_i.

Although the space $\mathcal{S}$ can be chosen freely, and in principle it would be interesting to compare different approaches, in this work we only consider the case of unfitted tensor-product spline approximation. Suppose that the manifold $\mathcal{M}$ of dimension d is embedded in $\mathbb{R}^n$, with $n \geq d$, and let H be the n-dimensional bounding box around $\overline{\Omega}$ such that the lengths of its sides are integer multiples of a parameter $h_{\mathcal{S}} > 0$:

$$\overline{\Omega} \subset H = [a_1, b_1] \times \cdots \times [a_n, b_n], \qquad b_i - a_i = N_i \, h_{\mathcal{S}} \quad \text{for all } i = 1, \dots, n.$$

For each dimension, let T_i be the uniform knot vector

$$T_i = \{a_i, \dots, a_i, a_i + h_{\mathcal{S}}, \dots, b_i - h_{\mathcal{S}}, b_i, \dots, b_i\}, \tag{4.42}$$

where the first and last knots are repeated q times, and let $\mathcal{S}_i$ be the univariate spline space of order q (degree $q-1$) defined by the knot vector T_i. We define $\mathcal{S}$ to be the restriction of the tensor-product spline space $\mathcal{S}_1 \otimes \cdots \otimes \mathcal{S}_n$ to $\overline{\Omega}$, and a natural basis for $\mathcal{S}$ is given by the restriction to $\overline{\Omega}$ of those tensor-product B-splines $s_1, \dots, s_M$, whose supports have non-empty intersection with Ω. In the case $d < n$, this is known as the *ambient*, or *immersed* approach [66]. Clearly, an upper bound for $M = \dim \mathcal{S}$ is given by

$$\prod_{i=1}^{n} \dim \mathcal{S}_i = \prod_{i=1}^{n} \big(N_i + q - 1\big),$$

although M will often only be a fraction of this upper bound, for example when $n > d$, or when $n = d$ but $|\Omega| \ll |H|$. We denote by D the interior of the union of the supports of all B-splines in $\mathcal{S}$. Then $\overline{D}$ may also be considered as the domain of definition of the splines in $\mathcal{S}$.

B-splines s_j have local supports contained in cubes with side length $qh_{\mathcal{S}}$, and so the matrices L and B are sparse as each of their rows has at most q^n nonzeros. Note that the assembly of L and B does not require testing whether the intersection of the support of s_j with Ω is non-empty: we can simply discard any basis function such that $s_j|_{Y \cup Z} \equiv 0$, as done in Algorithm 4.2.

Assume that $n = d$, such that $\Omega \subset D$ are domains in $\mathbb{R}^d$. As before, we denote by k_L and k_B the orders of the operators $\mathcal{L}$ and $\mathcal{B}$, respectively, with $\max\{k_L, k_B\} < q$.

Then it follows by [12, Theorem 5.1] that for any $u \in W^q_\infty(D)$ there exists $s \in \mathcal{S}$ such that simultaneously

$$\|\mathcal{L}(u-s)|_Y\|_\infty \leq C_L h_{\mathcal{S}}^{q-k_L} |u|_{W^q_\infty(D)} \quad \text{and} \quad \|\mathcal{B}(u-s)|_Z\|_\infty \leq C_B h_{\mathcal{S}}^{q-k_B} |u|_{W^q_\infty(D)},$$

where C_L, C_B are some positive constants independent of h_L, h_B and u. Therefore, we obtain from (4.40) and (4.41)

$$\Big| \int_\Omega f \, d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i) \Big| \leq \inf_{u \in \mathcal{U}_f} \Big\{ C_L \, \|w^*\|_1 \, h_{\mathcal{S}}^{q-k_L} + C_B \, \|v^*\|_1 \, h_{\mathcal{S}}^{q-k_B} \Big\} |u|_{W^q_\infty(D)}, \tag{4.43}$$

$$\Big| \int_{\partial\Omega} g \, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \leq \inf_{u \in \mathcal{U}_g} \Big\{ C_L \, \|w^*\|_1 \, h_{\mathcal{S}}^{q-k_L} + C_B \, \|v^*\|_1 \, h_{\mathcal{S}}^{q-k_B} \Big\} |u|_{W^q_\infty(D)}, \tag{4.44}$$

as long as $\mathcal{U}_f$ and $\mathcal{U}_g$ contain functions extensible to D with a finite seminorm $|u|_{W^q_\infty(D)}$.

Similar to Section 4.6.1, for a smooth domain $\Omega \subset \mathbb{R}^d$ and operators $\mathcal{L} = \Delta$, $\mathcal{B} = \partial_\nu$ with $k_L = 2$, $k_B = 1$, assuming that $\hat{f}$ and $\hat{g}$ are infinitely differentiable, we may use Proposition 2.1.4, the Sobolev embedding theorem, and the Stein extension theorem, to obtain the $\mathcal{O}(h_{\mathcal{S}}^{q-2})$ estimates

$$\Big| \int_\Omega f \, d\mu - \sum_{i=1}^{N_Y} w_i^* f(y_i) \Big| \leq C_1 \big(\|w^*\|_1 + \|v^*\|_1 \big) h_{\mathcal{S}}^{q-2} \|f\|_{H^s(\Omega)} \tag{4.45}$$

for $f \in H^s(\Omega)$ with $s > q - 2 + d/2$, and

$$\Big| \int_{\partial\Omega} g \, d\sigma - \sum_{i=1}^{N_Z} v_i^* g(z_i) \Big| \leq C_2 \big(\|w^*\|_1 + \|v^*\|_1 \big) h_{\mathcal{S}}^{q-2} \|g\|_{H^s(\partial\Omega)} \tag{4.46}$$

for $g \in H^s(\partial\Omega)$ with $s > q - 2 + (d+1)/2$, where the constants C_1 and C_2 are independent of f and g.

We refer to [66] for approximation results by ambient tensor-product splines when Ω is a compact closed hypersurface.

Elements of the basis of $\mathcal{S}$ obtained by restricting B-splines to Ω may have arbitrarily small support, and this is a known cause of instability in some numerical schemes of immersed type, see for example [15, 24, 28]. In the context of Framework 4.3.1, however, we did not encounter situations where instability of w and v would likely be caused by small cut elements, and so we did not investigate the use of any special stabilization techniques, see the remarks at the end of Chapter 5 for more information.

4.7 On the choice of minimization norm

As we have seen in Section 4.6, Framework 4.3.1 leads to quadrature formulas of high convergence order assuming sufficient regularity of the boundary value problem (4.15), the use of an appropriate numerical differentiation scheme, and stability of

the quadrature formulas, that is, $\|w\|_1 = \mathcal{O}(1)$ and $\|v\|_1 = \mathcal{O}(1)$. This suggests the use of the combined 1-norm

$$\|(w,v)\|_\sharp = \|(w,v)\|_1 = \|w\|_1 + \|v\|_1$$

in step 5 of the framework. However, the main reason why the 1-norm appears in the estimates is that in the error analysis it was convenient to use the 1-norms of the quadrature weight vectors w, v, and the ∞-norm of the error of numerical differentiation. In fact, Hölder's inequality (2.3) with any pair of conjugated exponents $p, p' \in [1,\infty]$ such that $1/p + 1/p' = 1$, leads to the estimates

$$\Big|\int_\Omega f\,d\mu - \sum_{i=1}^{N_Y} w_i f(y_i)\Big| \le \inf_{u\in\mathcal{U}_f}\Big\{\|w\|_p \big\|\mathcal{L}u_{|Y} - L\Psi(u)\big\|_{p'} + \|v\|_p \big\|\mathcal{B}u_{|Z} - B\Psi(u)\big\|_{p'}\Big\},$$

$$\Big|\int_{\partial\Omega} g\,d\sigma - \sum_{i=1}^{N_Z} v_i g(z_i)\Big| \le \inf_{u\in\mathcal{U}_g}\Big\{\|w\|_p \big\|\mathcal{L}u_{|Y} - L\Psi(u)\big\|_{p'} + \|v\|_p \big\|\mathcal{B}u_{|Z} - B\Psi(u)\big\|_{p'}\Big\},$$

where $p = 1$ and $p' = \infty$ as in (4.20) and (4.21) is not necessarily optimal.

Nevertheless, there are still good reasons why one may prefer the minimization of the 1-norm of the quadrature weights. First of all, the 1-norm is the standard way to assess stability of a quadrature formula: any perturbation of size $\varepsilon > 0$ in the values $f(y_i)$ and $g(z_i)$ is potentially amplified by $\|w\|_1$ and $\|v\|_1$ when computing the sums

$$\sum_{i=1}^{N_Y} w_i f(y_i) \quad \text{and} \quad \sum_{i=1}^{N_Z} v_i g(z_i).$$

Second, minimizing the 1-norm can be a way to obtain *positive formulas* with non-negative weights $w_i \ge 0$ or $v_i \ge 0$, which is a highly desirable property in many applications, although far from necessary for achieving stability of numerical integration in general. If we know the measures

$$|\Omega| = \int_\Omega 1\,d\mu, \qquad |\partial\Omega| = \int_{\partial\Omega} 1\,d\sigma,$$

of the domain and its boundary, then we can choose $\hat{f} \equiv 1$ and $\hat{g} \equiv -1$ as auxiliary functions, so that

$$\sum_{i=1}^{N_Y} w_i + \sum_{i=1}^{N_Z} v_i = |\Omega| + |\partial\Omega| \tag{4.47}$$

holds for all quadrature formulas (Y, w) and (Z, v) satisfying (4.18). Hence,

$$|\Omega| + |\partial\Omega| \le \|w\|_1 + \|v\|_1,$$

and equality is attained if and only if the weights w and v are non-negative. If a pair of non-negative weight vectors (w, v) satisfying system (4.18) exists, then it will be found as (w^*, v^*) by 1-norm minimization, and the combined 1-norm $\|(w^*, v^*)\|_1$ will be equal to $|\Omega| + |\partial\Omega|$, implying excellent stability. Alternatively, knowing $|\Omega|$

and $|\partial\Omega|$ we may ensure that both formulas (Y, w) and (Z, v) are exact for constants by requiring two separate conditions

$$\sum_{i=1}^{N_Y} w_i = |\Omega| \quad \text{and} \quad \sum_{i=1}^{N_Z} v_i = |\partial\Omega| \tag{4.48}$$

instead of (4.47). Note that when imposing one or both conditions in (4.48) there is no guarantee that 1-norm minimization delivers a positive formula for either w or v, even if they exist, because we minimize the sum $\|w\|_1 + \|v\|_1$ rather than $\|w\|_1$ or $\|v\|_1$ separately.

When using more than one auxiliary condition, for example by imposing exactness of the combined quadrature formula for two pairs of functions $(\hat{f}_1, \hat{g}_1)$ and $(\hat{f}_2, \hat{g}_2)$ as in (4.48), the error analysis in Proposition 4.2.1 can be modified to take this into account. The boundary value problem (4.15) is replaced by

$$\begin{cases} \mathcal{L}u = f - \alpha_1 \hat{f}_1 - \alpha_2 \hat{f}_2 & \text{in } \Omega, \\ \mathcal{B}u = g - \alpha_1 \hat{g}_1 - \alpha_2 \hat{g}_2 & \text{on } \partial\Omega, \end{cases} \tag{4.49}$$

the term $|\alpha|\,\hat{\varepsilon}$ in inequality (4.16) becomes

$$|\alpha_1|\,\hat{\varepsilon}_1 + |\alpha_2|\,\hat{\varepsilon}_2, \quad \text{with } \hat{\varepsilon}_1 = |\delta(\hat{f}_1, \hat{g}_1)| \text{ and } \hat{\varepsilon}_2 = |\delta(\hat{f}_2, \hat{g}_2)|,$$

and by taking the infimum over all pairs of coefficients α_1, α_2 satisfying the compatibility condition

$$\int_\Omega f - \alpha_1 \hat{f}_1 - \alpha_2 \hat{f}_2 \, d\mu = \int_{\partial\Omega} g - \alpha_1 \hat{g}_1 - \alpha_2 \hat{g}_2 \, d\sigma$$

and over all solutions u of (4.49) for these pairs (α_1, α_2), we may improve the estimate (4.16) compared to a single pair of auxiliary functions $(\hat{f}_1, \hat{g}_1)$ or $(\hat{f}_2, \hat{g}_2)$. However, in our numerical experiments in Section 5.1.4 we did not observe a significant improvement of the accuracy of the quadrature formulas when using more than one pair $(\hat{f}, \hat{g})$.

From a computational point of view, any 1-norm minimization problem can be reformulated as a linear program with twice as many unknowns, so any general purpose linear programming solver can be used for 1-norm minimization. When the simplex algorithm is used, *sparse* interior quadrature formulas w with no more than m nonzero weights can be obtained, with m being the total number of rows in the linear system (4.18).

The choice $p = 2$ is very attractive from the computational point of view: efficient linear algebra routines based on e.g. QR decomposition are available to find a solution of system (4.18) with minimal combined 2-norm

$$\|(w, v)\|_2 := \sqrt{\|w\|_2^2 + \|v\|_2^2}.$$

Moreover, the strict convexity of the 2-norm guarantees uniqueness of the optimal solution (w^*, v^*), a property that may not hold for the 1-norm. For a numerical comparison of 1-norm and 2-norm minimization, we refer to the experiments in Section 5.1.5.

More generally, one can consider a weighted 2-norm

$$\|(w,v)\|_{2,\Lambda} := \sqrt{\lambda_1 \|w\|_2^2 + \lambda_2 \|v\|_2^2}$$

to balance the asymptotic sizes of $\|w\|_2$ and $\|v\|_2$ as the spacing parameters h and $h_{\mathcal{S}}$ introduced in Section 4.6 approach zero, under the assumptions of quasi-uniform nodes and the expectation that

$$\|w\|_\infty = \mathcal{O}(N_Y^{-1}), \quad \|v\|_\infty = \mathcal{O}(N_Z^{-1}),$$

where $N_Z \ll N_Y$ as $Z \subset \partial\Omega$. Although this idea is theoretically sound because it is easy to see that the unweighted 2-norm is unbalanced, numerical experiments do not show significant improvements in the accuracy and stability of our quadrature formulas after switching to a suitably balanced weighted 2-norm. For this reason, and to keep complexity to a minimum, we have decided not to investigate the use of weighted 2-norms in this thesis.

4.8 Practical choices for effective algorithms

After discussing in Sections 4.4–4.7 various options available for the realization of Framework 4.3.1, we now describe two particular settings tested in the numerical experiments of Chapter 5 and recommended for practical applications.

The evidence gathered in our experiments suggests that the choices presented below in Algorithms 4.1 and 4.2 are essentially optimal among the ones that will be compared in Section 5.1, and therefore represent a good starting point for applications and future research. The first algorithm is based on meshless finite difference formulas, whereas the second one is based on collocation of tensor-product spline spaces defined on a bounding box around Ω. In this way we demonstrate that both the functional and meshless finite difference approaches have effective realizations.

We only consider domains Ω in $\mathbb{R}^d$, even if the methods are applicable to surfaces and other manifolds. We assume that $|\partial\Omega|$ is known, and choose

$$(\hat{f},\hat{g}) \equiv (0,1), \quad \mathcal{L} = \operatorname{div}, \quad \mathcal{B} = \gamma_\nu, \quad \|(w,v)\|_\sharp = \|(w,v)\|_2 = \sqrt{\|w\|_2^2 + \|v\|_2^2}.$$

For simplicity, we assume that the quadrature nodes in Y and Z, although irregular in general, are not intentionally generated with density varying over the domain or its boundary. Therefore we characterize their density by a single *spacing parameter* $h > 0$, meaning that Y and Z are produced by some node generation method with a uniform target spacing depending on h. We assume that the *packing spacing* of the node sets, defined as

$$h_{ps}(Y) := \left(\frac{|\Omega|}{N_Y}\right)^{1/d}, \quad h_{ps}(Z) := \left(\frac{|\partial\Omega|}{N_Z}\right)^{1/(d-1)}, \tag{4.50}$$

is proportional to the spacing parameter h of the node generation method. Note that $h_{ps}(Y)$ is the step size of a uniform Cartesian grid with N_Y nodes in a d-dimensional cube of measure $|\Omega|$, and similarly for $h_{ps}(Z)$. For some methods we may expect

that $h_{ps}(Y) \approx h$ and $h_{ps}(Z) \approx h$, but in other cases the packing spacing may be larger or smaller than the spacing parameter of the method by some fixed factor.

We assume that, in addition to supplying the quadrature nodes in Y and Z, the user provides sufficiently accurate outward-pointing unit normals at the boundary nodes of $Z \subset \partial\Omega$, and, in the case of Algorithm 4.1, is in position to generate an additional node set X in $\overline{\Omega}$ targeting a prescribed spacing parameter $h_X > 0$.

Note that in order to increase the performance of Algorithm 4.1, one should assemble the sparse matrices L^T and B^T directly, instead of precomputing matrices L_k and D_k for each spatial dimension separately. Moreover, the index sets $S_{L,i}$ and $S_{B,i}$ can be determined on the fly during the assembly procedure. Nevertheless, we have chosen to present Algorithm 4.1 in the form above to enhance readability, and to clearly show how L^T and B^T can be put together in the divergence case, which has not been detailed out in Section 4.6. Indeed, we have assumed in that section that functions in $\mathcal{D}$ are scalar valued, but in the case of $\mathcal{L} = \operatorname{div}$ and $\mathcal{B} = \gamma_\nu$ the functions in $\mathcal{D}$ are actually vector valued.

In the scalar case, the discretization operator $\Psi\colon \mathcal{D} \to \mathbb{R}^{N_X}$ is defined by pointwise evaluation over X, see (4.33), and so the elements in the vector $\Psi(u)$ follow the order of the nodes in X. In the vector valued case, the discretization operator $\Psi(F) \in \mathbb{R}^{dN_X}$ is defined by pointwise evaluation over X of all d components of the vector field $F \in \mathcal{D}$, and so one must choose whether to order the elements of $\Psi(F)$ so that all d components for a fixed node are contiguous, as in

$$\Psi(F) = \big(F_1(x_1), \ldots, F_d(x_1), \;\; \ldots, \;\; F_1(x_{N_X}), \ldots, F_d(x_{N_X})\big)^T,$$

or so that all N_X pointwise values for a fixed component are contiguous, as in

$$\Psi(F) = \big(F_1(x_1), \ldots, F_1(x_{N_X}), \;\; \ldots, \;\; F_d(x_1), \ldots, F_d(x_{N_X})\big)^T.$$

In Algorithm 4.1 we have chosen the latter ordering, and although the discretization operator Ψ is not used in the algorithm, the chosen ordering for $\Psi(F)$ clearly determines the order of the columns of L and B, and hence their assembly. In a high performance implementation, the order that provides the fastest memory access should be preferred. In any case, the final values of w and v do not depend on the chosen ordering for $\Psi(F)$, because changing the order amounts to permuting the rows of the linear system $Ax = b$. Similar considerations about ordering also apply to the next algorithm, based on collocation of vector fields whose components are uniform tensor-product splines defined on a bounding box around Ω. A basis for this space of vector fields is defined component-by-component using the same scalar basis of B-splines for each component. Once again, the ordering of basis elements in $\Psi(F)$ is completely arbitrary, and the one that provides the fastest memory access should be preferred in practice.

Once again, we have written Algorithm 4.2 in a way that enhances readability rather than performance. We conclude this section with three remarks.

First, we note that the set X of Algorithm 4.1 may also be generated as a Cartesian grid over the bounding box H as defined in Algorithm 4.2. This version of MFD delivered acceptable results in our numerical experiments, although it is inferior in accuracy and stability to the approach described in the algorithm. Generating a node set $X \subset \overline{\Omega}$ targeting a prescribed spacing parameter h_X is not a restrictive

Algorithm 4.1 – Approach based on meshless finite difference formulas (MFD)

Input: Sets of quadrature nodes $Y \subset \overline{\Omega}$ and $Z \subset \partial\Omega$ with spacing parameter $h > 0$.
Input: Outward-pointing unit normals $\nu_{|Z}$ at the nodes in Z.
Input: Measure $|\partial\Omega|$ of the boundary.
Input: Polynomial order $q \geq 2$.

Output: Quadrature weights w and v.

1: Set the spacing parameter $h_X = 1.6\,h$.
2: Generate a node set X of N_X nodes in $\overline{\Omega}$ according to the spacing parameter h_X.
3: Generate a k-d tree to facilitate nearest neighbors searches on X.
4: Set
$$n_L = 2\binom{q-1+d}{d} \quad \text{and} \quad n_B = 2\binom{q-2+d}{d}.$$
5: **for** $i = 1, \ldots, N_Y$ **do**
6: Query the k-d tree and store in $S_{L,i}$ the indices of the n_L nodes in X closest to y_i.
7: **end for**
8: **for** $i = 1, \ldots, N_Z$ **do**
9: Query the k-d tree and store in $S_{B,i}$ the indices of the n_B nodes in X closest to z_i.
10: **end for**
11: **for** $k = 1, \ldots, d$ **do**
12: Assemble the sparse matrix $L_k \in \mathbb{R}^{N_Y \times N_X}$ whose nonzero elements ℓ_{kij} are differentiation weights for the discretization of operator ∂_{x_k} on nodes X and Y:
$$\partial_{x_k} u(y_i) \approx \sum_{j \in S_{L,i}} \ell_{kij} u(x_j).$$
The weights ℓ_{kij} are generated using polyharmonic radial basis kernels with power $2q-1$ and polynomial term in Π_q^d, as explained in Section 4.6.1.
13: Assemble the diagonal matrix $D_k \in \mathbb{R}^{N_Z \times N_Z}$ whose elements are the k-th components of $\nu_{|Z}$.
14: **end for**
15: Assemble the sparse matrix $\tilde{B} \in \mathbb{R}^{N_Z \times N_X}$ whose nonzero elements b_{ij} are approximation weights for the pointwise values at the nodes in Z from the nodes of X:
$$u(z_i) \approx \sum_{j \in S_{B,i}} b_{ij} u(x_j).$$
The weights are generated using polyharmonic radial basis kernels with power $2q-3$ and polynomial term in Π_{q-1}^d, as explained in Section 4.6.1.
16: Concatenate $L_1, \ldots, L_d$ horizontally into $L \in \mathbb{R}^{N_Y \times dN_X}$.
17: Concatenate the products $D_1\tilde{B}, \ldots, D_d\tilde{B}$ horizontally into $B \in \mathbb{R}^{N_Z \times dN_X}$.
18: Assemble the sparse matrix
$$A = \begin{pmatrix} L^T & -B^T \\ 0 & \mathbb{1}^T \end{pmatrix},$$
with $\mathbb{1}$ being the all-ones column vector of dimension N_Z.
19: Assemble the right hand side $b = (0, |\partial\Omega|)^T$.
20: Compute the solution to $Ax = b$ with smallest 2-norm.
21: Split x into subvectors $w \in \mathbb{R}^{N_Y}$ and $v \in \mathbb{R}^{N_Z}$.

Algorithm 4.2 – Approach based on a tensor-product spline space (BSP)

Input: Sets of quadrature nodes $Y \subset \overline{\Omega}$ and $Z \subset \partial\Omega$ with spacing parameter $h > 0$.
Input: Outward-pointing unit normals $\nu_{|Z}$ at the nodes in Z.
Input: Measure $|\partial\Omega|$ of the boundary.
Input: Polynomial order $q \geq 2$.

Output: Quadrature weights w and v.

1: Set the knot spacing parameter $h_{\mathcal{S}} = 4\,h$.
2: Compute $H = [a_1, b_1] \times \cdots \times [a_d, b_d]$, a bounding box around $\overline{\Omega}$ such that the length of its sides are multiples of $h_{\mathcal{S}}$.
3: **for** $i = 1, \ldots, d$ **do**
4: Compute the knot vector T_i on $[a_i, b_i]$ with uniform spacing $h_{\mathcal{S}}$, as defined in (4.42).
5: **end for**
6: Let $\{s_1, \ldots, s_{N_{\mathcal{S}}}\}$ be the B-spline basis of the tensor-product space $\mathcal{S}_1 \otimes \cdots \otimes \mathcal{S}_d$, where each $\mathcal{S}_i$ is the univariate spline space with knot vector T_i and degree $q - 1$.
7: **for** $k = 1, \ldots, d$ **do**
8: Assemble the sparse collocation matrix $L_k \in \mathbb{R}^{N_Y \times N_{\mathcal{S}}}$ whose nonzero elements ℓ_{kij} are given by the evaluation of the k-th partial derivative of B-splines at the nodes of Y:

$$\ell_{kij} = \partial_{x_k} s_j(y_i)$$

For each node y_i, only q^d B-splines need to be evaluated, i.e. the ones whose support contains y_i.
9: Assemble the diagonal matrix $D_k \in \mathbb{R}^{N_Z \times N_Z}$ whose elements are the k-th components of $\nu_{|Z}$.
10: **end for**
11: Assemble the sparse collocation matrix $\tilde{B} \in \mathbb{R}^{N_Z \times N_{\mathcal{S}}}$ whose nonzero elements b_{ij} are given by evaluation of B-splines at the nodes of Z:

$$b_{ij} = s_j(z_i)$$

For each node z_i, only q^d B-splines need to be evaluated, i.e. the ones whose support contains z_i.
12: Concatenate $L_1, \ldots, L_d$ horizontally into $L \in \mathbb{R}^{N_Y \times dN_{\mathcal{S}}}$.
13: Concatenate the products $D_1\tilde{B}, \ldots, D_d\tilde{B}$ horizontally into $B \in \mathbb{R}^{N_Z \times dN_{\mathcal{S}}}$.
14: Assemble the sparse matrix

$$A = \begin{pmatrix} L^T & -B^T \\ 0 & \mathbb{1}^T \end{pmatrix},$$

with $\mathbb{1}$ being the all-ones column vector of dimension N_Z. Any row whose elements are all zeros is not included in A. Such rows come from B-splines whose support is disjoint from $Y \cap Z$.
15: Assemble the right hand side $b = (0, |\partial\Omega|)^T$.
16: Compute the solution to $Ax = b$ with smallest 2-norm.
17: Split x into subvectors $w \in \mathbb{R}^{N_Y}$ and $v \in \mathbb{R}^{N_Z}$.

hypothesis in practice, because, whenever node generation is deemed expensive due to the complexity of $\overline{\Omega}$, one may obtain X by thinning the set $Y \cup Z$.

Second, Algorithms 4.1 and 4.2 can be adapted to the setting where the quadrature nodes Y and Z are generated with locally varying density according to a *spacing function* $h(x)\colon \overline{\Omega} \to \mathbb{R}_+$. In this case, the set X of Algorithm 4.1 may be generated by using a scaled spacing function $h_X(x) = c\,h(x)$ for some $c > 1$, either from scratch, or by a suitable subsampling algorithm, see e.g. [63]. In the case of Algorithm 4.2, the spacing function $h(x)$ may be used to guide local refinement of the tensor-product spline space.

Third, the choice of the coefficients in $h_X = 1.6\,h$ and $h_{\mathcal{S}} = 4\,h$ is justified by the numerical experiments of Chapter 5. For now, we point out that these choices lead by design to underdetermined systems $Ax = b$. To see why, let us consider the case of Algorithm 4.1. Assuming that $N_Y \gg N_Z$, the matrix A has approximately N_Y columns and dN_X rows. By the definitions of h and h_X, assuming that $\left(\frac{|\Omega|}{N_X}\right)^{1/d} \approx h_X$, we have

$$\frac{dN_X}{N_Y} \approx \frac{d\,|\Omega|\,h_X^{-d}}{|\Omega|\,h^{-d}} = d(1.6)^{-d},$$

and so the ratio of rows to columns is always smaller than 1 for all $d \in \mathbb{N}$, with a peak of about 0.8 for $d = 2$. A similar argument shows that the linear system $Ax = b$ assembled in Algorithm 4.2 is also underdetermined for small enough h.

CHAPTER 5

Numerical tests of moment-free quadrature

In this chapter we numerically evaluate the accuracy and stability of quadrature formulas (Y, w) and (Z, v) produced by Algorithms 4.1 and 4.2 on several 2D and 3D domains (denoted by $\Omega_1, \ldots, \Omega_6$, see Figure 5.4). Let Ω be a bounded Lipschitz domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with piecewise smooth boundary. Quadrature errors of (Y, w) and (Z, v) are computed for two test functions $f_1, f_2 \in C^\infty(\overline{\Omega})$, and, respectively, their restrictions to the boundary $g_1 = f_1|_{\partial\Omega}$ and $g_2 = f_2|_{\partial\Omega}$. The first test function f_1 is a multidimensional generalization of Runge's function centered at a domain-dependent point x_R in the interior of Ω:

$$f_1(x, x_R) = \frac{1}{1 + 25\,\|x - x_R\|_2^2}.$$

The second test function f_2 is a scaled and translated version of the Franke function [37] in 2D, and its 3D generalization by Renka defined in [89]. The Franke and Renka functions are meant to be evaluated over $[0,1]^d$, but our domains are centered at the origin. For this reason, we compose these functions with the affine mapping

$$(x_1, \ldots, x_d) \mapsto \left(\frac{x_1 + 1}{2}, \ldots, \frac{x_d + 1}{2}\right), \quad d = 2, 3,$$

between $[0,1]^d$ and $[-1,1]^d$.

The accuracy of the quadrature formulas is assessed by computing the relative errors

$$e(f_1) := |\delta(f_1, 0)| \,/\, |I(f_1, 0)| \ , \ \ldots \ , \ e(g_2) := |\delta(0, g_2)| \,/\, |I(0, g_2)| \,.$$

In all tests, the denominators are large enough to make relative errors meaningful. Relative errors were preferred to absolute errors to make results comparable across different test functions and domains. As will be clear from the convergence plots of Section 5.2 that go below 10^{-15} in some instances, we need very accurate reference values of the integrals

$$I(f_1, 0), \quad I(f_2, 0), \quad I(0, g_1), \quad I(0, g_2),$$

in order to reliably compute at least one significant digit of the true relative errors. For every test domain, parametrizations of all smooth pieces of $\partial\Omega$ are known in closed form, with the parametric domain given by either an interval for 2D domains, or a rectangle for 3D domains. The integrals of f_1 and f_2 over Ω were computed by choosing vector fields F_1 and F_2 whose divergence is f_1 and f_2, then using the divergence theorem to turn the integrals over Ω into integrals over $\partial\Omega$, and finally integrating over the parametric domain of each smooth boundary piece with

MATLAB's adaptive quadrature routines `integral` and `integral2`. To meet the required accuracy target, absolute and relative tolerances were set to $4 \cdot 10^{-16}$.

Note that even though the test functions f_1 and f_2 are both infinitely differentiable and of a simple shape, we can distinguish their smoothness because the partial derivatives of f_1 grow as the factorial of their order, whereas those of f_2, as en entire function, grow much slower as an exponent of the order. This makes f_1, that actually stems from the famous Runge example for polynomial interpolation, a more difficult test function for high order methods than f_2.

The stability of the quadrature formulas is assessed by computing the *normalized stability constants*

$$K_w := \frac{1}{|\Omega|} \sum_{i=1}^{N_Y} |w_i| \,, \qquad K_v := \frac{1}{|\partial\Omega|} \sum_{i=1}^{N_Z} |v_i| \,,$$

that measure the sensitivity of the quadrature to the maximum absolute error in the function values, such that formulas with smaller K_w and K_v are more stable. Again, the normalization helps to compare the stability across all test functions and domains. For any quadrature formula exact on constants we have $K_w \geq 1$ or $K_v \geq 1$, and the equality holds if and only if the formula is positive. Without exactness for constants, but with the relative quadrature error for the constant function less than some $\varepsilon \in (0, 1)$, we get $K_w > 1 - \varepsilon$ and $K_v > 1 - \varepsilon$. Therefore, the best stability is attained when K_w and K_v are close to one. Note that Algorithms 4.1 and 4.2 may produce quadrature weights such that $K_w < 1$, because exactness for constants is only enforced for (Z, v) by the choice $(\hat{f}, \hat{g}) \equiv (0, 1)$. If $|\Omega|$ is known, in addition to $|\partial\Omega|$, then one may also enforce exactness of (Y, w) for constants by adding one more row to the linear system (4.18), but this does not provide any tangible benefit, as will be shown in Section 5.1.4.

In what follows we first demonstrate in Section 5.1 the effectiveness of the settings suggested in Algorithms 4.1 and 4.2, which serves as a kind of a tuning step for the free parameters in the algorithms. We do the testing mostly for Algorithm 4.1, because the results for Algorithm 4.2 are very similar. A second batch of numerical experiments in Section 5.2 shows that the quadrature errors of both algorithms converge to zero as $h \to 0$ on all test domains $\Omega_1, \ldots, \Omega_6$ at the rate determined by the order of the numerical differentiation scheme.

In all tests the weights of the meshless finite difference formulas are computed using the open source library mFDlab [20], and those for the tensor-product spline differentiation by MATLAB's Curve Fitting Toolbox.

5.1 Validation of recommended parameters

5.1.1 Choice of quadrature nodes

In the first test we demonstrate the robust performance of the MFD method described in Algorithm 4.1 with respect to the choice of nodes in the sets X, Y, Z. These experiments are performed on an elliptical domain Ω_1 in $\mathbb{R}^2$ with semi-axes of length

1 and 3/4 centered at the origin:

$$\Omega_1 = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; x_1^2 + \frac{x_2^2}{(3/4)^2} < 1 \right\}.$$

A domain in $\mathbb{R}^d$ with piecewise smooth boundary may be discretized by scattered nodes using a wide range of techniques, see the survey [107]. We consider here two important cases: a meshless advancing front node generation algorithm, as implemented in Algorithm 3.6, and a rejection sampling algorithm.

As explained in Section 3.2, the advancing front method works by first generating a set of boundary nodes Z from a parametric description of $\partial\Omega$, possibly consisting of multiple patches, and then placing interior nodes Y_{int} in Ω by advancing the front inside the domain starting from Z. Node sets Z and Y_{int} are disjoint, and their nodes are spaced according to locally varying *spacing functions* $h_Z \colon \partial\Omega \to \mathbb{R}^+$ and $h_Y \colon \Omega \to \mathbb{R}^+$, although we restrict ourselves to constant spacing

$$h_Z \equiv h_Y \equiv h \in \mathbb{R}^+$$

for all numerical experiments, which implies that the nodes are quasi-uniform. Outward-pointing unit normals $\nu_{|Z}$ are also computed and stored, because they are required to assemble matrix B in Algorithms 4.1 and 4.2.

Rejection sampling is based on either a uniform Cartesian grid, or the quasi-random Halton sequence, or pseudo-random uniformly distributed samples, and works in the same way regardless of the node distribution: samples are first generated in a bounding box H containing Ω, and then are either accepted, projected to the boundary, or discarded. More precisely, a set Y_H of $N_H = \mathrm{round}\big(|H|\, h^{-d}\big)$ nodes is initially generated in the bounding box H, and then only nodes inside Ω are kept by using a level set function φ such that

$$\Omega = \{x \in H \colon \varphi(x) < 0\}, \qquad Y_{int} = \{y \in Y_H \colon \varphi(y) < 0\}.$$

Nodes in a first-order approximation of a tubular neighborhood of $\partial\Omega$ of width $2h$ are projected to the zero set of φ along the direction parallel to the gradient of φ, and then a thinning operation is applied to the projected points to ensure that no pair of nodes in Z has distance smaller than h.

By the uniform density of the initial nodes used in the rejection sampling process, the packing spacing $h_{ps}(Y)$ of the generated internal nodes, see (4.50), ends up being very close to the spacing parameter h. The same, however, is not true for the advancing front method: for a fixed value of h, the two approaches produce a significantly different number of internal nodes (about 15% more for rejection sampling in 2D, compared to advancing front), and this is due to different definitions of h in two node placement techniques. For the sake of notational simplicity, we have decided to use the same variable h in all cases. However, whenever necessary, we tweak h in our numerical experiments to ensure that the sets Y and Z have approximately the same size across all node generation methods.

Regardless of how the sets Y_{int} and Z are generated, we produce both *open* quadrature formulas, for which $Y \cap \partial\Omega = \emptyset$, and *closed* quadrature formulas, for which $Y \cap \partial\Omega \neq \emptyset$. In all of our numerical experiments, the former are obtained

by taking $Y = Y_{int}$, whereas the latter are obtained by taking $Y = Y_{int} \cup Z$. Note that the accuracy and stability of v are also affected by the choice of Y, because the weight vectors w and v are computed simultaneously.

Unless stated otherwise, the set X of discretization nodes used to define the numerical differentiation scheme is generated in the same way as the closed version of Y, but with a larger spacing $h_X = 1.6\,h$. Alternatively, the set X is chosen as an unfitted uniform Cartesian grid over a bounding box H around Ω, large enough to ensure that no set of influence $S_{L,i}$ or $S_{B,i}$ reaches the boundary of H. The step size of the grid is slightly larger than $1.6\,h$, so that the number of rows in the matrix A stays approximately the same.

Node generation may be a stochastic process, which applies in particular to the advancing front method and rejection-sampling of random nodes, or a deterministic process, whose starting conditions are arbitrary and may therefore be randomized, like rejection-sampling of Halton nodes. Even in the case of Cartesian grids, some randomness can be included in the node generation process by a random shift of the entire grid in the range $[0, h)$ along each Cartesian dimension. In either case, node sets X, Y, Z can be understood as functions of a *seed*, an integer variable i_s that initializes the pseudo-random number generators. A naive comparison of quadrature errors based on a single choice of seed can lead to misleading results, because all errors are subject to random noise, and so any particular node distribution may overperform or underperform compared to the others, sometimes by an order of magnitude. Numerical quadrature is much more affected by this than, for example, the data fitting problem, where the maximum or the root mean square error over an entire domain is computed, which significantly reduces the influence of the noise in the pointwise errors.

Therefore we use average errors in all parameter validation tests. They are computed as follows. Let $e(\cdot, i_s)$ be the relative quadrature error for a given seed $i_s \in \mathbb{N}$. We denote by $e_{rms}(\cdot)$ the root mean square values (RMS) of $e(\cdot, i_s)$ for $i_s = 1, \ldots, n_s$:

$$e_{rms}(\cdot) = \left(\frac{1}{n_s} \sum_{i_s=1}^{n_s} e(\cdot, i_s)^2 \right)^{1/2},$$

where we take $n_s = 64$. The stability constants K_w and K_v are obtained by averaging the values

$$K_w(i_s) = \frac{1}{|\Omega|} \sum_{j=1}^{N_Y(i_s)} |w_j(i_s)| \quad \text{and} \quad K_v(i_s) = \frac{1}{|\partial\Omega|} \sum_{j=1}^{N_Z(i_s)} |v_j(i_s)|$$

over all seeds $i_s = 1, \ldots, n_s$.

Table 5.1 reports the RMS quadrature errors and average stability constants for open and closed quadrature formulas computed by Algorithm 4.1 on Ω_1 and $\partial\Omega_1$ using the four node generation methods described above. Numerical differentiation is performed by meshless finite difference formulas of polynomial order $q = 5$, and the Runge function f_1 is centered at $x_R = (0, 0)$. In all rows of Table 5.1, care was taken to tweak the value of h so that $N_Y \approx 2500$. The values of N_Y and N_Z are also averages over 64 seeds, but have been rounded to the nearest integer to enhance readability.

Table 5.1: Comparison of RMS quadrature errors and average stability constants for different node distributions. MFD algorithm on ellipse Ω_1 with polynomial order $q = 5$ and $n_s = 64$ distinct seeds. The set X is chosen in the same way as the closed version of Y, but with a larger spacing $h_X = 1.6\,h$.

Node distribution	h	N_Y	N_Z	$e_{\mathrm{rms}}(f_1)$	$e_{\mathrm{rms}}(g_1)$	$e_{\mathrm{rms}}(f_2)$	$e_{\mathrm{rms}}(g_2)$	K_w	K_v
Adv. front closed	2.91e−2	2507	189	6.98e−6	3.10e−9	3.38e−7	1.93e−7	1.41	1.002
Adv. front open	2.81e−2	2492	196	8.26e−6	5.37e−9	1.39e−6	3.14e−7	2.32	1.001
Cartesian grid closed	3.19e−2	2475	181	1.12e−5	1.03e−8	4.00e−7	2.78e−7	1.53	1.003
Cartesian grid open	3.07e−2	2473	188	1.54e−5	5.13e−9	4.49e−7	2.17e−7	2.12	1.005
Halton closed	3.19e−2	2499	183	9.59e−6	5.18e−9	4.29e−7	4.08e−7	1.97	1.003
Halton open	3.07e−2	2494	191	9.78e−6	5.15e−9	3.80e−7	3.42e−7	2.46	1.004
Random closed	3.19e−2	2488	175	6.12e−5	1.51e−8	1.68e−6	1.20e−6	4.99	1.004
Random open	3.07e−2	2494	182	6.93e−5	2.35e−8	1.90e−6	1.68e−6	10.00	1.109

We observe that nodes generated by the advancing front algorithm lead to the most accurate and stable quadrature formulas, although Cartesian nodes, Halton nodes, and even random nodes remain competitive. This suggests that our quadrature formulas are robust with respect to the exact placement of the input nodes in Y and Z. On average, closed quadrature formulas are more accurate and stable than their open counterparts, and so they will be used in all subsequent experiments. Note that the errors for the worst of 64 seeds are less than three times higher than the RMS errors in the table in all cases except for the open quadrature on random nodes, where they may get to almost 8 times higher. The worst stability constant K_w is 1.71 for the closed quadrature on advancing front nodes and is significantly higher than the average only for random nodes, with 8.29 for the closed and 233.21 for the open formulas. The largest constant K_v for the open formulas on random nodes is 12.72 and never exceeds 1.07 in other cases.

Table 5.2 reports the errors for the same numerical experiments as in Table 5.1, except that the set X is chosen as an unfitted uniform Cartesian grid with step size $2h$ when Y and Z are generated by the advancing front method, and step size $1.8\,h$ when rejection sampling is used. Smaller step sizes, such as $1.6\,h$, may lead to overconstrained linear systems, because the sets of influence $S_{L,i}$ and $S_{B,i}$ extend outside of $\overline{\Omega}$, and this increases significantly the number of constraints; we have found $2h$ and $1.8\,h$ to roughly equalize the size of A across the two tables.

Errors are up to two orders of magnitude larger, and the resulting quadrature formulas are much less stable. Therefore, we recommend to choose X in the same way as the closed version of Y in practical applications, as described in Algorithm 4.1.

5.1.2 Choice of spacing parameters h_X and $h_{\mathcal{S}}$

In order to keep the linear system (4.19) underdetermined and solvable, the spacing parameters h_X and $h_{\mathcal{S}}$ must be sufficiently large, because they determine the number of rows of the system matrix. Nevertheless, as seen by the considerations in Section 4.6, they should remain comparable to h to ensure that the numerical differentiation scheme is asymptotically accurate, in the sense that

$$\varepsilon_L(u) = \mathcal{O}(h^{q-k}) \quad \text{and} \quad \varepsilon_B(u) = \mathcal{O}(h^{q-k}).$$

Table 5.2: Comparison of RMS quadrature errors and average stability constants for different node distributions. MFD algorithm on ellipse Ω_1 with polynomial order $q = 5$ and $n_s = 64$ distinct seeds. The set X is chosen as an unfitted uniform Cartesian grid with step size about 0.057 over the bounding box $H = [-2, 2] \times [-2, 2]$ around Ω_1.

Node distribution	h	N_Y	N_Z	$e_{\text{rms}}(f_1)$	$e_{\text{rms}}(g_1)$	$e_{\text{rms}}(f_2)$	$e_{\text{rms}}(g_2)$	K_w	K_v
Adv. front closed	$2.91e{-2}$	2507	189	$1.74e{-4}$	$2.00e{-8}$	$4.06e{-6}$	$1.55e{-6}$	6.72	1.188
Adv. front open	$2.81e{-2}$	2492	196	$4.13e{-4}$	$7.53e{-8}$	$9.78e{-6}$	$4.07e{-6}$	22.01	2.415
Cartesian grid closed	$3.19e{-2}$	2475	181	$8.69e{-5}$	$3.12e{-8}$	$2.50e{-6}$	$1.55e{-6}$	6.21	1.115
Cartesian grid open	$3.07e{-2}$	2473	188	$5.56e{-4}$	$1.01e{-7}$	$1.60e{-5}$	$1.07e{-5}$	35.03	3.448
Halton closed	$3.19e{-2}$	2499	183	$8.27e{-5}$	$1.24e{-8}$	$2.78e{-6}$	$9.34e{-7}$	5.68	1.090
Halton open	$3.07e{-2}$	2494	191	$4.05e{-4}$	$6.94e{-8}$	$1.26e{-5}$	$4.75e{-6}$	27.13	2.746
Random closed	$3.19e{-2}$	2488	175	$4.98e{-4}$	$6.16e{-8}$	$9.34e{-6}$	$4.09e{-6}$	21.86	2.006
Random open	$3.07e{-2}$	2494	182	$2.52e{-3}$	$3.71e{-7}$	$5.48e{-5}$	$2.96e{-5}$	128.64	7.002

Therefore, we suggest to choose h_X and $h_{\mathcal{S}}$ to be directly proportional to h, and investigate how the errors of the quadrature formulas depend on the ratios h_X/h and $h_{\mathcal{S}}/h$.

In Algorithms 4.1 (MFD) and 4.2 (BSP) we recommend to choose the spacing parameters as $h_X = 1.6\,h$ and $h_{\mathcal{S}} = 4\,h$ based on numerical evidence that we have gathered in our numerical experiments on several 2D and 3D domains. The case of the ellipse Ω_1 is presented in Figure 5.1, where quadrature errors are plotted as a function of the ratios h_X/h and $h_{\mathcal{S}}/h$ for $h = 0.025$ and $q = 5$. Figure 5.2 shows a similar test on the ellipsoid Ω_4 in $\mathbb{R}^3$ with semi-axes of length 1, 0.7, 0.7,

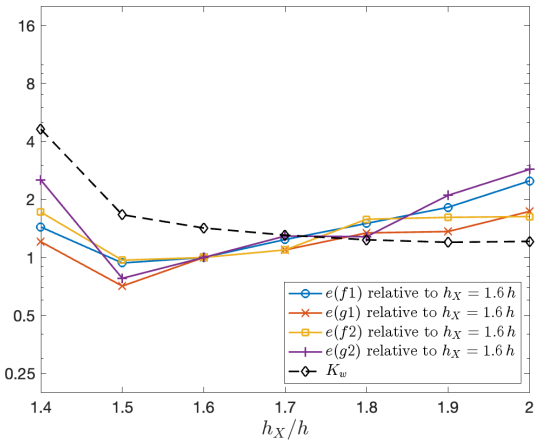
$$\Omega_4 = \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 \;\middle|\; x_1^2 + \frac{x_2^2}{0.7^2} + \frac{x_3^2}{0.7^2} < 1 \right\},$$

and with $h = 0.05$ and $q = 5$.

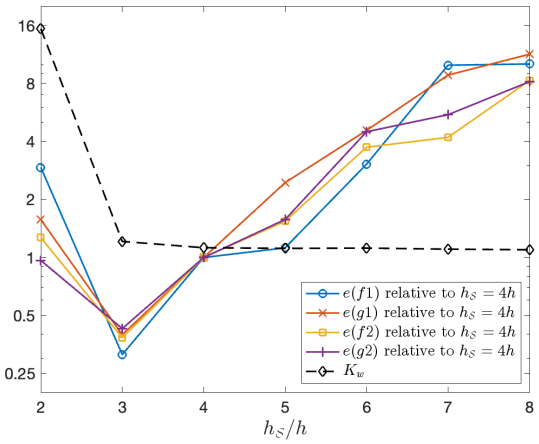
In both cases, we use node sets X, Y, Z generated by the advancing front algorithm, with $Z \subset Y$. As in the previous section, data points are the RMS of 64 quadrature errors given by different seeds. The Runge function f_1 is centered at the origin. To aid visualization, instead of plotting the absolute values of the errors, we plot their relative sizes compared to the choices $h_X = 1.6\,h$ and $h_{\mathcal{S}} = 4\,h$. The vertical axes are in logarithmic scale. In addition, we plot the stability constant K_w. In the BSP case, the size of the bounding box H has been chosen depending on $h_{\mathcal{S}}$ to avoid boundary effects, and to ensure that the length of each side of H is a multiple of $h_{\mathcal{S}}$.

The two figures clearly show that quadrature errors decrease as the ratios h_X/h and $h_{\mathcal{S}}/h$ get smaller, but only up to a point: when the ratios get too small, the errors and the stability constant K_w quickly rise up again. Reducing the ratios even further leads to overdetermined systems for which no exact solution exists, and so their errors are not included in the plots. The stability constant K_v is not included in the figures because its dependence on h_X/h and h_S/h is very weak. The ratios $h_X = 1.6\,h$ and $h_{\mathcal{S}} = 4\,h$ were found to be a good compromise between accuracy and stability, as they are close to the minimum of the curves in the two figures, but leaning on the side of stability, so that a safety margin is left in Algorithms 4.1 and 4.2. This choice of the ratios was also validated by running the convergence tests of Section 5.2 with slightly different values of h_X/h and $h_{\mathcal{S}}/h$, and comparing the results.

Even though increasing the ratios reduces the number of rows of the system

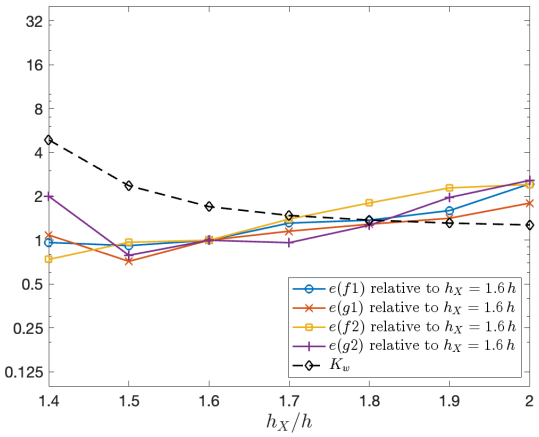


(a) MFD in 2D: Errors and stability constant K_w as functions of the ratio h_X/h; errors are relative to their values for $h_X/h = 1.6$.

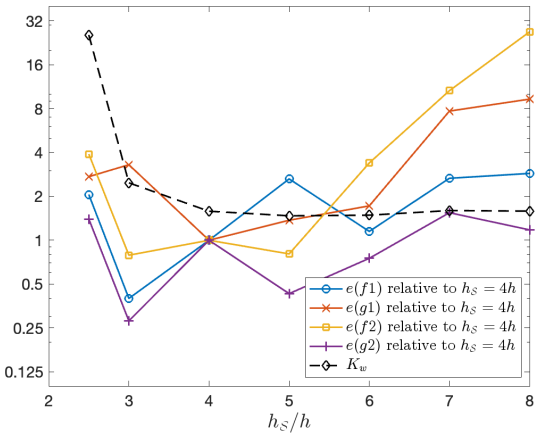


(b) BSP in 2D: Errors and stability constant K_w as functions of the ratio h_S/h; errors are relative to their values for $h_S/h = 4$.

Figure 5.1: RMS quadrature errors on ellipse Ω_1 with $h = 0.025$, $q = 5$ and 64 different seeds. The dashed lines show the average stability constant. On the left: MFD algorithm. On the right: BSP algorithm.



(a) MFD in 3D: Errors and stability constant K_w as functions of the ratio h_X/h; errors are relative to their values for $h_X/h = 1.6$.



(b) BSP in 3D: Errors and stability constant K_w as functions of the ratio h_S/h; errors are relative to their values for $h_S/h = 4$.

Figure 5.2: RMS quadrature errors on ellipsoid Ω_4 with $h = 0.05$, $q = 5$ and 64 different seeds. The dashed lines show the average stability constant K_w. On the left: MFD algorithm. On the right: BSP algorithm.

matrix A, it does not change the total number of nonzeros in A. Indeed, the number of nonzeros in every column of the horizontal concatenation $\begin{pmatrix} L^T & -B^T \end{pmatrix}$ depends in the MFD case only on the size n_L or n_B of the set of influence of the corresponding node in Y or Z, as chosen in Step 4 of Algorithm 4.1. Likewise, in the BSP case the number of nonzeros in every column in general equals the number q^d of B-splines of degree $q-1$ that include the corresponding quadrature node in the interior of their support. Taking into account the last row of A corresponding to the non-homogeneous constraint, we get the following upper bound for the number of nonzeros:

$$\operatorname{nnz}(A) \leq N_Z + \begin{cases} n_L N_Y + n_B N_Z, & \text{for Algorithm 4.1,} \\ q^d (N_Y + N_Z), & \text{for Algorithm 4.2.} \end{cases} \tag{5.1}$$

Note that we do not test how the performance of the MFD quadrature depends on the parameters n_L and n_B, leaving this to future work, where also more sophisticated methods for the selection of the sets of influence could be investigated. As mentioned in Section 4.6.1, we rely on the safe and simple but possibly not optimal selection of K nearest neighbors in X to a node in Y or Z, with K being twice the dimension of the corresponding polynomial space. As a result, for a fixed polynomial order q, $\operatorname{nnz}(A)$ for BSP is about twice as large compared to MFD in 3D, but somewhat smaller than it in 2D, compare numerical results in Table 5.6.

5.1.3 Choice of operators $\mathcal{L}$ and $\mathcal{B}$

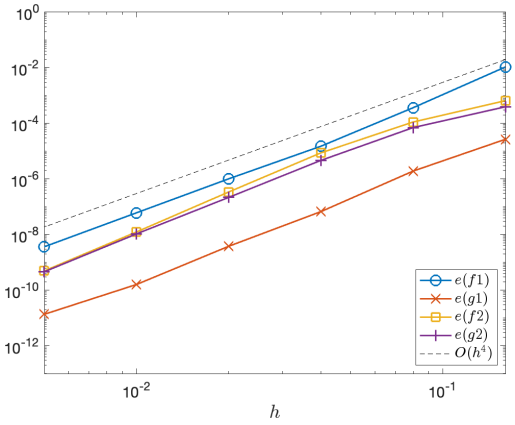
In Section 4.5 we discussed two possible ways to choose operators $\mathcal{L}$ and $\mathcal{B}$ for our scheme: the *elliptic approach* with $\mathcal{L} = \Delta$ and $\mathcal{B} = \partial_\nu$, and the *divergence approach*, with $\mathcal{L} = \operatorname{div}$ and $\mathcal{B} = \gamma_\nu$. Theoretical considerations indicate that the elliptic approach should be appropriate for domains with smooth boundary. However, the theory of the elliptic approach breaks down when the boundary is not smooth. At the same time, the divergence approach looks promising for piecewise smooth domains, even though the theory to justify it is far from complete because of the lack of results on the regularity of the corresponding boundary value problem (4.32).

In this section we compare the numerical performance of both approaches on two 2D domains, the ellipse Ω_1 with a smooth boundary, and the disk sector Ω_2 defined in the polar coordinates (r, θ) centered at the origin by

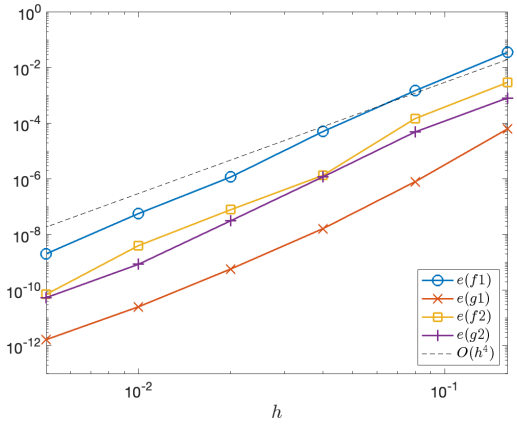
$$\Omega_2 = \left\{ (r, \theta) \in \mathbb{R}^2 \;\middle|\; 0 < r < 1 \text{ and } 0 < \theta < 3\pi/2 \right\}.$$

The domain Ω_2 has non-smooth boundary, with a reentrant corner at $(0, 0)$, and serves as a standard benchmark for testing numerical methods for the elliptic boundary value problem (2.5) that take into account the reduced smoothness of the solution at reentrant corners.

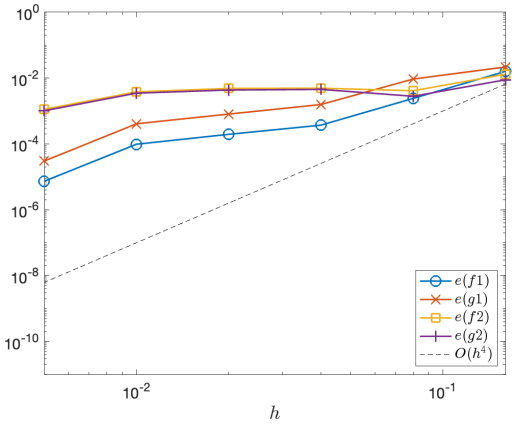
We use numerical differentiation schemes (Ψ, L, B) based on meshless finite difference formulas for both approaches, the same sets of quadrature nodes Y and Z generated by the advancing front algorithm with $Z \subset Y$, and the same non-homogeneous constraints with $(\hat{f}, \hat{g}) \equiv (0, 1)$. In the divergence approach we generate X with the usual advancing front method and spacing parameter $h_X = 1.6\, h$, whereas in the elliptic approach we pick $X = Y$. We have found this simple choice to



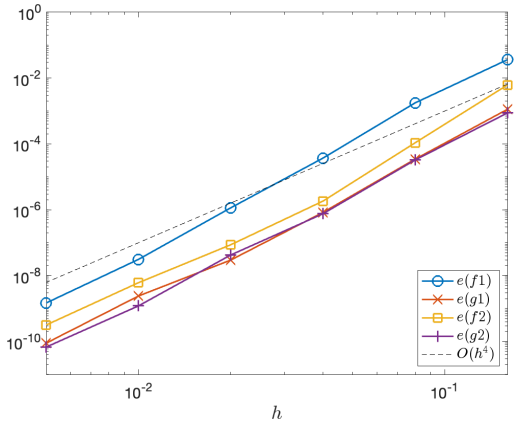
(a) Errors on ellipse Ω_1 for $\mathcal{L} = \Delta$ and $\mathcal{B} = \partial_\nu$.



(b) Errors on ellipse Ω_1 for $\mathcal{L} = \mathrm{div}$ and $\mathcal{B} = \gamma_\nu$.



(c) Errors on disk sector Ω_2 for $\mathcal{L} = \Delta$ and $\mathcal{B} = \partial_\nu$.



(d) Errors on disk sector Ω_2 for $\mathcal{L} = \mathrm{div}$ and $\mathcal{B} = \gamma_\nu$.

Figure 5.3: Comparison of RMS quadrature errors for the elliptic (left) and divergence (right) approaches on the ellipse Ω_1 (top) and disk sector Ω_2 (bottom), averaged over 8 seeds. Numerical differentiation based on MFD with polynomial order $q = 4 + k$, with $k = 2$, the order of Δ, (left) and $k = 1$, the order of div, (right).

be more accurate and reliable compared to the generation of a coarser set of nodes X. (Figure 5.1 is specific to the divergence approach.) The Runge function f_1 is centered at $x_R = (0, 0)$ in the case of the ellipse Ω_1, and at $x_R = (\cos(3\pi/4)/2, \sin(3\pi/4)/2)$ in the case of the disk sector Ω_2. We have chosen the polynomial order as $q = 4 + k$, where k is the order of the differential operator $\mathcal{L}$, so that the errors are expected to decay as h^4 as $h \to 0$, unless the convergence order is saturated by the low regularity of the solution u of the boundary value problem (4.15).

Figure 5.3 presents relative quadrature errors as a function of spacing parameter h for both the elliptic and the divergence approach for the ellipse Ω_1 (top row) and for the disk sector Ω_2 (bottom row). Relative errors are the RMS of 8 outcomes given by distinct seeds.

Comparing the plots in the top row of Figure 5.3 for the smooth domain Ω_1, we observe that although the errors are smaller for the divergence approach, the

decay is h^4 in both cases, as predicted by the theory presented in Sections 4.5 and 4.6.1. In contrast to this, the plots in the bottom row for the non-smooth disk sector Ω_2 show that in the case of the elliptic approach the quadrature errors are barely decreasing as $h \to 0$, whereas the divergence approach still delivers the expected h^4 convergence order. This is the very compelling reason why the divergence approach is recommended in Algorithms 4.1 and 4.2, despite the additional complexity introduced by the numerical differentiation of a vector field instead of a scalar function. Moreover, the divergence approach can achieve the same convergence order $q - k$ with a smaller polynomial order q, which means that smaller sets of influence can be used.

Note that the stability constants K_w and K_v of the quadrature formulas generated by the elliptic approach on the disk sector never exceed 1.1 in the tests of Figure 5.3(c). This provides further evidence that the large quadrature errors are not caused by the instability of the computed formulas, but rather by the ineffectiveness of the numerical differentiation scheme on solutions u to the boundary value problem (2.5) on the disk sector due to their low regularity near the reentrant corner, which leads to large recovery errors $\varepsilon_L(u)$ and $\varepsilon_B(u)$ and influences all estimates of Section 4.1 starting from (4.16).

In addition to the experiments in this section, those in Section 5.2 for $4 \leq q \leq 8$ on the disk sector Ω_2 and an L-shaped 3D domain (see Figures 5.6 and 5.9), also confirm that the divergence approach does not suffer from saturation of the convergence order of the quadrature formulas. This can be interpreted as numerical evidence in support of our conjecture that the boundary value problem (4.32) for the divergence operator has solutions of high regularity order on Lipschitz domains with piecewise smooth boundary, whenever the right hand side is highly regular, see Section 4.5 for a more detailed discussion.

5.1.4 Choice of non-homogeneous constraints

At least one non-homogeneous constraint (4.25) has to be included in the linear system (4.19), or else the quadrature will not be accurate for f, g with nonzero combined integral $I(f, g) \neq 0$, and the minimum-norm solution will be trivial. In Table 5.3 we have compared quadrature errors and stability constants for different combinations of the non-homogeneous constraints introduced in Section 4.4:

$$(\hat{f}, \hat{g}) \equiv (1, 0), \qquad (\hat{f}, \hat{g}) \equiv (0, 1), \qquad (\hat{f}, \hat{g}) \equiv (1, -1),$$
$$(\hat{f}, \hat{g}) \equiv (0, \partial_\nu \Phi(\cdot, x_0)) \text{ with } x_0 \in \Omega.$$

The integration domains are the ellipse Ω_1 and the disk sector Ω_2, as defined previously. The spacing parameter is $h = 0.025$ and the polynomial order is $q = 5$. Nodes in X, Y, Z are generated by the advancing front algorithm with $Z \subset Y$, and relative errors are the RMS of 64 quadrature errors given by different seeds. The Runge function f_1 is centered at $x_R = (0, 0)$ in the case of the ellipse Ω_1, and at $x_R = (\cos(3\pi/4)/2, \sin(3\pi/4)/2)$ in the case of the disk sector Ω_2. The fundamental solution is centered at $x_0 = (0.1, 0.05)$, a point that does not belong to any axis of symmetry of the two domains.

The results in the table show that quadrature errors depend quite weakly on the choice of non-homogeneous constraints. In particular, on the ellipse, whose

Table 5.3: Comparison of RMS quadrature errors and average stability constants for different non-homogeneous constraints. MFD algorithm on ellipse Ω_1 and disk sector Ω_2 with polynomial order $q = 5$ and 64 distinct seeds.

Domain	Constraints $(\hat{f}, \hat{g})$	h	$e_{\mathrm{rms}}(f_1)$	$e_{\mathrm{rms}}(g_1)$	$e_{\mathrm{rms}}(f_2)$	$e_{\mathrm{rms}}(g_2)$	K_w	K_v
Ω_1	$(1, 0)$	2.5e−2	3.31e−6	2.00e−9	1.92e−7	1.10e−7	1.4231	1.0009
Ω_1	$(0, 1)$	2.5e−2	3.24e−6	2.00e−9	1.91e−7	1.05e−7	1.4223	1.0009
Ω_1	$(1, -1)$	2.5e−2	3.31e−6	1.94e−9	1.92e−7	1.10e−7	1.4231	1.0009
Ω_1	$(0, \partial_\nu\Phi)$	2.5e−2	3.24e−6	1.97e−9	1.91e−7	1.05e−7	1.4224	1.0009
Ω_1	$(1, 0)$ and $(0, 1)$	2.5e−2	3.60e−6	1.98e−9	1.87e−7	1.11e−7	1.4159	1.0009
Ω_1	$(1, 0)$ and $(0, \partial_\nu\Phi)$	2.5e−2	3.57e−6	1.81e−9	1.87e−7	1.09e−7	1.4156	1.0009
Ω_1	$(0, 1)$ and $(0, \partial_\nu\Phi)$	2.5e−2	3.47e−6	1.80e−9	1.91e−7	1.08e−7	1.4161	1.0009
Ω_2	$(1, 0)$	2.5e−2	3.14e−6	1.27e−7	2.95e−7	1.10e−7	1.3480	1.0005
Ω_2	$(0, 1)$	2.5e−2	3.14e−6	9.94e−8	2.85e−7	9.51e−8	1.3449	1.0005
Ω_2	$(1, -1)$	2.5e−2	3.13e−6	9.61e−8	2.89e−7	9.94e−8	1.3480	1.0005
Ω_2	$(0, \partial_\nu\Phi)$	2.5e−2	4.62e−6	3.36e−6	3.47e−6	3.41e−6	1.3449	1.0005
Ω_2	$(1, 0)$ and $(0, 1)$	2.5e−2	3.09e−6	9.30e−8	3.07e−7	9.51e−8	1.3534	1.0005
Ω_2	$(1, 0)$ and $(0, \partial_\nu\Phi)$	2.5e−2	3.04e−6	1.25e−7	2.97e−7	9.99e−8	1.3493	1.0005
Ω_2	$(0, 1)$ and $(0, \partial_\nu\Phi)$	2.5e−2	3.02e−6	9.92e−8	2.90e−7	8.99e−8	1.3505	1.0005

boundary is smooth, there is essentially no difference in the errors, whereas on the disk sector, with a non-smooth boundary, a difference of one order of magnitude or more can be observed between enforcing exactness for constants and enforcing exactness for the normal derivative of the fundamental solution centered at $x_0 \in \Omega$. This is the reason to recommend the choice $(\hat{f}, \hat{g}) \equiv (0, 1)$ in Algorithms 4.1 and 4.2 whenever the measure of $\partial\Omega$ is known or can be sufficiently accurately approximated at low cost, as in the case of domains whose boundary is described by parametric patches with standard parameter domains for which accurate quadrature formulas are available. Whenever this is challenging, the purely moment-free approach based on the fundamental solution remains effective, although a smaller spacing parameter h may be needed to achieve the same errors on domains with non-smooth boundary. As far as stability is concerned, all the non-homogeneous constraints perform equally well. Boundary quadrature formulas are close to being positive, but no constraint can consistently deliver non-negative weights across all 64 seeds.

5.1.5 Choice of minimization norm

In Section 4.7, we have discussed relative merits of the norms

$$\|(w, v)\|_1 = \|w\|_1 + \|v\|_1 , \qquad \|(w, v)\|_2 = \sqrt{\|w\|_2^2 + \|v\|_2^2}$$

as candidates for the optimization norm $\|(w, v)\|_\sharp$ to be used in the final step of Framework 4.3.1, where the quadrature weights w^* and v^* are computed by minimizing $\|(w, v)\|_\sharp$ over all solutions to the non-homogeneous linear system $Ax = b$ of (4.19). We now present numerical experiments to compare their performance and support our choice of the 2-norm for the practical algorithms of Section 4.8.

The 1-norm minimization problem can be reformulated as the linear program

$$\text{Minimize} \quad \mathbb{1}^T x^+ + \mathbb{1}^T x^- \quad \text{subject to} \quad Ax^+ - Ax^- = b \quad \text{and} \quad x^+, x^- \geq 0, \quad (5.2)$$

with $x = x^+ - x^-$, and solved by e.g. the dual simplex algorithm or an interior-point method [82, 52]. For our numerical tests, we have used their implementations

Table 5.4: Comparison of RMS quadrature errors and average stability constants for different minimization norms and solvers. MFD algorithm on ellipse Ω_1 with polynomial order $q = 5$ and 8 distinct seeds. Column conv reports how many runs have converged out of 8.

Norm and solver	h	conv	$e_{\mathrm{rms}}(f_1)$	$e_{\mathrm{rms}}(g_1)$	$e_{\mathrm{rms}}(f_2)$	$e_{\mathrm{rms}}(g_2)$	K_w	K_v
L^1 dual simplex	$1.0e{-}1$	8/8	$2.88e{-}3$	$1.98e{-}6$	$1.90e{-}4$	$1.35e{-}4$	1.003	1
L^1 interior-point	$1.0e{-}1$	7/8	$2.26e{-}3$	$1.17e{-}6$	$2.13e{-}4$	$1.08e{-}4$	1.004	1
L^2 SuiteSparseQR	$1.0e{-}1$	8/8	$3.57e{-}3$	$3.26e{-}6$	$5.67e{-}4$	$8.55e{-}5$	1.359	1.006
L^1 dual simplex	$5.0e{-}2$	8/8	$1.88e{-}4$	$5.90e{-}8$	$6.50e{-}6$	$4.51e{-}6$	1.020	1
L^1 interior-point	$5.0e{-}2$	6/8	$1.35e{-}4$	$4.76e{-}8$	$3.52e{-}6$	$3.22e{-}6$	1.014	1
L^2 SuiteSparseQR	$5.0e{-}2$	8/8	$2.52e{-}4$	$4.68e{-}8$	$7.65e{-}6$	$6.37e{-}6$	1.372	1.001
L^1 dual simplex	$2.5e{-}2$	8/8	$2.15e{-}6$	$1.40e{-}9$	$1.49e{-}7$	$6.89e{-}8$	1.069	1
L^1 interior-point	$2.5e{-}2$	0/8	-	-	-	-	-	-
L^2 SuiteSparseQR	$2.5e{-}2$	8/8	$2.38e{-}6$	$1.45e{-}9$	$1.56e{-}7$	$8.03e{-}8$	1.427	1.002

provided by the 2024a release of MATLAB's Optimization Toolbox. For the solution of the 2-norm minimization problem, we have used a direct method based on the sparse QR factorization provided by the SuiteSparseQR library [18], version 4.3.3 available from [19].

The integration domain for these tests is the ellipse Ω_1, the polynomial order is $q = 5$, and the sets of nodes X, Y, Z are generated as usual by the advancing front algorithm with $Z \subset Y$. The Runge function f_1 is centered at $x_R = (0, 0)$.

Table 5.4 reports quadrature errors and stability constants of the formulas produced by Algorithm 4.1 when minimizing the norms above, denoted in the table by L^1 and L^2. Separate rows are devoted to the 1-norm minimization computed by the dual simplex algorithm and the interior-point method, because their results are significantly different. For the same tests, Table 5.5 reports further information such as the runtime t_{solve} of the solver, the number of the rows m of the system matrix A, and the number of nonzero weights in the quadrature formulas w and v. The computations have been performed on an Intel NUC Mini PC system with a Core i7-1165G7 CPU and 64 GB of DDR4 RAM. For each value of the spacing parameter h, results are averaged over 8 different seeds. To enhance readability, we have rounded to the nearest integer the values that are are meant to be integers, such as the number m of the rows of the system matrix.

It turned out that the interior-point method for 1-norm minimization may stall or lose feasibility when solving the linear program (5.2). For this reason, we have included the column "conv" in the tables that reports how many runs have converged out of 8, and all statistics are computed only for the seeds for which convergence has been achieved.

As we see in Table 5.4, minimization of the 1-norm leads on average to smaller quadrature errors compared to the 2-norm, although the advantage becomes less significant as h decreases. The L^1 methods also deliver more stable formulas, especially on the boundary: the appearance of the exact value 1 for K_v indicates that a positive formula (Z, v) has been found for all seeds, because Algorithm 4.1 relies on the non-homogeneous constraint $(\hat{f}, \hat{g}) \equiv (0, 1)$. The L^2 method only delivers positive formulas on the boundary for a few seeds.

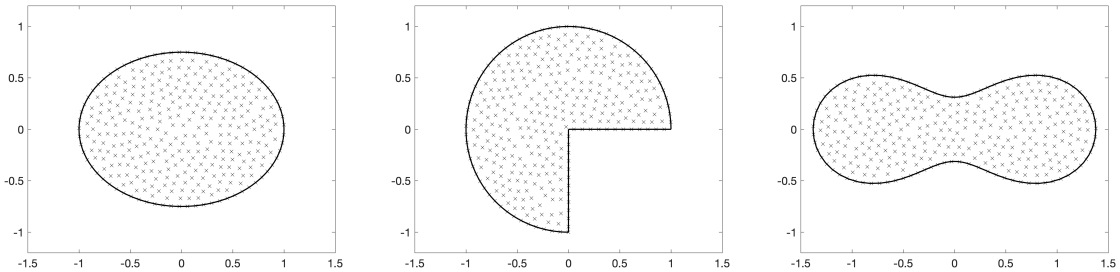
Table 5.5: Comparison of solver runtime and sparsity of the quadrature formulas for different minimization norms and solvers. MFD algorithm on ellipse Ω_1 with polynomial order $q = 5$ and 8 distinct seeds. Column conv reports how many runs have converged out of 8, m denotes the number of rows of the system matrix.

Norm and solver	h	conv	t_{solve}	m	N_Y	nnz(w)	N_Z	nnz(v)
L^1 dual simplex	$1.0e{-1}$	8/8	$2.52e{-1}$	195	232	148	55	41
L^1 interior-point	$1.0e{-1}$	7/8	$6.52e{-2}$	195	232	232	55	55
L^2 SuiteSparseQR	$1.0e{-1}$	8/8	$7.00e{-3}$	195	232	232	55	55
L^1 dual simplex	$5.0e{-2}$	8/8	$6.80e{+0}$	706	874	622	110	80
L^1 interior-point	$5.0e{-2}$	6/8	$3.78e{-1}$	706	874	874	110	110
L^2 SuiteSparseQR	$5.0e{-2}$	8/8	$2.44e{-2}$	706	874	874	110	110
L^1 dual simplex	$2.5e{-2}$	8/8	$2.29e{+3}$	2698	3395	2537	221	155
L^1 interior-point	$2.5e{-2}$	0/8	-	2698	3395	-	221	-
L^2 SuiteSparseQR	$2.5e{-2}$	8/8	$1.17e{-1}$	2698	3395	3395	221	221

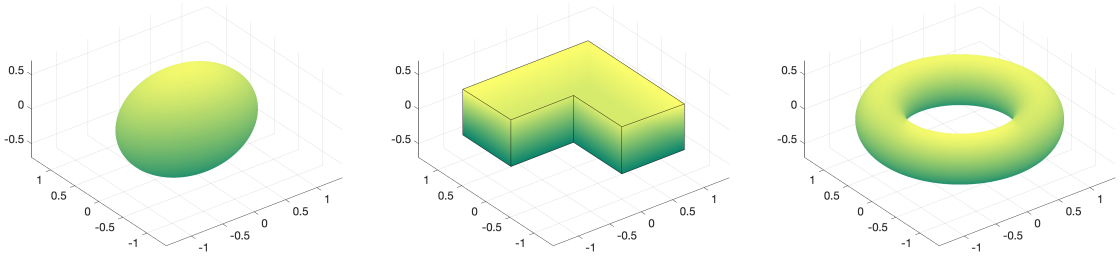
The L^1 methods, however, struggle with computational efficiency and robustness. The interior-point method does not converge for any seed when $h = 0.025$, even if parameters such as `OptimalityTolerance`, `ConstraintTolerance`, and `MaxIterations` are increased significantly from their default values. The dual simplex algorithm always terminates successfully in these tests, but its running time is orders of magnitude longer compared to SuiteSparseQR. This is the reason why the 2-norm is recommended in Algorithms 4.1 and 4.2 for general use.

The two algorithms for the minimization of the 1-norm that we have considered so far are not the only iterative algorithms that struggle with the system $Ax = b$: the ones for the minimization of the 2-norm, such as LSQR [86], also require too many iterations to be practical, unless an effective preconditioning strategy is employed. (Recall that the condition number of the minimum-norm least squares problem naturally grows as $h \to 0$.) Since out-of-the-box performance of standard left and right preconditioners such as symmetric successive over-relaxation and incomplete Cholesky factorization was not found to be satisfactory, we opted for a direct solver based on a sparse QR factorization of A^T, as implemented in the command `spqr_solve` provided by the MATLAB interface to the open source library SuiteSparseQR [18]. Fill-in during sparse QR factorization is mitigated by the reordering algorithms used by SuiteSparseQR, which were found to be quite effective, especially in the case of 2D domains. The QR factorization provided by SuiteSparseQR is rank-revealing, and the numerical rank of A is taken into account when computing the solution to $Ax = b$ with minimal 2-norm. To maximize accuracy, the tolerance 10^{-15} was passed to `spqr_solve` for numerical rank computation in all tests in this chapter, unless stated otherwise.

By the well-known properties of the linear program (5.2), its solution is not necessarily unique, and there is at least one solution x with the number of nonzero components not exceeding the number m of rows in the matrix A. The dual simplex algorithm is known to terminate at a solution with this property, and indeed we see that $\text{nnz}(w) + \text{nnz}(v)$ is less than m in all cases when this method is used, meaning that the combined quadrature weights $x = (w, v)^T$ have a significant proportion of zeros, which may be seen as an advantage. Moreover, the quadrature formulas



(a) Ellipse Ω_1. **(b)** Disk sector Ω_2. **(c)** Cassini oval Ω_3.



(d) Ellipsoid Ω_4. **(e)** L-shaped domain Ω_5. **(f)** Torus Ω_6.

Figure 5.4: Integration domains for the convergence tests. The node set Y generated by the advancing front method for a closed quadrature formula with $h = 0.08$ is displayed as crosses on top of the 2D domains.

(Y, w) and (Z, v) are also sparse when considered individually, because $\mathrm{nnz}(w) < N_Y$ and $\mathrm{nnz}(v) < N_Z$. In contrast to this, all weights coming from the interior-point method and the 2-norm minimization are always nonzero. Even sparser weights can be obtained by the dual simplex method with a larger ratio h_X/h, although the accuracy of the resulting quadrature formulas will be reduced.

5.2 Convergence tests in 2D and 3D

Now that numerical evidence for the choices made in Algorithms 4.1 and 4.2 has been given, we move on to the evaluation of relative quadrature errors for $h \to 0$, i.e. for an increasing number of quasi-uniform quadrature nodes, on the six test domains depicted in Figure 5.4.

The first two domains are the ellipse Ω_1 and the disk sector Ω_2 already defined in Sections 5.1.1 and 5.1.3. The third and last 2D domain is the region enclosed by a Cassini oval, a quartic plane curve defined as the locus of points such that the product of the distances to two fixed points $(-a, 0)$ and $(a, 0)$, called *foci*, is a constant $b^2 \in \mathbb{R}^+$:

$$\Omega_3 = \Big\{(x_1, x_2) \in \mathbb{R}^2 \;\Big|\; \big((x_1 + a)^2 + x_2\big)^2 \big((x_1 - a)^2 + x_2\big)^2 - b^4 < 0\Big\}.$$

For $a < b < a\sqrt{2}$, the domain Ω_3 has smooth boundary and is simply connected, although it is not convex: its shape resembles the number eight rotated sideways. For our numerical experiments, we have taken $a = 0.95$ and $b = 1$ to stress its non-convex shape and to ensure that its area is close to that of Ω_1 and Ω_2. As far

Table 5.6: Number of quadrature nodes and the size and sparsity of system matrix A for the tests on the ellipse Ω_1 and ellipsoid Ω_4, with polynomial order $q = 5$. Variables m_{MFD} and m_{BSP} denote the number of rows in A, and nnz_{MFD} and nnz_{BSP} the number of nonzero elements in A, produced by MFD and BSP algorithms, and $N = N_Y + N_Z$ is the number of columns of A.

Domain	h	N_Y	N_Z	m_{MFD}	m_{MFD}/N	nnz_{MFD}	m_{BSP}	m_{BSP}/N	nnz_{BSP}
Ω_1	1.60e−1	96	34	83	0.64	7.12e+3	85	0.65	6.53e+3
Ω_1	8.00e−2	350	69	285	0.68	2.38e+4	181	0.43	2.10e+4
Ω_1	4.00e−2	1338	138	1081	0.73	8.59e+4	393	0.27	7.39e+4
Ω_1	2.00e−2	5258	276	4161	0.75	3.27e+5	1199	0.22	2.77e+5
Ω_1	1.00e−2	20802	552	16321	0.76	1.27e+6	3869	0.18	1.07e+6
Ω_1	5.00e−3	82702	1105	64889	0.77	5.01e+6	13489	0.16	4.19e+6
Ω_4	1.00e−1	2066	697	1681	0.61	5.18e+5	1324	0.48	1.04e+6
Ω_4	7.94e−2	3979	1105	3193	0.63	9.69e+5	1903	0.37	1.91e+6
Ω_4	6.30e−2	7675	1739	6073	0.65	1.82e+6	3337	0.35	3.53e+6
Ω_4	5.00e−2	15011	2753	11674	0.66	3.49e+6	4627	0.26	6.66e+6
Ω_4	3.97e−2	29476	4389	22579	0.67	6.72e+6	7447	0.22	1.27e+7
Ω_4	3.15e−2	57966	6944	44107	0.68	1.30e+7	11185	0.17	2.43e+7
Ω_4	2.50e−2	114498	11033	86347	0.69	2.54e+7	18181	0.14	4.71e+7

as 3D domains are concerned, Ω_4 is an ellipsoid with semiaxes $(1, 0.7, 0.7)$, Ω_5 is an L-shaped domain obtained by cutting one quadrant from a rectangular cuboid with sides of length $(2, 2, 2/3)$, and Ω_6 is the solid torus with major radius $R = 1$ and minor radius $r = 0.32$.

For all tests we use node sets X, Y, Z generated by the advancing front algorithm of Section 3.2, with $Z \subset Y$ and spacing parameter h ranging from 0.005 to 0.16 on 2D domains and from 0.025 to 0.1 on 3D domains. Note that the size of 2D domains has been chosen so that their areas are approximately equal, and the same has been done for 3D domains and their volumes. This means that the advancing front algorithm generates a similar number N_Y of internal quadrature nodes for a given spacing parameter $h > 0$, and so the plots of interior quadrature errors can be directly compared across different domains of the same dimension. The exact number of quadrature nodes in Y and Z, the size of the system matrix A, and the number of its nonzero elements are reported in Table 5.6 for the domains Ω_1 and Ω_4, exemplifying 2D and 3D domains, respectively. The data for the other four domains is very similar to Ω_1 or Ω_4, except that the size of Z varies somewhat as it depends on the measure of $\partial\Omega$. Variables m_{MFD} and m_{BSP} denote the number of rows of A in the case of Algorithms 4.1 and 4.2, respectively, whereas nnz_{MFD} and nnz_{BSP} denote the respective number of nonzero elements in A. These quantities are reported for the order $q = 5$. The ratios m_{MFD}/N and m_{BSP}/N are below one, and this confirms that the linear system $Ax = b$ is underdetermined. Numbers for nnz_{MFD} and nnz_{BSP} agree with the estimates in (5.1).

Results for each test domain are presented in Figures 5.5 to 5.10. Relative quadrature errors for the polynomial orders q between 4 and 8 are shown as functions of the spacing parameter h. Results obtained by Algorithms 4.1 and 4.2 are shown in logarithmic scale side by side, with the same range for the horizontal and vertical axis to aid visualization and comparisons. Reference slopes corresponding to different convergence orders $q - 1$ are shown using dotted lines. The center x_R of the Runge function f_1 for each test domain is specified in the caption of its corresponding

figure, and is chosen so that $x_R \in \Omega$. Boundary quadrature errors for the test function g_2 were omitted for the sake of brevity, as these values are very similar to the ones for g_1 in all cases. Unlike the numerical tests in Section 5.1, relative quadrature errors are now reported for a single seed. Because of this, the error curves in Figures 5.5–5.10 look somewhat noisy compared to the convergence plots for e.g. the numerical solution of boundary value problems, where typically the RMS error over the entire domain is reported, which averages out the noise present in individual pointwise errors. We have chosen to plot quadrature errors for a single seed to stress that convergence is still evident despite the noise induced by the node generation process, and that no averaging with respect to the seed is needed in practice to get accurate and reliable results.

Note that quadrature errors are dropped from the plots whenever the linear system $Ax = b$ is overdetermined, i.e. the matrix A has more rows than columns. Moreover, in all MFD tests, values are dropped from the plots whenever $n_L > N_X$, that is, when the target size of the sets of influence for the discretization of $\mathcal{L}$ exceeds the size of the set X. These two measures can be understood as rough safeguards against ill-chosen parameters h and q, and are only needed on the coarsest node distributions.

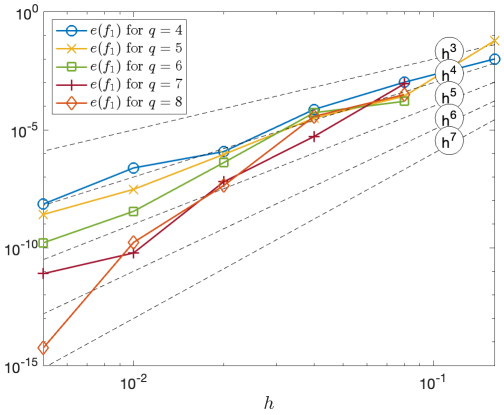
We indeed observe in all the figures a high order of convergence for both methods MFD and BSP, including the non-smooth domains Ω_2 and Ω_5, with the slopes of the error plots increasing with q and roughly corresponding to the expected h^{q-1} convergence order, in accordance with error estimates (4.35), (4.36), (4.43) and (4.44), if the noise induced by the node generation process is factored out. In the 2D setting, errors close to the accuracy limits of double-precision floating-point numbers are reached in all tests, with values around 10^{-15} being typical for the highest order methods on the largest number of nodes considered for the tests. To reach such small errors in 2D tests, the rank detection tolerance used by SuiteSparseQR (see Section 5.1.5) had to be manually set to 10^{-15}, because with default tolerances the errors would plateau around 10^{-12}. In the 3D setting, where such high precision is not reached, the rank detection tolerance used by SuiteSparseQR was kept to its automatically assigned value (about 10^{-10} for most tests), a choice motivated by better errors and stability constants for coarser sets of nodes.

Concerning stability constants, the worst-case values of K_w and K_v across all 2D and 3D test domains are plotted for $q = 5$ and $q = 8$ as a function of h in Figure 5.11 for the MFD approach, and in Figure 5.12 for the BSP approach. The figures confirm the overall stability of the quadrature formulas produced by both algorithms, and show that stability constants improve as h or q are reduced. For the smallest values of h considered in the tests, K_w never exceeds 3, and K_v is very close to 1. Moreover, for $q = 5$, all stability constants are quite small even for the coarsest sets of nodes. In all tests, the boundary quadrature formula is significantly more stable than the interior one, with values barely above 1 for $q = 5$, regardless of h.

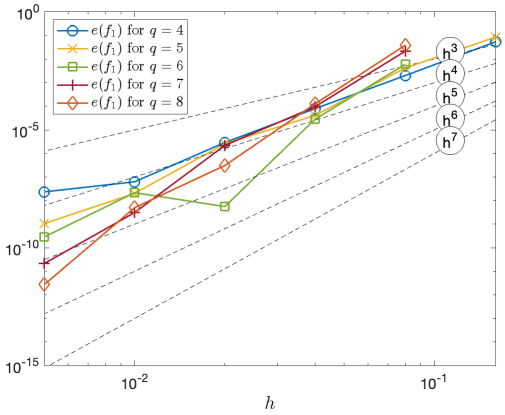
Comparison of the results for MFD and BSP reveals similar behavior between the two approaches, with comparable quadrature errors across all tests for methods with the same polynomial order q, and comparable stability constants, except for the case $q = 8$ and $h > 0.05$. The main difference in accuracy, although small, is that the MFD method tends to perform better than BSP on 3D domains with non-smooth boundary such as Ω_5, whereas the BSP method tends to perform better

than MFD for the integration of the Franke function (the smoother one among f_1 and f_2). Note that the numbers in the columns m_{MFD} and m_{BSP} of Table 5.6 indicate that the system matrices for MFD are much larger than those for BSP, which increases the fill-in for the direct method of SuiteSparseQR. On the other hand, as can be seen in Table 5.6 and the estimate (5.1), the total number of nonzeros in the larger system matrix for MFD is significantly smaller compared to the BSP matrix in the more computationally demanding 3D setting, even though we have not attempted to optimize the size of the sets of influence in MFD, as discussed at the end of Section 5.1.2. This may bring computational advantages for MFD when using iterative methods to solve the system. Another advantage of MFD is that values of h larger than 0.04 can be used with high polynomial orders such as $q = 6, 7, 8$ without a significant increase in the stability constants.

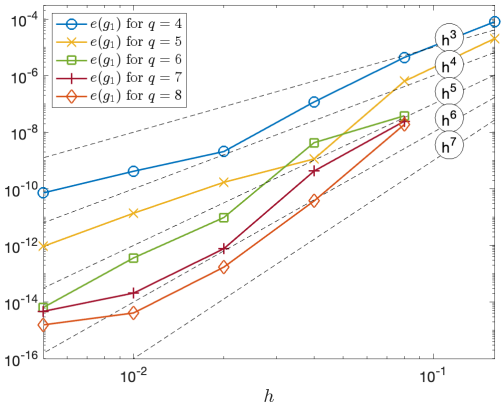
In the case of Algorithm 4.2, the support of some B-splines may have small intersection with Ω, and this is known to be a source of instability in immersed methods, see e.g. [15, 24, 28]. In our numerical experiments, however, we have not observed any instability as $h \to 0$, and a likely explanation is that the minimization of 2-norm has a regularizing effect on the computation of quadrature weights. Nevertheless, stabilization techniques from the literature on immersed methods may further improve the accuracy and stability of the BSP approach.



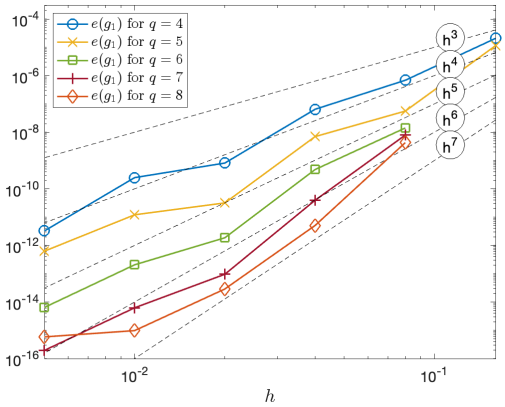
(a) Errors for f_1 with MFD weights.



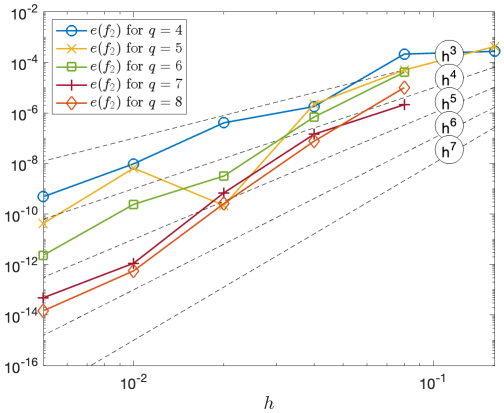
(b) Errors for f_1 with BSP weights.



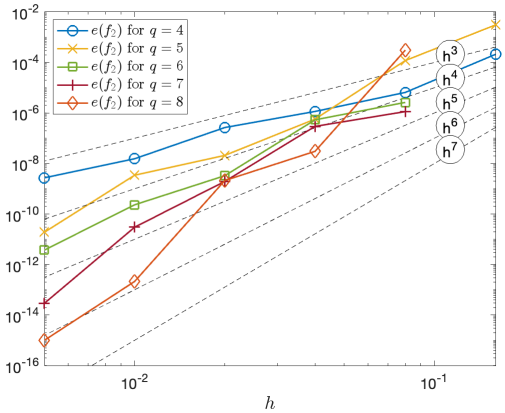
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

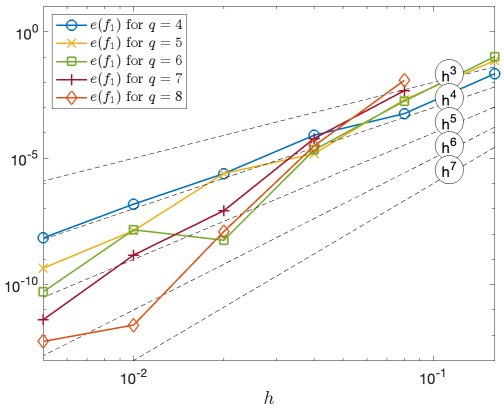


(e) Errors for f_2 with MFD weights.

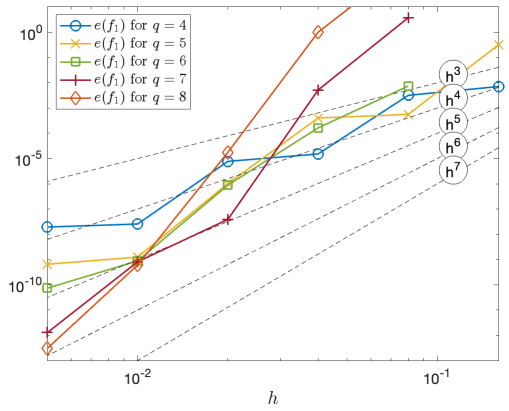


(f) Errors for f_2 with BSP weights.

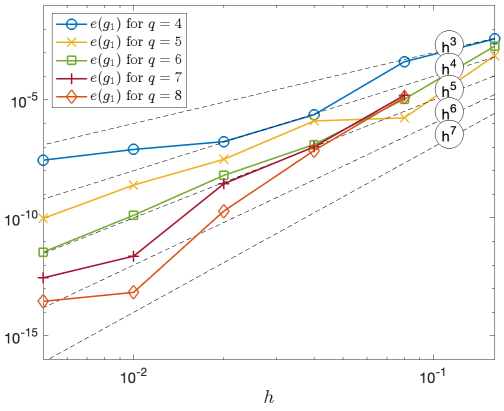
Figure 5.5: Relative quadrature errors on 2D domain Ω_1 (ellipse) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (0, 0)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.



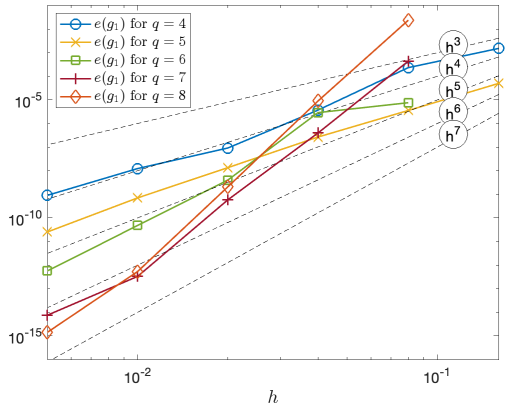
(a) Errors for f_1 with MFD weights.



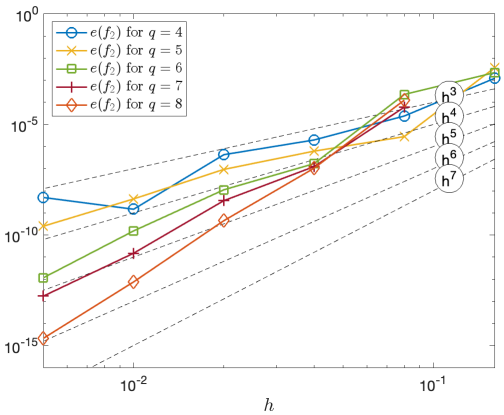
(b) Errors for f_1 with BSP weights.



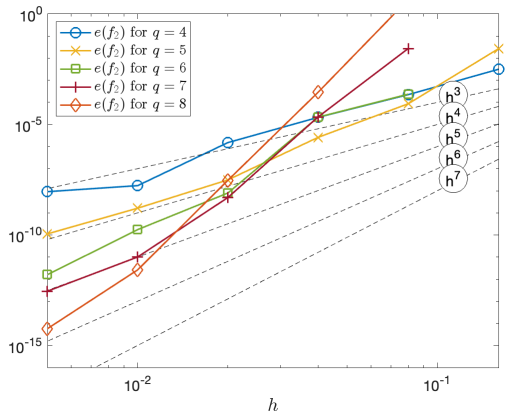
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

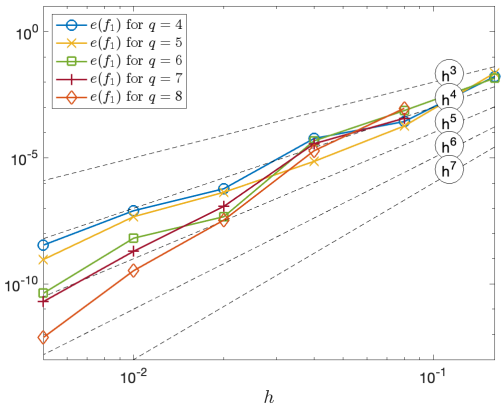


(e) Errors for f_2 with MFD weights.

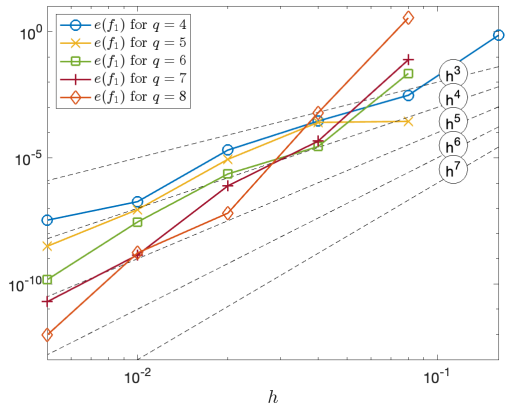


(f) Errors for f_2 with BSP weights.

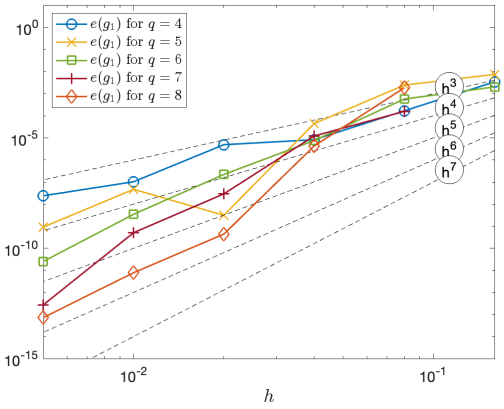
Figure 5.6: Relative quadrature errors on 2D domain Ω_2 (disk sector) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (\cos(3\pi/4)/2, \sin(3\pi/4)/2)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.



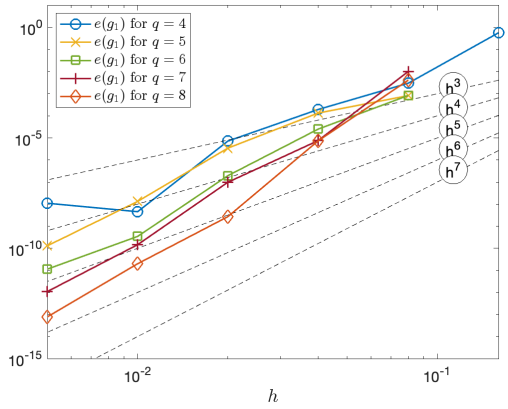
(a) Errors for f_1 with MFD weights.



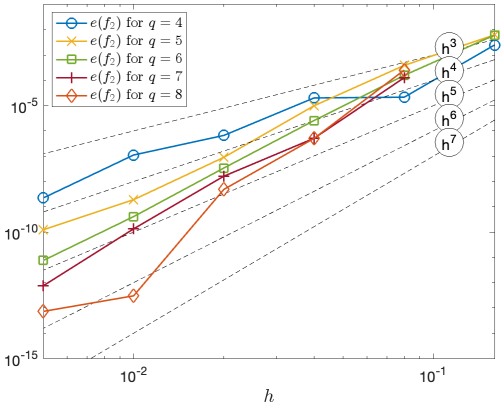
(b) Errors for f_1 with BSP weights.



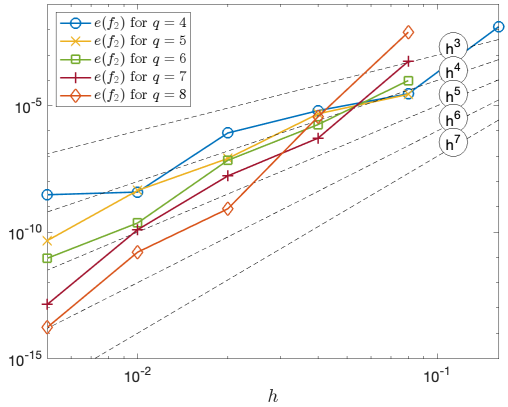
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

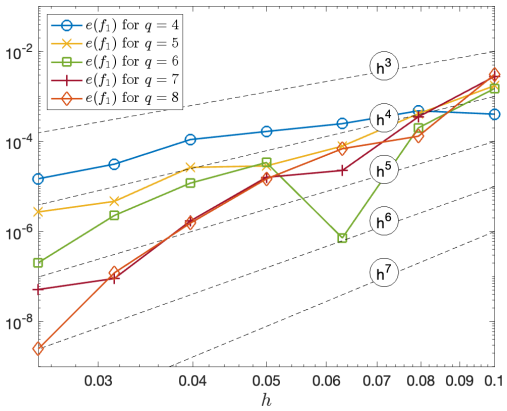


(e) Errors for f_2 with MFD weights.

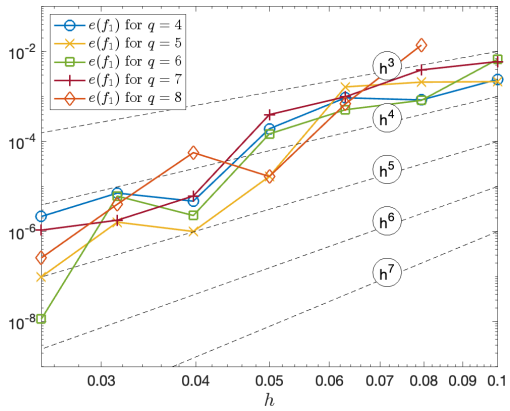


(f) Errors for f_2 with BSP weights.

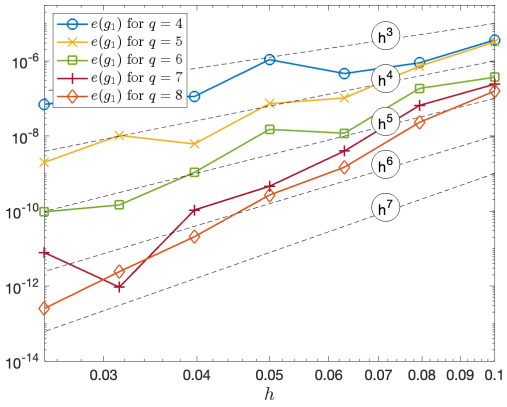
Figure 5.7: Relative quadrature errors on 2D domain Ω_3 (Cassini oval) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (0, 0)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.



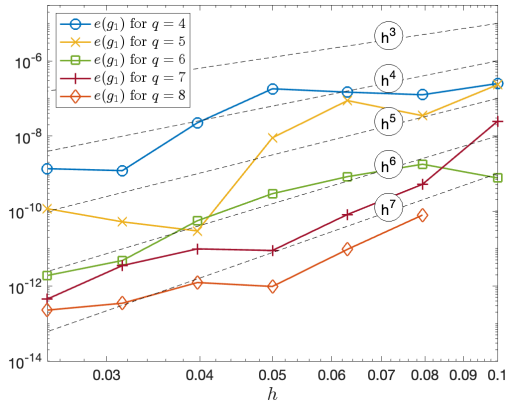
(a) Errors for f_1 with MFD weights.



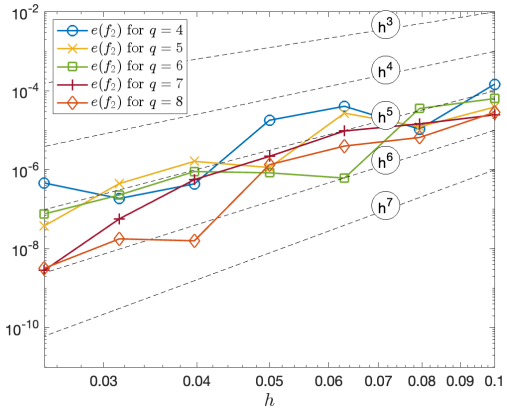
(b) Errors for f_1 with BSP weights.



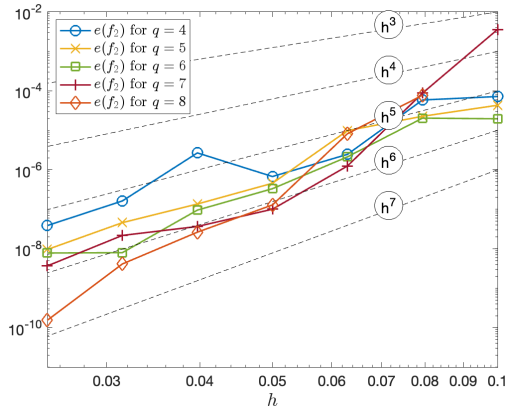
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

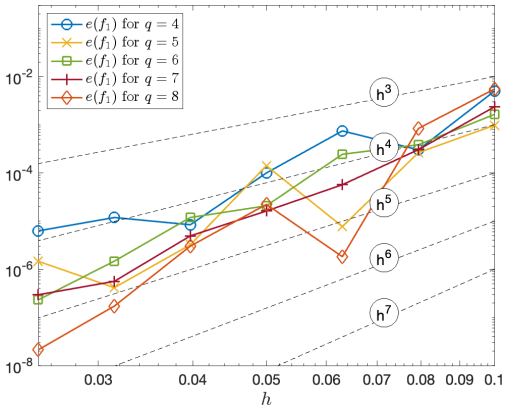


(e) Errors for f_2 with MFD weights.

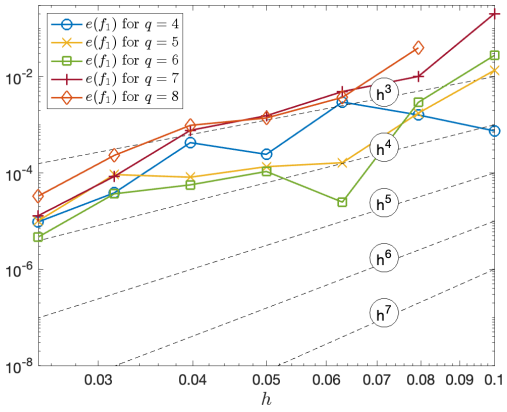


(f) Errors for f_2 with BSP weights.

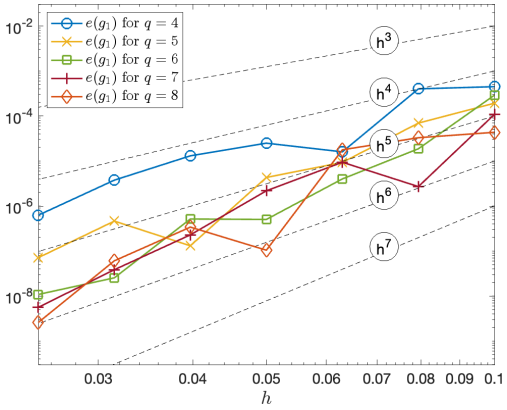
Figure 5.8: Relative quadrature errors on 3D domain Ω_4 (ellipsoid) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (0, 0, 0)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.



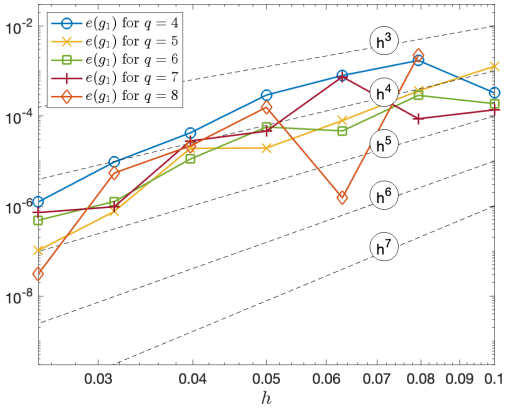
(a) Errors for f_1 with MFD weights.



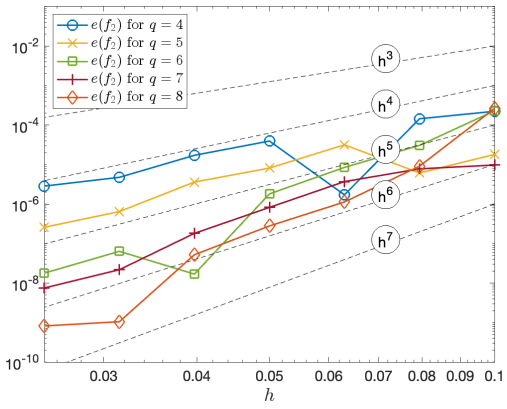
(b) Errors for f_1 with BSP weights.



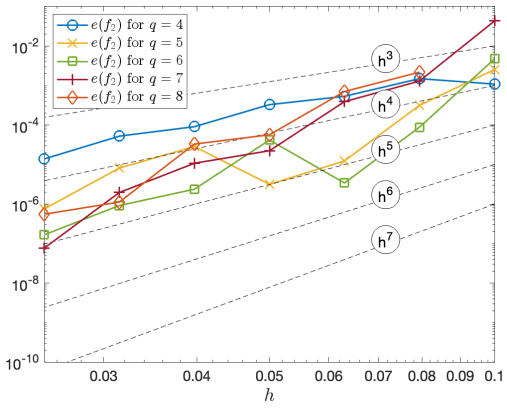
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

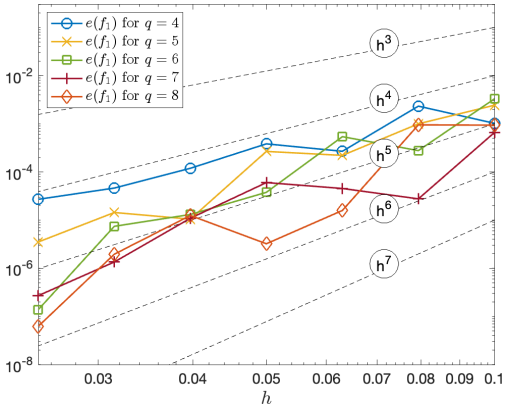


(e) Errors for f_2 with MFD weights.

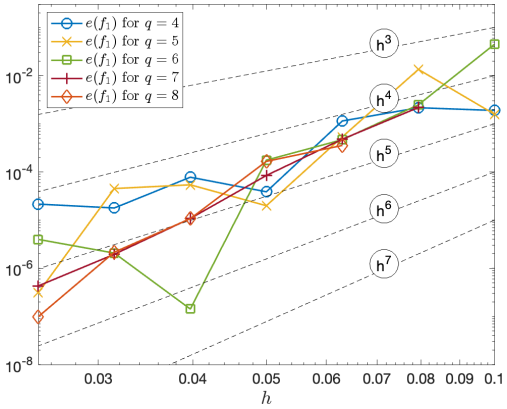


(f) Errors for f_2 with BSP weights.

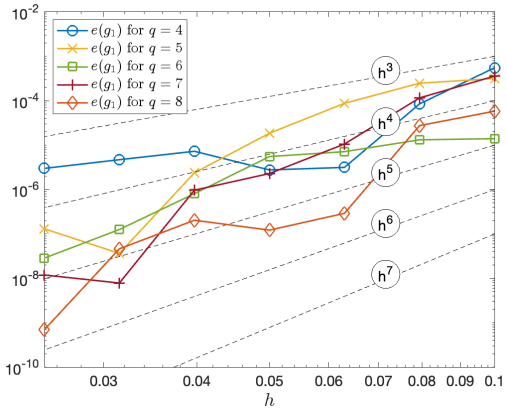
Figure 5.9: Relative quadrature errors on 3D domain Ω_5 (L-shaped) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (1/2, 1/2, 0)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.



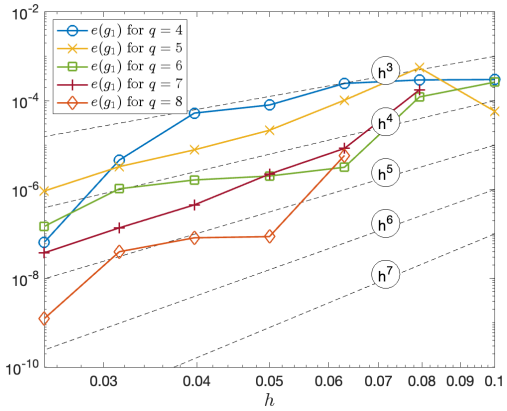
(a) Errors for f_1 with MFD weights.



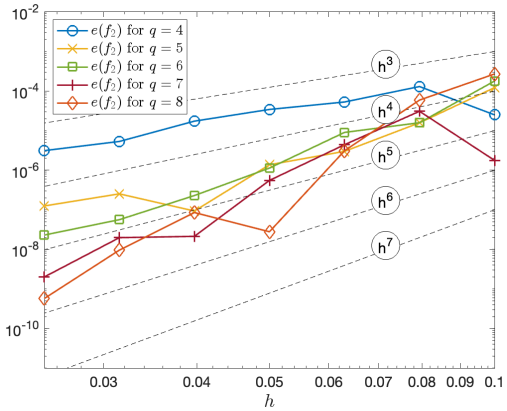
(b) Errors for f_1 with BSP weights.



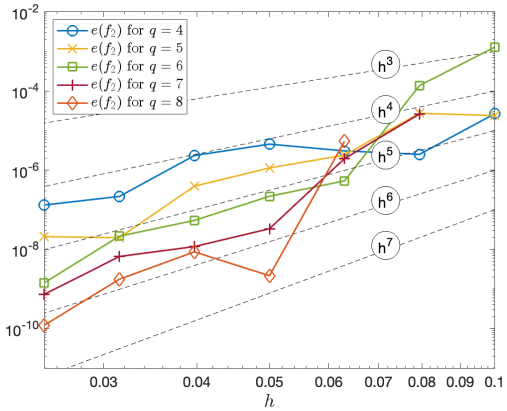
(c) Errors for g_1 with MFD weights.



(d) Errors for g_1 with BSP weights.

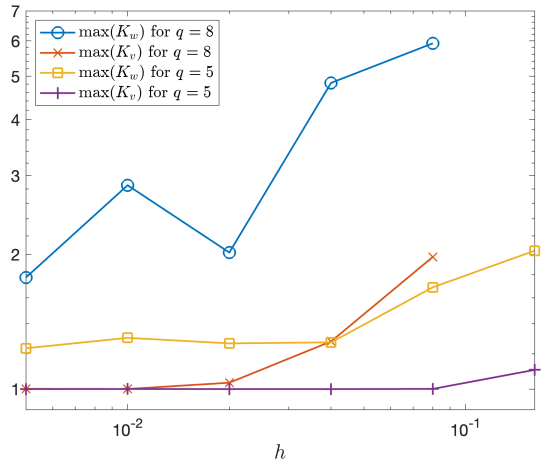


(e) Errors for f_2 with MFD weights.

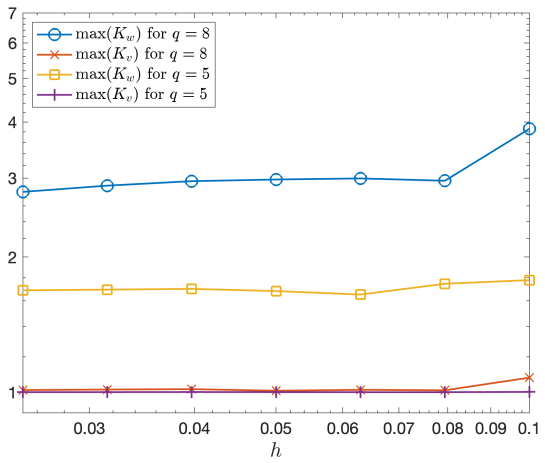


(f) Errors for f_2 with BSP weights.

Figure 5.10: Relative quadrature errors on 3D domain Ω_6 (torus) and its boundary, plotted as a function of h. Nodes are generated with an advancing front method. Runge function f_1 is centered at $x_R = (1, 0, 0)$. Results of Algorithms 4.1 and 4.2 are compared for polynomial orders $q = 4, \ldots, 8$. Reference slopes of convergence order $q - 1$ are shown using dotted lines.

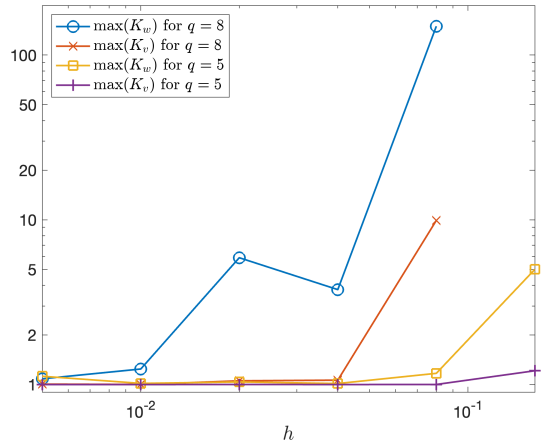


(a) Maximum stability constants produced by MFD algorithm on 2D domains for $q = 5$ and $q = 8$.

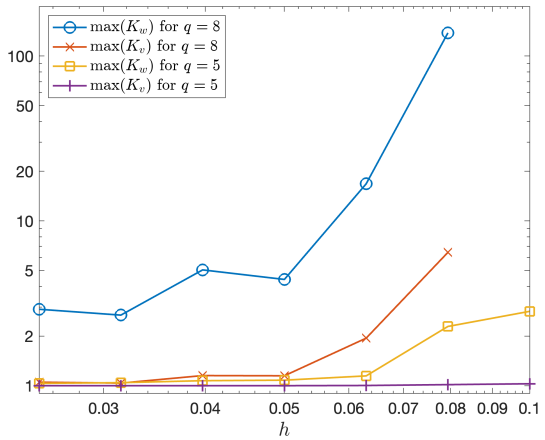


(b) Maximum stability constants produced by MFD algorithm on 3D domains for $q = 5$ and $q = 8$.

Figure 5.11: Maximum values of the normalized stability constants $K_w = \|w\|_1 / |\Omega|$ and $K_v = \|v\|_1 / |\partial\Omega|$ for the MFD convergence tests across all domains $\Omega_1, ..., \Omega_6$ and polynomial orders $q = 5, 8$, plotted as a function of h.



(a) Maximum stability constants produced by BSP algorithm on 2D domains for $q = 5$ and $q = 8$.



(b) Maximum stability constants produced by BSP algorithm on 3D domains for $q = 5$ and $q = 8$.

Figure 5.12: Maximum values of the normalized stability constants $K_w = \|w\|_1 / |\Omega|$ and $K_v = \|v\|_1 / |\partial\Omega|$ for the BSP convergence tests across all domains $\Omega_1, ..., \Omega_6$ and polynomial orders $q = 5, 8$, plotted as a function of h.

CHAPTER 6

Positive moment-free quadrature

Nonnegativity of the weights is one of the most desirable features for a quadrature formula (Y, w), because it implies not only stability, but also mimetic properties such as *monotonicity* for any pair of functions f_1, f_2 defined on a set that contains Y,

$$f_1 \leq f_2 \quad \Rightarrow \quad \sum_{i=1}^{N_Y} w_i f_1(y_i) \leq \sum_{i=1}^{N_Y} w_i f_2(y_i),$$

and positive semi-definiteness of the symmetric bilinear form

$$(\cdot,\cdot)_w \colon \mathbb{R}^{N_Y} \times \mathbb{R}^{N_Y} \to \mathbb{R}, \qquad (U,V)_w := \sum_{i=1}^{N_Y} w_i U_i V_i,$$

which serves as a discrete counterpart to the L^2 inner product in many contexts. If the weights are nonnegative, we speak of *positive* formulas, and if the weights are strictly positive, we speak of *strictly positive* formulas.

As demonstrated by the numerical experiments in the previous chapter, the high-order, moment-free quadrature formulas on scattered nodes produced by Algorithms 4.1 and 4.2 are stable, although they generally lack positivity. Both algorithms achieve stability by minimizing the 2-norm of the weights (w, v), a choice motivated by computational efficiency. However, replacing the 2-norm with the 1-norm yields only marginal improvements in stability constants. Since minimization of the 1-norm is guaranteed to yield positive weights when such solutions exist (under the assumption of exactness for constants, see Section 4.7), this provides numerical evidence that the linear system (4.18) assembled by these algorithms does not have positive solutions $(w, v) \geq 0$.

Algorithms 4.1 and 4.2 are merely specific instances of Framework 4.3.1; in this chapter we characterize positive formulas in the context of the whole framework, and prove the existence of positive formulas for special choices of nodes and differentiation schemes.

Our proofs are constructive and rely on the existence of positive differentiation formulas for the Laplacian and normal derivative operators. On scattered nodes, this requirement essentially limits the order of convergence of the quadrature formulas to one, but on more structured nodes, such as grids of Shortley–Weller type, second order can be reached, up to a logarithmic factor.

6.1 Positive quadrature from positive differentiation

Recall that, according to Framework 4.3.1, quadrature weights w and v are computed as solutions to the linear system $Qx = b$, with

$$Q = \begin{pmatrix} L^T & -B^T \\ \hat{f}|_Y^T & -\hat{g}|_Z^T \end{pmatrix}, \quad x = \begin{pmatrix} w \\ v \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \end{pmatrix}, \tag{6.1}$$

and that the consistency of $Qx = b$ can be characterized in terms of a certain *discrete incompatibility condition*, see Proposition 4.3.2. The simultaneous existence of positive quadrature weights w and v can be characterized similarly.

Proposition 6.1.1. *The linear system $Qx = b$ given by (6.1) has a nonnegative solution $x = (w, v) \geq 0$ if and only if every vector $y = (c, \lambda) \in \mathbb{R}^M \times \mathbb{R}$ that satisfies the system of inequalities*

$$\begin{cases} Lc \geq -\lambda \hat{f}_{|Y} \\ -Bc \geq \lambda \hat{g}_{|Z} \end{cases}$$

also satisfies

$$\lambda \left(\int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \right) \geq 0.$$

Proof. The characterization is an immediate consequence of Farkas' lemma, which states that exactly one of the following assertions about Q and b is true:

1. There exists $x \in \mathbb{R}^{N_Y + N_Z}$ such that $Qx = b$ and $x \geq 0$.
2. There exists $y \in \mathbb{R}^{M+1}$ such that $Q^T y \geq 0$ and $b^T y < 0$.

Expanding the products $Q^T y$ and $b^T y$ in the second assertion completes the proof. □

An important class of differentiation schemes for which we can prove that $Qx = b$ has nonnegative solutions is given by *positive numerical differentiation formulas* for operators $\mathcal{L} = -\Delta$ and $\mathcal{B} = \partial_\nu$. The notion of positive differentiation formulas is defined in Section 2.3, and has been investigated in the context of meshless methods by several authors [29, 68, 69, 90, 91]. To avoid a clash of notation with Section 2.3, matrix A in Framework 4.3.1 has been renamed to Q. In this section, we define A as the vertical concatenation of matrices L and $-B$. We can now prove that $Qx = b$ has nonnegative solutions under positivity of the numerical differentiation formulas and a few more technical assumptions.

Proposition 6.1.2. *Let X, Y, Z be nonempty scattered sets of nodes such that*

$$Y = \{y_i\}_{i=1}^{N_Y} \subset \overline{\Omega}, \quad Z = \{z_i\}_{i=1}^{N_Z} \subset \partial\Omega, \quad X = \{x_i\}_{i=1}^{N_X} = Y \cup Z, \quad Y \cap Z = \emptyset,$$

and partition the range $\{1, \ldots, N_X\}$ into two sets of indices $I(Y)$ and $I(Z)$ such that

$$Y = \{x_i \in X \mid i \in I(Y)\} \quad \text{and} \quad Z = \{x_i \in X \mid i \in I(Z)\}.$$

Let $L \in \mathbb{R}^{N_Y \times N_X}$ and $B \in \mathbb{R}^{N_Z \times N_X}$ be matrices associated to numerical differentiation formulas for operators $\mathcal{L} = -\Delta$ and $\mathcal{B} = \partial_\nu$ such that

1. *Matrices $-L$ and B have positive type.*
2. *All the rows of $-L$ and B sum to zero. In other words, the differentiation formulas are exact for constants.*
3. *For all $i \in I(Z)$, there exists $j \in I(Y)$ such that vertex i is connected to vertex j in the matrix graph of the square matrix A (see Definition 2.3.4).*

Then, for all positive functions $\hat{f}\colon \overline{\Omega} \to \mathbb{R}^+$ bounded away from zero, i.e.

$$\hat{f}(x) \geq \hat{f}_{\min} > 0 \quad \text{for some constant } \hat{f}_{\min} \in \mathbb{R},$$

the linear system $Qx = b$ given by (6.1) with $\hat{g} \equiv 0$ has a nonnegative solution $x \geq 0$.

Proof. Let $(c, \lambda) \in \mathbb{R}^{N_X} \times \mathbb{R}$ be a pair that satisfies the system of inequalities

$$\begin{cases} Lc \geq -\lambda \hat{f}_{|Y} \\ -Bc \geq \lambda \hat{g}_{|Z} \end{cases}$$

as in Proposition 6.1.1, and suppose, for the sake of contradiction, that

$$\lambda \left(\int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma \right) < 0,$$

which is equivalent to $\lambda < 0$. When $\hat{f}$ is bounded away from zero, we can prove a discrete counterpart of the strong maximum principle for elliptic operators:

$$\max\{c_i \mid i \in I(Y)\} < \max\{c_i \mid i \in I(Z)\}. \tag{6.2}$$

Indeed, suppose that for some $i \in I(Y)$ it holds that $c_i \geq c_j$ for all $j = 1, \ldots, N_X$. The i-th row of inequality $Lc \geq -\lambda \hat{f}_{|Y}$ reads

$$\sum_{j \in S_{L,i}} L_{ij}\, c_j \geq -\lambda \hat{f}(y_i).$$

Then, by the assumptions on λ, $\hat{f}$, c_i, and $-L$,

$$0 < -\lambda \hat{f}_{\min} < -\lambda \hat{f}(y_i) \leq \sum_{j \in S_{L,i}} L_{ij}\, c_j \leq \sum_{j \in S_{L,i}} L_{ij}\, c_i = c_i \sum_{j \in S_{L,i}} L_{ij} = 0.$$

This contradiction proves inequality (6.2). Now, let $x_{i^*} = z_i \in Z$ be the node on $\partial\Omega$ where c attains its maximum c_{i^*}, and choose a path in the matrix graph of A that connects the i^*-th node to an internal node in Y. By the maximality of c_{i^*}, the values of c along the path cannot exceed c_{i^*}, and by (6.2) the value of c at the end of the path must be strictly smaller than c_{i^*}. Therefore, there exists $x_{\ell^*} = z_\ell \in Z$ such that x_{ℓ^*} is the last node along the path such that $c_{\ell^*} = c_{i^*}$. The ℓ-th row of inequality $-Bc \geq \lambda \hat{g}_{|Z}$ is equivalent to

$$\sum_{j \in S_{B,\ell}} B_{\ell j}\, c_j \leq 0.$$

Then, by the assumptions on B and the maximality of c_{ℓ^*},

$$0 \geq \sum_{j \in S_{B,\ell}} B_{\ell j}\, c_j = \sum_{j \in S_{B,\ell}} B_{\ell j}\, (c_j - c_{\ell^*}) > 0.$$

The rightmost inequality sign is strict because, by construction, $c_j < c_{\ell^*}$ for some $j \in S_{B,\ell}$. We have finally reached a contradiction, and the proof is complete. □

An analogous statement with auxiliary functions $\hat{f} \equiv 0$ and $\hat{g}\colon \partial\Omega \to \mathbb{R}^-$ such that

$$\hat{g}(x) \leq \hat{g}_{\min} < 0 \text{ for some constant } \hat{g}_{\min} \in \mathbb{R}$$

also holds, and can be proved with similar techniques, although a different assumption on the connectivity of the matrix graph of A is required, namely that for all $i \in I(Y)$ there exists $j \in I(Z)$ such that vertex i is connected to vertex j.

Since an L-matrix whose rows sum to zero is automatically a weakly diagonally dominant matrix, one might as well assume that any two rows of A are connected, and try to prove existence of nonnegative solutions to $Qx = b$ using Proposition 2.3.6 and M-matrix techniques instead. This alternative approach turns out to be successful, and leads to *strictly* positive quadrature weights. We start with the following lemma.

Lemma 6.1.3. *Suppose that the matrix $A = (a_{ij})_{i,j=1}^N$ is singular and becomes an M-matrix if any one of its rows is replaced by e_i, the unit vector with one on the diagonal. Then there exists $\gamma \in \ker A^T$ with $\gamma_i > 0$ for all $i = 1, \ldots, N$.*

Proof. Since A is singular, there is a vector $\gamma \in \ker A^T \setminus \{0\}$ with at least one positive component. Without loss of generality assume that $\gamma_1 > 0$. Denote by $a_1, \ldots, a_N$ the columns of A^T, and by e_i the unit vector with i-th component equal to one. Then $0 = A^T\gamma = \sum_{i=1}^N \gamma_i a_i$ implies

$$a_{11}e_1 + \sum_{i=2}^{N} \frac{\gamma_i}{\gamma_1} a_i = a_{11}e_1 - a_1,$$

from which it follows that

$$\begin{pmatrix} e_1^T \\ a_2^T \\ \vdots \\ a_N^T \end{pmatrix}^T \begin{pmatrix} a_{11} \\ \gamma_2/\gamma_1 \\ \vdots \\ \gamma_N/\gamma_1 \end{pmatrix} = a_{11}e_1 - a_1.$$

The matrix in the left hand side is the transpose of an M-matrix, hence all entries of its inverse are nonnegative. The vector in the right hand side is nonnegative because its first component is zero, and the remaining components cannot be negative by the definition of M-matrix, and we conclude that $\gamma_i/\gamma_1 \geq 0$, $i = 2, \ldots, N$. This proves that $\gamma \geq 0$.

To prove the strict positivity of γ, we start with the observation that A must have rank $N - 1$, because changing one row of A is enough to bring its rank to N. Therefore, the kernel of A^T must have dimension one, and must be spanned by γ. Suppose on the contrary that $\gamma_i = 0$ for some index $i \in \{2, \ldots, N\}$. Then

$$e_i \perp \gamma \quad \Rightarrow \quad e_i \perp \ker A^T \quad \Rightarrow \quad e_i \in \operatorname{Im} A \quad \Rightarrow \quad Av = e_i \text{ for some } v \in \mathbb{R}^N.$$

It follows in particular that $a_i^T v = 1$. Let $\tilde{A}_i$ be the M-matrix obtained by replacing the i-th row of A with e_i^T. We have

$$\tilde{A}_i v = Av + (e_i^T - a_i^T)ve_i = e_i + (v_i - a_i^T v)e_i = v_i e_i.$$

Let $\eta \in \mathbb{R}^N$ be a nonzero vector in $\ker A$. Then the rows of A are orthogonal to η and hence $\tilde{A}_i \eta = \eta_i e_i$. Therefore

$$\tilde{A}_i(\eta_i v - v_i \eta) = 0,$$

which implies $\eta_i v = v_i \eta$ since $\tilde{A}_i$ is regular. But then

$$\eta_i e_i = \eta_i A v = v_i A \eta = 0$$

and hence $\eta_i = 0$, which however implies $\eta \in \ker \tilde{A}_i$, a contradiction with the regularity of $\tilde{A}_i$. This shows that $\gamma_i \neq 0$ for all $i = 2, \dots, N$, and hence γ must be strictly positive. □

The linear system $Qx = b$ defined by (6.1) is split into several homogeneous equations and one non-homogeneous equation. Since any $\gamma \in \mathrm{Ker}(A^T)$ automatically satisfies all the homogeneous ones, a solution $x = (w, v)$ can be computed by scaling γ to satisfy the only non-homogeneous constraint left.

Proposition 6.1.4. *Let X, Y, Z be nonempty scattered sets of nodes such that*

$$Y = \{y_i\}_{i=1}^{N_Y} \subset \overline{\Omega}, \quad Z = \{z_i\}_{i=1}^{N_Z} \subset \partial\Omega, \quad X = \{x_i\}_{i=1}^{N_X} = Y \cup Z, \quad Y \cap Z = \emptyset,$$

and let $L \in \mathbb{R}^{N_Y \times N_X}$ and $B \in \mathbb{R}^{N_Z \times N_X}$ be matrices associated to numerical differentiation formulas for operators $\mathcal{L} = -\Delta$ and $\mathcal{B} = \partial_\nu$ such that

1. *Matrices $-L$ and B have positive type.*
2. *All the rows of $-L$ and B sum to zero.*
3. *Any two rows of $A = (L^T, -B^T)^T$ are connected.*

Then, for all nonnegative functions $\hat{f} \colon \overline{\Omega} \to \mathbb{R}^+$ and all nonpositive functions $\hat{g} \colon \partial\Omega \to \mathbb{R}^-$ such that

$$(\hat{f}_{|Y}^T, -\hat{g}_{|Z}^T) \neq 0 \quad \text{and} \quad \int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma > 0$$

the linear system $Qx = b$ given by (6.1) has a unique solution $x = (w, v)$, and $x > 0$.

Proof. The square matrix $-A$ has positive type and all of its rows sum to zero, so it is a singular weakly diagonally dominant L-matrix. Moreover, any two rows of $-A$ are connected because A and $-A$ have the same matrix graph. Therefore, it follows from Proposition 2.3.6 and Lemma 6.1.3 that there exists $\gamma \in \mathrm{Ker}(-A^T) = \mathrm{Ker}(A^T)$ with $\gamma > 0$.

We now search for a solution to $Qx = b$ in the form $x = \alpha\gamma$, with $\alpha > 0$. By construction, $\alpha\gamma$ satisfies all the homogeneous equations in (6.1). Imposing equality also for the non-homogeneous equation, we obtain

$$\alpha = \frac{\int_\Omega \hat{f}\, d\mu - \int_{\partial\Omega} \hat{g}\, d\sigma}{(\hat{f}_{|Y}^T, -\hat{g}_{|Z}^T)\gamma} > 0.$$

Uniqueness of x follows from the observation in Lemma 6.1.3 that the kernel of A^T has dimension one. □

As explained in Section 4.7, positive quadrature weights (w, v), whenever they exist, can be found by minimizing the combined 1-norm $\|w\|_1 + \|v\|_1$ with non-homogeneous constraints $(\hat{f}, \hat{g}) \equiv (1, -1)$. Under the assumptions of Proposition 6.1.4, the strictly positive solution of $Qx = b$ can be found more efficiently by solving the regular square system

$$\begin{pmatrix} A^T & \mathbb{1} \\ \mathbb{1}^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = b. \tag{6.3}$$

Indeed, let x be the solution of $Qx = b$ with $\hat{f} \equiv 1$ and $\hat{g} \equiv -1$. Then $(x, 0)$ is a solution of (6.3). Conversely, let (x_1, λ_1) and (x_2, λ_2) be two solutions of (6.3). Since all the rows of A sum to zero,

$$\begin{aligned} A^T(x_1 - x_2) + \mathbb{1}(\lambda_1 - \lambda_2) = 0 \quad &\Rightarrow \quad \mathbb{1}^T A^T(x_1 - x_2) + \mathbb{1}^T\mathbb{1}(\lambda_1 - \lambda_2) = 0 \\ &\Rightarrow \quad \mathbb{1}^T\mathbb{1}(\lambda_1 - \lambda_2) = 0, \end{aligned}$$

which proves that $\lambda_1 = \lambda_2$. Now, by (6.3), it must be that $Qx_1 = b$ and $Qx_2 = b$, which implies $x_1 = x_2$ by the uniqueness result in Proposition 6.1.4.

To sum up, there exist sufficient conditions under which Framework 4.3.1 applied to positive numerical differentiation formulas for operators $\mathcal{L} = -\Delta$ and $\mathcal{B} = \partial_\nu$ leads to strictly positive quadrature weights which can be computed as the unique solution of the square and regular system (6.3). These conditions are very easy to satisfy in practice: the second condition in Proposition 6.1.4 follows from the exactness of the differentiation formulas for constants, and the third condition about connectedness of the matrix graph of A holds for any reasonable choice of sets of influence $S_{L,i}$ and $S_{B,i}$, as long as the sets are sufficiently large and the nodes in X are quasi-uniform.

Moment-free quadrature arising from positive numerical differentiation formulas has excellent theoretical properties, but its use in practice is severely limited by the low order of convergence that can be achieved. According to the error analysis in Chapter 4 and the numerical experiments in Chapter 5, we can expect the same order of convergence as the *consistency order* of the numerical differentiation formulas (equal to the order of polynomial exactness q minus the order k of the differential operator, see Definition 2.3.1). Unfortunately, it can be proved that there are no positive differentiation formulas for $\mathcal{L} = -\Delta$ with polynomial order $q \geq 5$, and no positive differentiation formulas for $\mathcal{B} = \partial_\nu$ with polynomial order $q \geq 3$, which limits a priori the order of convergence of our quadrature formulas to two [90].

A point which we have not addressed yet is whether positive numerical differentiation formulas exist on given sets X, Y, Z of scattered nodes in $\mathbb{R}^d$. Sufficient conditions for the existence of positive differentiation formulas for $\mathcal{L} = \Delta$ and $\mathcal{B} = \partial_\nu$ with consistency order 1 on scattered sets of nodes are given in [90]. In the same work, necessary conditions on the placement of nodes are also described. However, the question of whether numerical differentiation formulas with consistency order 2 can be reliably computed on scattered nodes is still open, even in $\mathbb{R}^2$.

It is well known that on tensor product grids with uniform step size h, the five-point stencil for $-\Delta$ in 2D

$$-\Delta u(x, y) \approx \frac{-u(x - h, y) - u(x, y - h) + 4u(x, y) - u(x + h, y) - u(x, y + h)}{h^2} \tag{6.4}$$

and its generalizations to higher dimensions have consistency order 2. One may therefore wonder if quadrature formulas of the same order can be obtained by placing nodes on a tensor product grid in the interior of the integration domain Ω, so that the five-point stencil formula can be used at all nodes sufficiently far from the boundary. The answer is positive, as proved in the next section.

6.2 Gridded nodes of Shortley-Weller type

Let Ω be a smooth domain in $\mathbb{R}^2$, and consider the nodes of a Cartesian grid with step size $h > 0$ lying inside Ω. This set of nodes Y can divided into two disjoint subsets: Y_{fb} containing interior nodes that lie sufficiently far from the boundary such that the segments connecting them to their left, right, top and bottom grid neighbors are included in Ω, and Y_{nb} containing the remaining grid nodes near the boundary of Ω. Moreover, let us consider a set of nodes Z on the boundary obtained as the first intersections with $\partial\Omega$ of the rays emanating from a node in Y_{nb} in the direction of its left, right, top or bottom grid neighbors for which the connecting segments are not contained in Ω. For a more precise description of this construction, see [51, Section 4.8.1]. In the following, we will refer to the union

$$X = \{x_i\}_{i=1}^{N_X} = Y_{\mathrm{fb}} \cup Y_{\mathrm{nb}} \cup Z, \tag{6.5}$$

as *gridded nodes of Shortley-Weller type*, and we will assume that the elements in X are ordered so that the nodes in Y_{fb}, Y_{nb}, and Z appear in the same order as in (6.5).

Consider now the Neumann problem with homogeneous boundary conditions

$$\text{Find } u \in C^4(\overline{\Omega}) \text{ such that } \quad \begin{cases} \Delta u = f & \text{in } \Omega, \\ \partial_\nu u = 0 & \text{on } \partial\Omega, \end{cases} \tag{6.6}$$

where f is sufficiently smooth and satisfies the compatibility condition $\int_\Omega f\,dx = 0$. The finite difference scheme specifically tailored to gridded nodes of Shortley-Weller type described in [14] generates a matrix

$$A = (A_{\mathrm{fb}}^T, A_{\mathrm{nb}}^T, A_{\mathrm{b}}^T)^T \in \mathbb{R}^{N_X \times N_X}$$

depending only on Ω and X, with the number of rows in the blocks $A_{\mathrm{fb}}, A_{\mathrm{nb}}, A_{\mathrm{b}}$ corresponding to the number of nodes in $Y_{\mathrm{fb}}, Y_{\mathrm{nb}}, Z$, respectively, and the entries of these submatrices given as coefficients of the equations in (2.4), (2.6) and (2.15) of [14]. In particular, the rows of A_{fb} are all defined as the five-point stencil (6.4). It is proved in the same paper that, for a sufficiently small grid step size $h > 0$, the matrix A has the following properties:

1. A is sparse, with five nonzero elements in each row of A_{fb} and A_{nb}, and at most four nonzero elements in each row of A_{b}.

2. A is singular, with $\ker A = \operatorname{span}\{\mathbb{1}\}$, see equations (2.19) and (2.20) in [14].

3. $-A$ becomes an M-matrix if any one of its rows is replaced by the unit vector with one on the diagonal.

Let y_{i_0} be an arbitrary element of Y_{fb}, and let $e_{i_0} \in \mathbb{R}^{N_X}$ be the unit vector whose i_0-th component is equal to one. The discrete numerical solution $\hat{u} \approx u_{|X}$ of the Neumann problem (6.6) is computed as the solution of the nonsingular linear system (compare [14, Eq. (2.17)])

$$\tilde{A}\hat{u} = \begin{pmatrix} \tilde{f}_{|Y_{\mathrm{fb}}} \\ f_{|Y_{\mathrm{nb}}} \\ Df_{|Z} \end{pmatrix}, \quad \text{with } \tilde{A} := \begin{pmatrix} \tilde{A}_{\mathrm{fb}} \\ A_{\mathrm{nb}} \\ A_{\mathrm{b}} \end{pmatrix},$$

where D is a diagonal matrix depending only on Ω and X, the matrix $\tilde{A}_{\mathrm{fb}}$ is obtained by replacing the i_0-th row of A_{fb} with $e_{i_0}^T$, and the vector $\tilde{f}_{|Y_{\mathrm{fb}}}$ is obtained by setting the i_0-th element of $f_{|Y_{\mathrm{fb}}}$ to zero. The matrix D is a key part of the construction of an $\mathcal{O}(h^2)$ approximation of positive type to the normal derivative at nodes in Z described in Section 2 of [14]. An exhaustive description of D is given in [13], which goes into more detail than [14], but for our purposes it is enough to know that D is a diagonal matrix with strictly positive diagonal entries, as explained in the following remark.

Remark 6.2.1. Let us switch to the notation used in [13] until the end of the remark. The set C_h denotes the set of nodes placed on $\partial\Omega$. For all nodes $P \in C_h$, Equation (2.11) in [13] with $\alpha = 0$ and $g = 0$ defines a matrix D such that the element of $Df_{|C_h}$ corresponding to node P is a multiple of $f(P)$, and more specifically, it is equal to

$$f(P) \sum_{i=1}^{3} a_i \frac{y_i^2}{2}. \tag{6.7}$$

In the paragraphs above Equation (2.11) it is explained that, for a sufficiently small h, all variables a_i and y_i can be chosen to be strictly positive. Therefore, D is a diagonal and strictly positive matrix.

We now have all the prerequisites to define the weight vector w of a quadrature formula arising from the finite difference scheme on X. Unlike the construction in Framework 4.3.1, the following approach will only lead to an interior quadrature formula (X, w), and not to the usual pair of quadrature formulas (Y, w) and (Z, v).

Let $h > 0$ be small enough that the matrix A defined above satisfies properties (a)–(c). By Lemma 6.1.3, there exists $\gamma \in \ker A^T$ with $\gamma_i > 0$ for all $i = 1, \ldots, N_X$. From a computational point of view, γ can be obtained as the unique solution of the rectangular linear system

$$A^T\gamma = 0, \quad e_1^T\gamma = 1,$$

which can be reformulated as the sparse nonsingular square system

$$\begin{pmatrix} A^T & e_1 \\ e_1^T & 0 \end{pmatrix} \begin{pmatrix} \gamma \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

with a dummy variable $\lambda \in \mathbb{R}$, so that its solution satisfies $\lambda = 0$, as in (6.3). We split γ into subvectors γ_{fb}, γ_{nb}, γ_{b} corresponding to the nodes in Y_{fb}, Y_{nb}, Z, respectively, so that

$$\gamma = \begin{pmatrix} \gamma_{\mathrm{fb}} \\ \gamma_{\mathrm{nb}} \\ \gamma_{\mathrm{b}} \end{pmatrix}.$$

Assume that the integral $\hat{I} = \int_\Omega \hat{f}\,dx$ is known for a positive function $\hat{f} \in C^0(\overline{\Omega})$ bounded from below by a strictly positive constant. Without loss of generality, we can assume that $\hat{f}(x) \geq 1$ for all $x \in \overline{\Omega}$. Then it holds that $\hat{I} \neq 0$. Note that we may use $\hat{f} \equiv 1$ if the area of Ω is known, in which case $\hat{I} = |\Omega|$. We set

$$w = \frac{\hat{I}}{\gamma_D^T \hat{f}_{|X}} \gamma_D, \quad \text{where } \gamma_D := \begin{pmatrix} \gamma_{\mathrm{fb}} \\ \gamma_{\mathrm{nb}} \\ D\gamma_{\mathrm{b}} \end{pmatrix}. \tag{6.8}$$

The vector w is strictly positive since $\gamma > 0$, D is a diagonal matrix with strictly positive diagonal entries, and $\hat{f}_{|X} > 0$.

We can now prove an error estimate for the positive quadrature formula (X, w).

Proposition 6.2.2. *Assume that Ω is a 2D domain with smooth boundary and that the integral $\hat{I}$ of a function $\hat{f} \in H^s(\Omega)$ such that $s > 3$ and $\hat{f} \geq 1$ is known. For a sufficiently small $h > 0$, define the node set $X \subset \overline{\Omega}$ and the weight vector $w > 0$ as in* (6.5) *and* (6.8)*. Then, for all $f \in H^s(\Omega)$ with $s > 3$, it holds that*

$$\left| \int_\Omega f\,dx - \sum_{i=1}^{N_X} w_i f(x_i) \right| \leq c h^2 \left| \log h \right|,$$

for some constant c that is independent of h and only depends on Ω, $\hat{f}$, and f.

Proof. Choose a node $y_{i_0} \in Y_{\mathrm{fb}}$ such that

$$\gamma_{i_0} = \min\{\gamma_i \mid y_i \in Y_{\mathrm{fb}}\}.$$

Since

$$\sum_{y_i \in Y_{\mathrm{fb}}} \gamma_i \leq \sum_{y_i \in Y_{\mathrm{fb}}} \gamma_i \hat{f}(y_i) \leq \gamma_D^T \hat{f}_{|X}$$

and $|Y_{\mathrm{fb}}| \geq c_1/h^2$ for some $c_1 > 0$ independent of h because Ω is an open set, we have

$$\gamma_{i_0} \leq \frac{1}{c_1} h^2 \gamma_D^T \hat{f}_{|X}. \tag{6.9}$$

Let $\alpha = \hat{I}^{-1} \int_\Omega f\,dx$, so that $\int_\Omega (f - \alpha\hat{f})\,dx = 0$, and denote by u the solution of the boundary value problem

$$\begin{cases} \Delta u = f - \alpha\hat{f} & \text{in } \Omega, \\ \partial_\nu u = 0 & \text{on } \partial\Omega, \end{cases}$$

satisfying $u(y_{i_0}) = 0$. Let $\hat{u} \in \mathbb{R}^{N_X}$ be the unique solution of

$$\tilde{A}\hat{u} = \begin{pmatrix} \tilde{A}_{\mathrm{fb}} \\ A_{\mathrm{nb}} \\ A_{\mathrm{b}} \end{pmatrix} \hat{u} = \begin{pmatrix} \widetilde{(f - \alpha\hat{f})}_{|Y_{\mathrm{fb}}} \\ (f - \alpha\hat{f})_{|Y_{\mathrm{nb}}} \\ D(f - \alpha\hat{f})_{|Z} \end{pmatrix}.$$

By construction, $\hat{u}_{i_0} = 0$. According to the error estimate in [14, Eq. (2.41)] for the finite difference scheme that is being considered,

$$\left\| u_{|X} - \hat{u} \right\|_\infty \leq c_2 h^2 \left| \log h \right| \tag{6.10}$$

as soon as $u \in C^4(\overline{\Omega})$, where c_2 once again does not depend on h, but only on Ω, f, $\hat{f}$. By Proposition 2.1.4, the solution u of the boundary value problem belongs to $H^s(\Omega)$ with $s > 5$. Therefore, by the Sobolev embedding theorem, it follows that $u \in C^4(\overline{\Omega})$, as required by the error estimate. By the definition of γ, we have that

$$
\begin{aligned}
0 = \gamma^T A\hat{u} = \gamma^T (A - \tilde{A})\hat{u} + \gamma^T \tilde{A}\hat{u} &= \gamma_{i_0}\big((A\hat{u})_{i_0} - \hat{u}_{i_0}\big) + \gamma^T \begin{pmatrix} \widetilde{(f - \alpha\hat{f})|_{Y_{\mathrm{fb}}}} \\ (f - \alpha\hat{f})|_{Y_{\mathrm{nb}}} \\ D(f - \alpha\hat{f})|_Z \end{pmatrix} \\
&= \gamma_{i_0}(A\hat{u})_{i_0} + \gamma_D^T \begin{pmatrix} (f - \alpha\hat{f})|_{Y_{\mathrm{fb}}} \\ (f - \alpha\hat{f})|_{Y_{\mathrm{nb}}} \\ (f - \alpha\hat{f})|_Z \end{pmatrix} - \gamma_{i_0}\big(f(y_{i_0}) - \alpha\hat{f}(y_{i_0})\big) \\
&= \gamma_{i_0}\big((A\hat{u})_{i_0} - \Delta u(y_{i_0})\big) + \gamma_D^T f|_X - \alpha\gamma_D^T \hat{f}|_X,
\end{aligned}
$$

which implies, in view of (6.9), that

$$
\left|\gamma_D^T f|_X - \alpha\gamma_D^T \hat{f}|_X\right| = \gamma_{i_0}\left|(A\hat{u})_{i_0} - \Delta u(y_{i_0})\right| \le \frac{1}{c_1} h^2 \gamma_D^T \hat{f}|_X \left|(A\hat{u})_{i_0} - \Delta u(y_{i_0})\right|. \tag{6.11}
$$

By substituting the definitions of w and α, we get

$$
\gamma_D^T f|_X - \alpha\gamma_D^T \hat{f}|_X = \hat{I}^{-1}\gamma_D^T \hat{f}|_X \left(\sum_{i=1}^{N_X} w_i f(x_i) - \int_\Omega f\,dx\right),
$$

from which we can estimate the quadrature error using (6.11):

$$
\begin{aligned}
\left|\sum_{i=1}^{N_X} w_i f(x_i) - \int_\Omega f\,dx\right| &= \frac{\hat{I}}{\gamma_D^T \hat{f}|_X}\left|\gamma_D^T f|_X - \alpha\gamma_D^T \hat{f}|_X\right| \\
&\le \frac{\hat{I}}{c_1}\left|(A\hat{u})_{i_0} - \Delta u(y_{i_0})\right| h^2. \qquad (6.12)
\end{aligned}
$$

To complete the proof, the term $|(A\hat{u})_{i_0} - \Delta u(y_{i_0})|$ remains to be analyzed:

$$
\left|(A\hat{u})_{i_0} - \Delta u(y_{i_0})\right| \le \left|(Au|_X)_{i_0} - \Delta u(y_{i_0})\right| + \left|(A\hat{u})_{i_0} - (Au|_X)_{i_0}\right|.
$$

Let $\partial_{x_1}^4$ and $\partial_{x_2}^4$ denote the fourth partial derivatives along the Cartesian axes. Keeping in mind that $(Au|_X)_{i_0}$ is the five-point stencil estimate of $\Delta u(y_{i_0})$ and that the 1-norm of the i_0-th row $a_{i_0}^T$ of A is $8h^{-2}$, we obtain using (6.10),

$$
\begin{aligned}
\left|(A\hat{u})_{i_0} - \Delta u(y_{i_0})\right| &\le \frac{1}{12}h^2\left(\left\|\partial_{x_1}^4 u\right\|_{C(\overline{\Omega})} + \left\|\partial_{x_2}^4 u\right\|_{C(\overline{\Omega})}\right) + \left\|\hat{u} - u|_X\right\|_\infty \|a_{i_0}^T\|_1 \\
&\le \frac{1}{12}h^2\left(\left\|\partial_{x_1}^4 u\right\|_{C(\overline{\Omega})} + \left\|\partial_{x_2}^4 u\right\|_{C(\overline{\Omega})}\right) + c_2 h^2 \left|\log h\right| 8h^{-2} \le c_3 \left|\log h\right|.
\end{aligned}
$$

Substituting back in (6.12) completes the proof. □

Bibliography

[1] P. Alliez, S. Tayeb, and C. Wormser. 2D and 3D fast intersection and distance computation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0 edition, 2024.

[2] P. Antolin, A. Buffa, and M. Martinelli. Isogeometric analysis on V-reps: first results. *Computer Methods in Applied Mechanics and Engineering*, 355:976–1002, 2019.

[3] D. N. Arnold, L. R. Scott, and M. Vogelius. Regular inversion of the divergence operator with Dirichlet boundary conditions on a polygon. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 15(2):169–192, 1988.

[4] I. Babuška and J. M. Melenk. The partition of unity method. *International journal for numerical methods in engineering*, 40(4):727–758, 1997.

[5] J. A. Bærentzen and H. Aanaes. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, 2005.

[6] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *Journal of Computational Physics*, 332:257–273, 2017.

[7] C. Beltrán, N. Corral, and J. G. C. del Rey. Discrete and continuous Green energy on compact manifolds. *Journal of Approximation Theory*, 237:160–185, 2019.

[8] T. Belytschko, J.-S. Chen, and M. Hillman. *Meshfree and particle methods: fundamentals and applications*. John Wiley & Sons, 2023.

[9] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[10] Å. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[11] J. L. Blanco and P. K. Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. `https://github.com/jlblancoc/nanoflann`, 2014. Accessed: 2024-10-15.

[12] C. de Boor and G. Fix. Spline approximation by quasiinterpolants. *Journal of Approximation Theory*, 8(1):19–45, 1973.

[13] J. Bramble and B. Hubbard. Approximation of solutions of mixed boundary value problems for Poisson's equation by finite differences. *Journal of the ACM (JACM)*, 12(1):114–123, 1965.

[14] J. Bramble and B. Hubbard. A finite difference analog of the Neumann problem for Poisson's equation. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(1):1–14, 1965.

[15] B.-D. Chu, F. Martin, and U. Reif. Stabilization of spline bases by extension. *Advances in Computational Mathematics*, 48(3):23, 2022.

[16] P. G. Ciarlet. Discrete maximum principle for finite-difference operators. *Aequationes mathematicae*, 4(3):338–352, 1970.

[17] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[18] T. A. Davis. Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–22, 2011.

[19] T. A. Davis. SuiteSparse: A suite of sparse matrix algorithms. Version 7.7.0, `https://github.com/DrTimothyAldenDavis/SuiteSparse`, 2024.

[20] O. Davydov. mFDlab: A laboratory for meshless finite difference (mFD) methods, `https://bitbucket.org/meshlessFD/mfdlab`, 2020.

[21] O. Davydov. Approximation with conditionally positive definite kernels on deficient sets. In M. N. Gregory E. Fasshauer and L. L. Schumaker, editors, *Approximation Theory XVI: Nashville 2019*, pages 27–38. Springer Berlin Heidelberg, 2021.

[22] O. Davydov. Error bounds for a least squares meshless finite difference method on closed manifolds. *Advances in Computational Mathematics*, 49, 2023. Article number 48.

[23] O. Davydov, D. T. Oanh, and N. M. Tuong. Improved stencil selection for meshless finite difference methods in 3D. *Journal of Computational and Applied Mathematics, 115031*, 2023.

[24] O. Davydov, J. Prasiswa, and U. Reif. Two-stage approximation methods with extended B-splines. *Mathematics of Computation*, 83(286):809–833, 2014.

[25] O. Davydov and R. Schaback. Error bounds for kernel-based numerical differentiation. *Numerische Mathematik*, 132(2):243–269, 2016.

[26] O. Davydov and R. Schaback. Minimal numerical differentiation formulas. *Numerische Mathematik*, 140(3):555–592, 2018.

[27] O. Davydov and R. Schaback. Optimal stencils in Sobolev spaces. *IMA Journal of Numerical Analysis*, 39(1):398–422, 2019.

[28] F. de Prenter, C. V. Verhoosel, E. H. van Brummelen, M. G. Larson, and S. Badia. Stability and conditioning of immersed finite element methods: analysis and remedies. *Archives of Computational Methods in Engineering*, 30(6):3617–3656, 2023.

[29] L. Demkowicz, A. Karafiat, and T. Liszka. On some convergence results for fdm with irregular mesh. *Computer Methods in Applied Mechanics and Engineering*, 42(3):343–355, 1984.

[30] S. C. Divi, C. V. Verhoosel, F. Auricchio, A. Reali, and E. H. Van Brummelen. Error-estimate-based adaptive integration for immersed isogeometric analysis. *Computers & Mathematics with Applications*, 80(11):2481–2516, 2020.

[31] U. Duh, G. Kosec, and J. Slak. Fast variable density node generation on parametric surfaces with application to mesh-free methods. *SIAM Journal on Scientific Computing*, 43(2):A980–A1000, 2021.

[32] U. Duh, V. Shankar, and G. Kosec. Discretization of non-uniform rational b-spline (nurbs) models for meshless isogeometric analysis. *Journal of scientific computing*, 100(2):51, 2024.

[33] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[34] R. T. Farouki. Trimmed-surface algorithms for the evaluation and interrogation of solid boundary representations. *IBM Journal of Research and Development*, 31(3):314–334, 1987.

[35] B. Fornberg and N. Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2015.

[36] B. Fornberg and J. M. Martel. On spherical harmonics based numerical quadrature over the surface of a sphere. *Advances in Computational Mathematics*, 40:1169–1184, 2014.

[37] R. Franke. *A critical comparison of some methods for interpolation of scattered data*. Naval Postgraduate School Monterey, CA, 1979.

[38] Fraunhofer Society. MESHFREE. `https://www.meshfree.eu`.

[39] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[40] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[41] J. Glaubitz. Stable high-order cubature formulas for experimental data. *Journal of Computational Physics*, 447:110693, 2021.

[42] J. Glaubitz. Construction and application of provable positive and exact cubature formulas. *IMA Journal of Numerical Analysis*, 43(3):1616–1652, 2023.

[43] GrabCAD Community. Gear shaft (3D model in STEP file format). `https://grabcad.com/library/gear-shaft-49`.

[44] GrabCAD Community. Stanford bunny (3D model in STEP file format). `https://grabcad.com/library/stanford-bunny-solid-model`.

[45] GrabCAD Community. Turbine blade (3D model in STEP file format). https://grabcad.com/library/turbine-blade-22.

[46] GrabCAD Community. Winged corkscrew (3D model in STEP file format). https://grabcad.com/library/winged-corkscrew-2.

[47] P. Grisvard. *Elliptic Problems in Nonsmooth Domains*. Monographs and studies in mathematics. Pitman Advanced Pub. Program, 1985.

[48] D. Gunderman, K. Weiss, and J. A. Evans. High-accuracy mesh-free quadrature for trimmed parametric surfaces and volumes. *Computer-Aided Design*, 141:103093, 2021.

[49] D. Gunderman, K. Weiss, and J. A. Evans. Spectral mesh-free quadrature for planar regions bounded by rational parametric curves. *Computer-Aided Design*, 130:102944, 2021.

[50] W. Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95. Springer, 1994.

[51] W. Hackbusch. *Elliptic Differential Equations: Theory and Numerical Treatment*, volume 18. Springer, 2017.

[52] Q. Huangfu and J. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

[53] International Organization for Standardization. ISO 10303: Industrial automation systems and integration — Product data representation and exchange. https://www.iso.org/standard/51556.html, 2024. ISO Standard Series.

[54] T. Jacquemin, P. Suchde, and S. P. Bordas. Smart cloud collocation: geometry-aware adaptivity directly from cad. *Computer-Aided Design*, 154:103409, 2023.

[55] T. Jacquemin, S. Tomar, K. Agathos, S. Mohseni-Mofidi, and S. P. A. Bordas. Taylor-series expansion based numerical methods: A primer, performance benchmarking and new approaches for problems with non-smooth solutions. *Archives of Computational Methods in Engineering*, Aug 2019.

[56] P. S. Jensen. Finite difference techniques for variable grids. *Computers & Structures*, 2(1-2):17–29, 1972.

[57] D. S. Jerison and C. E. Kenig. The Neumann problem on Lipschitz domains. *Bull. Amer. Math. Soc.(NS)*, 4(1):203–207, 1981.

[58] J. T. Kajiya. Ray tracing parametric patches. *ACM SIGGRAPH Computer Graphics*, 16(3):245–254, 1982.

[59] E. J. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—i surface approximations and partial derivative estimates. *Computers & Mathematics with applications*, 19(8-9):127–145, 1990.

[60] E. J. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & mathematics with applications*, 19(8-9):147–161, 1990.

[61] A. Kolar-Požun, M. Jančič, and G. Kosec. A superconvergence result in the rbf-fd method. In *Journal of Physics: Conference Series*, volume 2766, page 012161. IOP Publishing, 2024.

[62] A. R. Krommer and C. W. Ueberhuber. *Computational integration*. SIAM, 1998.

[63] A. P. Lawrence, M. E. Nielsen, and B. Fornberg. Node subsampling for multilevel meshfree elliptic PDE solvers. *Computers & Mathematics with Applications*, 164:79–94, 2024.

[64] C. K. Lee, X. Liu, and S. C. Fan. Local multiquadric approximation for solving boundary value problems. *Computational Mechanics*, 30:396–409, 2003.

[65] J. M. Lee and J. M. Lee. *Smooth manifolds*. Springer, 2012.

[66] N. Lehmann, L.-B. Maier, S. Odathuparambil, and U. Reif. Ambient approximation on hypersurfaces. *Constructive Approximation*, 49:175–190, 2019.

[67] D. Levin. Stable integration rules with scattered integration points. *Journal of computational and applied mathematics*, 112(1-2):181–187, 1999.

[68] Z. Li. An overview of the immersed interface method and its applications. *Taiwanese journal of mathematics*, 7(1):1–49, 2003.

[69] T. Liszka, C. Duarte, and W. Tworzydlo. hp-meshless cloud method. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):263–288, 1996.

[70] T. Liszka and J. Orkisz. The finite difference method at arbitrary irregular grids and its application in applied mechanics. *Computers & Structures*, 11(1-2):83–95, 1980.

[71] G.-R. Liu. *Meshfree methods: moving beyond the finite element method*. CRC press, 2009.

[72] G.-R. Liu and Y.-T. Gu. *An introduction to meshfree methods and their programming*. Springer Science & Business Media, 2005.

[73] 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.

[74] R. Löhner and E. Onate. An advancing front point generation technique. *Communications in numerical methods in engineering*, 14(12):1097–1108, 1998.

[75] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, 82:1013–1024, 1977.

[76] B. Marussig and T. J. Hughes. A review of trimming in isogeometric analysis: challenges, data exchange and simulation aspects. *Archives of computational methods in engineering*, 25:1059–1127, 2018.

[77] C. A. Micchelli and T. J. Rivlin. Lectures on optimal recovery. In *Numerical Analysis Lancaster 1984: Proceedings of the SERC Summer School held in Lancaster, England, Jul. 15–Aug. 3, 1984*, pages 21–93. Springer, 1985.

[78] D. Mitrea, M. Mitrea, and S. Monniaux. The Poisson problem for the exterior derivative operator with Dirichlet boundary condition on nonsmooth domains. *Communications on Pure and Applied Mathematics*, 7(6):1295–1333, 2008.

[79] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational mechanics*, 10(5):307–318, 1992.

[80] P. Negi and P. Ramachandran. Sphgeom. `https://gitlab.com/pypr/sph_geom`, 2019.

[81] P. Negi and P. Ramachandran. Algorithms for uniform particle initialization in domains with complex boundaries. *Computer Physics Communications*, 265:108008, 2021.

[82] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

[83] NOGRID GmbH. NOGRID. `https://www.nogrid.com`.

[84] Open Cascade SAS. Open CASCADE Technology. `https://www.opencascade.com/open-cascade-technology/`.

[85] Open Cascade Technologies. Demo geometry from CAD Assistant (3D model in STEP file format). `https://www.opencascade.com/products/cad-assistant/`.

[86] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[87] N. Perrone and R. Kao. A general finite difference method for arbitrary meshes. *Computers & Structures*, 5(1):45–57, 1975.

[88] J. A. Reeger and B. Fornberg. Numerical quadrature over smooth surfaces with boundaries. *Journal of Computational Physics*, 355:176–190, 2018.

[89] R. J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software (TOMS)*, 14(2):139–148, 1988.

[90] B. Seibold. *M-Matrices in meshless finite difference methods*. Phd dissertation, University of Kaiserslautern, 2006.

[91] B. Seibold. Minimal positive stencils in meshfree finite difference methods for the poisson equation. *Computer Methods in Applied Mechanics and Engineering*, 198(3-4):592–601, 2008.

[92] L. F. Shampine. Matlab program for quadrature in 2D. *Applied Mathematics and Computation*, 202(1):266–274, 2008.

[93] V. Shankar, R. M. Kirby, and A. L. Fogelson. Robust node generation for mesh-free discretizations on irregular domains and surfaces. *SIAM Journal on Scientific Computing*, 40(4):A2584–A2608, 2018.

[94] C. Shu, H. Ding, and K. Yeo. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations. *Computer methods in applied mechanics and engineering*, 192(7-8):941–954, 2003.

[95] J. Slak and G. Kosec. On generation of node distributions for meshless pde discretizations. *SIAM journal on scientific computing*, 41(5):A3202–A3229, 2019.

[96] J. Slak and G. Kosec. Medusa: A c++ library for solving pdes using strong form mesh-free methods. *ACM Transactions on Mathematical Software*, 2021.

[97] S. L. Sobolev. Formulas for mechanical cubatures in n-dimensional space. In *Doklady Akademii Nauk*, volume 137, pages 527–530. Russian Academy of Sciences, 1961.

[98] S. L. Sobolev and V. Vaskevich. *The theory of cubature formulas*, volume 415. Springer Science & Business Media, 1997.

[99] H. Sohr. *The Navier-Stokes equations: An elementary functional analytic approach*. Birkhäuser, 2001.

[100] A. Sommariva and M. Vianello. Computing approximate Fekete points by QR factorizations of Vandermonde matrices. *Computers & Mathematics with Applications*, 57(8):1324 – 1336, 2009.

[101] A. Sommariva and M. Vianello. Gauss–Green cubature and moment computation over arbitrary geometries. *Journal of Computational and Applied Mathematics*, 231(2):886–896, 2009.

[102] A. Sommariva and M. Vianello. RBF moment computation and meshless cubature on general polygonal regions. *Applied Mathematics and Computation*, 409:126375, 2021.

[103] A. Sommariva and M. Vianello. inrs: implementing the indicator function of nurbs-shaped planar domains. *Applied Mathematics Letters*, 130:108026, 2022.

[104] A. Sommariva and M. Vianello. Low cardinality positive interior cubature on NURBS-shaped domains. *BIT Numerical Mathematics*, 63(2):22, 2023.

[105] J. Spainhour, D. Gunderman, and K. Weiss. Robust containment queries over collections of rational parametric curves via generalized winding numbers. *arXiv preprint arXiv:2403.17371*, 2024.

[106] A. H. Stroud. *Approximate calculation of multiple integrals*. Prentice Hall, 1971.

[107] P. Suchde, T. Jacquemin, and O. Davydov. Point cloud generation for meshfree methods: An overview. *Archives of Computational Methods in Engineering*, 30(2):889–915, 2023.

[108] P. Suchde and J. Kuhnert. A meshfree generalized finite difference method for surface PDEs. *Computers & Mathematics with Applications*, 78(8):2789–2805, 2019.

[109] T. Tao. *An introduction to measure theory*, volume 126. American Mathematical Soc., 2011.

[110] M. E. Taylor. *Partial differential equations I: Basic theory*. Springer, 1996.

[111] A. I. Tolstykh and D. Shirobokov. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. *Computational Mechanics*, 33(1):68–79, 2003.

[112] M. Turner, J. Peiró, and D. Moxey. A variational framework for high-order mesh generation. *Procedia Engineering*, 163:340–352, 2016.

[113] K. van der Sande. Node generation. `https://github.com/kierav/node_generation`, 2019.

[114] K. van der Sande and B. Fornberg. Fast variable density 3-d node generation. *SIAM Journal on Scientific Computing*, 43(1):A242–A257, 2021.

[115] H. Wendland. Fast evaluation of radial basis functions: Methods based on partition of unity. 2002.

[116] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004.

[117] G. B. Wright and B. Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. *Journal of Computational Physics*, 212(1):99–123, 2006.