# NODAL DISTRIBUTIONS ON THE HIGH-DIMENSIONAL SIMPLEX FOR HIGH-ORDER INTERPOLATION AND INTEGRATION

Albert Jiménez-Ramos



Doctoral Thesis
Advisor: Xevi Roca Navarro
Co-advisor: Abel Gargallo-Peiró
Barcelona, May 2023

Departament de Matemàtiques
Programa de Doctorat en Matemàtica Aplicada

*Dedicat a les persones i experiències del passat que em faran la persona que seré.*

## Abstract

### Nodal distributions on the high-dimensional simplex for high-order interpolation and integration

Albert Jiménez-Ramos

To simulate unsteady phenomena on complex moving geometry, computational scientists and engineers have been interested in unstructured space-time discretizations. These space-time discretizations aim to overcome some issues of standard time marching methods by considering a unique mesh that discretizes the steady space-time geometry described by the moving spatial geometry. Unfortunately, they have not been widely adopted for 3D unsteady high-fidelity simulations because many fundamental capabilities related to high-order approximation to geometry and solution must still be developed in 4D. These capabilities include performing high-order numerical interpolation and integration on complex space-time geometry.

To address these issues, this thesis aims to demonstrate interpolation-aware optimization of nodal distributions on the high-dimensional simplex constrained to fulfill either feasible interpolation or feasible integration with the point coordinates as design variables. To this end, it proposes the following contributions. First, to interpolate a curved subdivision model obtained from a linear triangular mesh, we devise an equispaced nodal geometry representation method. Second, to additionally preserve the required simulation intent, we devise a nodal modeling method. Third, to estimate the interpolation error of a nodal distribution, we propose a point refinement method. Fourth, to explore the nodal distributions that are the heuristically best local minima of an interpolation error proxy, we propose a specific-purpose optimization method. Fifth, to minimize the interpolation error of an initial nodal distribution, we propose a constrained optimization method. Finally, to enforce small interpolation error and exact integration, we propose a constrained optimization problem.

In conclusion, this thesis demonstrates interpolation-aware optimization of nodal distributions on the high-dimensional simplex for feasible interpolation or integration. To this end, it proposes novel nodal methods to model and represent the simulation intent and to explore and optimize nodal distributions. These novelties will contribute to performing high-order numerical interpolation and integration on complex 4D geometry. Hence, they will help to exploit the benefits of high-fidelity 4D space-time simulation for 3D moving complex geometries.

# Acknowledgments

I am the author of this thesis, but I am not the only one who wrote it. The following words are dedicated to those people.

Aquesta tesi està signada per mi, l'Albert Jiménez, però allà on posi el meu nom hi heu d'afegir uns quants més.

S'hi ha d'afegir el nom dels meus tutors. Abel i Xevi, Xevi i Abel, moltes gràcies per apostar per mi com a doctorand i deixar-me acompanyar-vos en aquesta etapa de la vostra carrera. Us agraeixo el vostre recolzament en l'àmbit professional i, sobre tot, personal. Gràcies per la vostra paciència, llibertat i empatia. Abel, et dono les gràcies per aquella hamburguesa a Mataró, l'entrepà de pernil a casa els teus avis i la companyia als migdies. Xevi, tot i que des del juny del 2021 et veig canviat, t'agraeixo la confiança que has dipositat en mi, la teva comprensió en els moments de dificultat i haver-te pogut explicar les meves preocupacions tant professionals com personals. Vau començar com a tutors i us heu convertit en amics, em sap greu.

S'hi ha d'afegir el nom de la meva familia. Papa y mama, gracias. Gracias por el esfuerzo que habéis hecho para poder darme la educación que tengo. Gracias por invertir vuestro tiempo en mí. Espero que, en un futuro, sepa valorar el esfuerzo que seguís haciendo por mí y estar a la altura. Marta, muchas gracias por todo tu apoyo, tu interés y confianza. Os quiero. Avis y yayas, muchas gracias por el soporte que me habéis dado. Familia Aragón Ramos, gracias por el respeto y confianza con el que siempre me habéis tratado. Espero que podáis decir con orgullo que he sido un buen hijo, un buen hermano, un buen nieto, un buen sobrino y un buen primo.

S'hi ha d'afegir el nom de la meva nova família. Anna, gràcies per fer-me costat en tots els moments de dificultat. Gràcies pels somriures, les abraçades i els petons. M'agradaria ser el teu company de pis durant molt de temps. Estic enamorat de tu. T'estimo molt.

S'hi ha d'afegir el nom dels meus amics i companys. Eloi Ruiz, moltes gràcies per intentar transmetre'm els teus coneixements, he gaudit molt les discussions amb tu. Guillem i Guillermo, ha estat un plaer compartir amb vosaltres les confidències i preocupacions durant la nostra etapa com a estudiants de doctorat. Sergi, Marc, Miquel, Agus, amb les nostres converses i trobades m'heu ajudat, de forma conscient o inconscient, a tirar endavant aquesta tesi. També signen aquesta tesi els meus amics que m'han acompanyat des de la carrera: Víctor, Miguel i Gerard, us hi estic molt agraït. Moltes inquietuds no haguessin sorgit si no fos per les converses tècniques que

hem tingut.

Finalment, també vull fer menció a la gent de la FME i del BSC que m'ha ajudat a finalitzar aquesta tesi. Carme Capdevila, gràcies per la teva ajuda en tots els temes burocràtics. Laura Gutiérrez, gràcies per fer-me la vida tan fàcil sempre que t'ho he demanat. Joan i Gerard, gràcies per la flexibilitat que m'heu donat en aquests últims mesos.

I ara, la tesi.

# Contents

# List of Figures

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Motivation and background

To simulate unsteady phenomena on complex moving geometry, computational scientists and engineers have been interested in unstructured space-time discretizations (Wang and Persson, 2015; Murman et al., 2016; Badia and Olm, 2017; Jayasinghe et al., 2018). This interest has been prompted by the potential to overcome some issues of standard time marching unstructured methods for unsteady simulations on moving geometries. To illustrate some of these issues, let us consider a time-marching simulation of the flow in a turbomachine. To obtain the fluid dynamics, standard time-marching unstructured methods perform a sequence of time steps, facilitating parallel computations in space but not in time. These time steps can be different at each step, but they must be the same for all the spatial regions. Thus, it is not possible to feature different time resolutions in different spatial locations. This fixed time resolution hampers the possibility of locally adapting the space and time resolution to the space-time solution. Furthermore, many non-fully implicit high-order time discretizations have stability issues beyond the fourth order (Wanner and Hairer, 1996). Finally, to simulate the motion of the rotor blades, the simulation code has to feature coupling techniques for non-conformal sliding meshes (Chaurasia, 2010).

In contrast, space-time discretizations consider a unique mesh that discretizes the steady space-time geometry described by the moving spatial geometry (Wang and Persson, 2015; Murman et al., 2016; Jayasinghe et al., 2018). Having a unique mesh allows parallelizing simultaneously in space and time, yet this parallelization is limited

by causality (Badia and Olm, 2017). Moreover, we can generate an initial space-time mesh featuring different time resolutions in different spatial regions. Then, it is possible to use this initial mesh to locally adapt the space-time mesh to feature different resolutions not only in space but also along the time dimension (Caplan et al., 2020; Belda-Ferrín et al., 2019, 2023). Moreover, high-order space-time discretizations allow stable discretizations beyond the fourth order in space as well as in time (Wanner and Hairer, 1996). Finally, because the motion of the spatial geometry results in a steady space-time geometry, there is no need to use sliding meshes. Unfortunately, although space-time discretizations have these potential advantages, they have not been widely adopted for 3D unsteady high-fidelity simulations because many fundamental capabilities related to high-order approximation to geometry and solution must still be developed in 4D. These capabilities include performing high-order numerical interpolation and integration on complex space-time geometry (Frontin et al., 2021).

*Developing methods for high-order numerical interpolation and integration on complex 4D geometry suitable for simulation is the main challenge of this thesis.* This development is relevant because it will contribute to enabling 4D high-order space-time discretizations for 3D unsteady high-fidelity simulation.

### 1.1.1 Representation and modeling of the simulation intent

Flow simulations with unstructured high-order methods require curved meshes that approximate a curved boundary representation preserving the simulation intent (Persson and Peraire, 2008; Chaurasia et al., 2012; Johnen et al., 2013; Gargallo-Peiró, 2014; Gargallo-Peiró et al., 2015c,b; Ruiz-Gironés et al., 2016b,a; Moxey et al., 2016; Thakur et al., 2009; White et al., 2005; Shapiro et al., 2011; Nolan et al., 2015). To model the previous features, it is standard to use CAD boundary representations based on trimmed NURBS, but these trim-based models might violate the simulation intent since they might present unintended gaps or discontinuities of the normal vector on irregular points and curves adjacent to trimmed surfaces. This issue can be circumvented using fine elements yet coarser than the model tolerance, but since these elements are planar, this approximation does not feature normal vector continuity through any triangular edge, and thus, it is not adequate for flow simulation. Nevertheless, we can convert the triangular mesh to a gap-free curved geometry model that features normal vector continuity by using successive applications of a subdivi-

sion scheme (Persson et al., 2006) or degree three Bézier polynomials (Loseille and Rochery, 2023). Unfortunately, no approach has devised a method to model the simulation intent and represent curved geometry without gaps with a $\mathcal{C}^1$-continuous piecewise polynomial parameterization.

*Representing and modeling the simulation intent on complex geometry is the first research problem of this thesis.* Without this feature, boundary representations based on straight-sided discretizations may be inadequate for flow simulations. This is so because linear triangulations do not explicitly feature curvature information and thus they fail to preserve the simulation intent.

### 1.1.2 Numerical interpolation

To reduce the interpolation error, nodal distributions in the simplex are obtained by optimizing an objective function that has as design variables either the point coordinates (Heinrichs, 2005; Briani et al., 2012; Rapetti et al., 2012; Roth, 2005) or a parameter controlling an explicit warp and blend of the boundary nodes (Blyth and Pozrikidis, 2005; Warburton, 2006; Luo and Pozrikidis, 2006; Isaac, 2020). The objective function corresponds either to the non-differentiable infinity norm of the interpolation operator, known as Lebesgue constant, or to a differentiable surrogate. When the minimization of a differentiable surrogate is considered (Fekete, 1923; Bos, 1983; Chen and Babuška, 1995; Hesthaven, 1998; Taylor et al., 2000; Roth, 2005; Briani et al., 2012; Van Barel et al., 2014; Chen and Babuška, 1996; Hesthaven and Teng, 2000), nodal distributions have been obtained only up to 3D. When the minimization of the Lebesgue constant is considered (Heinrichs, 2005; Briani et al., 2012; Rapetti et al., 2012; Roth, 2005), there are no works using the node coordinates as design variables and consistently exploiting first-order derivatives of the Lebesgue function.

*Numerical interpolation of functions in the simplex up to four dimensions is the second research problem of this thesis.* Nodal distributions with optimal interpolation properties have been used as a proxy of good nodal distributions for the nodal high-order unstructured methods such as the discontinuous (Hesthaven and Warburton, 2007) and continuous Galerkin (Karniadakis and Sherwin, 2005) methods. Without quasi-optimal interpolation nodal distributions, the accuracy of interpolating and solving the Vandermonde systems could be reduced because non-optimal distributions are not specifically devised for interpolation.

### 1.1.3 Numerical interpolation and integration

Linear problems with the mass matrix as system matrix appear in many problems such as in explicit time integration and $\mathcal{L}_2$ projections to approximate functions. The collocation of the interpolation points with the quadrature points yields a diagonal mass matrix. This fact is a key aspect of the spectral element method (Patera, 1984), it leads to evaluating the functions only on the interpolation points (Hesthaven and Warburton, 2007; Kopriva and Gassner, 2010), it improves the numerical stability for specific finite element methods (Jameson et al., 2012; Williams and Jameson, 2013; Williams, 2013; Witherden and Vincent, 2014), and it is also of interest in space-time discretizations (Williams et al., 2020). To impose the exact integration of polynomials, several numerical quadratures for the simplex have been studied (Lyness and Jespersen, 1975; Zhang et al., 2009; Wandzura and Xiao, 2003; Taylor et al., 2007; Williams et al., 2014; Witherden and Vincent, 2015; Keast, 1986; Shunn and Ham, 2012; Hammer et al., 1956; Hammer and Stroud, 1956; Silvester, 1970; Grundmann and Möller, 1978; Dan and Wang, 2009; Xiao and Gimbutas, 2010; Williams et al., 2020). Alternatively, there exist other approaches specifically devised to exploit the advantages of the spectral element method (Taylor and Wingate, 1999, 2000; Pasquetti and Rapetti, 2006, 2013; Giraldo and Taylor, 2006). Unfortunately, none of these approaches is ready for 4D symmetric numerical integration with positive weights and a simplicial number of closed points.

*Simultaneous numerical integration and interpolation of functions in the simplex up to four dimensions is the third research problem of this thesis.* Without these interpolation-aware numerical quadratures, the interpolation accuracy could be reduced. This is so because numerical quadratures are not specifically devised for interpolation.

## 1.2 Research opportunities and questions

The previous overview identifies the *research opportunity to enable high-order numerical interpolation and integration on complex geometry suitable for simulation up to four dimensions.* Although there are methods to obtain nodal distributions for numeral interpolation and integration up to four dimensions, *there are no known nodal distribution frameworks for complex geometry that consider simulation suitability as well as quasi-optimal interpolation error and numerical quadratures.* The overview

also identifies the following key research questions:

(Q1) How can we represent and model the simulation intent on complex geometry?

(Q2) How can we numerically interpolate functions on complex geometry up to four dimensions?

(Q3) How can we simultaneously numerically integrate and interpolate functions on complex geometry up to four dimensions?

*The answers to questions (Q1), (Q2), and (Q3) contribute to enable simulation on complex geometry up to four dimensions.* The question (Q1) deals with obtaining complex geometry suitable for simulation. Meanwhile, the next two questions address two key numerical aspects for simulation on complex geometry, interpolation (Q2) and integration (Q3).

## 1.3 Aim and objectives

To enable high-order numerical interpolation and integration in complex geometry, *this thesis aims to demonstrate interpolation-aware optimization of nodal distributions on the high-dimensional simplex constrained to fulfill either feasible interpolation or feasible integration with the point coordinates as design variables.* To this end, this thesis develops the following objectives:

(O1) To nodally represent boundary geometry for flow simulation (Jiménez-Ramos et al., 2020)

(O2) To model flow simulation intent with a nodal boundary representation, Chapter 3 (Jiménez-Ramos et al., 2022).

(O3) To estimate the interpolation error of a given point distribution, Chapter 4 (Jiménez-Ramos et al., 2023d).

(O4) To obtain candidate point distributions suitable for interpolation, Chapter 5.

(O5) To obtain point distributions with quasi-optimal interpolation error, Chapter 6.

(O6) To obtain integration points also suitable for interpolation, Chapter 7.

*The aim of this thesis addresses the research opportunity. Moreover, the combination of the objectives addresses the research questions.* For the research question (Q1), objective (O1) addresses the simulation intent representation, and objective (O2) accounts not only for the representation but also for the modeling of the simulation intent. Regarding the research question (Q2), objective (O3) addresses the evaluation of the interpolation error, objective (O4) addresses the exploration of the local minima of a proxy of the interpolation error, and objective (O5) addresses the optimization of a candidate nodal distribution to obtain quasi-optimal interpolation error. Finally, the research question (Q3) is addressed by objective (O6).

## 1.4 Scope

This thesis is focused on interpolation-aware optimization of nodal distributions on the high-dimensional simplex constrained to fulfill either feasible interpolation or feasible integration with the point coordinates as design variables. Specifically, this thesis focuses on symmetric nodal distributions on the high-dimensional simplex optimizing the Lebesgue constant, or a differentiable surrogate.

Following, we justify this scope, yet there are alternative choices. First, we consider interpolation and integration in the simplex because simplicial unstructured meshes have demonstrated to deal with complex geometry. Second, although the formulations are proposed for arbitrary dimensionality, we focus on moderate dimensions. This is so because we aim to simulate 4D space-time dynamics or to numerically solve the 6D Boltzmann equations. Third, we consider nodal distributions with a simplicial number of points because interpolation in the space of polynomials of degree less than or equal to $p$ in $d$ dimensions has a simplicial number of elements. Fourth, we are interested in symmetric nodal distributions so as to be invariant under permutation of the simplex vertices and avoid favoring any particular direction. Moreover, the number of points in each sub-simplex of the symmetry simplex is chosen to match that of the equispaced distribution because it seems to be the only one to provide a unisolvent set of nodes (Marchildon and Zingg, 2022). Fifth, deterministic optimization methods are favored because we want to exploit and understand the characteristics of the interpolation problem and obtain reproducible output for the same input. Sixth, to improve the interpolation properties of a nodal distribution, we optimize the Lebesgue constant since it is the infinity norm of the interpolation op-

erator. Seventh, to approximate the Lebesgue constant, we consider the proxy given by the vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials. Yet there exist other alternatives, we favor this proxy because it is twice-differentiable and seems to be more numerically stable.

## 1.5 Methodology

*To gradually meet the aim of this thesis, the research methodology sorts the objectives by sequential dependency and increasing difficulty.* Out of the scope of this thesis, our long-term goal is to represent and model 4D space-time geometry obtained by sweeping a 3D volume mesh. Accordingly, to obtain preliminary results for a simplified version of the problem in 3D, the first two objectives consider nodal boundary representation (O1) and modeling (O2). Then, following the 3D geometry interpolation results, the objectives (O3), (O4), and (O5) are focused on evaluating and optimizing the interpolation error of a nodal distribution. Specifically, objective (O3) pursues evaluating the interpolation error of a nodal distribution because it is crucial to optimize the coordinates of the distribution according to either a proxy (O4) or the interpolation error (O5). Finally, exploiting the previously developed optimization approaches for numerical interpolation, objective (O6) explores the possibility to obtain nodal distributions that not only properly interpolate but also exactly integrate polynomials.

*The methodology approaches are based on mathematical formulations and derivations, design of optimization methods, heuristics, computer implementations, runtime checks, and verification approaches.* First, the mathematical formulation and derivations allow stating the optimization problem formulation. Second, the proposed optimization methods are the base of the computer implementations. Third, heuristics are used to measure the distance to a limit model, explore hyper-volumes with an algebraic scaling with the dimension, stop point adaptivity according to a Lipschitz criterion, choose between different multiple minima of a Lebesgue constant proxy, use degree-aware trust region radius, and use previous Lebesgue maxima to sample first-order Lebesgue-function-based constraints. Fourth, the distributed computer implementations of the proposed optimization methods allow obtaining numerical results for nodal distributions for interpolation and integration. Fifth, the computer implementations check at runtime whether we obtain a descent direction and a feasi-

ble nodal distribution during the optimization process. Finally, to verify the results, we compare them with the best existing nodal distributions for each one of the dimensions (2D, 3D, and 4D).

## 1.6 Contributions and novelty

*The main contribution is to propose an optimization framework to obtain nodal distributions for high-order interpolation and integration in the high-dimensional simplex that have quasi-optimal interpolation accuracy.* These quasi-optimal nodal distributions can be used to represent the boundary of a complex geometry that is modeled to preserve the simulation intent (Q1). Regarding these boundary representations, the quasi-optimal nodal distributions can be used to approximate the target geometry through numerical interpolation (Q2) and to measure the geometric accuracy of the interpolative approximation through numerical integration (Q3). The framework heuristically explores different nodal distributions that are multiple minima of the interpolation error (Q2). Moreover, the resulting nodal distributions can be optimized to obtain numerical quadratures (Q3). To this end, addressing the previously stated objectives, this thesis contributes with novel methods:

(C1) Devising an equispaced nodal representation method to interpolate a curved subdivision model obtained from a linear triangular mesh. This contribution addresses the representation aspect of question (Q1) for equispaced nodal distributions. The main novelty is proposing a nodal curving method based on hierarchical subdivision with no need for the underlying target geometry. This contribution corresponds to the peer-reviewed conference paper Jiménez-Ramos et al. (2020).

(C2) Devising a nodal modeling method to preserve the required simulation intent that interpolates the subdivision curvature of the sharp and smooth features prescribed on a linear triangular mesh. This contribution addresses question (Q1). There are two main novelties. First, proposing a modeling technique to preserve or smooth the sharp features of a linear mesh model. Second, given a polynomial degree and nodal distribution, generating curved piecewise polynomial representations interpolating a subdivision limit model with no need

for the underlying target geometry. This contribution corresponds to the journal paper Jiménez-Ramos et al. (2022).

(C3) Proposing a point refinement method to estimate the Lebesgue constant of a nodal distribution on the $d$-dimensional simplex. This contribution addresses the estimation aspect of question (Q2). Regarding the devised adaptive method to estimate the interpolation error, the main novelties are the algebraic scaling with the dimensionality of the sampling points and the automatic Lipschitz-based stopping criterion. This contribution corresponds to the peer-reviewed conference paper Jiménez-Ramos et al. (2023d).

(C4) Proposing an optimization method to explore the nodal distributions that are the heuristically best local minima of a Lebesgue constant proxy. This contribution addresses question (Q2) for a proxy of the interpolation error. There are two main novelties. First, approximating the uphills of the energy landscape with the faces of a Delaunay triangulation on the $(d+1)$-sphere. Second, heuristically enforcing a tunnel effect through the uphills of the energy landscape to generate initial approximations with no grid-like structure. This contribution corresponds to the journal paper in preparation Jiménez-Ramos et al. (2023a).

(C5) Proposing a constrained optimization method to minimize the Lebesgue constant of an initial nodal distribution. This contribution addresses question (Q2). The main novelty is to deterministically minimize the Lebesgue constant in the high-dimensional simplex by consistently exploiting the first-order derivatives of the Lebesgue function in terms of the node coordinates. This contribution corresponds to the journal paper in preparation Jiménez-Ramos et al. (2023c).

(C6) Formulating and minimizing a constrained optimization problem enforcing small Lebesgue constant and exact integration of an initial nodal distribution. This contribution addresses question (Q3). The main novelty is coupling in a single optimization problem the minimization of the interpolation error and the constraints of exact integration of polynomials. This contribution corresponds to the journal paper in preparation Jiménez-Ramos et al. (2023b).

*In this thesis, there are two key central findings to obtain the interpolation and integration results on the d-dimensional simplex. First, to propose the heuristics and methods to obtain the results, it is key to rely on the claim that nodal distributions*

*that feature optimal interpolation accuracy are equidistributed in the $d+1$-sphere orthant* (Taylor et al., 2000; Roth, 2005). To fully exploit it as a heuristic, in Chapter 5, we propose to evaluate and optimize nodal distributions by setting on the $(d+1)$-sphere orthant the: coordinates of the adaptive sampling points, initial equispaced approximations of nodal distributions, distance between points as the geodesic length, Delaunay triangulations proxying the energy landscape uphills, and trust region radius. Accordingly, the equidistribution claim is a key ingredient in several of the proposed methods and, therefore, it is crucial to obtain our quasi-optimal nodal distributions results for numerical interpolation and integration. *Second, to efficiently explore nodal distributions that are local optima and do not feature a lattice-like structure, it is key to emulate a tunnel effect in the energy landscape of a free node on the $(d+1)$-sphere orthant.* Specifically, in Chapter 5, we propose to proxy the uphills of the energy landscape of the interpolation error for a free node with the faces of a Delaunay tessellation of the nodal distribution on the $(d+1)$-sphere orthant. Using this proxy, we can emulate a tunnel effect that heuristically enforces a node to go to the other side of an uphill energy landscape by simply relocating the node on the centroid of the element on the other side of the face. Then, continuously optimizing all the degrees of freedom of the nodal distribution we can obtain a new local minima. Using deterministic continuous optimization without this tunnel effect we cannot recover the non lattice-like structures in 2D of the best Lebesgue nodal distributions reported in the literature (Roth, 2005). Hence, the tunnel effect emulation is crucial to verify our 2D results and obtain our 3D and 4D results for numerical interpolation and integration.

## 1.7 Layout

To develop the previous contributions, the thesis starts with preliminary definitions and contents that are common to the different contributions, see Chapter 2. Then, the proposed contributions are successively presented. Finally, the appendices include the detail of auxiliary methods and results. Following, we summarize the contents of the contribution chapters.

In Chapter 3, we present a nodal interpolation method to approximate a subdivision model. The main application is to model and represent curved geometry without gaps and preserving the required simulation intent. Accordingly, we devise

the technique to maintain the necessary sharp features and smooth the indicated ones. This sharp-to-smooth modeling capability handles unstructured configurations of the simulation points, curves, and surfaces. The surfaces correspond to initial linear triangulations that determine the sharp point and curve features. The method automatically suggests a subset of sharp features to smooth which the user modifies to obtain a limit model preserving the initial points. This model reconstructs the curvature by subdivision of the initial mesh, with no need for an underlying curved geometry model. Finally, given a polynomial degree and a nodal distribution, the method generates a piecewise polynomial representation interpolating the limit model. We show numerical evidence that this approximation, naturally aligned to the subdivision features, converges to the model geometrically with the polynomial degree for nodal distributions with sub-optimal Lebesgue constant. We also apply the method to prescribe the curved boundary of a high-order volume mesh. We conclude that our sharp-to-smooth modeling capability leads to curved geometry representations with enhanced preservation of the simulation intent.

In Chapter 4, we propose a point refinement method to estimate the Lebesgue constant on the $d$-dimensional simplex. The proposed method features a smooth gradation of the point resolution, neighbor queries based on neighbor-aware coordinates, and a point refinement that algebraically scales as $(d+1)\,d$. Remarkably, by using neighbor-aware coordinates, the point refinement method is ready to automatically stop using a Lipschitz criterion. For different polynomial degrees and point distributions, we show that our automatic method efficiently reproduces the literature estimations for the triangle and the tetrahedron. Moreover, we efficiently estimate the Lebesgue constant in higher dimensions. Accordingly, up to six dimensions, we conclude that the point refinement method is well-suited to efficiently estimate the Lebesgue constant on simplices. In perspective, for a given polynomial degree, the proposed point refinement method might be relevant to optimize a set of simplex points that guarantees a small interpolation error.

In Chapter 5, we propose an optimization method to explore the heuristically best high-order interpolation nodal distributions in the $d$-dimensional simplex. We consider a twice-differentiable proxy of the Lebesgue constant with multiple local minima that are heuristically explored by means of node relocations and smooth optimizations. For a free node, a proxy of the energy landscape of the Lebesgue constant is obtained through a Delaunay triangulation on the $(d+1)$-sphere, where the opposite

faces of the simplices incident to the node determine an approximation of the uphill energy landscape of the functional around such node. Accordingly, to explore proximal distributions, we heuristically enforce a tunnel effect by relocating one node to the other side of the uphill of the energy landscape. To heuristically find the best local minima, we explore nodal distributions which always improve the current function value. To exploit the available computational resources, the nodal distributions with better function values are explored first. The results show that the considered heuristics drastically reduce the number of local minima to explore while not having an impact on the best values found. Moreover, in 2D, our nodal distributions present good interpolation properties, and in 3D and 4D, our nodal sets improve the current best interpolative nodal configurations. We conclude that the computed nodal distributions might be suitable for high-order interpolation in the high-dimensional simplex, yet they might be excellent initial approximations for methods optimizing the Lebesgue constant to further improve the interpolation properties.

In Chapter 6, we propose a specific-purpose deterministic method to optimize the Lebesgue constant of a given nodal distribution in the $d$-dimensional simplex. In a sequential linear programming approach, we fix relevant spatial points and compute a candidate descent direction consistently exploiting the first-order derivatives of the Lebesgue function with respect to the node coordinates. The derivatives are computed analytically and the Lebesgue constant is not differentiated in any case. We consider an advantageous representation of the nodes on the orthant of the $(d+1)$-sphere to ensure smooth dynamics during the optimization procedure. The method finds an optimum close to the given initial approximation. Accordingly, it is well-suited to further improve the Lebesgue constant of nodal distribution with already good interpolation properties, such as quasi-optimal nodal distributions minimizing the vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials. The method is validated by finding the optimal Lebesgue nodes in 1D. Furthermore, we reproduce the literature results in 2D, and find the nodal distributions with the lowest Lebesgue constant up to date in 3D and 4D. In perspective, we expect that the computed nodal distributions might be suitable for high-order interpolation in the high-dimensional simplex.

In Chapter 7, we propose a method to compute interpolation-aware numerical quadratures in the four-dimensional simplex. We minimize the interpolation error of a symmetric and closed nodal distribution subject to integrating exactly high-order

polynomials with positive integration weights. The design variables are the weights and coordinates of the nodal representatives. The interpolation error is approximated with the twice-differentiable vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials. Because we consider symmetric distributions, the constraints only impose exact integration of high-order symmetry-invariant polynomials. Solving the constrained problem using standard optimization methods, we obtain preliminary results in the two-, three-, and four-dimensional simplex. In perspective, we expect that our collocation of the interpolation points with the integration points will be used to improve the efficiency and stability of the spectral element method and other unstructured high-order methods.

# Chapter 2

# Preliminaries and definitions

Before proposing the new methods and results of this thesis, we introduce the common notation, concepts, and definitions. First, we introduce some notation regarding the nodal distributions, the barycentric coordinates system, and the parameterization of symmetric nodal distributions. Second, we define the Lebesgue constant and show the properties relevant to this thesis. Finally, we describe some implementation details regarding the evaluation of the Lebesgue constant and a numerically stable orthonormal polynomial basis.

## 2.1 Symmetric nodal distributions on the simplex

In this thesis, we are interested in interpolation and integration on complex geometry up to four dimensions discretized using simplices. Specifically, a nodal distribution of polynomial degree $p$ in the $d$-dimensional simplex consists of $N_{p,d} = \binom{d+p}{d}$ nodes and is denoted by $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1,\dots,N_{p,d}}$.

### 2.1.1 Barycentric coordinates system

To avoid a dependence on a specific straight-sided reference simplex, we exploit the barycentric coordinates system. We denote by $\boldsymbol{x} = (x_1, \dots, x_d)$ the cartesian coordinates of a point in the $d$-simplex $K^d \subset \mathbb{R}^d$. The same point is expressed in barycentric

coordinates as $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{d+1})$, where the components satisfy

$$\begin{aligned} &0 \leq \lambda_i \leq 1, \quad \text{for } i = 1, \ldots, d+1, \\ &\sum_{i=1}^{d+1} \lambda_i = 1, \\ &\boldsymbol{x} = \sum_{i=1}^{d+1} \lambda_i \boldsymbol{x}_i, \end{aligned}$$

with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d+1}$ being the cartesian coordinates of the $d+1$ vertices of the simplex. We highlight that the barycentric coordinates $\boldsymbol{\lambda}$ are obtained from a linear combinations of the cartesian coordinates $\boldsymbol{x}$ and allow our definitions and methods to be independent of a specific simplex vertices.

### 2.1.2 Parameterization of symmetric nodal distributions

In this thesis, we consider symmetric nodal distributions on the simplex. We favor symmetric over non-symmetric nodal sets because we want elements that are label invariant and orientation free. As a consequence, since in a simplicial mesh two adjacent simplices intersect along an entity of dimension $k$, for $k = 1, \ldots, d-1$, the nodal distribution of both elements coincides along their intersection. Moreover, on the entities of dimension $k$, we impose that the number of nodes on the entity matches the number of nodes of the $k$-dimensional simplex, for $k = 1, \ldots, d-1$. Therefore, the restriction of the nodal basis on an entity coincides with the nodal high-order basis of the lower-dimensional simplex. Finally, we consider unisolvent set of nodes, that is, the data values at the nodes define a unique polynomial.

Since the nodal distribution is symmetric, determining the position of a subset of the set of nodes is enough to describe the position of the whole distribution. These nodes that identify the entire distribution are called representative nodes. Moreover, the movement of the nodes presents several constraints. On the one hand, interior nodes may belong to a symmetry hyper-plane that restricts their dynamics. For instance, a node on the median of the triangle can only move along the median. On the other hand, boundary nodes have to remain on the boundary while describing a symmetric distribution on the face of the element. Thus, each representative belongs to a manifold which describes its domain. In conclusion, a small set of parameters is enough to determine the position of a nodal distribution. These parameters are meant to serve as design variables in the optimization problems. We encode the $n$

parameters describing a nodal configuration in a vector $\boldsymbol{y} \in \mathbb{R}^n$, and we denote by $\boldsymbol{\sigma}$ the linear mapping which generates the whole nodal set $\boldsymbol{Z}$ from the vector of degrees of freedom,

$$\begin{array}{rccc} \boldsymbol{\sigma}: & \mathbb{R}^n & \to & \mathbb{R}^d \times \overset{N_{p,d}}{\cdots} \times \mathbb{R}^d \\ & \boldsymbol{y} & \mapsto & \boldsymbol{Z} \end{array}.$$

In Appendix C, we carefully detail how we determine the nodal representative, the degrees of freedom, and the mapping $\boldsymbol{\sigma}$.

## 2.2 Lebesgue constant

Consider a compact domain $K^d \subset \mathbb{R}^d$, and let $\mathcal{P}_p^d$ be the finite dimensional vector space of polynomials in $d$ variables of degree not greater than $p$. Denote by $N_{p,d} := \dim \mathcal{P}_p^d = \binom{d+p}{p}$, and consider a unisolvent nodal set $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_{p,d}}\}$. Thus, given a continuous function $f \in \mathcal{C}\left(K^d\right)$, there exists a unique function $g \in \mathcal{P}_p^d$ such that $g\left(\boldsymbol{z}_i\right) = f\left(\boldsymbol{z}_i\right)$, for $i = 1, \ldots, N_{p,d}$. We highlight that the interpolation operator

$$\begin{array}{rccc} \mathcal{I}_{\boldsymbol{Z}}: & \mathcal{C}\left(K^d\right) & \longrightarrow & \mathcal{P}_p^d \subset \mathcal{C}\left(K^d\right) \\ & f & \longmapsto & g = \mathcal{I}_{\boldsymbol{Z}} f \end{array}$$

is linear and depends on the nodal set. The *Lebesgue constant* is defined as the norm of this operator

$$\Lambda \equiv \Lambda\left(\boldsymbol{Z}\right) := \left\|\mathcal{I}_{\boldsymbol{Z}}\right\|.$$

Using the infinity norm in $\mathcal{C}\left(K^d\right)$ defined as $\|h\|_\infty = \max_{\boldsymbol{x} \in K^d} |h\left(\boldsymbol{x}\right)|$, for $h \in \mathcal{C}\left(K^d\right)$, the Lebesgue constant can be written as

$$\Lambda = \sup\{\|\mathcal{I}_{\boldsymbol{Z}} f\|_\infty : \|f\|_\infty = 1, f \in \mathcal{C}\left(K^d\right)\}.$$

For the nodal set $\boldsymbol{Z}$, we consider the Lagrange interpolating polynomial basis $\{\phi_i\}_{i=1,\ldots,N_{p,d}}$ satisfying $\phi_i\left(\boldsymbol{z}_j\right) = \delta_{ij}$. Indeed, $\{\phi_i\}_{i=1,\ldots,N_{p,d}}$ is a basis of $\mathcal{P}_p^d$. Thus, the interpolative polynomial $\mathcal{I}_{\boldsymbol{Z}} f$ can be written in this basis as

$$\left(\mathcal{I}_{\boldsymbol{Z}} f\right)\left(\boldsymbol{x}\right) = \sum_{i=1}^{N_{p,d}} f\left(\boldsymbol{z}_i\right) \phi_i\left(\boldsymbol{x}; \boldsymbol{Z}\right).$$

Consequently, we obtain the following equivalent definition of the Lebesgue constant

$$\Lambda = \left\|\mathcal{I}_{\boldsymbol{Z}}\right\| = \max_{\boldsymbol{x} \in K^d} \sum_{i=1}^{N_{p,d}} \left|\phi_i\left(\boldsymbol{x}; \boldsymbol{Z}\right)\right|. \tag{2.1}$$

Note that, fixed the dimension $d$ and the polynomial degree $p$, the Lebesgue constant solely depends on the nodal distribution $\boldsymbol{Z}$, it does not depend on the function $f$ to be interpolated. Furthermore, if we denote by $f^\star \in \mathcal{P}_p^d$ the best approximating polynomial to $f$, from $f^\star = \mathcal{I}_{\boldsymbol{Z}} f^\star$ and the triangular inequality, it follows that

$$\|f - \mathcal{I}_{\boldsymbol{Z}} f\|_\infty \leq (1 + \Lambda) \|f^\star - f\|_\infty . \tag{2.2}$$

On the right-hand side, only the term $\|f^\star - f\|_\infty$ depends on the function to be interpolated. Thus, using nodal distributions with low Lebesgue constant leads to tighter upper bounds for the interpolation of any function $f$.

It is also worth studying the growth of the Lebesgue constant as the polynomial degree increases. According to Eq. (2.2), $\Lambda$ should grow slower in $p$ than $\|f^\star - f\|_\infty$ dies away to attain convergence of the interpolator to the target function $f$ (Bloom et al., 1992; Roth, 2005). In particular, uniform convergence is attained if

$$\lim_{p \to +\infty} \Lambda^{1/p} = 1.$$

Unfortunately, even in one dimension, equispaced nodal sets have Lebesgue constants that grow exponentially with the polynomial degree (Bos, 1983; Bloom, 1988) and, therefore, are not suitable for interpolation. As an example, interpolating with equispaced points the Runge function leads to unbounded interpolation error towards the endpoints of the interval (Runge, 1901).

In conclusion, our goal is to find nodal distributions in the simplex featuring a simplicial number of points, low Lebesgue constant, and numerical evidence that $\Lambda^{1/p} \to 1$ as the polynomial degree increases. Moreover, since we expect these nodes to be used in the interpolation of fields arising from numerical simulations, we only study symmetric nodal sets.

## 2.3 Orthonormal polynomial basis

To compute the Lebesgue constant, we should evaluate the Lagrange interpolating polynomials, see Eq. (2.1). Given a basis $\{b_j(\boldsymbol{x})\}_{j=1,\dots,N_{p,d}}$ of $\mathcal{P}_p^d$, due to the interpolating property of the Lagrange polynomials, the following relation holds

$$b_j(\boldsymbol{x}) = \sum_{i=1}^{N_{p,d}} b_j(\boldsymbol{z}_i)\, \phi_i(\boldsymbol{x}; \boldsymbol{Z}) .$$

Thus, to evaluate the Lagrange interpolating polynomials we aim at numerically solving a linear system with a Vandermonde matrix with entries $b_j(\boldsymbol{z}_i)$. Since the value of the Lebesgue constant is independent of the choice of the polynomial basis $\{b_j(\boldsymbol{x})\}$, we favor a numerically robust and stable basis. The orthogonal polynomial basis due to Koornwinder (1975) and Dubiner (1991) is a good candidate. Nevertheless, these polynomials are obtained via a transformation of the hypercube to the hypersimplex and present some singularities. Fortunately, it is possible to circumvent these issues by evaluating directly in the simplex coordinates by means of a recurrence relation (Kirby, 2010). The available expressions generate an orthogonal polynomial basis in the triangle and the tetrahedron, yet we modify these expressions to generate an orthonormal polynomial basis in the $d$-dimensional simplex, $d \geq 1$. Furthermore, since these expressions are simple, one can easily obtain the first and second-order derivatives to be used in the computation of the Lagrange interpolating polynomials derivatives.

# Chapter 3

# Interpolation of subdivision features for curved geometry modeling

## 3.1 Introduction

The capability to model and represent curved geometry preserving the simulation intent is critical for flow simulation with unstructured high-order methods. These high-order simulations require curved meshes that approximate a curved boundary representation (Persson and Peraire, 2008; Chaurasia et al., 2012; Johnen et al., 2013; Gargallo-Peiró, 2014; Gargallo-Peiró et al., 2015c,b; Ruiz-Gironés et al., 2016b,a; Moxey et al., 2016). Ideally, this boundary representation should be composed of smooth and sharp features agreeing with the simulation intent (Thakur et al., 2009; White et al., 2005; Shapiro et al., 2011; Nolan et al., 2015).

Flow simulation practitioners favor continuous normal vectors on smooth features where the intent is to obtain attached flow. In contrast, they only need model continuity on sharp features where the flow detaches. To illustrate both types of features, we can consider an aircraft model. We can find there smooth features such as the nose tip, leading edges, and wing surfaces; and sharp features such as trailing edges and points.

To model the previous features, it is standard to use CAD boundary represen-

tations based on trimmed NURBS. Unfortunately, these trim-based models might violate the simulation intent. This breach is so since they might present unintended gaps or discontinuities of the normal vector on irregular points and curves adjacent to trimmed surfaces. Nevertheless, together with virtual geometry (Sheffer et al., 2000; Sheffer, 2001; Foucault et al., 2008, 2013), CAD boundary representations have shown to be crucial for generating an unstructured triangular mesh approximating the curved model boundary.

If the element size is fine but coarser than the model tolerance, we obtain a fair second-order approximation of the CAD boundary representation without gaps. However, since the triangles are planar, this approximation does not feature the normal vector continuity through any triangular edges, and thus, it is not adequate for flow simulation. Nevertheless, we can convert the triangular mesh to a gap-free curved geometry model (Persson et al., 2006) that features normal vector continuity by using a subdivision scheme (Loop, 1987).

This subdivision-based conversion to a curved model is useful even when there is no underlying CAD model. The conversion only needs a model composed of triangulations, which boundaries determine the model points and polylines. Hence, this conversion is of practical interest in any application where the triangular mesh comes from legacy data or real samples, such as in onshore wind farm energy forecasting (Gargallo-Peiró et al., 2015a, 2018), transport of pollutants in urban areas (Gargallo-Peiró et al., 2016a), and bio-engineering.

In these applications, the subdivision conversion provides a curved limit model. We can query this limit model by successive refinements (Persson et al., 2006; Loop, 1987; Lane and Riesenfeld, 1980). However, this approach requires more refinement levels the closer a given query point is to an irregular point.

To avoid this unbalanced query, in our previous work (Jiménez Ramos, 2018; Jiménez-Ramos et al., 2020), we proposed a method to interpolate with a continuous piecewise quadratic or quartic mesh the limit model while exploiting the structure of iterative subdivision. Any posterior query to the interpolation model only uses the corresponding triangular-wise polynomial components, thus skipping the successive refinement step.

Although skipping posterior successive refinement, our previous approach only extends to interpolation degrees equal to powers of two on equispaced nodal distributions. Therefore, it does not allow using arbitrary polynomial degrees and nodal

distributions. Recall that beyond degree four equispaced nodal distributions feature large Lebesgue constants that might hamper the corresponding interpolation accuracy.

The main contribution of this chapter is to propose a method to address the previous accuracy issues while still skipping successive refinement on posterior geometry queries. Specifically, we propose a method to interpolate the subdivision model with any degree and nodal distribution. The method evaluates the limit model parameterization (Stam, 1998) once on each interpolation point to obtain the resulting nodal curved mesh model. Furthermore, we show that this nodal mesh is ready to prescribe the boundary of a curved high-order volume mesh.

We also propose an approximation of the distance between the interpolation and the limit model to check the geometric accuracy. To compute this distance, we only need to perform forward evaluations of the nodal parameterization. The distance allows studying how the nodal interpolation converges to the limit model in terms of the polynomial degree for different nodal distributions.

We finally propose an assisted sharp-to-smooth modeling capability. The resulting method automatically suggests a subset of sharp features to smooth, which the user modifies to obtain a limit model preserving the initial points. It aims to reduce the amount of human labor required to describe the simulation intent of the model. We illustrate this assisted modeling capability to assign sharp and smooth features to the union of triangulations describing an aircraft model in a high-lift configuration.

The organization of the rest of the chapter is as follows. First, in Sect. 3.2, we review the related literature. Then, in Sect. 3.3, we present the problem statement and the methodology. Next, in Sect. 3.4, we present some preliminary results on subdivision, and in Sect. 3.5 we detail the subdivision limit model. In Sect. 3.6, we detail the curved piecewise polynomial surface mesh interpolation of the limit model. Following, in Sect. 3.7, we present our subdivision-based mesh curving approach, detailing the sharp-to-smooth modeling process. Then, in Sect. 3.8, we present several results to illustrate the capabilities and main features of the presented methods, and in Sect. 3.9, we discuss some aspects of the method as well as future contributions. Lastly, in Sect. 3.10, we present some concluding remarks of this chapter.

## 3.2 Related work

It is well documented that to obtain a geometric model suitable for simulation, we need to simplify the original model (Thakur et al., 2009; White et al., 2005). To obtain a simplified model, one standard approach is to define a Virtual Topology on top of the original CAD model (Sheffer et al., 2000; Sheffer, 2001). A similar CAD simplification approach, emphasizing automation and posterior meshing needs, considers a Mesh Constraint Topology (Foucault et al., 2008, 2013). These simplification frameworks have been proven to be successful in many practical applications. However, during the simplification process, these frameworks might present some geometric issues. These issues can be fixed by maintaining a firm link between the original CAD and the simulation model (Shapiro et al., 2011). The key idea is to store the simplification steps in a high-level framework connecting the design and simulation geometry, and thus, facilitating CAD de-featuring (Shapiro et al., 2011; Nolan et al., 2015). Alternatively, we can automatically de-feature a faceted representation of the original CAD model (Quadros and Owen, 2012).

In mesh generation, Loop's subdivision surfaces have been already used to define surrogate geometry with a surface-wise continuous normal vector (Persson et al., 2006). This subdivision scheme has also been used to provide nodal quadratic, and quartic surface and volume meshes (Jiménez Ramos, 2018; Jiménez-Ramos et al., 2020). Alternatively, the butterfly subdivision scheme (Dyn et al., 1990) has been used to relocate the boundary nodes when the geometry is unavailable (Yang et al., 2019). However, the limit model does not feature continuous normal vectors on irregular points, and the volume is not curved using hierarchical blending. Finally, there are volume subdivision methods to generate curved volume meshes, featuring parallel implementations, but they need a curved volume mesh to define the surrogate geometry (Gargallo-Peiró et al., 2017).

There are alternatives to subdivision schemes for describing the surrogate geometry. The work presented in Jiao and Wang (2012) proposes two curving methods based on weighted least squares approximations and piecewise polynomial fittings to generate curved meshes of the target surfaces. However, the method only enforces the continuity of the model. Herein, the limit model ensures continuity in the sharp features and normal vector continuity on the smooth features.

The curved surface meshes provided by the method in Jiao and Wang (2012) can also be used to limit the mesh volumes and, thus, to generate curved high-order

meshes when the underlying curved geometry is not available. Furthermore, it is possible to remove sharp features by selecting one-by-one the mesh entities defining it (Ims et al., 2015). Besides the surface mesh curving method, the method herein has other differences: a hierarchical blending approach to curve the mesh volume, an all-in-one feature selection to perform sharp-to-smooth modeling, and a robust and tunable assisted smooth feature detector.

## 3.3 Problem statement and methodology

### 3.3.1 Problem statement

We consider the problem of converting a linear mesh model to a piecewise polynomial curved mesh of degree $q$ that preserves both the sharp and smooth features determined in the model. Our solution approximates a curved limit model based on a subdivision scheme of the linear model. The rest of this section describes the inputs and the output of the proposed method.

The input data is a linear tetrahedral mesh, a linear model $\Omega^1$, and a list of features to be smoothed. A model of degree $q$, $\Omega^q$, represents the geometry as the union of the feature points $\mathcal{P}_p$, curves $\mathcal{S}_c^q$, and surfaces $\mathcal{T}_s^q$:

$$\Omega^q = \bigcup_{p=1}^{n_p} \mathcal{P}_p \cup \bigcup_{c=1}^{n_c} \mathcal{S}_c^q \cup \bigcup_{s=1}^{n_s} \mathcal{T}_s^q.$$

A geometry feature is characterized by a set of entities of the boundary of the volume mesh with the same identifier. Then, a point feature describes a vertex to preserve, and it is characterized by the global identifier of the point to be preserved. A curve feature describes a smooth curve to preserve. A curve of $\Omega^q$ is described by the union of segments of polynomial degree $q$ forming a polycurve. These segments correspond to boundary edges of the tetrahedral mesh with the same curve identifier. Finally, a surface feature describes a smooth surface to preserve. A surface of $\Omega^q$ is described by the union of facets of polynomial degree $q$ forming a triangulation. These facets correspond to boundary triangles of the tetrahedral mesh with the same surface identifier.

Alternatively, we can obtain the model when only the surface features are described. That is, if only the boundary triangles are marked, we can retrieve the feature curves from the intersection of the boundary of two or more feature surfaces.

Similarly, point features can be determined by the intersection of two or more feature curves.

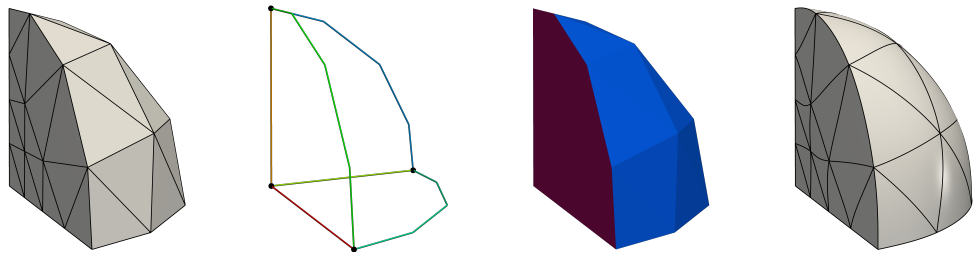
In addition to the tetrahedral mesh, the linear model, and the geometry features, we have an optional input determining the sharp features to smooth. Smoothing a geometry feature corresponds to removing it from the list of features to preserve and merging adjacent regions to ensure $\mathcal{C}^1$-continuity of the model along the removed feature. That is, if we smooth a curve, we remove the curve feature and merge the two adjacent surfaces. While when we smooth a point, we remove the point feature and merge the curves incident to the vertex. Since each geometry feature is associated with a unique identifier, the list of features to be smoothed is a sub-sequence of these unique identifiers.

The output of our method is a piecewise polynomial mesh of degree $q$ with a boundary preserving the sharp features of the model and satisfying three properties. First, high-order element vertices interpolate the initial linear mesh nodes. Second, the nodes of the high-order edges that belong to a feature curve and are not adjacent to a feature point (*inner curve edges*) interpolate a cubic $\mathcal{C}^2$-continuous curve. Third, the nodes of the high-order elements that belong to a feature surface and are not adjacent to a feature curve or point (*inner surface elements*) interpolate a $\mathcal{C}^1$-continuous surface. These properties provide regularity guarantees in the output mesh that are discussed in Sect. 3.6.2. The geometry features of the boundary of the generated tetrahedral mesh of polynomial degree $q$ characterize the model of polynomial degree $q$, $\Omega^q$.

### 3.3.2 Method: hierarchical subdivision and blending

The curved high-order mesh generation procedure proposed in this chapter is composed of five main steps:

0. **Sharp-to-smooth modeling**. Some applications need the capability to perform sharp-to-smooth geometry modeling. Then, as a preprocess, we can smooth the non-desired geometry features accordingly to the list of features to remove provided as input to obtain a smoother surrogate geometry. We devise a technique to automatically suggest the features to smooth in Sect. 3.6.4, and the smoothing process is detailed in Sect. 3.7.1.



(a) (b) (c) (d)

Figure 3.1: Method. Input: (a) a linear tetrahedral mesh and the associated model with (b) feature points, curves and (c) surfaces. Output: (d) a curved tetrahedral mesh of polynomial degree $q$, $q = 4$ in this case.

1. **Approximate a surrogate boundary**. Given a linear tetrahedral mesh, Fig. 3.1(a), we extract its boundary. The entities of the boundary mesh characterize the geometry features of the given linear model $\Omega^1$, see Fig. 3.1(b) and Fig. 3.1(c). The hierarchical subdivision of the geometry features determines a set of $\mathcal{C}^2$-continuous limit curves and $\mathcal{C}^1$-continuous limit surfaces that serves as surrogate geometry to generate a curved high-order triangular surface mesh, see Sect. 3.5. This curved surface mesh preserves the sharp features and smooth regions of the linear model $\Omega^1$, and interpolates it at the nodes of the high-order mesh. See details in Sect. 3.6.

2. **Substitute the boundary of the volume mesh**. We increase the polynomial degree of the volume mesh and replace the straight-sided boundary of the current high-order volume mesh by the high-order surface mesh obtained in the first step. See details in Sect. 3.7.1.

3. **Accommodate the curvature of the boundary**. We accommodate the curvature of the curved surface mesh to the boundary volume elements using an explicit hierarchical blending, see Fig. 3.1(d). See details in Sect. 3.7.2.

4. **Local smoothing and untangling**. If necessary, we optimize the low-quality elements locally following Gargallo-Peiró et al. (2015c,b).

## 3.4 Preliminaries: curve and surface mesh subdivision

In this section, we present the subdivision algorithms that determine the limit model. The boundary mesh is composed of points, curves, and surfaces. Then, the subdivision is performed hierarchically, that is, points remain fixed, curves are refined using a curve subdivision scheme, and surfaces are subdivided using a surface subdivision scheme.

In Sect. 3.4.1, we detail the curve subdivision algorithm we use herein, and Sect. 3.4.2 recalls Loop's subdivision surface scheme. Although these methods do not preserve the position of the initial control points, a simple modification leads to a smooth limit manifold interpolating these initial points, see Appendix A.

### 3.4.1 Curve subdivision scheme

The curve subdivision scheme used in this chapter was originally presented in Lane and Riesenfeld (1980). Successive applications of the algorithm generate finer polygons, all of them with the same limit curve determined by the initial mesh, referred to as control mesh. This curve is parametrized by a polynomial of degree three and it is $\mathcal{C}^2$-continuous.

We remark that there exists an element-wise parameterization of the limit curve that allows mapping any polyline point onto the limit curve. Specifically, consider $\mathcal{S}^1$ the $c$-th curve of the model $\Omega^1$. Let $\hat{S}$ be the master interval, $S_e$ the $e$-th segment of $\mathcal{S}^1$, $\boldsymbol{\lambda}$ the position in barycentric coordinates of a point in $\hat{S}$, and $\{\boldsymbol{x}_i^e\}_{i=1,\dots,4}$ the set composed of the nodes of the segment $S_e$ and its neighbors elements. Then, if $S_e$ is not adjacent to a feature vertex, we define the element-wise parameterization of the limit curve $\mathcal{S}^\infty$ as

$$\begin{aligned} \varphi_{c,e} \colon \hat{S} &\to S_e^\infty \subset \mathcal{S}^\infty \\ \boldsymbol{\lambda} &\mapsto \varphi_{c,e}(\boldsymbol{\lambda}) \equiv \varphi_{c,e}(\boldsymbol{\lambda}; \{\boldsymbol{x}_i^e\}_{i=1,\dots,4}), \end{aligned} \tag{3.1}$$

where $S_e^\infty$ is the section of the limit curve corresponding to $S_e$. The explicit expression of $\varphi_{c,e}$ can be found in Persson et al. (2006); De Boor (1978). We remark that the parameterization features third-degree polynomial components in barycentric coordinates. The case in which segment $S_e$ is adjacent to a feature vertex is detailed in Sect. 3.5.1.

### 3.4.2 Surface subdivision scheme

Loop's subdivision algorithm (Loop, 1987) is used to subdivide the feature surfaces of the model. Successive applications of the algorithm generate finer triangular meshes, all of them with the same limit surface determined by the initial control mesh. This surface is $\mathcal{C}^1$-continuous, attaining $\mathcal{C}^2$-continuity around regular vertices.

We start by introducing several definitions related to neighbor elements and nodes. The *neighbor elements* of a node $v$ are the elements incident to $v$, and the *neighbor nodes* of $v$ are the vertices of these elements. For the case of simplices, there exists an equivalent definition based on edges. The *neighbor edges* of a node $v$ are the edges incident to $v$, and the *neighbor nodes* of $v$ are the vertices of these edges. We say that a surface node is *regular* if it has six neighbor nodes. Otherwise, we say the node is *irregular* or *extraordinary*. Around regular nodes, the limit surface is parametrized by a polynomial of degree four and is $\mathcal{C}^2$-continuous, while on irregular nodes it is of class $\mathcal{C}^1$ (Zorin, 2000).

Similarly to the curve case, there also exists an element-wise parameterization of the limit surface for elements with at most one extraordinary vertex (Stam, 1998) and not adjacent to a sharp feature. Specifically, consider $\mathcal{T}^1$ the $s$-th surface of the model $\Omega^1$. Let $\hat{T}$ be the master triangle, $T_f$ the $f$-th triangle of $\mathcal{T}^1$, $\boldsymbol{\lambda}$ the position in barycentric coordinates of a point in $\hat{T}$, and $\{\boldsymbol{x}_i^f\}_{i=1,\dots,N}$ the $N$-point set composed of the nodes of the triangle $T_f$ and its neighbors elements. Assume $T_f$ has at most one irregular vertex and is not adjacent to a sharp feature. Then, we define the element-wise parameterization of the limit surface $\mathcal{T}^\infty$ as

$$\begin{aligned} \varphi_{s,f} \colon \hat{T} &\to T_f^\infty \subset \mathcal{T}^\infty \\ \boldsymbol{\lambda} &\mapsto \varphi_{s,f}\left(\boldsymbol{\lambda}\right) \equiv \varphi_{s,f}\left(\boldsymbol{\lambda}; \{\boldsymbol{x}_i^f\}_{i=1,\dots,N}\right). \end{aligned} \tag{3.2}$$

The details of this parameterization can be found in Stam (1998). We remark that the parameterization features fourth-degree polynomial components in barycentric coordinates.

In order to compute the parameterization of an element featuring more than one extraordinary vertex, one subdivision step is performed. Since the new edge points obtained from Loop's algorithm are regular, one subdivision of the mesh ensures that every element contains at most one irregular vertex and, therefore, the parameterization can be evaluated for all the elements of the subdivided mesh. Moreover, this

parameterization can be applied to elements not adjacent to a sharp feature. The case in which triangle $T_f$ is adjacent to a sharp feature is detailed in Sect. 3.5.1.

## 3.5 The limit model

The model describes the geometry to represent and is defined by the union of feature points, curves, and surfaces. For each of the feature curves (surface) in the model, the curve (surface) subdivision scheme determines a limit curve (surface), see Sect. 3.4. The union of feature points, limit curves, and limit surfaces determines the limit model $\Omega^\infty$,

$$\Omega^\infty = \bigcup_{p=1}^{n_p} \mathcal{P}_p \cup \bigcup_{c=1}^{n_c} \mathcal{S}_c^\infty \cup \bigcup_{s=1}^{n_s} \mathcal{T}_s^\infty.$$

This limit model preserves the sharp features of the linear model, inherits the smoothness properties of the limit curves and surfaces, and serves as surrogate geometry for the mesh curving problem. In this section, we detail the parameterization of the limit model, Sect. 3.5.1, and study its smoothness, Sect. 3.5.2.

### 3.5.1 Parameterization of the limit model

The limit model can be queried using the element-wise parameterizations of the limit curves and limit surfaces. As noted in Sect. 3.4, the evaluation of these parameterizations can only be computed in certain configurations of the mesh. Specifically, they can only be computed for inner elements and, in the case of surfaces, for triangles with at most one extraordinary vertex. However, in our models, there are segments (triangles) adjacent to feature points (feature points or curves), and triangles with more than one irregular vertex. Therefore, in order to compute the parameterization of an element in one of these two configurations the following procedures are applied. First, we successively subdivide a segment (triangle) adjacent to a feature point (feature point or curve) until the evaluation point belongs to an inner segment (triangle) of the subdivided polyline (triangular mesh). Second, if an inner triangle has more than one extraordinary vertex, we subdivide once the stencil needed for the evaluation of the parameterization of the limit surface. Since new edge points generated with Loop's scheme are regular, all the triangles of the subdivided patch feature at most one irregular vertex. In conclusion, the parameterization of the limit model can be computed for any element of the mesh.

**Algorithm 3.1** Parameterization of the limit model.

**Input:** Linear surface mesh $\mathcal{M}$, Linear model $\Omega^1$, Triangle $T_f^1$, Point $\boldsymbol{\lambda} \in \hat{T}$
**Output:** Point $\boldsymbol{x}^\infty$

1: **function** `MapOntoLimitManifold`($\mathcal{M}$, $\Omega^1$, $T_f^1$, $\boldsymbol{\lambda}$)
2: $\quad\boldsymbol{x} \leftarrow$ `ComputePhysicalCoordinate`($T_f^1$, $\boldsymbol{\lambda}$)
3: $\quad$**switch** Type of feature containing $\boldsymbol{x}$ **do**
4: $\quad\quad$**case** $\boldsymbol{x}$ is feature point
5: $\quad\quad\quad\boldsymbol{x}^\infty = \boldsymbol{x}$
6: $\quad\quad$**case** $\boldsymbol{x}$ belongs to the interior of $c$-th curve $\mathcal{S}^1$ of $\Omega^1$
7: $\quad\quad\quad\boldsymbol{x}^\infty \leftarrow$ `MapOntoLimitCurve`($\mathcal{M}$, $\Omega^1$, $\mathcal{S}^1$, $\boldsymbol{\lambda}$, $\boldsymbol{x}$)
8: $\quad\quad$**case** $\boldsymbol{x}$ belongs to the interior of $s$-th surface $\mathcal{T}^1$ of $\Omega^1$
9: $\quad\quad\quad\boldsymbol{x}^\infty \leftarrow$ `MapOntoLimitSurface`($\mathcal{M}$, $\Omega^1$, $\mathcal{T}^1$, $T_f^1$, $\boldsymbol{\lambda}$, $\boldsymbol{x}$)
10: $\quad$**return** $\boldsymbol{x}^\infty$
11: **end function**

Following, we detail the evaluation of the limit model for all the possible configurations. Consider a linear surface mesh $\mathcal{M}$, a triangle $T_f^1$ belonging to a surface $\mathcal{T}^1$ of the linear model $\Omega^1$, and denote by $\hat{T}$ the master triangle. Then, we define the element-wise parameterization of the limit model $\Omega^\infty$ as

$$\begin{aligned} \phi_f^\infty &: \hat{T} \to \mathcal{T}^\infty \subset \Omega^\infty \\ &\boldsymbol{\lambda} \mapsto \phi_f^\infty(\boldsymbol{\lambda}) \equiv \phi^\infty\left(\mathcal{M}, T_f^1, \boldsymbol{\lambda}\right). \end{aligned} \tag{3.3}$$

The computation of function $\phi^\infty$ is detailed in Algorithm 3.1. Given a linear surface mesh $\mathcal{M}$, a linear model $\Omega^1$, a triangle $T_f^1$ of $\mathcal{M}$, and a point in the master triangle with barycentric coordinates $\boldsymbol{\lambda}$, the function `MapOntoLimitManifold` maps $\boldsymbol{\lambda}$ onto the limit model using the parameterization on triangle $T_f^1$. Firstly, in Line 2, we map $\boldsymbol{\lambda}$ onto the linear triangle $T_f^1$, and obtain the point $\boldsymbol{x} \in T_f^1$. Now, we distinguish three cases in terms of the type of feature the point $\boldsymbol{x}$ belongs to. If $\boldsymbol{x}$ belongs to a feature:

- point, the limit position does not change, $\boldsymbol{x}^\infty = \boldsymbol{x}$, Line 5.
- curve, say the $c$-th curve $\mathcal{S}^1$, the parameterization of its limit curve `MapOntoLimitCurve` is computed, see Algorithm 3.2. First, we get the segment $S_e$ of the curve $\mathcal{S}^1$ that contains the point $\boldsymbol{x}$, Line 2. Then, we define as $\mathcal{S}_e$ the polygonal mesh composed of the segment $S_e$ and the neighbor segments of the curve, and as $\Omega_e$ the associated linear model, Line 3, see Fig. 3.2(a). Then,

**Algorithm 3.2** Parameterization of a limit curve of the limit model.

**Input:** Linear surface mesh $\mathcal{M}$, Linear model $\Omega^1$, Curve $\mathcal{S}^1$, Point $\boldsymbol{\lambda} \in \hat{T}$, Point $\boldsymbol{x} \in \mathcal{S}^1$

**Output:** Point $\boldsymbol{x}^\infty$

1: **function** `MapOntoLimitCurve`($\mathcal{M}$, $\Omega^1$, $\mathcal{S}^1$, $\boldsymbol{\lambda}$, $\boldsymbol{x}$)
2: $S_e \leftarrow$ `GetContainerSegment`($\mathcal{S}^1$, $\boldsymbol{x}$)
3: $\mathcal{S}_e$, $\Omega_e \leftarrow$ `SubCurve`($\mathcal{M}$, $\Omega^1$, $S_e$)
4: **while** $S_e$ is adjacent to feature point **do**
5: $\mathcal{S}'$, $\Omega' \leftarrow$ `CurveSubdivision`($\mathcal{S}_e$, $\Omega_e$)
6: $S_e \leftarrow$ `GetContainerSegment`($\mathcal{S}'$, $\boldsymbol{x}$)
7: $\mathcal{S}_e$, $\Omega_e \leftarrow \mathcal{S}'$, $\Omega'$
8: **end while**
9: $\boldsymbol{\lambda}_e \leftarrow$ `ComputeBarycentricCoordinates`($S_e$, $\boldsymbol{x}$)
10: $\{\boldsymbol{x}_i^e\}_{i=1,\dots,4} \leftarrow$ `GetCurveStencil`($\mathcal{S}_e$, $S_e$)
11: $\boldsymbol{x}^\infty \leftarrow \varphi_{c,e}\left(\boldsymbol{\lambda}_e; \{\boldsymbol{x}_i^e\}_{i=1,\dots,4}\right)$
12: **return** $\boldsymbol{x}^\infty$
13: **end function**

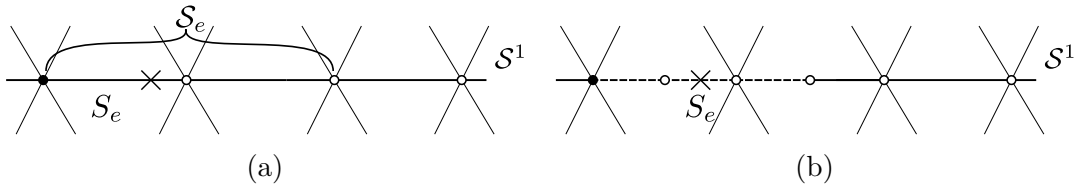



Figure 3.2: Subdivision to compute the element-wise parameterization of the limit curve on a point (cross) adjacent to a feature vertex (bold). Container edge in the: (a) original, and (b) subdivided meshes. Dashed segments represent the curve stencil.

while $S_e$ is adjacent to a feature point, in Line 5, we subdivide $\mathcal{S}_e$ using the curve subdivision scheme, and compute the segment of the subdivided polygonal mesh to which $\boldsymbol{x}$ belongs, Line 6. Note that if the original segment $S_e$ is not adjacent to a feature point, no subdivisions are performed. Now, since element $S_e$ is not adjacent to a feature point, see Fig. 3.2(b), in Line 9, we compute the barycentric coordinates of the point $\boldsymbol{x}$ with respect to $S_e$, $\boldsymbol{\lambda}_e$, and the stencil $\{\boldsymbol{x}_i^e\}_{i=1,\dots,4}$ needed for the evaluation of the limit curve, Line 10. Finally, in Line 11, we map $\boldsymbol{\lambda}_e$ onto its limit position, $\boldsymbol{x}^\infty = \varphi_{c,e}\left(\boldsymbol{\lambda}_e; \{\boldsymbol{x}_i^e\}_{i=1,\dots,4}\right)$.

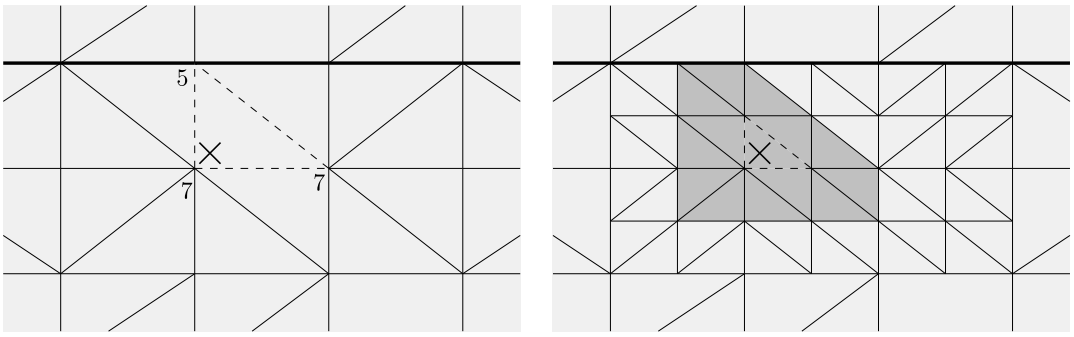
- surface, say the $s$-th surface $\mathcal{T}^1$, the parameterization of its limit surface `MapOntoLimitSurface` is computed, see Algorithm 3.3. First, we set $T_f \leftarrow T_f^1$. Then, we define as $\mathcal{T}_f$ the triangular mesh composed of the triangle $T_f$ and its

**Algorithm 3.3** Parameterization of a limit surface of the limit model.

**Input:** Linear surface mesh $\mathcal{M}$, Linear model $\Omega^1$, Surface $\mathcal{T}^1$, Triangle $T_f^1$, Point $\boldsymbol{\lambda} \in \hat{T}$, Point $\boldsymbol{x} \in \mathcal{T}^1$

**Output:** Point $\boldsymbol{x}^\infty$

1: **function** `MapOntoLimitSurface`($\mathcal{M}$, $\Omega^1$, $\mathcal{T}^1$, $T_f^1$, $\boldsymbol{\lambda}$, $\boldsymbol{x}$)
2: $T_f \leftarrow T_f^1$
3: $\mathcal{T}_f$, $\Omega_f \leftarrow$ `SubSurface`($\mathcal{M}$, $\Omega^1$, $T_f$)
4: **if** $T_f$ has more than one irregular vertex **then**
5: $\mathcal{T}'$, $\Omega' \leftarrow$ `SurfaceSubdivision`($\mathcal{T}_f$, $\Omega_f$)
6: $T_f \leftarrow$ `GetContainerTriangle`($\mathcal{T}'$, $\boldsymbol{x}$)
7: $\mathcal{T}_f$, $\Omega_f \leftarrow \mathcal{T}'$, $\Omega'$
8: **end if**
9: **while** $T_f$ is adjacent to feature point or curve **do**
10: $\mathcal{T}'$, $\Omega' \leftarrow$ `SurfaceSubdivision`($\mathcal{T}_f$, $\Omega_f$)
11: $T_f \leftarrow$ `GetContainerTriangle`($\mathcal{T}'$, $\boldsymbol{x}$)
12: $\mathcal{T}_f$, $\Omega_f \leftarrow \mathcal{T}'$, $\Omega'$
13: **end while**
14: $\boldsymbol{\lambda}_f \leftarrow$ `ComputeBarycentricCoordinates`($T_f$, $\boldsymbol{x}$)
15: $\{\boldsymbol{x}_i^f\}_{i=1,\dots,N} \leftarrow$ `GetSurfaceStencil`($\mathcal{T}_f$, $T_f$)
16: $\boldsymbol{x}^\infty \leftarrow \varphi_{s,f}\left(\boldsymbol{\lambda}_f; \{\boldsymbol{x}_i^f\}_{i=1,\dots,N}\right)$
17: **return** $\boldsymbol{x}^\infty$
18: **end function**




(a) (b)

Figure 3.3: Subdivision to compute the element-wise parameterization of the limit surface on a point (cross) adjacent to a feature curve (bold). Container triangle (dashed edges) in the: (a) original, and (b) subdivided meshes. Dark gray region represents the surface stencil.

neighbor triangles of the surface, and as $\Omega_f$ the associated linear model, Line 3, see Fig. 3.3(a). Next, we check if the triangle has more than one irregular

vertex. If so, in Line 5, we subdivide the triangular mesh $\mathcal{T}_f$ to isolate the irregularity, and, in Line 6, compute the triangle $T_f$ of the subdivided mesh containing the point $\boldsymbol{x}$, see Fig. 3.3(b). Then, analogously to the curve case, while element $T_f$ is adjacent to a sharp feature, we subdivide $\mathcal{T}_f$, Line 10, and get the triangle of the subdivided mesh which $\boldsymbol{x}$ belongs to, Line 11. Now, since the element $T_f$ is not adjacent to a sharp feature, see Fig. 3.3(b), in Line 14, we compute the parameter $\boldsymbol{\lambda}_f$ and the stencil $\{\boldsymbol{x}_i^f\}_{i=1,\dots,N}$ needed for the evaluation of the limit surface, Line 15. Finally, we map the point onto its limit position, $\boldsymbol{x}^\infty = \varphi_{s,f}\left(\boldsymbol{\lambda}_f; \{\boldsymbol{x}_i^f\}_{i=1,\dots,N}\right)$, Line 16.

### 3.5.2 Smoothness of the limit model

In this section, we analyze the smoothness of the limit curves and surfaces that compose the limit model.

On the one hand, the curve subdivision scheme, see Sect. 3.4.1, ensures that in a feature curve the union of the inner edges, the ones not adjacent to a feature point, determines a cubic and $\mathcal{C}^2$-continuous parameterization of the limit curve. On the contrary, curved edges of the control mesh that are incident to a feature point do not feature that cubic parameterization. The limit curve is only of class $\mathcal{C}^0$ on the feature point.

On the other hand, Loop's subdivision scheme, see Sect. 3.4.2, ensures that in a feature surface the union of the inner triangle, the ones not adjacent to a feature point or curve, determines a $\mathcal{C}^1$-continuous parameterization of the limit surface. Around regular vertices of the mesh, the limit surface features a quartic and $\mathcal{C}^2$-continuous parameterization. The surface triangles that are adjacent to a feature point or a curve determine the region where the limit surface is $\mathcal{C}^0$-continuous. We remark that this discontinuity in the derivatives is confined. Fig. 3.4 shows a regular mesh featuring inner triangles in dark gray and triangles adjacent to a feature curve (bold) in light gray. In this configuration, the limit surface is of class $\mathcal{C}^2$ only on the dark gray region. The limit curve determined by the bold edges is also of class $\mathcal{C}^2$.

This smooth limit model serves as surrogate geometry for the mesh curving problem. In Sect. 3.6, we approximate it by means of piecewise polynomial surface meshes that, in some cases, inherit its continuity.

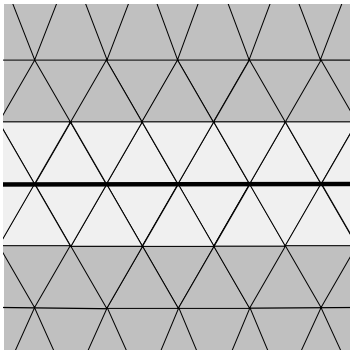


Figure 3.4: Regular mesh around a feature curve (bold). Elements adjacent to the feature curve are colored with light gray, while the elements not adjacent to the curve are colored in dark gray.

## 3.6 Approximation of the limit model

This section is devoted to the generation and analysis of piecewise polynomials approximations of the limit model. In Sect. 3.6.1, we exploit the parameterization of the limit model to generate nodal high-order surface meshes interpolating the limit model. Next, in Sect. 3.6.2, we analyze the smoothness of the high-order surface mesh. In Sect. 3.6.3, we use the parameterization of the limit model to compute the distance between the mesh and the limit model and, in Sect. 3.6.4, we detail the measure used to automatically suggest to the practitioners the geometry features to smooth.

### 3.6.1 Generation of high-order surface meshes

In this section, we present one of the main contributions of this chapter. Given a linear surface mesh, we generate a surface mesh of polynomial degree $q$ such that the nodes of the high-order element interpolate the limit model. Since there exists an explicit parameterization of the limit model, see Sect. 3.5.1, it is possible to map any point in the domain onto the limit model. In Jiménez-Ramos et al. (2020), we proposed a methodology that enabled us to use subdivision schemes to determine high-order meshes of $q = 2^k$, $k \geq 1$, with equispaced distribution of nodes. These meshes inherited the continuity properties of subdivision schemes but were limited by the structure of those schemes. By using the parameterization of the limit model, we can generalize the usage of subdivision schemes to generate high-order meshes of arbitrary polynomial degree and nodal distribution. Thus, nodal sets with better interpolation

**Algorithm 3.4** Generate high-order surface mesh interpolating the limit model.

**Input:** Linear surface mesh $\mathcal{M}$, Linear model $\Omega^1$, Polynomial degree $q$, Interpolation nodes $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$

**Output:** High-order surface mesh $\mathcal{M}^q$, High-order model $\Omega^q$

1: **function** `GenerateHOSurfaceMesh`($\mathcal{M}$, $\Omega^1$, $q$, $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$)
2: $\quad\mathcal{M}_C \leftarrow$ `GenerateNewControlMesh`($\mathcal{M}$, $\Omega^1$)
3: $\quad\{T_i^q\} \leftarrow$ `BuildHighOrderTopology`($\mathcal{M}$, $q$)
4: $\quad$**for** each triangle $T_i^1$ of $\mathcal{M}$ **do**
5: $\quad\quad\boldsymbol{x}_{T_i^q}^{\infty} \leftarrow$ `MapOntoLimitManifold`($\mathcal{M}_C$, $\Omega^1$, $T_i^1$, $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$)
6: $\quad$**end for**
7: $\quad\mathcal{M}^q \leftarrow$ `SetMesh`($\{\boldsymbol{x}_i^{\infty}\}$, $\{T_i^q\}$)
8: $\quad\Omega^q \leftarrow$ `GenerateHOModel`($\mathcal{M}$, $\Omega^1$, $\mathcal{M}^q$)
9: $\quad$**return** $\mathcal{M}^q$, $\Omega^q$
10: **end function**

properties can be used, as it is the case of the quasi-Lebesgue distributions presented in Warburton (2006).

The main algorithm is described in Algorithm 3.4. Given a linear triangular surface mesh $\mathcal{M}$, a linear model $\Omega^1$, a polynomial degree $q$, and a nodal distribution of degree $q$ in the master triangle in barycentric coordinates, $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$, the function `GenerateHOSurfaceMesh` returns a surface mesh of polynomial degree $q$ interpolating the limit model, and the associated high-order model $\Omega^q$. First, in Line 2, we cast the triangular surface mesh to a new control mesh to ensure that the location of the initial vertices is preserved in the high-order mesh, as detailed in Appendix A. Following, in Line 3, we generate the topology $\{T_i^q\}$ of the high-order mesh. Then, in Line 5, for each triangle $T_i^1$ of the linear mesh, we map the interpolation nodes onto the limit model by means of the evaluation of the element-wise parameterization described in Sect. 3.5.1. The position of the interpolation nodes on the limit model determines the position of the nodes of the high-order element. Finally, in Line 7, we set the mesh of polynomial degree $q$, and compute the associated high-order model $\Omega^q$, Line 8.

We highlight that the surrogate geometry for the generation of the high-order mesh is determined by the initial linear mesh and the given geometry features. However, as we detail in Sect. 3.6.2, prior subdivision steps improve the smoothness of the curved high-order mesh that approximates the surrogate. Therefore, if desired, after computing the new control mesh in Line 2, a new finer straight-sided mesh could be generated by applying several subdivision steps. From this point, the curving

procedure would continue as detailed in Algorithm 3.4.

We note that if the interpolation nodes follow an equispaced distribution and the polynomial degree is $q = 2^k$, $k \geq 1$, this method provides the same result as the algorithm presented in Jiménez-Ramos et al. (2020). In that method, the high-order nodes are generated explicitly, and therefore, it is more efficient in terms of computational time.

### 3.6.2 Smoothness of the high-order surface mesh

In this section, we analyze the smoothness of the generated high-order meshes. We highlight that the smoothness of the mesh depends on the chosen polynomial degree. Three cases can be distinguished: the quadratic, the cubic, and any higher polynomial degree.

The generated meshes of polynomial degree $q = 2$ approximate the limit model by interpolating the limit curve (surface) with third-order accuracy. Since the limit curve (surface) features a cubic (quartic) parameterization around inner regular configurations, meshes of polynomial degree 2 are strictly $\mathcal{C}^0$-continuous, and no guarantees of the $\mathcal{C}^2$-continuity are given by the proposed subdivision-based curving method.

Next, we analyze the smoothness of the meshes of polynomial degree three. First, the inner edges of the feature curves of the high-order mesh exactly capture the $\mathcal{C}^2$-continuous limit curve. This is so because the limit curve is parameterized by a third-degree polynomial, and the elements are described by shape functions of degree three too. For edges incident to feature points, we can only guarantee $\mathcal{C}^0$-continuity. Regarding the limit surface, it is interpolated at the nodes and approximated with fourth-order accuracy.

Following, we study the smoothness of the meshes of polynomial degree $q \geq 4$. As for meshes of polynomial degree three, the high-order mesh exactly captures the $\mathcal{C}^2$-continuous limit curve in the inner edges. Similarly to the curve case, the nodes of the surface mesh also interpolate the limit surface. However, the surface mesh does not inherit the smoothness of the limit surface straight-forwardly as in the curve case. This is so since the limit surface is parameterized element-wise, but the parameterization is of degree four only in a regular element, that is, in an element where its three vertices have six neighbors.

In the interior of an element, the mesh is of class $\mathcal{C}^\infty$. Therefore, the smoothness of the surface mesh is to be analyzed along the edges (interfaces between two inner

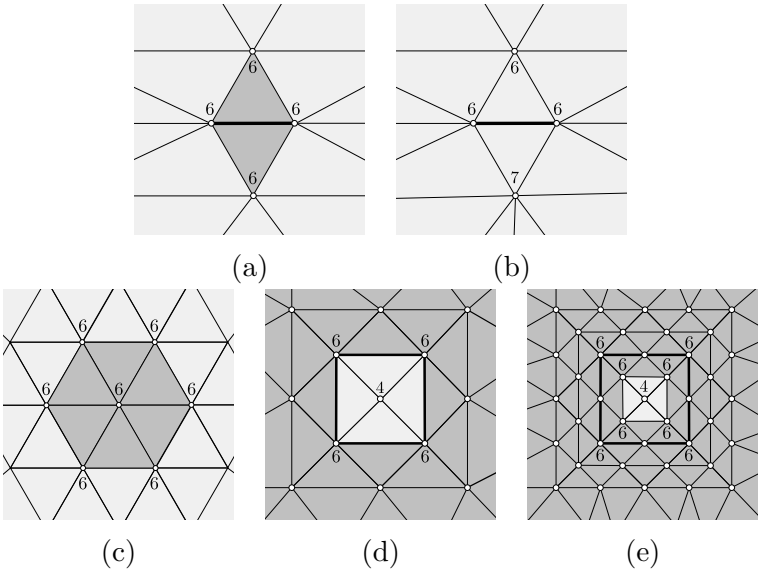



Figure 3.5: Mesh configurations to ensure $\mathcal{C}^2$ smoothness (dark gray). (a) Regular edge (bold). (b) Irregular edge (bold). (c) Regular patch around a regular vertex. (d) Irregular patch (light gray) around an irregular vertex. (e) Mitigation of the irregular patch after subdividing the mesh.

surface elements) and vertices (interfaces between more than two inner surface elements). We can only guarantee $\mathcal{C}^0$-continuity on the edges and vertices of triangles adjacent to sharp features. For the case of elements not adjacent to sharp features, we first analyze the case between two elements that share an edge. We say an edge is a *regular edge* if all the vertices of the two triangles that share such edge are regular, *i.e.* if the vertices have six neighbors, Fig. 3.5(a). In this case, the two elements of degree $q \geq 4$ that share the edge interpolate exactly the quartic limit surface. Hence both elements are exactly equal to the limit surface and, since the limit surface is $\mathcal{C}^2$-continuous, the interface (edge) between the two elements also is. In general, no guarantee of the continuity of the derivatives can be deduced along an edge that is not regular, see Fig. 3.5(b).

Now, we analyze the smoothness of the mesh around the vertices of the inner surface elements. Given a regular vertex (with six neighbors), if all the edges incident to it are regular then the surface mesh is of class $\mathcal{C}^2$ around such vertex. In particular, if all the edges incident to a regular vertex are regular, then all its neighbor vertices are also regular, as observed in Fig. 3.5(c). In such regions, colored in Fig. 3.5(c) in dark gray, the surface mesh captures exactly the limit surface and inherits all its features. The presence of an irregular vertex, as illustrated in Fig. 3.5(d), implies the surface

mesh approximates the limit surface, rather than exactly capturing it. Therefore, on the one hand, around regular patches, we are able to interpolate and exactly capture the limit surface and obtain a $\mathcal{C}^2$-continuous surface mesh. On the other hand, around irregular patches, the limit surface is interpolated and approximated with accuracy of order $q+1$ but not matched exactly.

Some prior subdivisions can be applied to the initial linear mesh to improve the smoothness of the high-order mesh. The linear mesh, Fig. 3.5(d), can be subdivided before generating the mesh of degree $q$. As observed in Fig. 3.5(e) in contrast to Fig. 3.5(d), the light gray irregular region where the limit surface (and subsequently its smoothness) is not exactly captured is reduced. Exploiting prior refinements of the linear mesh, the regions where the high-order surface mesh is not $\mathcal{C}^2$-continuous can be successively reduced. This new finer high-order mesh determines a better approximation of the surrogate geometry, and consequently, its smoothness is also improved.

### 3.6.3 Distance to the limit model

In this section, we use the parameterization of the limit model, see Sect. 3.5.1, to measure the accuracy of the generated high-order mesh with respect to the limit model.

For the model $\Omega^q$, the $s$-th surface $\mathcal{T}^q$ is a triangulation of polynomial degree $q$ composed of triangular facets, $\mathcal{T}^q = \bigcup_{f=1}^{n_f} T_f^q$. Then, we define the parameterization of the surface $\mathcal{T}^q$ element-wise for each $T_f^q$ as

$$\begin{aligned} \phi_f^q \colon \hat{T} &\to \mathcal{T}^q \subset \Omega^q \\ \boldsymbol{\lambda} &\mapsto \phi_f^q(\boldsymbol{\lambda}), \end{aligned}$$

where $\phi_f^q$ is the isoparametric mapping between the master triangle and the high-order element $T_f^q$ of $\mathcal{T}^q$.

Denote by $\mathcal{T}^\infty$ the $s$-th surface of the limit model, see Eq. (3.3). We define the distance between the limit surface $\mathcal{T}^\infty$ and the approximation given by the high-order surface mesh $\mathcal{T}^q$ as

$$d(\mathcal{T}^\infty, \mathcal{T}^q) = \max_{f=1,\dots,n_f} \sup_{\boldsymbol{\lambda}\in\hat{T}} \left\| \phi_f^\infty(\boldsymbol{\lambda}) - \phi_f^q(\boldsymbol{\lambda}) \right\|_2, \tag{3.4}$$

where the *supremum*, the least upper bound, is computed for the points $\boldsymbol{\lambda} \in \hat{T}$, $\|\cdot\|_2$ is the Euclidean norm, and the maximum is taken for all the triangles of the

triangulation. In practice, this value is approximated by the maximum distance over a fine grid of points $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_d}$ in the reference domain,

$$\sup_{\boldsymbol{\lambda}\in\hat{T}} \left\| \phi_f^{\infty}(\boldsymbol{\lambda}) - \phi_f^{q}(\boldsymbol{\lambda}) \right\|_2 \approx \max_{j=1,\dots,N_d} \left\| \phi_f^{\infty}(\boldsymbol{\lambda}_j) - \phi_f^{q}(\boldsymbol{\lambda}_j) \right\|_2 .$$

Then, the distance between the limit model $\Omega^{\infty}$ and the model $\Omega^q$ is defined as

$$d(\Omega^{\infty}, \Omega^q) = \frac{1}{L(\Omega^{\infty})} \max_{s=1,\dots,n_s} d(\mathcal{T}_s^{\infty}, \mathcal{T}_s^q), \tag{3.5}$$

the maximum of the distances between the $s$-th surface $\mathcal{T}_s^{\infty}$ and $\mathcal{T}_s^q$, Eq. (3.4), $s = 1,\dots,n_s$, adimensionalized with the characteristic length of the model $L(\Omega^{\infty})$.

We highlight that, as detailed in Sect. 3.6.2, the generated high-order mesh interpolates the limit model on the nodes, and therefore, if the polynomial degree is greater than three, $q \geq 3$, the polylines exactly coincide with the limit curves. If the polynomial degree is greater than four, $q \geq 4$, the mesh exactly coincides with the limit surface around the regular vertices. Thus, the distance between a mesh of polynomial degree $q \geq 4$ and the limit model is non-zero only around irregular vertices and along the interfaces between sharp features.

### 3.6.4 Automatic feature detection

As previously introduced in Sect. 3.3.1, the input linear model is composed of the union of features points, curves, and surfaces describing the geometry to represent. However, these features may not reproduce the simulation intent, and it may be necessary to smooth them. This process has to be performed by the user and may require significant human labor, depending on the complexity of the input model. We propose to study the continuity of the normal (tangent) vector along (at) a feature curve (point) to decide whether it is suggested to be smoothed.

Given a linear mesh which interpolates a surface or curve, we want to reconstruct the implicit high-order information of the discretization not fully available in the linear model. To this aim, we use the limit model from subdivision since it takes into account the stencil, and therefore, it reconstructs part of the initial high-order information. Consequently, to automatically detect the smooth features we propose to use the high-order model $\Omega^q$ which approximates the limit model and provides direct and explicit access to the approximation of the high-order information.

Specifically, consider the $c$-th curve $\mathcal{S}^q$ of the model $\Omega^q$ composed of the union of poly-segments of degree $q$, $\mathcal{S}^q = \bigcup_{e=1}^{n_e} S_e^q$. For each segment $S_e^q$, there are two

triangles $T_i^q$ and $T_j^q$ such that $S_e^q = T_i^q \cap T_j^q$. Then, for a point $\boldsymbol{x} \in S_e^q$, we consider the inner angle function

$$\alpha(\boldsymbol{x}) = \arccos\left(\boldsymbol{n}_i(\boldsymbol{x}) \cdot \boldsymbol{n}_j(\boldsymbol{x})\right), \tag{3.6}$$

where $\boldsymbol{n}_k(\boldsymbol{x})$ denotes the unitary normal vector defined from triangle $T_k^q$. This function accounts for the angle between the normal vectors at point $\boldsymbol{x}$. If the normal vectors at $\boldsymbol{x}$ are continuous, then $\boldsymbol{n}_i(\boldsymbol{x}) = \boldsymbol{n}_j(\boldsymbol{x})$, and thus, $\alpha(\boldsymbol{x}) = 0$. On the contrary, if $\boldsymbol{n}_i(\boldsymbol{x}) \neq \boldsymbol{n}_j(\boldsymbol{x})$, then $\alpha(\boldsymbol{x}) > 0$. In particular, the image of $\alpha$ is $[0, \pi]$, and it attains the minimum in a flat configuration, when $\boldsymbol{n}_i(\boldsymbol{x}) = \boldsymbol{n}_j(\boldsymbol{x})$, and the maximum in a reversal configuration, when $\boldsymbol{n}_i(\boldsymbol{x}) = -\boldsymbol{n}_j(\boldsymbol{x})$.

Thus, for each curve $\mathcal{S}$ of the model $\Omega^q$, we compute

$$\alpha_{\mathcal{S}} = \frac{\int_{\mathcal{S}} \alpha(\boldsymbol{x})\,\mathrm{d}x}{\int_{\mathcal{S}} \mathrm{d}x}.$$

Integrating along the curve provides an average and therefore, we avoid undesired detection due to spurious values that may arise when performing the computation edge-wise. The set $R_\delta = \{\mathcal{S}\colon \alpha_{\mathcal{S}} < \delta\}$ is composed of the feature curves with an angle below a desired threshold $\delta$. This set contains the potentially side features that are suggested to the practitioners to smooth. In particular, the two different surfaces adjacent to a curve feature $\mathcal{S} \in R_\delta$ are incident with an angle less than $\delta$ and, therefore, it may indicate that the curve feature has to be smoothed.

Once this set of curves has been identified, we smooth them in the linear model $\Omega^1$ and generate a surface mesh of polynomial degree $q$ and a new model $\Omega^q$ to study the feature points to be potentially smoothed. We distinguish several cases depending on the number of curves incident to the feature point. If no curves are incident, we suggest smoothing this point. If two curves are incident to a feature point, we proceed analogously as for curves but with the tangent vector instead. We compute the tangent vector at the feature point from the two incident edges and, if the angle between the two tangent vectors is below a desired threshold, we suggest smoothing this feature point. If one or more than two curves are incident to a feature point, the smoothing operation has to be performed manually.

We highlight that each of the feature points (curves) is characterized by a global point (curve) identifier, and therefore, the smoothing operation consists in providing a list of the identifiers of the points (curves) to smooth.

**Algorithm 3.5** Curve volume mesh smoothing features.

**Input:** Linear volume mesh $\mathcal{M}$, Linear model $\Omega^1$, Polynomial degree $q$, Features to smooth R, Interpolation nodes $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$
**Output:** High-order volume mesh $\mathcal{M}^q$, High-order model $\Omega^q$
1: **function** `CurveVolumeMesh`($\mathcal{M}$, $\Omega^1$, $q$, R, $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$)
2: $\Omega^1 \leftarrow$`SmoothFeatures`($\Omega^1$, R)
3: $\mathcal{M}^q$, $\Omega^q \leftarrow$ `GenerateHOVolumeMesh`($\mathcal{M}$, $\Omega^1$, $q$, $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$)
4: **return** $\mathcal{M}^q$, $\Omega^q$
5: **end function**

## 3.7 Curved volume mesh approximating the limit model

In this section, we detail how a linear tetrahedral mesh with marked boundary entities is curved while preserving the sharp features of the model. In Sect. 3.7.1, we detail the sharp-to-smooth modeling of the geometry features and the replacement of the straight-edged boundary of the linear mesh by the curved boundary mesh. In Sect. 3.7.2, the curvature on the boundary is accommodated to the interior using a blending technique. This procedure leads to a high-order tetrahedral mesh where its boundary approximates a surrogate geometry composed of feature surfaces with an interior that is $\mathcal{C}^1$-continuous and $\mathcal{C}^2$-continuous almost everywhere. In addition, the vertices of the high-order mesh are kept in the same position as in the initial linear mesh.

### 3.7.1 Substitute the boundary of the volume mesh

In this section, we detail the subdivision-based curving of a high-order tetrahedral mesh. This process is presented in Algorithm 3.5. First, in Line 2, we smooth the desired feature entities. Then, in Line 3, we generate a high-order volume mesh preserving the sharp features provided by the new model once the original entities have been smoothed. These processes, denoted as `SmoothFeatures` and `GenerateHOVolumeMesh`, are next detailed in Algorithms 3.6 and 3.7.

The first step to curve the volume mesh is to smooth, if necessary, the geometry features present in the original model. Recall that points, curves, and surfaces are characterized by a unique identifier. Thus, in order to smooth a sharp feature, it is enough to provide its identifier. That is, in Algorithm 3.6, the variable `R` contains a

**Algorithm 3.6** Smooth geometry features.

**Input:** Model $\Omega^q$, Features to smooth R
**Output:** Model $\Omega^q$

```
1: function SmoothFeatures(Ω^q, R)
2:   for each feature f in R do
3:     Ω^q ← RemoveFeatureFromList(Ω^q, f)
4:     Ω^q ← MergeIncidentFeatures(Ω^q, f)
5:   end for
6:   return Ω^q
7: end function
```

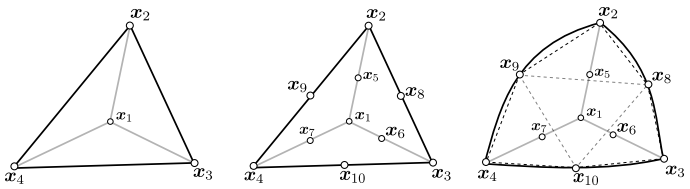
**Algorithm 3.7** Generate high-order volume mesh.

**Input:** Linear volume mesh $\mathcal{M}$, Linear model $\Omega^1$, Polynomial degree $q$, Interpolation nodes $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$
**Output:** High-order volume mesh $\mathcal{M}^q$, High-order model $\Omega^q$

```
1: function GenerateHOVolumeMesh(M, Ω^1, q, {λ_j}_{j=1,...,N_q})
2:   ∂M ← ExtractBoundary(M)
3:   ∂M^q, Ω^q ← GenerateHOSurfaceMesh(∂M, Ω^1, q, {λ_j})
4:   M^q ← IncreasePolynomialDegree(M, q)
5:   M^q ← ReplaceBoundary(M^q, ∂M^q)
6:   M^q ← AccommodateCurvature(M^q, M)
7:   if M^q is low quality then
8:     M^q ← OptimizeMesh(M^q)
9:   end if
10:  return M^q, Ω^q
11: end function
```

list of the identifiers of the feature points and curves to be smoothed. Specifically, the smoothing of a feature is composed of two steps: removing the feature from the list of features to preserve, Line 3; and merging the features incident to such feature, Line 4. Since each feature is described by a unique identifier, the process of merging the incident features reduces to assigning the same identifier to these features. The features to smooth are manually provided by the user, however, to reduce the human labor of manually selecting the features to smooth, the indicator proposed in Sect. 3.6.4 can be used to determine the features that have to be potentially smoothed, as illustrated in Sect. 3.8.5.

Next, the curving method based on hierarchical subdivision and blending is performed. The generation of a high-order volume mesh is described in Algorithm 3.7. First, given a linear tetrahedral mesh, the linear model $\Omega^1$, the polynomial degree




(a) (b) (c)

Figure 3.6: Curving of the boundary for a boundary element of polynomial degree $q = 2$. (a) Linear physical element, where the face (2 3 4) belongs to the boundary. (b) Straight-edged physical element of polynomial degree two. (c) Curved boundary element of polynomial degree two, displaying with dashed lines the four elements from the subdivision scheme applied to the boundary.

$q$, and the interpolation nodes of the master tetrahedron in barycentric coordinates $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$, Fig. 3.6(a), we extract its boundary, Line 2. The boundary is a surface mesh that inherits the geometry features of the volume mesh. Next, in Line 3, we call the function `GenerateHOSurfaceMesh` to generate a surface mesh of polynomial degree $q$ preserving the sharp features and interpolating the limit model at the boundary nodes of the interpolation set $\{\boldsymbol{\lambda}_j\}_{j=1,\dots,N_q}$, see Sect. 3.6.1. Third, we generate a straight-edged high-order volume mesh, Line 4, illustrated in Fig. 3.6(b). Following, in Line 5, we replace the boundary of the straight-edged mesh by the curved surface mesh, see Fig. 3.6(c). Then, in Line 6, the curvature of the surface is accommodated to the elements adjacent to the boundary, using a blending technique to be described in Sect. 3.7.2. Finally, if the mesh contains low-quality elements, it is optimized using the method in Gargallo-Peiró et al. (2015c,b), see Line 8.

The methodology proposed in this chapter can also be used to generate, given an initial linear mesh, finer linear meshes that successively improve the approximation of the surrogate geometry. To do so, the generated high-order mesh can be reinterpreted as a linear mesh by the decomposition of each high-order element into linear elements. Specifically, the reference high-order element is decomposed into several structured linear elements determined by the high-order nodes. If the linear mesh contains low-quality elements, the optimization procedure described in Gargallo-Peiró et al. (2015c) is applied to ensure a valid mesh.

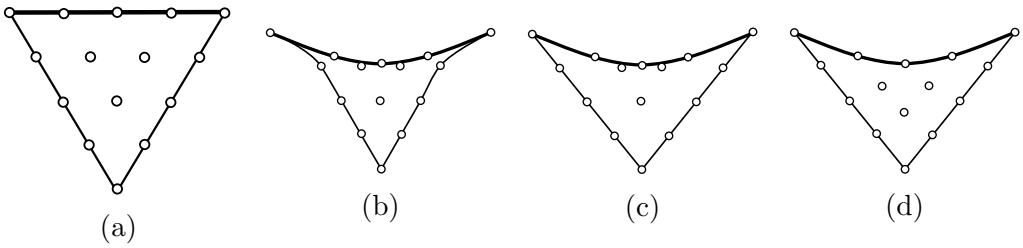



Figure 3.7: Accommodating the curvature to a triangular element of polynomial degree $q = 4$. (a) Straight-edged triangle with a boundary edge (in bold). (b) Triangle with curved boundary. (c) Transfinite interpolation applied to the edges, (d) and to the face.

### 3.7.2 Accommodation of the curvature of the boundary

In Algorithm 3.7, after replacing the curved boundary in the straight-sided high-order mesh, Line 5, the obtained high-order mesh may contain low-quality or tangled elements, see Fig. 3.7(a) and 3.7(b). In isotropic mesh generation, only boundary tetrahedra are affected and therefore, the number of invalid elements is small compared with the scale of the generated meshes. Thus, similarly to Johnen et al. (2018), as an attempt to improve the mesh quality in a fast and explicit manner, in Line 6 in function `AccommodateCurvature`, we use Transfinite Interpolation (TFI) (Perronnet, 1998) to accommodate the curved surface to those entities of the boundary elements not present in the surface mesh. Specifically, given a boundary element, we use transfinite interpolation hierarchically on its entities to accommodate the curving of the boundary. That is, first we relocate the nodes on edges, then nodes on faces, and finally, nodes in the interior of tetrahedra. We explicitly detail the position of all the interior nodes of the boundary elements in Appendix B.

The analogous procedure for triangular meshes is illustrated in Fig. 3.7. In Fig. 3.7(a), we show a straight-sided boundary triangle of degree $q = 4$, with the boundary edge highlighted in bold. After the curving of the boundary, the boundary nodes have been relocated, see Fig. 3.7(b). In Fig. 3.7(c), we illustrate the relocation of the nodes on the interior edges of the triangle, and the interior nodes in Fig. 3.7(d) account for the curvature propagated through the edges.

We remark that this method does not guarantee to repair the invalid elements, neither ensures an increase in the element quality. However, it is an explicit and fast formulation, which in practice represents a good initial condition for mesh curving methods when no geometry is available (Gargallo-Peiró et al., 2015b; Ruiz-Gironés

et al., 2019; Moxey et al., 2016; Toulorge et al., 2013; Persson and Peraire, 2008). In all the tested applications, see Sect. 3.8, the procedure improves significantly the quality of the meshes. Once the TFI-based relocation process is finalized, if low quality or inverted elements are present, we perform the non-linear quality optimization procedure presented in Gargallo-Peiró et al. (2015b).

## 3.8 Results

In this section, we present several examples to illustrate the main features of the methods presented in this chapter. As a proof of concept, the proposed algorithms have been developed in Anaconda Python (Van Rossum and Drake Jr, 1995). The prototyping code is sequential (one execution thread) and non-vectorized. All the examples have been run on a MacBook Pro (with one dual-core Intel Core i5 CPU, a clock frequency of 2.3 GHz, and a total memory of 16 GBytes).

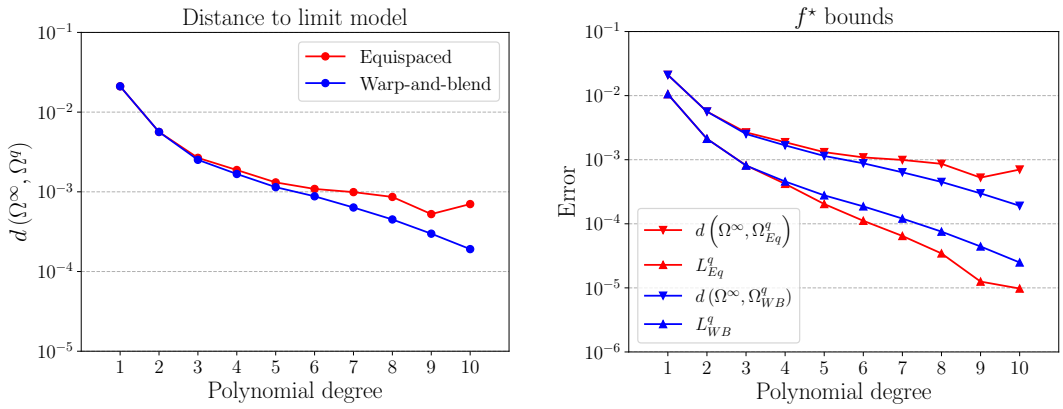
In all the examples, we validate both the high-order boundary and volume meshes using the Jacobian-based distortion measure proposed in Gargallo-Peiró et al. (2016b, 2015c). In particular, the quality of a high-order element is computed with respect to the corresponding original straight-edged element in the linear mesh.

### 3.8.1 Convergence of the high-order surface mesh to the limit model

In this example, we analyze the convergence of the surface meshes of polynomial degree $q$ to the limit model when $q$ is increased, $q = 1, \dots, 10$. This kind of convergence is useful in such applications where the initial partition of elements has to be preserved. We consider a sphere of radius 1 discretized with a linear triangular mesh composed of 450 nodes and 896 elements, see Fig. 3.9(a). The mesh is colored according to the distance to the limit model, see Eq. (3.5), with $L(\Omega^\infty) = 1$, the radius of the sphere. Note that the distance at the vertices is zero due to the interpolation property of the scheme. Then, for each polynomial degree $q$, we generate two surface meshes with two different nodal distributions: one equispaced, and the other defined using a warp-and-blend technique aimed at quasi-minimizing interpolation errors presented in Warburton (2006).

| $q$ | $\Lambda_{\text{Eq}}$ | $d\left(\Omega^{\infty}, \Omega^{q}_{\text{Eq}}\right)$ | $L_{\text{Eq}}$ | $\Lambda_{\text{WB}}$ | $d\left(\Omega^{\infty}, \Omega^{q}_{\text{WB}}\right)$ | $L_{\text{WB}}$ |
|---|---|---|---|---|---|---|
| 1 | 1.00 | $2.10 \cdot 10^{-2}$ | $1.05 \cdot 10^{-2}$ | 1.00 | $2.10 \cdot 10^{-2}$ | $1.05 \cdot 10^{-2}$ |
| 2 | 1.66 | $5.64 \cdot 10^{-3}$ | $2.12 \cdot 10^{-3}$ | 1.66 | $5.64 \cdot 10^{-3}$ | $2.12 \cdot 10^{-3}$ |
| 3 | 2.27 | $2.67 \cdot 10^{-3}$ | $8.15 \cdot 10^{-4}$ | 2.11 | $2.52 \cdot 10^{-3}$ | $8.10 \cdot 10^{-4}$ |
| 4 | 3.47 | $1.88 \cdot 10^{-3}$ | $4.20 \cdot 10^{-4}$ | 2.66 | $1.67 \cdot 10^{-3}$ | $4.57 \cdot 10^{-4}$ |
| 5 | 5.45 | $1.31 \cdot 10^{-3}$ | $2.04 \cdot 10^{-4}$ | 3.12 | $1.14 \cdot 10^{-3}$ | $2.78 \cdot 10^{-4}$ |
| 6 | 8.75 | $1.09 \cdot 10^{-3}$ | $1.11 \cdot 10^{-4}$ | 3.70 | $8.79 \cdot 10^{-4}$ | $1.87 \cdot 10^{-4}$ |
| 7 | 14.35 | $9.92 \cdot 10^{-4}$ | $6.46 \cdot 10^{-5}$ | 4.27 | $6.36 \cdot 10^{-4}$ | $1.21 \cdot 10^{-4}$ |
| 8 | 24.01 | $8.63 \cdot 10^{-4}$ | $3.45 \cdot 10^{-5}$ | 4.96 | $4.50 \cdot 10^{-4}$ | $7.55 \cdot 10^{-5}$ |
| 9 | 40.92 | $5.26 \cdot 10^{-4}$ | $1.25 \cdot 10^{-5}$ | 5.74 | $2.98 \cdot 10^{-4}$ | $4.43 \cdot 10^{-5}$ |
| 10 | 70.89 | $7.02 \cdot 10^{-4}$ | $9.76 \cdot 10^{-6}$ | 6.67 | $1.91 \cdot 10^{-4}$ | $2.49 \cdot 10^{-5}$ |

Table 3.1: Interpolation with equispaced and non-equispaced nodes of a sphere limit model for polynomial degree $q$, $q = 1, \ldots, 10$: Lebesgue constant, distance to the limit model, and lower bound of the distance.




(a) (b)

Figure 3.8: Convergence of the bounds and the distance to the limit model for the sphere meshes of polynomial degree $q$, $q = 1, \ldots, 10$, with equispaced and non-equispaced distribution sets: (a) distance, and (b) lower and upper bounds of the distance of the best approximating polynomial.

For each polynomial degree and interpolation set, we compute the distance between the high-order surface mesh and the limit model, see Eq. (3.5). In Table 3.1, we report the distances between the meshes and the limit model, for the different degrees and nodal distributions. In Fig. 3.8(a), we plot the logarithm of the distance in terms of the polynomial degree using red for the data of the equispaced distribution and blue for the data of the warp-and-blend distribution. We observe that for the

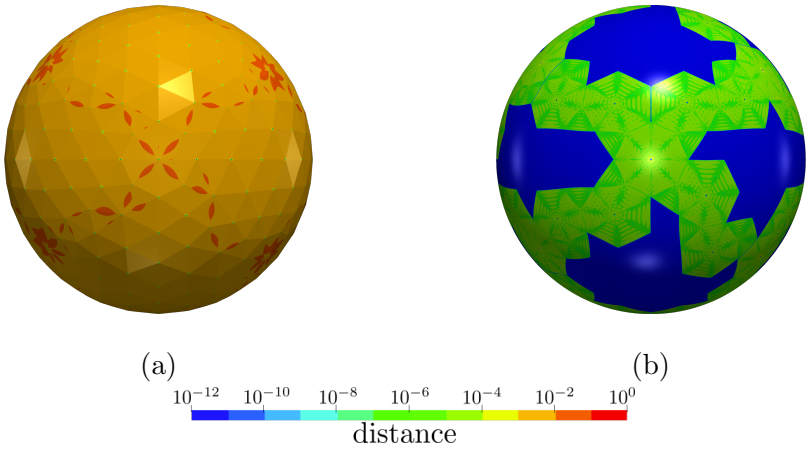



Figure 3.9: Mesh of a sphere of polynomial degree (a) one, $q = 1$, and (b) ten, $q = 10$, both colored according to the distance to the limit model.

warp-and-blend distribution, the logarithm of the distance seems to converge linearly with the polynomial degree $q$. In Table 3.1, we also show the Lebesgue constant of the equispaced distribution, $\Lambda_{\text{Eq}}$, and of the warp-and-blend distribution, $\Lambda_{\text{WB}}$. If we denote by $f^\star$ the best approximating polynomial of the limit model $\phi^\infty$, since it is optimal, we have the inequality

$$\|\phi^\infty - f^\star\|_\infty \leq \|\phi^\infty - \phi^q\|_\infty .$$

Furthermore, for a given interpolation set, the Lebesgue constant bounds the interpolation error in terms of the error made by the best approximating polynomial,

$$\|\phi^\infty - \phi^q\|_\infty \leq (1 + \Lambda) \|\phi^\infty - f^\star\|_\infty . \tag{3.7}$$

Combining these inequalities, we bound the error made by $f^\star$,

$$\frac{\|\phi^\infty - \phi^q\|_\infty}{1 + \Lambda} \leq \|\phi^\infty - f^\star\|_\infty \leq \|\phi^\infty - \phi^q\|_\infty . \tag{3.8}$$

The term on the left is denoted as $L^q_{\text{Eq}}$ and $L^q_{\text{WB}}$ for the equispaced and warp-and-blend distribution, respectively, and they are also reported in Table 3.1. In Fig. 3.8(b), we plot the lower and upper bound for the two interpolation sets. For low polynomial degrees, both distributions provide similar results. However, as the polynomial degree increases, the equispaced distribution becomes more uncertain, while the warp-and-blend distribution is more accurate and the region is more restricting. The error of the best approximating polynomial $f^\star$ is between the two bounds and, more concretely,

in the more restricting region. In this particular example, this region is given by the bounds of the warp-and-blend distribution and, in general, we would expect the same behavior since the warp-and-blend distribution has better interpolation properties.

We might intuitively expect that for the higher polynomial degrees, equidistant point distributions lead to significant error oscillations. Nevertheless, in this example, we can see that both point distributions lead to almost comparable errors. To understand this error similarity, we can use the upper bound of the interpolation error in terms of the Lebesgue constant, see Eq. (3.7). In particular, for polynomial degree 10, if we divide the upper bound of the error for both point distributions, we see that equidistant points feature an error at most 71/7 times larger than the error for warp and blend points. This error relation, around 10, agrees with the numerical results of Table 3.1.

Finally, in Fig. 3.9(b), we illustrate the mesh of polynomial degree $q = 10$ colored according to the pointwise distance to the limit model. It can be observed that, in the neighborhood of the regular vertices, the mesh exactly captures the limit model and, consequently, the distance is zero. On the contrary, around irregular vertices, the mesh approximates the limit model and the distance is non-zero.

### 3.8.2 Volume curving of a structured topography mesh

In this example, we illustrate the features of our method with a structured linear mesh that discretizes the topography on the Escudo mountain range (Spain). The original data is provided as a level curve map that is cast to a structured mesh with the level values on the nodes.

We consider a tetrahedral mesh that discretizes the region enclosed by the topography and a planar ceiling, located at the desired height. The tetrahedral mesh is regular and is generated using the mesher presented in Gargallo-Peiró et al. (2015a, 2018). From the linear mesh, we generate a curved high-order mesh of polynomial degree four using the procedure detailed in Sect. 3.7. Since the surface nodes are regular, the high-order topography surface mesh is $\mathcal{C}^2$-continuous and exactly captures the limit surface, as detailed in Sect. 3.6.2. To check the implementation, the distance between the mesh of polynomial degree four and the limit model, see Eq. (3.5), has been computed, verifying that the result is zero.

In Fig. 3.10(a), we show the initial linear mesh, composed of $4.8 \cdot 10^5$ nodes and $2.6 \cdot 10^6$ tetrahedra. The high-order boundary mesh is generated in 106 minutes and is

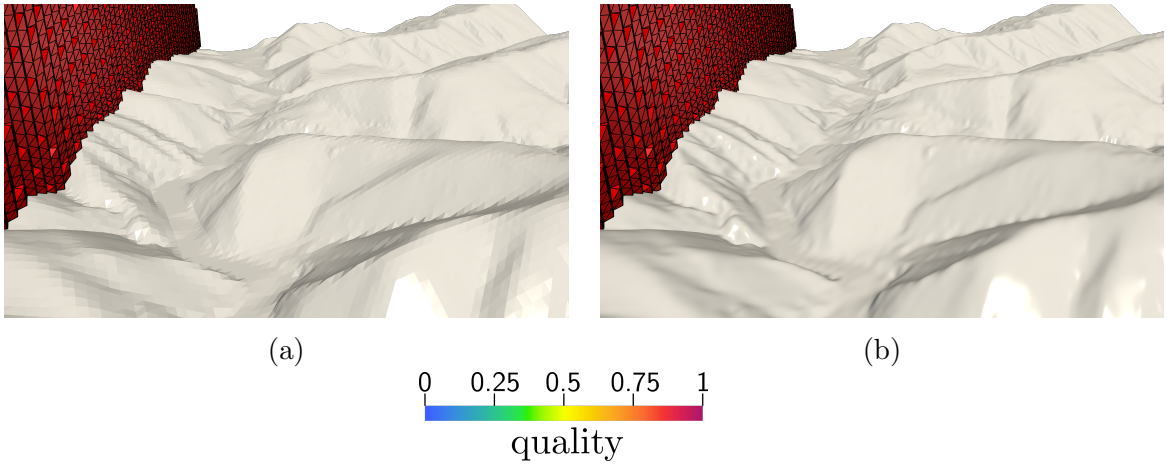



Figure 3.10: Curving of a tetrahedral mesh of the Escudo mountain range (Spain). (a) Linear mesh. (b) Curved mesh of polynomial degree $q = 4$. Elements of the volume meshes are colored with their elemental quality.

| | Min Q | # inv |
|---|---|---|
| Boundary | 0.93 | 0 |
| Volume (no TFI) | 0 | 169 |
| Volume (TFI) | 0.80 | 0 |

Table 3.2: Quality statistics of a mesh of polynomial degree $q = 4$ for the Sierra del Escudo (Spain) topography.

composed of $1.1 \cdot 10^6$ nodes and $1.4 \cdot 10^5$ triangles, featuring a minimum quality of 0.93. The curved high-order volume mesh takes 18 minutes to be generated and contains 169 inverted elements. The process of accommodating the curvature of the boundary detailed in Sect. 3.7.2 is performed to $4.1 \cdot 10^5$ elements abutting the boundary. This blending untangles all the invalid elements in 7 minutes, attaining a minimum quality of 0.8. Finally, in Fig. 3.10(b), we show the mesh of polynomial degree four composed of $2.9 \cdot 10^7$ nodes and $2.6 \cdot 10^6$ elements. The statistics regarding the mesh quality on the different steps of the mesh generation procedure are presented in Table 3.2.

### 3.8.3 Smoothness of the boundary mesh

In this example, we analyze the smoothness of the interpolative model $\Omega^q$ for the mesh of the sphere in Sect. 3.8.1. When the polynomial degree is greater than four, $q \geq 4$, the surface mesh features $\mathcal{C}^2$-continuity around regular patches, see Sect. 3.6.2. Thus,

| $q$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\max_e \alpha_{\infty,e}$ | 9.00 | 5.49 | 1.72 | 1.54 | 1.02 |

Table 3.3: Maximum angle between the normal vectors of adjacent elements for the surface mesh of the sphere of polynomial degree $q$, $q = 1, \ldots, 5$.

normal vectors are continuous along regular edges. On the contrary, we may observe discontinuous normal vectors along edges around irregular points which indicate that the mesh is not $\mathcal{C}^1$-continuous.

To quantify the smoothness of the surface mesh, we compute the distortion of the normal vector. Specifically, consider the $s$-th surface $\mathcal{T}^q$ composed of triangular facets, $\mathcal{T}^q = \bigcup_{f=1}^{n_f} T_f^q$. Then, for each internal edge $S_e^q$ such that $S_e^q = T_i^q \cap T_j^q$, we compute

$$\alpha_{\infty,e} = \max_{k=1,\ldots,N} \alpha\left(x_k\right),$$

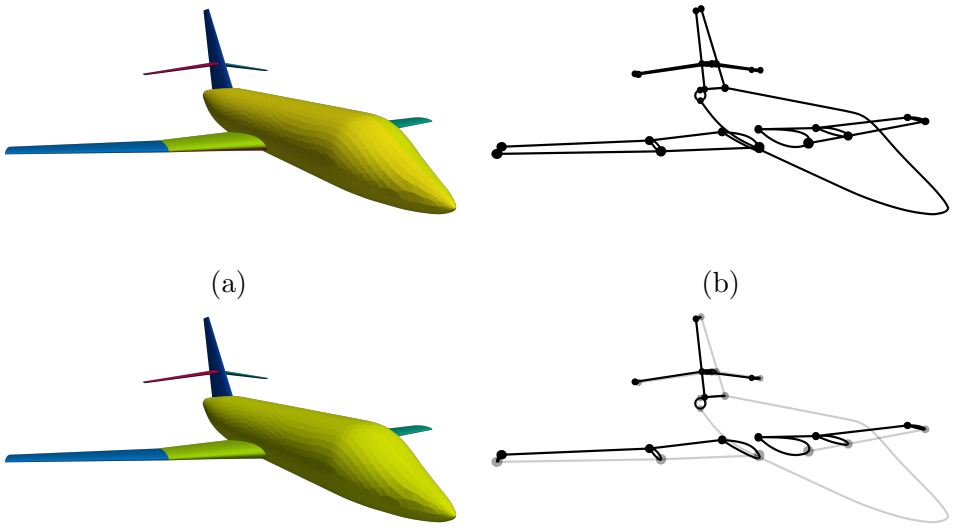
where $x_k \in S_e^q$ is a sampling point, and $\alpha\left(x\right)$ is defined in Eq. (3.6). The value $\alpha_{\infty,e}$ approximates the maximum angle formed by the normal vectors along the edge $S_e^q$.

In Table 3.3, we show the maximum angle between normal vectors for the mesh of the sphere in Sect. 3.8.1 for different polynomial degrees. We observe a significant improvement with cubic meshes with respect to the original linear mesh. For quartic and quintic polynomial meshes, the maximum angle formed by two normal vectors from adjacent elements is less than two degrees.

### 3.8.4 Sharp-to-smooth modeling and mesh volume curving

In this example, we illustrate the capability of our method to perform a sharp-to-smooth modeling in different features of the geometry. To preserve the simulation intent, we smooth some of the feature entities present in the original model and thus provide a new model improving the smoothness of the surrogate geometry. Each feature point (node of the mesh), curve (set of edges of the mesh), and surface (set of triangles of the mesh) is associated with a unique identifier. Therefore, to smooth a feature, it is enough to know its identifier.

We consider a linear tetrahedral mesh from a CAD legacy model of a simplified Falcon aircraft. The boundary triangles are marked identifying the feature surfaces. The complete linear model is composed of 28 surfaces, 54 curves, and 34 points. As shown in Fig. 3.11(a) and Fig. 3.11(b), the main part of the fuselage is composed of



(a) (b)

(c) (d)

Figure 3.11: Initial and final linear mesh model of a Falcon aircraft. Initial model: (a) surface features colored with their surface identifier, and (b) curve and point features. Final model: (c) virtual surface features colored with their surface identifier, (d) curve and point features smoothed (gray) and preserved (black).

two surfaces and a curve. However, such curve is not desirable since, ideally, we would desire a smooth model along each section of the fuselage. To address this issue, we smooth the curve indicating its unique identifier. The first step consists in removing the curve from the list of feature curves. Following, the two surfaces initially incident to this curve, see Fig. 3.11(a), are merged by identifying the identifiers of the two surfaces with a new but equal identifier, see Fig. 3.11(c). As a result, the whole fuselage is modeled as a smoother virtual surface.

Similarly, we observe that each section of the wing is described by a top and bottom surface, and a leading and trailing edge curve. To preserve the simulation intent, we smooth the feature curve describing the leading edge. This way, the surface at the top and the bottom are merged and join smoothly in the front part of the wing. We highlight that the curve describing the trailing edge is maintained as a sharp feature.

Note that the lateral wing joins the fuselage in a profile described by two curves (top and bottom) and two vertices (front and back). We smooth the feature point in the front. Thus, this point is removed from the list of feature points, and the two curves are merged by identifying their identifiers as a unique one. As a result, we

(a) (b)

Figure 3.12: Close-up view of the leading edge of a Falcon aircraft wing. Zebra mapping on the mesh of polynomial degree four with (a) the initial model, and (b) the final model with the leading edge smoothed.

obtain a single closed curve with a sharp endpoint on the trailing edge.

Similar changes are made in similar features of the mesh to generate the proper model for flow simulation, see Fig. 3.11(c) and Fig. 3.11(d). As highlighted in Sect. 3.7.1, once the identifiers of all the features to smooth are located, the smoothing process is straight-forward. Given the list of identifiers, the smoothing process consists in removing these features from the list of features to preserve and automatically identify the features adjacent to the smoothed feature as a single one. We remark that the smoothing of some of the geometry features does not modify the mesh, only the number of feature points, curves, and surfaces changes. Specifically, the original model of the presented Falcon aircraft contains 34 feature points, 54 curves, and 28 surfaces; while the model with the smoothed features contains 20 points to preserve, 32 curves, and 20 surfaces.

To illustrate the difference between these two models, we take a close look at the leading edge of the wing. In Fig. 3.12(a), we show a zebra mapping on the mesh of polynomial degree $q = 4$ generated with the initial marks. Specifically, we see the isophote bands for a spotlight in front of the leading edge. We observe a discontinuity in the normal vector of the wing along the leading edge. In Fig. 3.12(b), we show a mesh generated with a model in which the leading edge has been smoothed. The nodes originally present in the leading edge are still on the leading edge, but the new points are generated to interpolate the almost everywhere $\mathcal{C}^2$-continuous surrogate geometry. The second mesh model is smoother than the original one along those regions where the features have been smoothed. The leading edge now belongs to

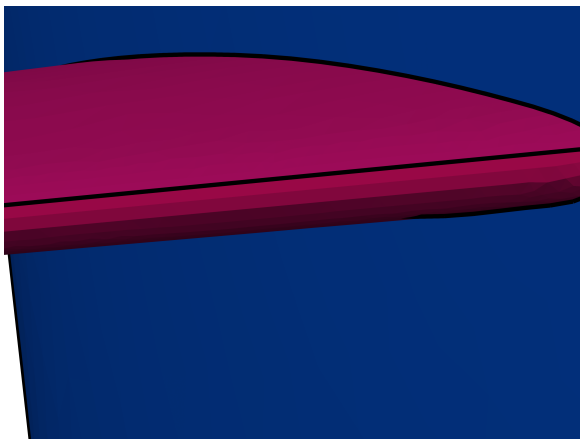


Figure 3.13: Close-up view of the leading edge of a Falcon aircraft rear wing. Surfaces are colored according to their surface identifier.

| $q$ | $\Lambda_{\mathrm{WB}}$ | $d\left(\Omega^{\infty}, \Omega^{q}_{\mathrm{WB}}\right)$ | $L_{\mathrm{WB}}$ |
|---|---|---|---|
| 1 | 1.00 | $1.48 \cdot 10^{-2}$ | $7.42 \cdot 10^{-4}$ |
| 2 | 1.66 | $4.09 \cdot 10^{-4}$ | $1.54 \cdot 10^{-4}$ |
| 3 | 2.11 | $1.98 \cdot 10^{-4}$ | $6.38 \cdot 10^{-5}$ |
| 4 | 2.66 | $9.93 \cdot 10^{-5}$ | $2.71 \cdot 10^{-5}$ |
| 5 | 3.12 | $5.14 \cdot 10^{-5}$ | $1.25 \cdot 10^{-5}$ |

Table 3.4: Interpolation with non-equispaced nodes of a Falcon aircraft limit model for polynomial degree $q$, $q = 1, \ldots, 5$: Lebesgue constant, distance to the limit model, and lower bound of the distance.

the interior of the surface, and therefore, all the nodes interpolate a $\mathcal{C}^1$-continuous surface.

The modeling capability not only allows for the smoothing of geometric features that prevented the model from describing the simulation intent, but it also removes the sharp features that were artificially created by CAD engines. That is, it is possible to repair a CAD model with artificial features. In this example, we observe that the intersection of the two surfaces (top and bottom) describing the rear wing does not occur exactly at the leading edge of the wing, but at the top of the wing, see Fig. 3.13. Indeed, this curve does not appropriately describe the simulation intent and should be smoothed. To this end, we proceed as described above by locating the curve identifier, removing the curve from the list of feature curves to preserve, and automatically assigning the same identifier to the adjacent surfaces.

In Table 3.4, for each polynomial degree $q$, $q = 2, \ldots 5$, we report the distances

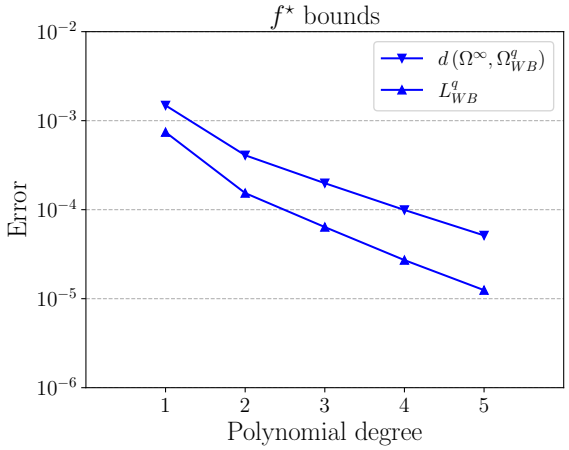



Figure 3.14: Lower and upper bounds of the distance of the best approximating polynomial in terms of the polynomial degree $q$ for the Falcon aircraft.

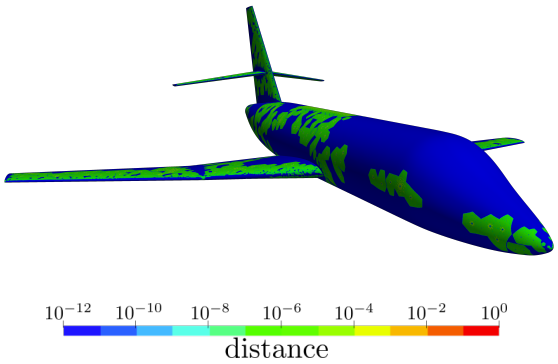


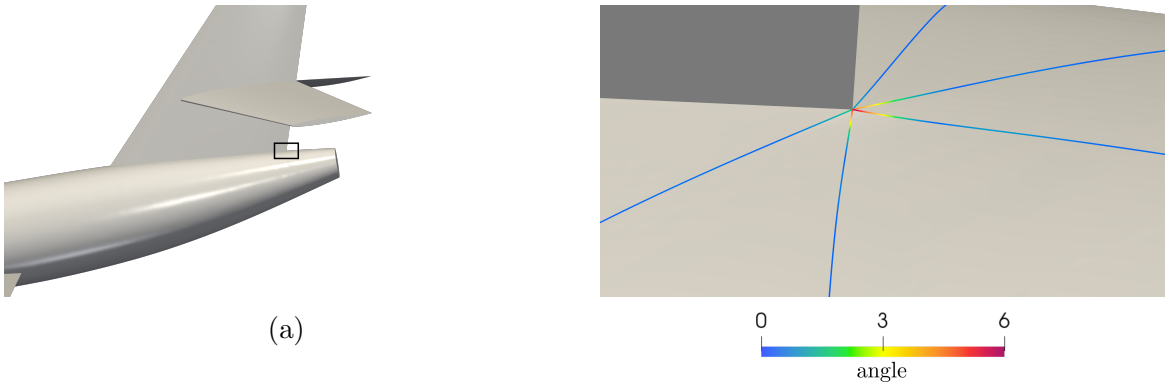

Figure 3.15: Mesh of polynomial degree $q = 5$ of a Falcon aircraft colored according to the distance to the limit model.

between the limit model and the high-order surface meshes, see Eq. (3.5). In this example, $L\,(\Omega^\infty)$ is set to the aircraft length. Note that as the polynomial degree $q$ increases, the distance to the limit model is reduced. In Fig. 3.14, we plot the lower and upper bound of the best approximation polynomial, see Eq. (3.8). The upper curve coincides with the logarithm of the distance which seems to converge linearly with the polynomial degree. In Fig. 3.15, we show the point-wise distance between the mesh of polynomial degree five and the limit model. Note that around the regular vertices, the distance between the limit model and the surface mesh of polynomial five is zero. Around the irregular vertices, the distance is smaller than $10^{-4}$.

The maximum angle between normal vectors along interior edges is reduced significantly as we increase the polynomial degree, see Table 3.5. The linear mesh features

| $q$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\max_e \alpha_{\infty,e}$ | 63.92 | 24.08 | 8.19 | 5.04 | 5.60 |

Table 3.5: Maximum angle between the normal vectors of adjacent elements for the surface mesh of the Falcon aircraft of polynomial degree $q$, $q = 1, \ldots, 5$.




(a) (b)

Figure 3.16: Close look of the rear point of the vertical stabilizer of the mesh of polynomial degree $q = 5$ of a Falcon aircraft. (a) Rear part of the aircraft. (b) Angle between the normal vectors along the interior edges incident to the sharp point.

an angle of 60 degrees along the leading edge of the rear wing. In contrast, the mesh of polynomial degree five attains a maximum angle smaller than 2 degrees in the same region. However, the interpolative property of the method affects the smoothness of the high-order mesh. For instance, consider the rear point where the vertical stabilizer joins the fuselage, see Fig. 3.16. All the high-order nodes of the triangles around the sharp point interpolate the $\mathcal{C}^1$-continuous limit surface determined by the surface subdivision scheme, except for the vertices that coincide with the sharp point, which remain fixed. In that region, we can only guarantee $\mathcal{C}^0$-continuity and the angle between the normal vectors is greater than 5 degrees.

A summary of the mesh quality can be found in Table 3.6. The linear mesh is composed of $3.1{\cdot}10^5$ nodes and $1.7{\cdot}10^6$ tetrahedra, and the volume mesh of polynomial degree $q = 5$ is generated in 6.5 hours and is composed of $3.6 \cdot 10^7$ nodes and $1.7 \cdot 10^6$ elements. This mesh, prior to the blending technique, contains 2179 tangled elements. In this example, there are $3.0{\cdot}10^5$ boundary elements and in 8 minutes the TFI reduces to 24 the number of invalid elements, that is, 99% of the invalid elements have been untangled. Now, we apply the optimization technique presented in Gargallo-Peiró

| | Min Q | # inv |
|---|---|---|
| Boundary | 0.55 | 0 |
| Volume (no TFI) | 0 | 2179 |
| Volume (TFI) | 0 | 24 |
| Volume (TFI + Optimization) | 0.69 | 0 |

Table 3.6: Quality statistics of a mesh of polynomial degree $q = 5$ for a Falcon aircraft.

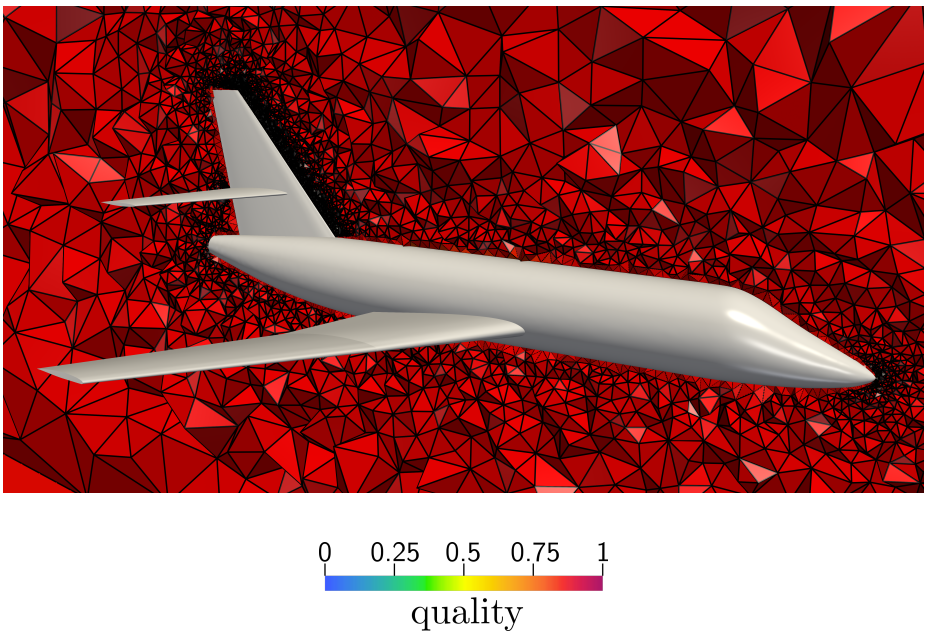
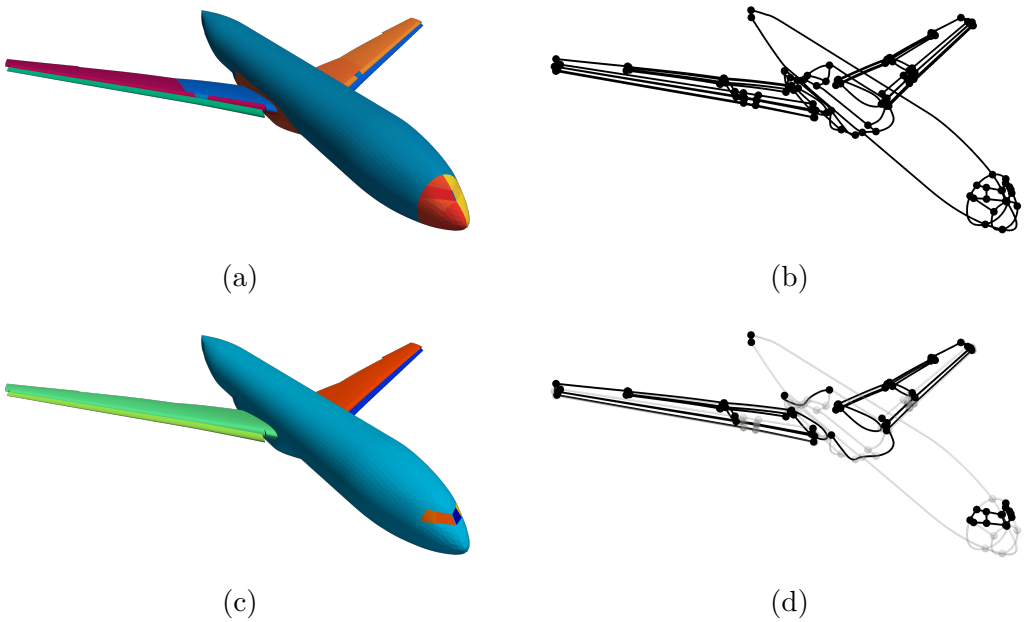



Figure 3.17: Curved tetrahedral mesh of polynomial degree $q = 5$ of a Falcon aircraft with no invalid elements.

et al. (2015c,b) to optimize locally the quality of the inverted elements. Since the mesh after the TFI is close to optimal, it is a good initial condition for the implicit optimization. This process takes 41 minutes and the mesh becomes valid achieving a minimum quality of 0.69. In Fig. 3.17, we show the valid curved tetrahedral mesh of polynomial degree $q = 5$ with the volume elements colored according to their quality.

### 3.8.5 Assisted sharp-to-smooth boundary modeling

As detailed in Sect. 3.6.4, it is possible to automatically suggest to the practitioners the features to smooth. In this example, we consider a mesh-based boundary



(a) (b)

(c) (d)

Figure 3.18: Initial and final linear mesh model of an aircraft in high-lift configuration. Initial model: (a) surface features colored with their surface identifier, and (b) curve and point features. Final model: (c) virtual surface features colored with their surface identifier, (d) curve and point features smoothed (gray) and preserved (black).

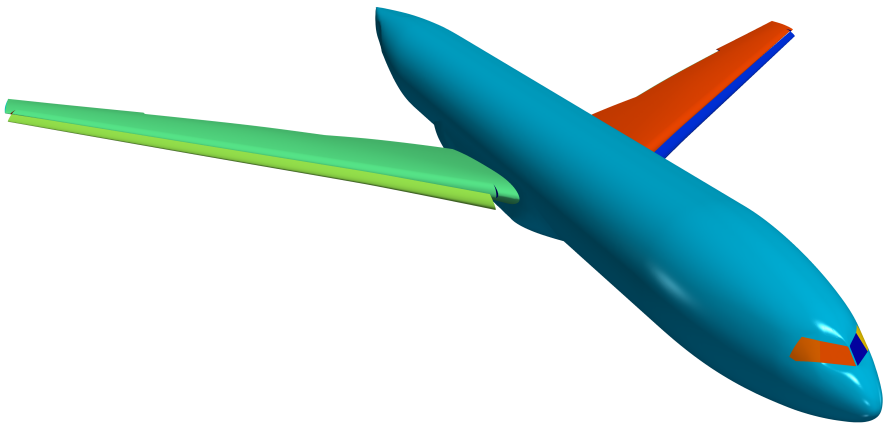


Figure 3.19: Curved triangular mesh of polynomial degree $q = 4$ of an aircraft in high-lift configuration colored with the surface identifiers.

representation of an aircraft in high-lift configuration, a common research model presented in the 3rd High-Lift Prediction Workshop (Rumsey et al., 2019). The model,

see Fig. 3.18(a) and Fig. 3.18(b), is represented by 182 feature points, 282 curves, and 118 surfaces. For models with a significant complexity, we use the automatic feature detection capability detailed in Sect. 3.6.4 to reduce the human labor required to design a model preserving the simulation intent.

First, using the original model, we generate a surface mesh of polynomial degree $q = 4$. Then, for each feature curve, we compute the normal vector along the curve and decide whether this curve has to be smoothed. The threshold used in this example is $\delta = 17$ degrees. We highlight that this is a parameter that can be easily tuned to determine the curves in the set $R_\delta$. Next, we manually decide if these candidates have to be indeed smoothed or not. Some of the curves present in this set are the ones representing the leading edge of the wings and flaps that indeed have to be smoothed. However, there are some curves in the set that we do not want to smooth. For example, we do not smooth the two curves that describe where the swept wing starts to angle, and the curves that represent the cabin windows. Thus, these curves are not smoothed. Second, once the model has been updated, we generate again the mesh to detect the smooth points. The final model is composed of 122 feature points, 169 curves, and 67 surfaces, see Fig. 3.18(c) and Fig. 3.18(d).

In Fig. 3.19, we show the generated mesh of polynomial degree four using the smoothed model where the surfaces have been colored with their surface identifier. Since the curves describing the leading edge have been automatically smoothed, the wing surface describes the simulation intent. On the contrary, we can appreciate the desired discontinuity on the normal vectors of the surface along the curves describing the cabin windows.

## 3.9 Discussion

After detailing the methods and results to interpolate the subdivision features for curved geometry modeling, we present a discussion on many aspects related to: $\mathcal{C}^1$-continuous shape functions, evaluating a limit curve around a feature point, the feature detection threshold, the TFI node relocation, polynomial degrees, and exploiting successive subdivisions.

**$\mathcal{C}^1$-continuous shape functions** In this work, we indirectly approximate the $\mathcal{C}^1$-continuity of the limit surface by using interpolative triangular elements. An alterna-

tive approach might be to directly impose $\mathcal{C}^1$-continuity using the Argyris et al. (1968) or the Bell (1969) triangular elements. Both elements have polynomial degree 5 and feature one degree of freedom per edge, and six degrees of freedom per vertex. The degrees of freedom of the vertices correspond to the value, the two first derivatives, and the three second derivatives required to determine the symmetric Hessian. Since the limit surface has $\mathcal{C}^1$-continuity everywhere, it is possible to query at the vertices the value and the first derivatives from the limit surface, and set the corresponding vertex degrees of freedom. However, since at the irregular points the limit surface might not be $\mathcal{C}^2$-continuous, the Hessian at the irregular points of the limit surface is not defined. Since it is undefined, to apply this approach, one would need to decide a criterion to force a Hessian at the irregular nodes. Herein, to not force the Hessian, we favor the numerical approximation of the $\mathcal{C}^1$-continuous limit surface.

**Limit curve around a sharp point** To evaluate the limit curve close to a sharp point, we can replace the successive refinement in Algorithm 3.2 with an explicit evaluation. To perform this evaluation, we can consider a stencil featuring three segments and four points. To this end, one straightforward approach is to add to the original control poly-line a ghost segment on the other side of each sharp point. Each ghost segment is limited by the original sharp point and an additional point having the same coordinates. Now, we can evaluate the standard limit curve expression on this new control poly-line since the original segment becomes internal. Other evaluation alternatives are detailed in De Boor (1978).

**Feature detection threshold** The automatic feature detector from Sect. 3.6.4 uses the same threshold $\delta$ for the whole mesh. Even though the current normal disparity is a helpful tool to set up a ranking of sharp features, it might be worth considering a variable weight accounting for the local mesh size. This improvement may be studied in the near future.

**TFI node relocation** In Sect. 3.7.2, we present a technique to accommodate the curvature of the surface mesh to the volume elements adjacent to the boundary. By discarding the TFI node relocation, we could simplify the mesh curving procedure by directly untangling and optimizing the volume mesh. However, the TFI node relocation pre-process helps, for isotropic meshes, to accelerate the whole curving process. This is so since the TFI relocation is explicit, fast, and provides a fair initial approx-

imation. This approximation reduces the number of non-linear iterations required by the implicit and slower optimization solver. Finally, note that in all the examples, the TFI relocation reduces the number of invalid elements.

**Polynomial degree** In this work, we present a method to interpolate the limit model with arbitrary polynomial degree. In practice, one might prefer polynomial degree four. This preference for degree four is so since the interpolation reproduces the $\mathcal{C}^2$-continuity of the limit model around the regular points. Furthermore, we have seen that around the irregular points with degree four, the interpolative approximation to the limit model has a sufficiently small error. If we need a smaller error, we can increase the polynomial degree of the limit model approximation. Nevertheless, this approximation cannot be guaranteed to be $\mathcal{C}^1$-continuous around irregular points. Thus, since the limit model is just a geometry surrogate, degree four could be adequate for practical purposes.

**Exploiting successive subdivisions** Renouncing to interpolate the limit model, we could simplify the interpolation of subdivision features by skipping the limit model parameterization (Stam, 1998). To this end, we could interpolate the limit model with a polynomial degree equal to a power of two on equispaced nodal distribution exploiting the structure of the subdivision scheme (Jiménez-Ramos et al., 2020). Then, we would interpolate this surrogate with an inferior polynomial degree to obtain an alternative approximation. The resulting surface mesh might not interpolate the limit model, but it might provide similar accuracy measurements.

## 3.10 Concluding remarks

We have proposed a method to interpolate a subdivision model with arbitrary degree and nodal distribution. The method evaluates one time the limit model and thus, skips successive refinement on posterior geometry queries. The results show that we can generate, from an initial straight-edged mesh, a piecewise polynomial mesh that approximates smooth curves and surfaces while preserving the required sharp features. We also show that the resulting curved high-order surface mesh is ready to prescribe the boundary in mesh curving methods. We have to highlight that for

regular meshes, as it is the case for digitized topographies, the resulting curved surface mesh has class $\mathcal{C}^2$.

We have numerical evidence that the proposed distance from the nodal interpolation to the limit model converges geometrically with the polynomial degree for nodal distributions with sub-optimal Lebesgue constant. Hence, we can expect accurate approximations of the limit model for sufficiently high polynomial degrees. We understand that this geometrical convergence rate, to a parameterization with up to continuous derivatives, seems to be originated by the natural alignment of the triangulation entities with the subdivision continuity interfaces. That is, irregular points of the initial triangulation are aligned with $\mathcal{C}^1$ irregular points of the limit model. Analogously, the interior of segments and triangles of the initial triangulation are aligned with the interior of the $\mathcal{C}^2$ segments and the $\mathcal{C}^\infty$ triangles of the limit model, respectively.

The results show that the proposed assisted sharp-to-smooth modeling capability reduces the human labor required to prescribe the simulation intent. Specifically, it facilitates assigning sharp and smooth features to a linear model composed of points, polylines, and triangulations. To illustrate it, we have assigned the desired flow simulation intent to an aircraft model in a high-lift configuration.

In conclusion, we have presented a methodology to model and represent curved geometry of practical interest for flow simulation with unstructured high-order methods. In perspective, high-order methods might benefit from using curved meshes that approximate our curved boundary representation, which we devised to describe the flow simulation intent.

# Chapter 4

# Refining simplex points for scalable estimation of the Lebesgue constant

## 4.1 Introduction

In approximation theory, one of the key problems is obtaining a set of simplex points for a given polynomial degree that guarantees a small interpolation error. To solve this problem, it is standard to obtain interpolation points featuring a small *Lebesgue constant*. This constant is usually denoted by $\Lambda$ and defined as

$$\Lambda = \max_{\boldsymbol{x} \in K^d} \sum_{i=1}^{N_{p,d}} \left| \phi_i \left( \boldsymbol{x} \right) \right| . \tag{4.1}$$

It corresponds to the maximum on the $d$-dimensional simplex $K^d$ of the summation of the absolute values of the Lagrange polynomials $\phi_i$ associated with each of the $N_{p,d}$ interpolation points, a summation that is Lipschitz because the absolute value terms are Lipschitz, a summation that is called the *Lebesgue function*. The maximum of this function appears in the upper bound of the interpolation error. Specifically, given a point distribution, for any function $f$, the error of the polynomial interpolation satisfies

$$\left\| f - I \left( f \right) \right\| \leq \left( 1 + \Lambda \right) \left\| f - f^{\star} \right\| ,$$

where $I(f)$ denotes the Lagrange interpolator, and $f^\star$ the best polynomial approximation. According to the previous inequality, the smaller the Lebesgue constant, the smaller the bound of the interpolation error. Moreover, the Lebesgue constant exclusively depends on the position of the interpolation points. For instance, equispaced distributions of points, which lead to larger errors for higher polynomial degrees, feature large values of the Lebesgue constant, yet non-uniform distributions of points, with improved interpolation error, feature sub-optimal values of the Lebesgue constant (Angelos et al., 1989; Roth, 2005; Warburton, 2006). Accordingly, to guarantee small interpolation errors, the Lebesgue constant has to be evaluated and thus, it is key to estimate on the simplex the maximum of the Lebesgue function.

To approximate this maximum, it is critical to automatically generate on the $d$-dimensional simplex a finite number of sample points – exactly the goal of this work. Then, for those points, the approximation is the maximum of the function evaluations. Note that these sample points are used to estimate the Lebesgue constant, but they do not correspond to the interpolation points that define the Lagrange polynomials in Eq. (4.1). To estimate the Lebesgue constant, there are general zeroth-order optimization (Paulavičius and Žilinskas, 2014) and Lebesgue-specific (Roth, 2005; Warburton, 2006; Briani et al., 2012) methods. These approaches are mainly different because the latter family exploits the structure of the Lebesgue function. Specifically, because the Lebesgue function presents several similar local maxima, specific-purpose methods successfully favor smooth gradations of the point resolution.

Nevertheless, both families of methods share some aspects. They feature the same stopping criterion, add sample points, and have computational costs scaling with the number of points. To stop the maximum approximation process, all the previous methods terminate after a fixed number of iterations. The two families of methods generate sample points statically or dynamically, statically by adding in one shot a grid of points (Briani et al., 2012), dynamically by adding at each iteration new points (Roth, 2005; Warburton, 2006; Paulavičius and Žilinskas, 2014). Because the Lebesgue function is evaluated at these sample points, the computational cost of the maximum estimation scales with the number of points. This scaling depends on the method, the polynomial degree, and the simplex dimension. Next, we see how to automatically stop the optimization iterations and the need for neighbor queries and scalable point refinement.

Unfortunately, to automatically stop the optimization iterations, neither the gen-

eral nor the specific-purpose methods exploit that the Lebesgue function is Lipschitz. Although all the previous methods can improve the estimation of the maximum by increasing the number of iterations, none of them automatically stops when the optimal of the Lebesgue function is sufficiently converged. To measure the convergence and stop the iterations, first- and second-order optimization methods check if the approximated candidate is sufficiently flat, a successful condition that can be emulated if the function is Lipschitz — precisely the case for the Lebesgue function.

To emulate the sufficiently flat condition on a sample point, it is key to query for point neighbors. Thus, using all these points to evaluate the function, a local estimation of the Lipschitz constant is the quotient of the function difference and the distance of the neighbor points. Then, using this constant and a flatness tolerance, the automatic stopping criterion can be incorporated by only evaluating the function. The specific-purpose (Roth, 2005; Warburton, 2006) and the general (Paulavičius and Žilinskas, 2014) methods incorporate neither the stopping criterion nor the neighbor queries.

To efficiently estimate the Lebesgue constant in the simplex, it is critical to use scalable point refinement techniques. In this manner, the Lebesgue function can be finely sampled only on the interest regions and coarsely sampled otherwise, an adaptive strategy that reduces the number of needed points (Roth, 2005; Warburton, 2006; Paulavičius and Žilinskas, 2014). As we said before, the computational cost scales with the number of points, so the local refinement reduces the cost of approximating the Lebesgue constant. Unfortunately, when specific-purpose methods refine the resolution (Roth, 2005; Warburton, 2006; Briani et al., 2012), the number of points scales exponentially with the dimensionality. This exponential scaling is affordable for two and three dimensions, but impractical for higher dimensions. Fortunately, some general methods do not scale exponentially with the dimension. Specifically, the DiSimpl (Paulavičius and Žilinskas, 2014) method adds only two new points per point refinement, a refinement scaling that is well-suited for higher dimensions.

Summarizing, for more than three dimensions, there is no specific-purpose method to estimate the Lebesgue constant in the simplex. For two and three dimensions, the specific-purpose methods successfully estimate the Lebesgue constant, but their extensions to arbitrary dimensions do not scale well with the dimensionality. For arbitrary dimensions, optimization methods for general functions scale well with the dimensionality, but they are not specifically devised to estimate the Lebesgue con-

stant. That is, they do not control size gradation. Neither general nor specific-purpose methods feature neighbor queries. Thus, they are not ready to stop automatically when the optimal candidate features sufficient flatness.

To address the previous issues, the main contribution of this chapter is to propose a new specific-purpose point refinement method. The proposed method features a smooth gradation of the resolution, neighbor queries based on neighbor-aware point coordinates, and a point refinement that scales algebraically with the dimension as $(d+1)d$. The main novelty of the proposed smooth point refinement method is not only that it scales algebraically with the dimension but also that it is ready to use an automatic Lipschitz stopping criterion.

The main application of the proposed point refinement method is to estimate the Lebesgue constant on the simplex. Accordingly, the results check whether the proposed point refinement method reproduces the literature estimations for the triangle and the tetrahedron. Moreover, the results assess whether the method is well-suited for Lebesgue constant approximations on the simplex for mid-range dimensionality.

The rest of the chapter is organized as follows. First, in Sect. 4.2, we review the related literature. Then, in Sect. 4.3, we describe the system of coordinates that allow the neighbor queries and the core point refinement operations. Next, in Sect. 4.4, we detail the adaptive minimization method. In Sect. 4.5, we illustrate with several examples the main features of the presented method. Lastly, in Sect. 4.6, we present some concluding remarks of this chapter.

## 4.2 Related work

One of the most immediate estimates of the Lebesgue constant is given by the maximum value of the function in an equispaced grid of points. The size of the sampling determines the accuracy of the estimation at the expense of increasing the number of function evaluations. Alternatively, it is possible to use a sequence of admissible meshes (Calvi and Levenberg, 2008) as sampling points (Briani et al., 2012). Admissible meshes have the property that the maximum value at this finite set of points bounds the infinity norm of a polynomial of certain degree overall the simplex. Unfortunately, none of these methods is adaptive.

An alternative is to estimate the Lebesgue constant by means of a non-deterministic adaptive method (Warburton, 2006). The method starts with a random sample of

points in the simplex. Next, the function is evaluated, and the points are sorted in terms of their function value. Then, new random samples are generated inside boxes centered at the points with the largest values. At each iteration, the box edge-length is halved to capture the maximum more accurately, and thus, a smooth gradation in the sampling resolution is obtained. This process is repeated until a prescribed number of iterations is reached. Finally, the estimate for the Lebesgue constant is the largest value at a sample point.

To compute the Lebesgue constant, an alternative adaptive method (Roth, 2005) named DiTri modifies the DiRect algorithm (Jones et al., 1993) to work in triangles. The method starts with the evaluation of the function at the centroid of the triangle. Next, the triangle is subdivided using a quadtree strategy. Then, the function is evaluated at the centroid of the three new smaller triangles. At each iteration, the algorithm chooses a set of potentially optimal triangles to refine in terms of their size and the function value at their centroid. After the refinement step, additional elements are refined to ensure a smooth gradation of the element size. When a prescribed number of iterations is reached, the centroid of the triangle with the largest function value determines an estimation of the Lebesgue constant. Remarkably, the method exploits the simplicity of the triangle by uniquely identifying each element with a triplet of integers, and therefore, no explicit mesh connectivity structure is needed.

Similarly to the DiRect algorithm but based on simplices, the method named DiSimpl also considers a Lipschitzian optimization approach (Paulavičius and Žilinskas, 2014). DiSimpl is devised to find the global minimum of an arbitrary function whose domain is a hypercube or simplex, and performs particularly well when the function presents symmetries. Initially, the search space is decomposed into simplices. Then, two approaches are considered. In one case, the function is evaluated at the centroid of the simplex, and two hyper-planes cutting the longest edge subdivide the potentially optimal simplex into three smaller simplices. In the other case, the function is evaluated at the vertices of the simplex, and one hyper-plane cutting the longest edge generates two smaller simplices. Interestingly, in any of the approaches, only two function evaluations per refinement are performed. Finally, the algorithm stops when a prescribed number of iterations is reached.

These adaptive methods outperform grid-based methods but they are not devised to estimate the Lebesgue constant in the $d$-dimensional simplex. The non-

deterministic method (Warburton, 2006) starts sampling the function in 10 000 points and generates 10 samples per each 2D box. Thus, to keep the same resolution we would need $10^{d/2}$ points inside the $d$-box. Even in 2D, a considerable amount of approximately 200 000 sample points are required to accurately estimate the Lebesgue constant, and we expect a higher value for higher dimensions. Moreover, the non-deterministic nature of the algorithm makes it difficult to query neighbor points. The quad-tree subdivision-based method (Roth, 2005) is devised to estimate the Lebesgue constant in the triangle. The natural extension of this method to higher dimensions would subdivide the $d$-simplex into $2^d$ subelements and, consequently, the number of sample points would increase exponentially. Furthermore, the simplicity of the triangle case to uniquely identify each element could not be exploited. Finally, the DiSimpl algorithm (Paulavičius and Žilinskas, 2014) has not been tested against the Lebesgue function and the subdivision strategy becomes complicated in high dimensions. Moreover, the resulting mesh is not conformal and the method does not feature access to neighbor elements.

The method proposed in this chapter is deterministic and exploits a rational barycentric system of coordinates to uniquely identify each sample point. The method considers a discrete set of directions to refine which are parallel to the simplex edges. Every time a point is refined, we evaluate the function at most $(d+1)\,d$ times. Then, after refining the potentially optimal points, we generate new sample points to ensure a smooth gradation of the sampling resolution. Moreover, the system of coordinates allows accessing to the adjacent points with no need for storing the neighbor structure, which enables a stopping criterion based on the sampling density and the local Lipschitz constant around the extremum.

## 4.3 Neighbor-aware coordinates for point refinement

Even though the main application is computing the maximum of the Lebesgue function, we present the method in a minimization framework. In Sect. 4.3.1, we schematically illustrate the method in 2D. Then, we detail the system of coordinates in Sect. 4.3.2, and the core refinement operations in Sect. 4.3.3 and Sect. 4.3.4.

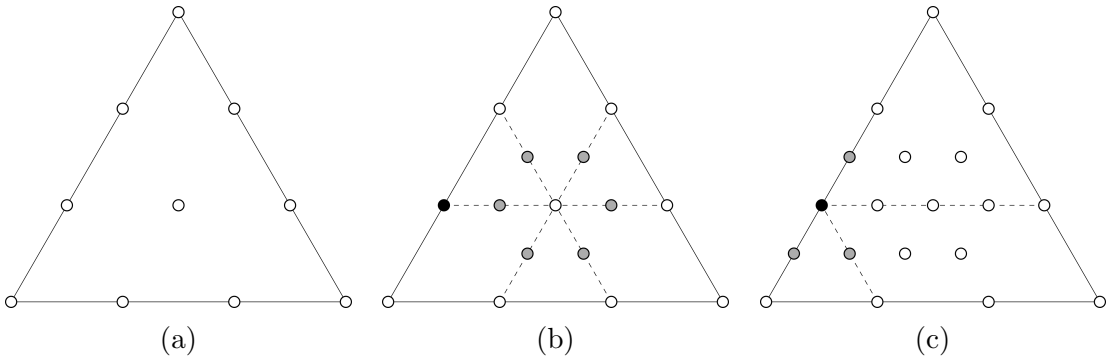



Figure 4.1: Illustration of the method. (a) Initial sampling. (b) Refining the point at the barycenter by generating six new points (in gray) around it parallel to the triangle edges. (c) To refine the black point we only evaluate the target function at three new points (in gray).

### 4.3.1 Outline

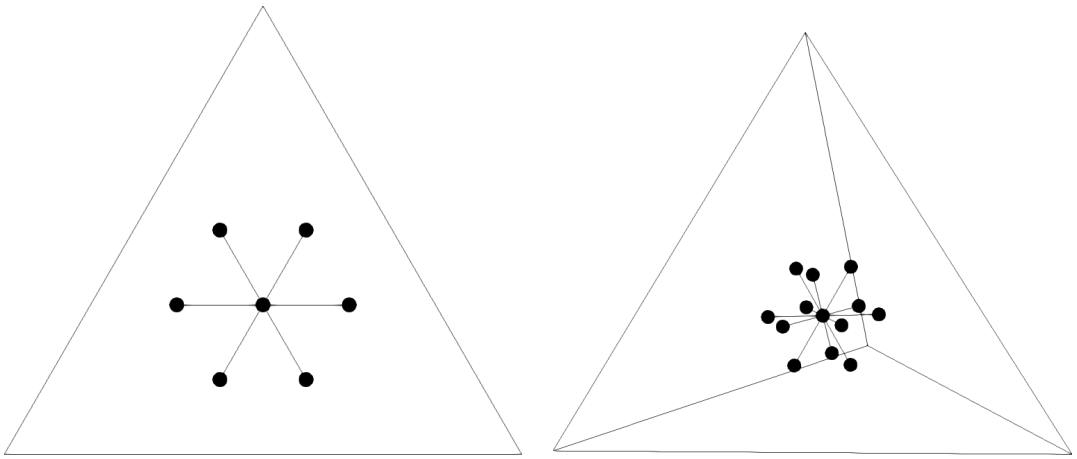
To estimate the minimum of the target function, we propose using neighbor-aware sample points. Consider the set of sample points shown in Fig. 4.1(a) and assume that the point at the barycenter of the triangle is our minimum candidate. To improve the estimation of the minimum, we refine the sampling around it by generating new sample points parallel to the simplicial edges, see Fig. 4.1(b). Analogously, if our next minimum candidate is the black point in Fig. 4.1(b), we generate new sample points at positions parallel to the triangle edges, see Fig. 4.1(c). However, we only generate three new points (in gray) since one of them already exists. Applying successively this refinement operation to potentially optimal points, we expect to finely sample the target function and find an accurate estimate of the minimum.

### 4.3.2 Neighbor-aware coordinates

Since we work with simplicial domains, we exploit the barycentric coordinates system. More precisely, we consider an equispaced sampling with $q+1$ points on each edge of the simplex, and we uniquely determine a sample point $\boldsymbol{x} \in K^d$ by a set of rational barycentric coordinates of the form

$$\boldsymbol{x} = \left( \frac{\lambda^1}{2^r q}, \ldots, \frac{\lambda^{d+1}}{2^r q} \right), \tag{4.2}$$

with $\sum_{i=1}^{d+1} \frac{\lambda^i}{2^r q} = 1$, and non-negative integers $r$ and $\lambda^i$, $i = 1, \ldots, d+1$. For each barycentric coordinate, the numerator indicates the position on a uniform grid and



(a) (b)

Figure 4.2: For a central point (black dot), surrounding stencil points (black dots) for the refinement directions (gray segments) parallel to the simplex edges (gray edges) in (a) 2D and (b) 3D.

the denominator represents the level of refinement on the grid. Thus, the higher the denominator, the higher the resolution of the sampling around the point.

Aligned with this system of coordinates, for each simplicial edge we choose a refinement direction. Each refinement direction has two possible orientations: forward and backward. Thus, we consider $n_D = 2n_E$ vectors, where $n_E$ is the number of edges of $K^d$. Each vector is identified by a pair of integers $(i, j)$ and is written in rational barycentric coordinates as

$$\boldsymbol{u}_{(i,j)} = \frac{1}{q} \left( \boldsymbol{e}_i - \boldsymbol{e}_j \right),$$

with $i, j = 1, \ldots, d+1$, $i \neq j$, and where the $(d+1)$-dimensional vector $\boldsymbol{e}_k$ is a vector with a one in the $k$-th position and zeros elsewhere. The set of direction vectors

$$\mathcal{U} = \{\boldsymbol{u}_{(i,j)} \text{ for } i, j = 1, \ldots, d+1, i \neq j\}$$

defines a canonical stencil that is used for generating new sample points. More concretely, we generate at most $(d+1)\,d$ new sample points per point refinement. These $(d+1)\,d$ refinement positions provide a reasonable scaling for medium dimensionality while sampling sufficiently fine the neighborhood of the point to refine. In particular, for the equilateral triangle, the six refinement positions are top-left, top-right, left, right, bottom-left, and bottom-right, see Fig. 4.2(a), while for the tetrahedron there are twelve positions, see Fig. 4.2(b).

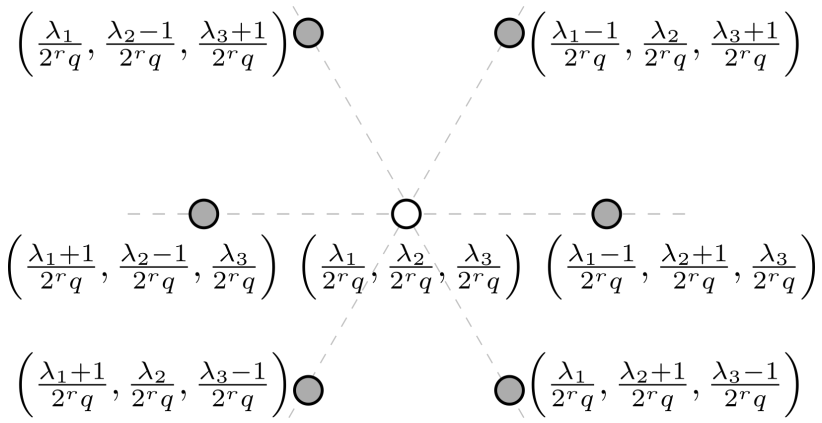



Figure 4.3: Coordinates of a point of resolution $r$ and its neighbors.

The value $r$ in Eq. (4.2) is strongly related to the resolution of the sampling. Without loss of generality, consider the sample point $\boldsymbol{x} = \left(\frac{1}{2}, \frac{1}{2}\right)$ in the one-dimensional interval $[0, 1]$, and the direction vector $\boldsymbol{u}_{(1,2)} = \left(\frac{1}{2}, \frac{-1}{2}\right)$. The point

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}_{(1,2)} = \left(\frac{2}{2}, \frac{0}{2}\right)$$

corresponds to the point zero in cartesian coordinates, while the point $\boldsymbol{z} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{u}_{(1,2)} = \left(\frac{3}{4}, \frac{1}{4}\right)$ is between the points $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus, scaling the vector $\boldsymbol{u}_{(1,2)}$ with a factor of the form $2^{-r}$, $r \geq 0$, leads to the generation of closer points along the direction described by the vector $\boldsymbol{u}_{(1,2)}$.

The value of $q$ determines the density of the initial sampling, see Eq. (4.2). In practice, we favor setting $q$ equal to one. When $q$ is one, the initial grid contains only the simplex vertices as sampling points. That is, the initial number of sampling points is $d + 1$, and thus, it scales linearly with the number of dimensions. Bigger values determine denser initial samplings that might require fewer adaptation iterations, but the initial number of sampling points scales exponentially with the number of dimensions. This initial offset might be reflected in a larger number of final sampling points required to seek the target function minimum.

We store the points in a hash table built from the rational coordinates. Hence, the point $\left(\frac{\lambda^1}{2^r q}, \ldots, \frac{\lambda^{d+1}}{2^r q}\right)$ and the point $\left(\frac{2^k \lambda^1}{2^{r+k} q}, \ldots, \frac{2^k \lambda^{d+1}}{2^{r+k} q}\right)$ are identified as the same point. Moreover, this system of rational barycentric coordinates allows to easily access the neighbor points with no need for storing the neighbor structure. As depicted in Fig. 4.3, coordinates of neighbor points with the same denominator only differ in one unit. Thus, to query if a neighbor exists, we simply add one to one component,

subtract one from another component, and search in the set of points. Therefore, there is no need for storing explicit connectivity information.

### 4.3.3 Point refinement

Besides the resolution, it is also useful to classify the points in terms of the completeness of the stencil. On the one hand, we say a point $\boldsymbol{x}$ is *incomplete of resolution* $r$ if the sample point $\boldsymbol{x} + 2^{-r}\boldsymbol{u}_{(i,j)}$ exists, for some $\boldsymbol{u}_{(i,j)} \in \mathcal{U}$. Alternatively, this means that at least one point of the stencil of resolution $r$ centered at $\boldsymbol{x}$ exists. A sample point $\boldsymbol{x}$ may be incomplete in several resolutions, but it is for the highest resolution when the representation of the function around $\boldsymbol{x}$ is more accurate.

On the other hand, if all the points of the stencil exist, we say this point is complete. More precisely, a point $\boldsymbol{x}$ is *complete of resolution* $r$ if the sample point $\boldsymbol{x} + 2^{-r}\boldsymbol{u}_{(i,j)}$ exists, for all $\boldsymbol{u}_{(i,j)} \in \mathcal{U}$. Similarly to the incomplete case, the higher the resolution, the more accurate the representation of the function around the point.

We remark that a complete point of resolution $r$ provides a finer discretization than an incomplete point of resolution $r'$ with $r' \leq r$, since the neighborhood is denser and sampled along all the considered directions. Furthermore, a point may be complete and incomplete at the same time both providing meaningful information. For instance, consider a complete point of resolution $r$ which is also incomplete of resolution $r+1$. In this case, not only the neighborhood is fully sampled at resolution $r$, but additional partial information of the function at resolution $r+1$ is known.

Around an incomplete point, this partial information has to be enhanced to obtain a more accurate representation of the target function. Accordingly, we consider an operation that completes the stencil around an incomplete point. Specifically, to complete an incomplete point of resolution $r$, we propose to generate all the missing points of the stencil of resolution $r$. Thus, the resulting point is no longer incomplete at level $r$. In Fig. 4.4, for the two-dimensional case, we illustrate the completion step for an incomplete point of resolution $r$. Since three points of the stencil exist, Fig. 4.4(a), we only generate the remaining missing points to complete the stencil. Once completed, the point becomes complete of resolution $r$, Fig. 4.4(b).

When the information gathered from a complete point indicates that there is a minimum nearby, we should sample the function in a smaller neighborhood to capture it. Thus, we need a point refinement operation. Refining a complete point of resolution $r$ consists in generating all the points of the stencil of resolution $r+1$.

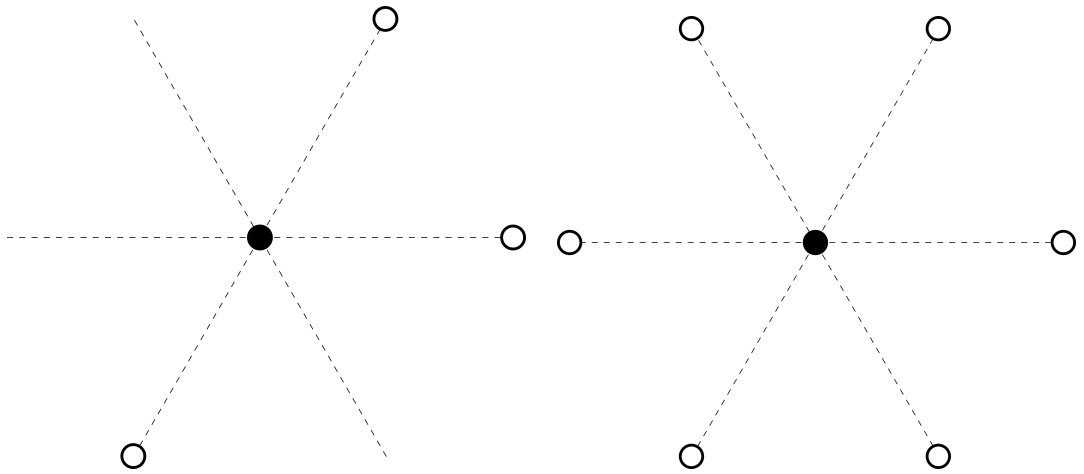



Figure 4.4: Completing an incomplete sample point. An (a) incomplete point of resolution $r$ becomes (b) complete of resolution $r$.

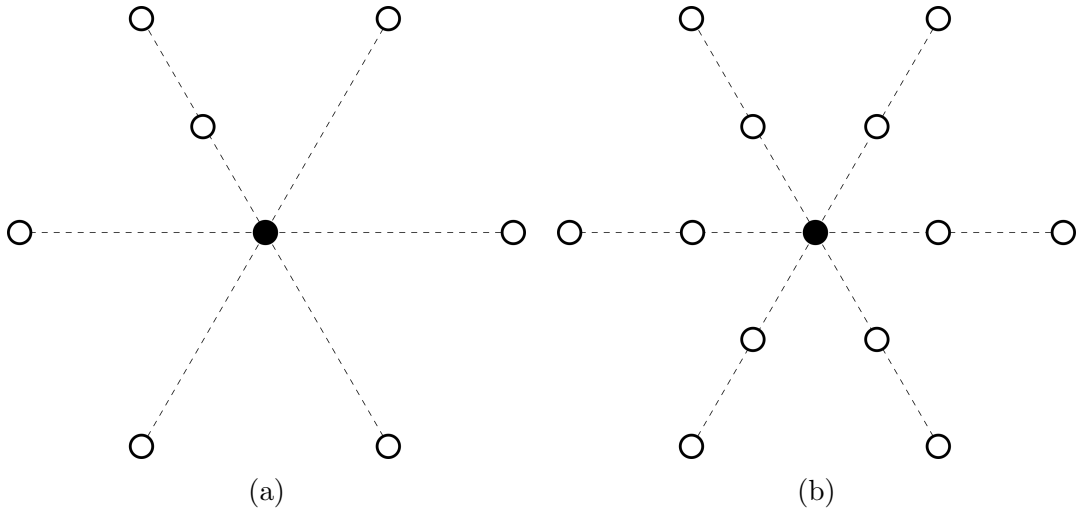



Figure 4.5: Refining a complete sample point. A (a) complete point of resolution $r$ becomes (b) complete of resolution $r + 1$.

Thus, the point becomes complete of resolution $r + 1$. We highlight that if the point is incomplete of level $r + 1$, we only generate those points needed to complete the stencil of resolution $r + 1$ and, therefore, we avoid repeated function evaluation. In Fig. 4.5, we illustrate, for the two-dimensional case, the refinement of a complete point of resolution $r$ which is also incomplete of resolution $r + 1$. Since one point of the stencil of resolution $r+1$ exists, see Fig. 4.5(a), we generate five points to complete the stencil. Then, the point becomes complete of resolution $r + 1$, see Fig. 4.5(b).

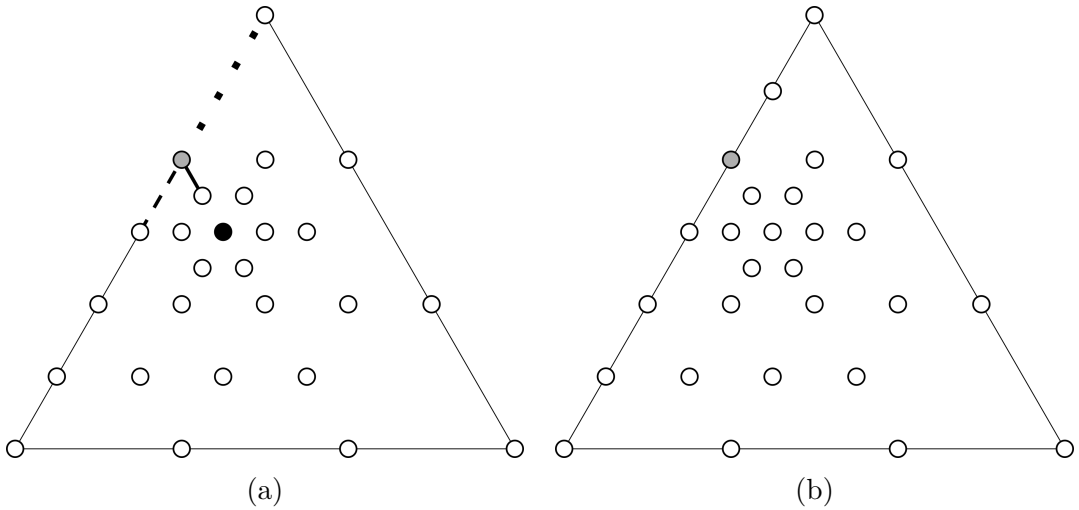



Figure 4.6: Smooth gradations of the resolutions. (a) The gray point is complete of resolution $r$ (dotted line) and incomplete of resolutions $r+1$ (dashed line) and $r+2$ (solid line). (b) Smooth sampling after refining the gray point until it becomes complete of resolution $r+1$.

### 4.3.4 Smooth gradation

To obtain smooth discretizations of the target function, we need smooth gradations of the resolution of the sampling points. Accordingly, we only consider sampling configurations where the resolution between neighbors differs at most by one unit. More precisely, assume that the finest complete resolution of a point is $r$, and the highest incomplete resolution is $r'$, $r' > r$. Then, the sampling is smooth if $r' = r+1$. Thus, after completing or refining a point we check if we have a smooth gradation of points. If there is a point such that $r' > r+1$, we smooth it by refining until resolution $r'-1$.

In Fig. 4.6(a), we show the sampling after refining the black point. We observe that the gray point is complete of resolution $r$ (dotted stencil), but it is also incomplete of resolutions $r+1$ (dashed stencil) and $r+2$ (solid stencil), and therefore, this sampling configuration is non-smooth. To obtain a smooth discretization, we refine the gray point until it becomes complete of resolution $r+1$ by generating one new point, see Fig. 4.6(b). Now, there is a smooth gradation of the point resolution.

**Algorithm 4.1** Approximating the minimum by sampling.

**Input:** Function $F$, Domain $K^d$
**Output:** Minimum $\boldsymbol{x}^\star$, $F(\boldsymbol{x}^\star)$

```
1: function ComputeMinimum(F, K^d)
2:   Σ ← InitializeSamplePoints(F, K^d)
3:   x* ← GetMinimum(Σ)
4:   while x* is not a minimum of F do
5:     Σ_C ← GetCompletePoints(Σ)
6:     {x_{C_i}} ← GetPointsToRefine(Σ_C)
7:     RefinePoints(F, Σ, {x_{C_i}})
8:     Σ_I ← GetIncompletePoints(Σ)
9:     {x_{I_j}} ← GetPointsToComplete(Σ_I)
10:    CompletePoints(F, Σ, {x_{I_j}})
11:    SmoothSampling(F, Σ)
12:    x* ← GetMinimum(Σ)
13:  end while
14:  return x*, F(x*)
15: end function
```

## 4.4 Adaptive point refinement

In this section, we first present our adaptive method to estimate the minimum of a function defined in the $d$-dimensional simplex, see Sect. 4.4.1. The rational barycentric coordinate system described in Sect. 4.3 is the core of our method since an explicit point connectivity structure is not needed. Then, in Sect. 4.4.2, we detail the stopping criterion.

### 4.4.1 Algorithm

The adaptive point refinement is detailed in Algorithm 4.1. Given the function to minimize, $F$, and the simplicial domain where it is defined, $K^d$, the first step of the method is to initialize the set of sample points denoted as $\Sigma$, Line 2. We remark that function $F$ is an arbitrary target function, yet in our main application it corresponds to minus the Lebesgue function. In the second step, the method gets the first minimum approximation on the initial sample points, Line 3. This initialization allows iterating to seek a better approximation of the minimum until convergence, Line 4. To improve the minimum approximation, the iterative process successively refines and completes the potentially optimal sample points and smooths the gradation of

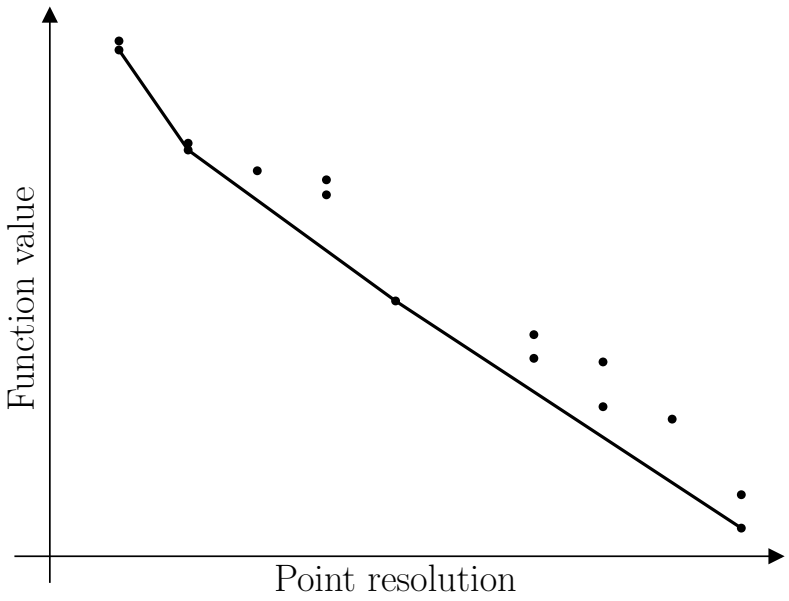



Figure 4.7: Two-dimensional representation of the complete points in terms of the resolution and function value. The lower boundary of the convex hull determines the points to complete.

the sampling point resolution.

First, the method refines the candidate points. To this end, in Line 5, we retrieve the set of complete points $\Sigma_C$ and choose the points to refine, Line 6. We determine the points to refine in terms of their resolution and function value. More concretely, each complete sample point is represented in a graph by a dot with the horizontal coordinate given by its resolution, and the vertical coordinate given by its function value. If a point $\boldsymbol{x}$ is complete of resolutions $r_1, \ldots, r_k$, with $r_1 < r_2 < \cdots < r_k$, then it is represented by a dot at position $(r_k, F(\boldsymbol{x}))$. In Fig. 4.7, we show this graph in an intermediate stage of the algorithm. Similarly to the DiRect algorithm (Jones et al., 1993), we choose the points to refine by exploring multiple Lipschitz constants which, in practice, reduces to computing the lower boundary of the convex hull of this point cloud. Then, in Line 7, we refine the chosen points $\{\boldsymbol{x}_{C_i}\}$.

Second, the method completes the incomplete points. To this end, in Line 8, we obtain the set of incomplete points $\Sigma_I$ and choose the points to complete, Line 9. Let $\boldsymbol{x} \in \Sigma_I$ be an incomplete point of resolutions $r_1, \ldots, r_k$, with $r_1 < r_2 < \cdots < r_k$, which is either not complete or complete with finest resolution $r$, $r < r_1$. Incomplete points provide information about the function in a local sense since they have been

sampled along a particular direction only. In contrast, complete points have been sampled along all the directions and, therefore, global information is known. Since we prefer to have first a big picture of the function landscape before focusing on the higher-resolution detail, we represent the incomplete point $\boldsymbol{x}$ by a dot with coordinates $(r_1, F(\boldsymbol{x}))$ instead of $(r_k, F(\boldsymbol{x}))$. Then, we obtain a point cloud similar to the one shown in Fig. 4.7. The lower part of the convex hull of this representation of $\Sigma_I$ determines the points $\{\boldsymbol{x}_{I_j}\}$ to be completed. Finally, in Line 10, we complete these points.

Third, the method smooths the gradation of the sampling point resolution. Specifically, in Line 11, we generate the points needed to ensure the sampling is smooth, see Sect. 4.3.4, and retrieve the minimum point $\boldsymbol{x}^\star$ from the sampling $\Sigma$, Line 12. These steps are repeated until the point $\boldsymbol{x}^\star$ is a minimum, see Line 4. The details on the stopping criterion are to be described in Sect. 4.4.2. Finally, the algorithm returns the minimum point and the function value at the minimum, Line 14.

We highlight that the function is evaluated only in the generation of new sample points, that is, in the refinement, completion, and smoothing steps. Further, in the point data structure, we store the point coordinates and the function value, so it can be immediately obtained when needed avoiding repeated calculations. To easily access the neighbor points, the point data structure also contains an updated list of the complete and incomplete resolutions.

### 4.4.2 Stopping criterion

In zeroth-order minimization, it is standard to stop seeking a minimum when a fixed number of iterations is reached or when the minimum approximation is numerically close to a known minimum value. In our case, only the value of the function is known, yet the sample structure allows obtaining an indicator of the flatness of the function. Accordingly, we can consider a stopping criterion accounting for the function flatness as in first- and second-order optimization methods. The user specifies spatial and functional tolerances, and the method automatically stops when a minimum below these thresholds is found.

The spatial tolerance controls the resolution of the sampling in the neighborhood of the minimum sample point. Specifically, given a spatial tolerance $\delta$, there exists a resolution $R$ such that the distance between a point $\boldsymbol{x}$ and the point $\boldsymbol{x} + 2^{-r}\boldsymbol{u}_{(i,j)}$ is smaller than $\delta$ for all $r \geq R$ and vector $\boldsymbol{u}_{(i,j)} \in \mathcal{U}$. Note that a complete point of

resolution $r$, $r \geq R$, satisfies this criterion.

The functional tolerance $\varepsilon$ is used to assess the flatness of the function around a point in terms of an estimate of the local Lipschitz constant. Specifically, consider a complete point $\boldsymbol{x}$ of resolution $r$, and denote by $\boldsymbol{y}$ the neighbor along the direction $(i,j)$, $\boldsymbol{y} = \boldsymbol{x} + 2^{-r}\boldsymbol{u}_{(i,j)}$. We estimate the Lipschitz constant of resolution $r$ around $\boldsymbol{x}$ along the direction $(i,j)$ as

$$\tilde{K}_{(i,j)}(\boldsymbol{x}) = \frac{F(\boldsymbol{x}) - F(\boldsymbol{y})}{d(\boldsymbol{x},\boldsymbol{y})},$$

where $d(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|_2$ is the distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that we allow negative Lipschitz constant estimations. In particular, $\tilde{K}_{(i,j)}(\boldsymbol{x})$ is negative if and only if $F(\boldsymbol{x}) < F(\boldsymbol{y})$. Moreover, the magnitude of the Lipschitz constant is strongly related to the flatness of the function around $\boldsymbol{x}$. Thus, the point $\boldsymbol{x}$ is a minimum candidate if $\tilde{K}_{(i,j)}(\boldsymbol{x})$ is negative and

$$\left|\tilde{K}_{(i,j)}(\boldsymbol{x})\right| < \varepsilon,$$

for all the directions $\boldsymbol{u}_{(i,j)} \in \mathcal{U}$.

In Algorithm 4.1, at the end of the loop, there exists a sample point $\boldsymbol{x}^\star$ such that $F(\boldsymbol{x}^\star) \leq F(\boldsymbol{y})$, for all $\boldsymbol{y} \in \Sigma$. Then, in Line 4, we check if the point $\boldsymbol{x}^\star$ is complete of resolution $r$, $r \geq R$, and the local estimates of the Lipschitz constant along all the possible directions for resolution $r$ are less than $\varepsilon$. If so, we assume that the neighborhood of $\boldsymbol{x}^\star$ has been sufficiently sampled and the function is sufficiently flat there. Thus, the point $\boldsymbol{x}^\star$ is considered an estimate of the function minimum and the algorithm stops.

Alternatively, it is also possible to limit the number of iterations. This limit allows the user to obtain an approximation of the minimum before the tolerance-based stopping criterion is satisfied. In both cases, the algorithm returns the sample point $\boldsymbol{x}^\star$ with the smallest function value.

## 4.5 Results: estimation of the Lebesgue constant

The main application of the method presented in Sect. 4.4 is the estimation of the Lebesgue constant in the $d$-dimensional simplex. The Lebesgue constant is used to assess the interpolation capabilities of a nodal distribution and is defined as the maximum of the Lebesgue function, see Eq. (4.1). Due to the absolute value, the function

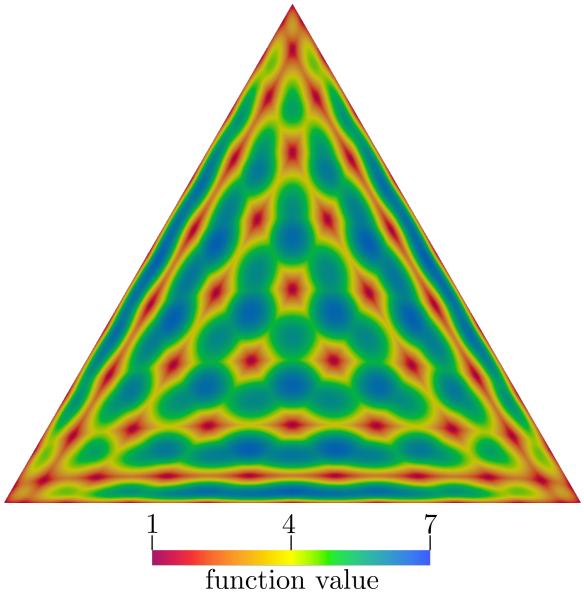



Figure 4.8: Lebesgue function of the warp-and-blend nodal distribution of polynomial degree 10 in the triangle (Warburton, 2006).

is non-differentiable and, hence, a zeroth-order method is required to compute the maximum.

In Fig. 4.8, for a triangle of polynomial degree 10, we show the Lebesgue function for a warp-and-blend nodal distribution (Warburton, 2006). Since this nodal family is symmetric, the Lebesgue function is symmetric, too. Consequently, it is enough to find the maximum inside the sextant of the triangle. More precisely, we consider the symmetric tile of the $d$-dimensional simplex determined by the points with barycentric coordinates $\left(\lambda^1, \ldots, \lambda^{d+1}\right)$, $\sum_{i=1}^{d+1} \lambda^i = 1$, such that $\lambda^i \geq \lambda^j$ if $i \geq j$.

### 4.5.1 Verification in 2D and 3D

To verify the estimated values of the Lebesgue constant found using our method, we compare our results with those reported in Warburton (2006). In Table 4.1, we report the value of the Lebesgue constant for the equispaced and the warp-and-blend distribution (Warburton, 2006) for several polynomial degrees $p$, $p = 2, \ldots, 15$, in the triangle. The initial sampling is composed of the three vertices of the domain, $q = 1$. We set $\delta = 10^{-4}$ and $\varepsilon = 10^{-3}$ for the stopping criterion, and in all cases the minimum is found before the limit of 50 iterations is reached. We also list the number of sample points needed. In general, as the polynomial degree increases, also does the

| | Equispaced | | Warp-and-blend | |
|---|---|---|---|---|
| $p$ | $\Lambda_{\mathrm{Eq}}$ | # points | $\Lambda_{\mathrm{WB}}$ | # points |
| 2 | 1.67 | 219 | 1.67 | 221 |
| 3 | 2.27 | 302 | 2.11 | 292 |
| 4 | 3.47 | 280 | 2.66 | 283 |
| 5 | 5.45 | 280 | 3.12 | 483 |
| 6 | 8.75 | 424 | 3.70 | 404 |
| 7 | 14.34 | 356 | 4.27 | 378 |
| 8 | 24.01 | 409 | 4.96 | 668 |
| 9 | 40.92 | 533 | 5.74 | 611 |
| 10 | 70.89 | 397 | 6.67 | 685 |
| 11 | 124.53 | 427 | 7.90 | 497 |
| 12 | 221.41 | 538 | 9.36 | 747 |
| 13 | 397.70 | 422 | 11.47 | 735 |
| 14 | 720.69 | 412 | 13.97 | 1142 |
| 15 | 1 315.89 | 599 | 17.65 | 885 |

Table 4.1: Number of sample points needed to estimate the Lebesgue constant using the equispaced distribution, $\Lambda_{\mathrm{Eq}}$, and the warp-and-blend distribution (Warburton, 2006), $\Lambda_{\mathrm{WB}}$, of polynomial degree $p = 2, \ldots, 15$ as interpolation set in the triangle.

number of points. This is so because, for high polynomial degrees, the basins of the Lebesgue function that contain the minima are smaller and deeper and, consequently, more sample points are needed to capture the minimum with the same precision. In spite of this fact, we remark that our method is able to compute a good estimate of the Lebesgue constant using less than 1200 sample points, yet the values coincide with those reported in Warburton (2006) up to the second decimal place.

In Fig. 4.9, we show the final sampling used to capture the maximum of the Lebesgue function associated with the warp-and-blend distribution of polynomial degree 10 represented in Fig. 4.8. Since this function features triangle symmetry, the search space is simply the sextant. We remark that regions with higher values, blueish areas in Fig. 4.8, present a finer sampling in Fig. 4.9. We also see that there are three local minima with similar function values, yet the global minimum is the one in the interior of the domain.

In Table 4.2, we report the maximum value and the number of sample points needed to estimate the Lebesgue constant for the equispaced and the warp-and-blend distribution (Warburton, 2006) for several polynomial degrees $p$, $p = 2, \ldots, 15$, in the tetrahedron. We use the same initial sampling and the same tolerances $\delta = 10^{-4}$ and

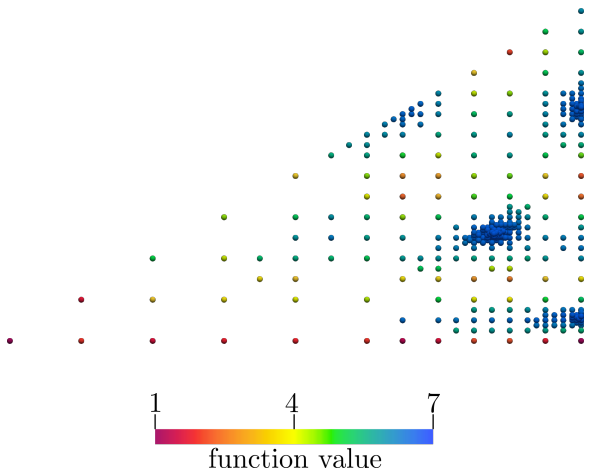



Figure 4.9: Final sampling used to capture the maximum of the Lebesgue function of the warp-and-blend nodal distribution of polynomial degree 10 in the triangle (Warburton, 2006).

| | Equispaced | | Warp-and-blend | |
|---|---|---|---|---|
| $p$ | $\Lambda_{\text{Eq}}$ | # points | $\Lambda_{\text{WB}}$ | # points |
| 2 | 2.00 | 398 | 2.00 | 398 |
| 3 | 3.02 | 565 | 2.93 | 635 |
| 4 | 4.88 | 536 | 4.07 | 722 |
| 5 | 8.09 | 581 | 5.32 | 990 |
| 6 | 13.66 | 690 | 7.01 | 1040 |
| 7 | 23.38 | 675 | 9.21 | 1671 |
| 8 | 40.55 | 751 | 12.54 | 854 |
| 9 | 71.15 | 708 | 17.02 | 1651 |
| 10 | 126.20 | 779 | 24.36 | 2412 |
| 11 | 225.99 | 798 | 36.35 | 1644 |
| 12 | 408.15 | 853 | 54.18 | 1707 |
| 13 | 742.69 | 860 | 84.62 | 2594 |
| 14 | 1 360.49 | 843 | 135.75 | 2635 |
| 15 | 2 506.95 | 926 | 217.71 | 3519 |

Table 4.2: Number of sample points needed to estimate the Lebesgue constant using the equispaced distribution, $\Lambda_{\text{Eq}}$, and the warp-and-blend distribution (Warburton, 2006), $\Lambda_{\text{WB}}$, of polynomial degree $p = 2, \ldots, 15$ as interpolation set in the tetrahedron.

$\varepsilon = 10^{-3}$ for the stopping criterion, and in all cases the minimum is found before the limit of 50 iterations is reached. As in the two-dimensional case, more sample points are required to estimate the Lebesgue constant of higher polynomial degrees.

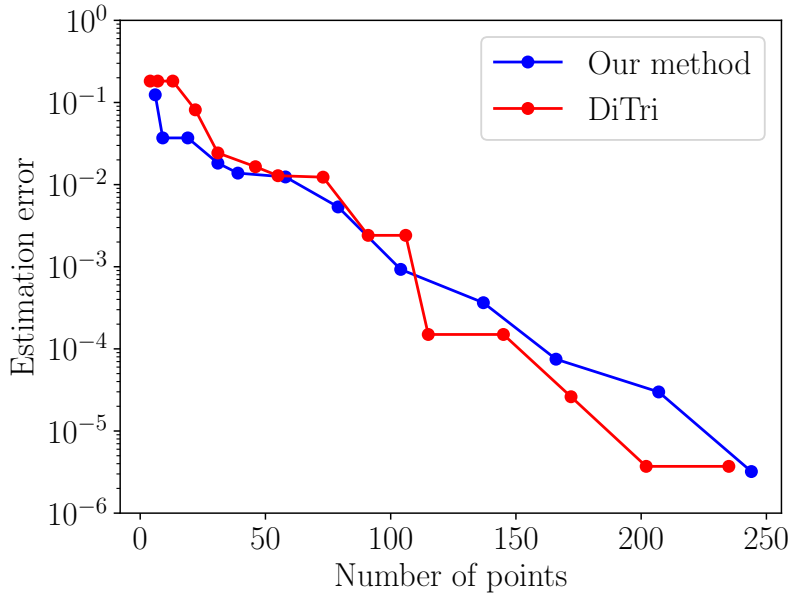



Figure 4.10: Error in the estimation of the Lebesgue constant in terms of the number of sample points using our method (blue) and DiTri (Roth, 2005) (red) for the warp-and-blend distribution of polynomial degree 10 in the triangle.

We highlight that the values coincide with those reported in Warburton (2006) up to the second decimal place, and only 3519 points are needed to compute an estimate of the Lebesgue constant for the warp-and-blend distribution of polynomial degree 15. In contrast, using an admissible mesh of $3519^{(1/3)} \approx 16$ points per line, the estimated value of the Lebesgue constant is 211.07.

### 4.5.2 Performance comparison in 2D

To check the performance, we compare the results of our method with the results of our implementation of the DiTri algorithm (Roth, 2005). For both methods, we compute in a triangle the Lebesgue constant for the warp-and-blend symmetric nodal distribution of polynomial degree 10. To do so, we report, at the end of each iteration, the number of sample points and the relative error of the maximum estimation. In Fig. 4.10, we show the evolution of our method, in blue, and the DiTri algorithm, in red. We observe that both methods show similar evolution. Moreover, to capture the maximum with a relative error below $10^{-4}$, both methods need less than 200 sample points. Note that we do not consider the non-deterministic method (Warburton, 2006) because the initial sampling already consists of 10 000 sample points.

| | Dimension 4 | | | Dimension 5 | | | Dimension 6 | | |
|---|---|---|---|---|---|---|---|---|---|
| $p$ | # points | $\Lambda$ | $\tilde{\Lambda}$ | # points | $\Lambda$ | $\tilde{\Lambda}$ | # points | $\Lambda$ | $\tilde{\Lambda}$ |
| 6 | 1126 | 19.22 | 19.05 | 2030 | 25.49 | 19.90 | 1807 | 32.63 | 25.45 |
| 7 | 1075 | 34.08 | 33.51 | 1545 | 46.54 | 34.43 | 1790 | 61.00 | 50.46 |
| 8 | 1033 | 60.86 | 55.75 | 1578 | 85.24 | 65.06 | 2466 | 114.13 | 96.97 |
| 9 | 1175 | 109.43 | 90.72 | 1677 | 156.62 | 126.31 | 2252 | 213.76 | 180.24 |
| 10 | 1572 | 198.08 | 150.71 | 1667 | 288.82 | 241.51 | 2381 | 400.93 | 323.42 |

Table 4.3: Estimation of the Lebesgue constant of the equispaced distribution of polynomial degree $p = 6, \ldots, 10$ in the $d$-simplex, $d = 4, \ldots, 6$. Number of sample points needed to compute our estimation $\Lambda$, and approximation using an admissible mesh $\tilde{\Lambda}$.

Although the evolution of both methods is similar in 2D, our method scales better in higher dimensions. We highlight that to refine a point using our method, we generate at most 6 new sample points, while DiTri always requires 3 new sample points. This difference is almost irrelevant in 2D and, consequently, the two curves follow a similar trend. However, this would not be the case in higher dimensions since we generate at most $(d+1)\,d$ new sample points per refinement, while an extension to higher dimensions of DiTri would require $2^d - 1$ new sample points. Hence, for each method, the number of sample points scales differently with the dimension $d$, exponentially for an extension to arbitrary dimensions of DiTri, quadratically for our method.

Moreover, since we only need a point data structure, the refinement, completion, and smoothing operations are implemented for arbitrary dimensions and no dimension-specific considerations are required. Finally, the system of rational barycentric coordinates allows easily accessing the adjacent points with no need for storing the neighbor structure, which enables using a stopping criterion based on the flatness of the function.

### 4.5.3 Results in 4D, 5D, and 6D

The values reported in Sect. 4.5.1 for 2D and 3D coincide with the ones found in the literature (Warburton, 2006). Thus, we believe that our method is capable of estimating the Lebesgue constant accurately using a moderate amount of sample points. In Table 4.3, we show our estimation $\Lambda$ of the Lebesgue constant of the

equispaced nodal distribution of polynomial degree $p$ in the $d$-simplex, $p = 6, \ldots, 10$, $d = 4, \ldots, 6$. We also show the number of required sample points. As expected, we observe that the values increase with the polynomial degree and the dimension.

As an alternative to our method, we can use an admissible mesh (Calvi and Levenberg, 2008) to estimate the Lebesgue constant. Since in dimension $d = 4$ we provided an estimate using at most 1572 sample points, we approximate the Lebesgue constant using an admissible mesh of approximately $1572^{1/4}$ points per line. Analogously, in 5D and 6D, we compute an estimate using approximately $2030^{1/5}$ and $2466^{1/6}$ points per line, respectively. In Table 4.3, we denote by $\tilde{\Lambda}$ the maximum function value at this grid of sample points. We observe that with the same number of points, our method captures a higher value and, therefore, it is more suitable to estimate the Lebesgue constant in small and moderate dimensionality.

## 4.6 Concluding remarks

To estimate the Lebesgue constant on the simplex, we have proposed a new specific-purpose point refinement method. The proposed method features a smooth gradation of the resolution, neighbor queries based on neighbor-aware coordinates, and a point refinement that algebraically scales with dimensionality. Remarkably, by using neighbor-aware coordinates, the point refinement method is ready to automatically stop using a Lipschitz criterion.

In mid-range dimensionality, we conclude that the point refinement is well-suited to automatically and efficiently estimate the Lebesgue constant on simplices. Specifically, for different polynomial degrees and point distributions, our results efficiently have reproduced the literature estimations for the triangle and the tetrahedron. Moreover, we have adaptively estimated the Lebesgue constant up to six dimensions.

In perspective, for a given polynomial degree, the proposed point refinement might be relevant to obtaining a set of simplex points that guarantees a small interpolation error. That is, it efficiently estimates the Lebesgue constant, an estimation that is helpful in two ways. First, to assess the quality of a given set of interpolation points. Second, to evaluate the Lebesgue constant when optimizing the interpolation error for the point distribution as a design variable. We also think the method might be well-suited to seek optima in the simplex for functions behaving as the Lebesgue function.

# Chapter 5

# Exploring locally optimal nodal distributions of a Lebesgue constant proxy

## 5.1 Introduction

In computational science and engineering, point distributions featuring optimal interpolation error for unknown functions have a major interest. This interest is prompted by the different applications of optimal interpolation point distributions. In this thesis, we are interested in interpolating curved geometry obtained as the boundary of a 3D representation of the discretized domain (Jiménez-Ramos et al., 2020, 2022), see Chapter 3, or as the 3-dimensional boundary of a space-time 4D geometry. Moreover, we are also interested in using optimal interpolation points as a proxy of good nodal distributions for the nodal high-order unstructured methods such as the discontinuous (Hesthaven and Warburton, 2007) and continuous Galerkin (Karniadakis and Sherwin, 2005) methods for 4D space-time solutions.

In these high-dimensional applications, it is important to specifically obtain point distributions on the simplex. This is so because complex geometries can be discretized using automatic triangular and tetrahedral mesh generators such as the advancing front method (Peraire et al., 1987; Löhner and Parikh, 1988) and the Delaunay method (George et al., 1990; Shewchuk, 2005; Si, 2015). Beyond three dimensions,

there also exist mesh generators featuring Delaunay implementations (Barber et al., 1996; The CGAL Project, 2023).

Fair point distributions for interpolation in the simplex are obtained by optimizing an objective function that has as design variables either the point coordinates or a parameter controlling an explicit warp and blend of the points on the boundary. The objective function, in turn, is either the non-differentiable infinity norm of the interpolation operator, known as Lebesgue constant, or a differentiable surrogate. These alternatives have yielded different approaches. On the one hand, for up to three dimensions, we have methods optimizing the point coordinates as design variables. In one dimension, it is possible to obtain the optimal Lebesgue point distributions with deterministic optimization methods (Angelos et al., 1989). In two dimensions, it is possible to explore first the local minima of a surrogate of the Lebesgue constant (Fekete, 1923; Bos, 1983; Chen and Babuška, 1995; Hesthaven, 1998; Taylor et al., 2000; Roth, 2005; Briani et al., 2012; Van Barel et al., 2014) and, then, optimize the Lebesgue constant of the candidates with a non-deterministic genetic algorithm (Roth, 2005). Nonetheless, it is also possible to optimize directly the Lebesgue constant by means of a sequential quadratic programming approach (Briani et al., 2012; Rapetti et al., 2012) or a damped Newton's method (Heinrichs, 2005). In 3D, some local minima of a differentiable surrogate have been found (Chen and Babuška, 1996; Hesthaven and Teng, 2000), yet optimizing the point coordinates to minimize the Lebesgue constant has not been considered. On the other hand, for up to three dimensions, there are methods optimizing the parameter of a warp and blend function aiming to reduce the Lebesgue constant (Warburton, 2006). Alternatively, there also exist explicit point distributions generated by blending a prescribed nodal distribution on the edges into the interior of the triangle and the tetrahedron (Blyth and Pozrikidis, 2005; Luo and Pozrikidis, 2006) or the $d$-dimensional simplex (Isaac, 2020).

The methods using the point coordinates as variables lead to fairer point distributions for interpolation when higher is the interpolation degree. This is so because they are able to obtain points that break the grid structure of an initial equidistributed distribution, a mechanism that has been identified as primordial to obtain better interpolation points (Taylor et al., 2000; Roth, 2005). Remarkably, these suboptimal sets cluster the points around the boundary, yet present an equidistributed arrangement on the sphere (Roth, 2005). Unfortunately, beyond three dimensions,

there are no methods that optimize an objective function with the point coordinates as design variables.

Accordingly, the goal of this chapter is to obtain fair point distributions for interpolation on the high-dimensional simplex by optimizing an objective function with the point coordinates as design variables. We expect that the obtained point distributions feature good interpolation properties and provide the required accuracy for the aforementioned applications in complex geometry.

Our main contribution is to devise a specific-purpose approach that systematically and deterministically explores the heuristically best local minima of a proxy of the Lebesgue constant up to dimension four. To explore proximal distributions, we heuristically relocate one node to the other side of the uphill of the energy landscape. A proxy of the energy landscape is obtained through a Delaunay triangulation on the $(d+1)$-sphere, where the opposite faces of the simplices incident to a node determine an approximation of the uphill energy landscape of the functional around such node. After these nodal dislocations, these point configurations are locally optimized using Newton's method equipped with a trust-region globalization taking into account the point resolution. Finally, the optimized nodal sets are represented as nodes of a tree that are heuristically explored to obtain the best proxy values.

Our solution presents four main benefits. First, the local nodal dislocation technique is deterministic and allows reaching local minima non-reachable with continuous minimization. Second, it is a generic formulation on the spatial dimension and thus, we are able to optimize up to dimension four using the available computational resources. Third, the tree structure handles the unicity of local optima using a specific hash function. Fourth, the heuristic exploration of the tree nodes enforces reducing the value of the functional.

The rest of the chapter is organized as follows. In Sect. 5.2, we recall some preliminaries regarding some properties of the Lebesgue constant and the parameterization of symmetric nodal distributions in the simplex. Next, in Sect. 5.3, we describe the connection between the simplex and the sphere, and illustrate the relation between the influence regions of the Lagrange interpolating polynomials and the euclidean Delaunay mesh. The nodal dislocation technique and the exploration method are presented in Sect. 5.4. Then, in Sect. 5.5, we include discussions and results. The chapter ends with the concluding remarks in Sect. 5.6.

## 5.2 Preliminaries

Following, we recall some notation and definitions introduced in Chapter 2 used throughout this chapter regarding the Lebesgue constant and the parameterization of symmetric nodal distributions in the simplex. We refer the reader to Chapter 2 for a full detailed description.

For a given polynomial degree $p$ and a set of interpolation nodes $\boldsymbol{Z} = \{\boldsymbol{z}_j\}_{j=1,\dots,N_{p,d}}$ in a simplex $K^d \subset \mathbb{R}^d$, the Lebesgue constant $\Lambda$ corresponds to the infinity norm of the interpolation operator $\mathcal{I}_{\boldsymbol{Z}}$ and appears in the upper bound of the interpolation error of a function $f$,

$$\left\| f - \mathcal{I}_{\boldsymbol{Z}} f \right\|_\infty \leq (1 + \Lambda) \left\| f^\star - f \right\|_\infty ,$$

where $f^\star$ denotes the best approximating polynomial. Accordingly, to attain convergence of the interpolator $\mathcal{I}_{\boldsymbol{Z}} f$ to the target function $f$, $\Lambda$ should grow slower in $p$ than $\left\| f^\star - f \right\|_\infty$ dies away. In particular, uniform convergence is attained if

$$\lim_{p \to +\infty} \Lambda^{1/p} = 1.$$

Remarkably, the Lebesgue constant can be expressed as

$$\Lambda(\boldsymbol{Z}) = \max_{\boldsymbol{x} \in K^d} \sum_{i=1}^{N_{p,d}} \left| \phi_i(\boldsymbol{x}; \boldsymbol{Z}) \right| ,$$

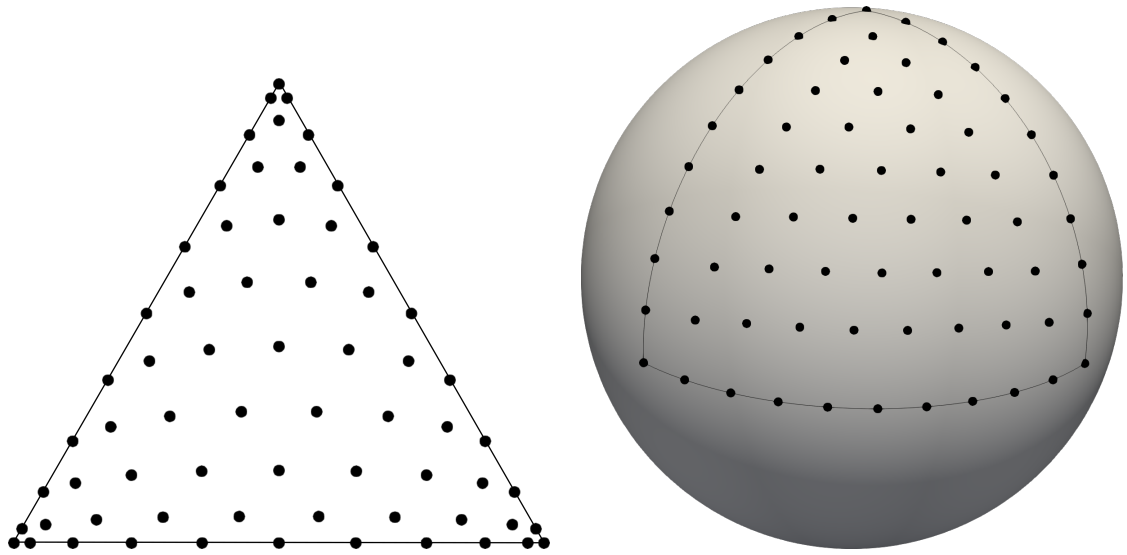
where $\phi_j$ is the Lagrange interpolating polynomial associated to the node $\boldsymbol{z}_j$, $j = 1, \dots, N_{p,d}$.

In this thesis, the set of interpolation nodes $\boldsymbol{Z}$ is symmetric. Thus, as detailed in Sect. 2.1, only $n$ degrees of freedom encoded in a vector $\boldsymbol{y}$ are enough to describe the position of the whole set of nodes. We recall that $\boldsymbol{\sigma}$ is the function mapping the vector of degrees of freedom $\boldsymbol{y}$ to the whole nodal distribution,

$$\begin{array}{rccc} \boldsymbol{\sigma}: & \mathbb{R}^n & \to & \mathbb{R}^d \times \overset{N_{p,d}}{\cdots} \times \mathbb{R}^d \\ & \boldsymbol{y} & \mapsto & \boldsymbol{\sigma}(\boldsymbol{y}) = \boldsymbol{Z} \end{array} .$$

## 5.3 Spherical simplex

Up to moderate polynomial degree, close to optimal interpolation nodal distributions present a grid structure. In contrast, for high polynomial degrees, quasi-optimal sets



(a) (b)

Figure 5.1: Close to optimal nodal distributions of degree $p = 10$ (a) in the triangle, and (b) the enhanced representation on the sphere.

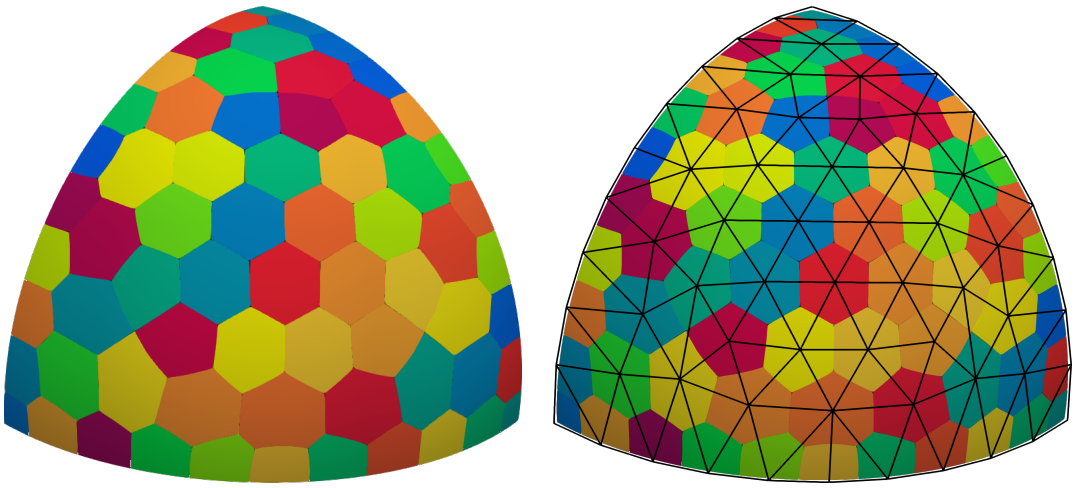
tend to cluster points towards the boundary and do not follow a uniform distribution, see Fig. 5.1(a). Remarkably, there exists a mapping representing close to optimal interpolation nodes in the simplex as uniformly distributed points on the sphere $\mathcal{S}^d \subset \mathbb{R}^{d+1}$, see Fig. 5.1(b). More precisely, a point $P \in K^d$ expressed in barycentric coordinates as $P = (\lambda^i)_{i=1,\dots,d+1}$, with $\lambda^i \geq 0$, and $\sum_{i=1}^{d+1} \lambda^i = 1$, is mapped onto the sphere $\mathcal{S}^d$ via

$$\begin{array}{rccc} \varphi: & K^d \subset \mathbb{R}^d & \longrightarrow & \mathcal{S}^d \subset \mathbb{R}^{d+1} \\ & P = (\lambda^i)_{i=1,\dots,d+1} & \longmapsto & \varphi(P) = \left(\sqrt{\lambda^i}\right)_{i=1,\dots,d+1}. \end{array} \tag{5.1}$$

The region $\varphi\left(K^d\right)$, which we call *spherical simplex*, corresponds to the orthant of the sphere $\mathcal{S}^d$, namely the surface of the sphere where all the coordinates are non-negative. Remarkably, uniformly distributed points on the orthant are distributed in the simplex with density

$$\rho = \prod_{i=1}^{d+1} \frac{1}{2\sqrt{\lambda^i}}$$

via the mapping $\varphi$ (Roth, 2005). This density coincides with the extremal measure of the simplex (Baran, 1995) and, in the interval, Fekete points follow this distribution (Bloom et al., 1992). For higher dimensions, it is an open question whether there exist Fekete points following the density $\rho$ (Bloom et al., 1992), but it is conjectured that



(a) (b)

Figure 5.2: (a) Lagrangian influence zones, each represented by a different color, and (b) the skeleton of the Delaunay mesh using the Euclidean distance.

nodal sets with good interpolatory qualities in the simplex are distributed uniformly over the orthant (Roth, 2005). Unfortunately, to obtain good interpolations points, uniform distributions over the orthant are not sufficient. For instance, equispaced distributions of points using spherical averaging (Buss and Fillmore, 2001) feature large Lebesgue constant and do not exhibit uniform convergence (Roth, 2005).

Nevertheless, considering a nodal interpolative distribution as a set of points on the sphere is an advantageous representation since it provides the nodes with connectivities. To infer a structure on the spherical simplex, we first consider the influence zone of a Lagrange polynomial $\phi_i$, that is, the set of points in the simplex where the polynomial $\phi_i$ dominates. Formally, the influence zone $V_i$ of a Lagrange polynomial $\phi_i$ is defined as

$$V_i = \{\boldsymbol{x} \in K^d \colon \left|\phi_i\left(\boldsymbol{x}\right)\right| \geq \left|\phi_j\left(\boldsymbol{x}\right)\right|, \ \forall j = 1, \ldots, N_{p,d}\}.$$

For the distribution represented in Fig. 5.1(b), we plot in Fig. 5.2(a) the sets $\varphi\left(V_i\right)$, for all $i = 1, \ldots, N_{p,d}$. For close to optimal nodal distributions, we observe that the sets $\varphi\left(V_i\right)$ resemble euclidean Voronoi cells. Thus, we approximate the regions $\varphi\left(V_i\right)$ by the constrained Voronoi diagram on the spherical simplex. Accordingly, the dual of this diagram, the constrained Delaunay mesh, provides a reasonable set of point connectivities, see Fig. 5.2(b).

Furthermore, the skeleton of the Delaunay mesh approximates the uphills of the

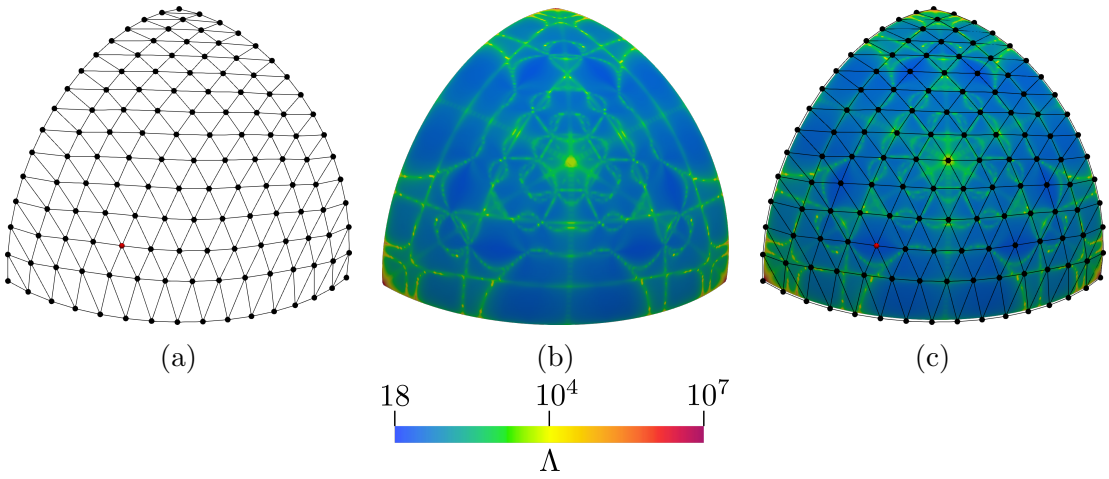



Figure 5.3: (a) Skeleton of the Delaunay mesh of a close to optimal nodal distribution of polynomial degree $p = 15$. (b) Value of the function $\Lambda$ evaluated at the nodal distribution resulting from placing the red node at every position of the spherical simplex preserving symmetry. (c) Superposition of the skeleton of the Delaunay mesh and Fig. 5.3(b).

Lebesgue constant energy landscape for a free node. Let us consider the nodal distribution $\boldsymbol{Z}$ of polynomial degree $p = 15$ shown in Fig. 5.3(a), and compute the Lebesgue constant when the red node moves through the interior of the triangle. Specifically, in Fig. 5.3(b), we plot the estimation of the Lebesgue constant of the nodal distribution obtained as a result of relocating the red node from Fig. 5.3(a) to every position in the interior of the triangle while preserving the symmetry. Remarkably, the skeleton of the Delaunay mesh approximates the uphills of this landscape, see Fig. 5.3(c). More precisely, let $\mathcal{E}$ be the set containing the mesh elements of the constrained Delaunay mesh generated from the nodal set $\boldsymbol{Z}$, and define $\mathcal{I}(\boldsymbol{z}_i)$ as the set of elements incident to node $\boldsymbol{z}_i$,

$$\mathcal{I}(\boldsymbol{z}_i) := \{e \in \mathcal{E} : \boldsymbol{z}_i \in e\}.$$

Then, the boundary of $\mathcal{I}(\boldsymbol{z}_i)$ determines a discrete hyper-sphere that approximates the energy uphills surrounding node $\boldsymbol{z}_i$. Going through these uphills is key to obtain quasi-optimal interpolation nodal distributions, as it is to be detailed in the next section.

According to the previous observation, we consider minor modifications of the method to estimate the Lebesgue constant proposed in Chapter 4 to account for the importance of this result. Specifically, the search space is not the simplex but the orthant of the $(d+1)$-sphere $\mathcal{S}^d$. Accordingly, sample points are computed by

means of spherical averaging of the orthant vertices (Buss and Fillmore, 2001), where the weight of each vertex is given by the corresponding component of the rational barycentric coordinate. Nevertheless, sample points are mapped from the sphere onto the simplex via $\varphi^{-1}$ to evaluate the Lebesgue function. Finally, in the Lipschitz-based automatic stopping criterion, we consider the geodesic distance between sample points to compute a local estimate of the Lipschitz constant.

## 5.4 Exploring local minima

The number of local minima of the Lebesgue constant increases with the polynomial degree. For low polynomial degrees, the Buss-Fillmore distribution (Buss and Fillmore, 2001) belongs to the convergence basin of the optimal Lebesgue configuration but, for high polynomial degrees, these two distributions are basins apart. Thus, to find the optimal Lebesgue configuration, one can optimize the Lebesgue constant starting the optimization procedure from different initial approximations and, among these local optima, the nodal distribution with the lowest Lebesgue constant determines an approximation of the global Lebesgue optimum. Unfortunately, optimizing the Lebesgue constant is not straight-forward. Thus, in order to explore as many minima as possible in a fast and systematic manner, we present an exploration method based on nodal dislocations, optimization of the Chen-Babuska functional (Chen and Babuška, 1995) and featuring an analogy to standard graph search algorithms. First, in Sect. 5.4.1, we present the Chen-Babuska functional used as a proxy of the Lebesgue constant, later, in Sect. 5.4.2, we introduce the concept of nodal dislocation, and finally, in Sect. 5.4.3, we explicitly detail the exploration algorithm.

### 5.4.1 Optimization of Chen-Babuska functional

Let $\boldsymbol{y}$ be the degrees of freedom describing the node position of a symmetric nodal distribution, see Sect. 5.2. To approximate the minimization of the Lebesgue constant, we consider the fast minimization of the Chen-Babuska functional (Chen and Babuška, 1995). This functional corresponds to the square of the vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials,

$$\beta\left(\boldsymbol{y}\right) := \frac{1}{\mathrm{vol}\left(K^d\right)} \int_{K^d} \sum_{i=1}^{N_{p,d}} \phi_i^2\left(\boldsymbol{x}; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \mathrm{d}\boldsymbol{x}. \tag{5.2}$$

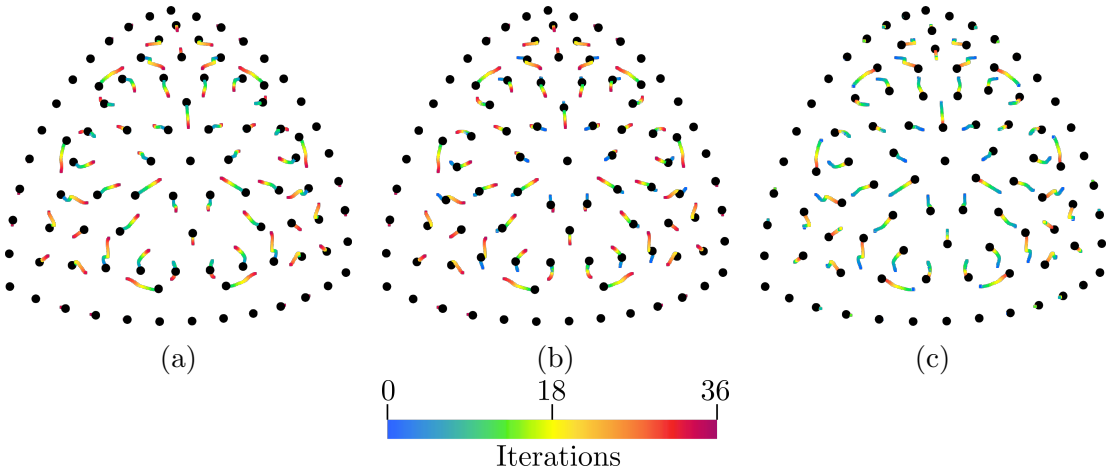



Figure 5.4: Nodal dynamics during the optimization of the Chen-Babuska functional for a nodal distribution of polynomial degree 12 in the triangle. (a) Initial nodal set, (b) nodal set after 8 iterations, and (c) nodal set at the end of the optimization process. Trajectories are colored in terms of the iteration number.

When this functional is used as a proxy of the Lebesgue function, it allows efficiently obtaining new nodal distributions with approximately improved Lebesgue constants. That is, given an initial nodal distribution, we can compute the closer local minimum of the Chen-Babuska functional with standard second-order methods. This computationally efficient possibility is so since there are analytical expressions for evaluating the functional value and the first and second derivatives, see Appendix D. Specifically, we solve for the minimum of $\beta(\boldsymbol{y})$ by means of Newton's method equipped with a trust-region globalization (Nocedal and Wright, 2006) implemented in Mogensen and Riseth (2018). The trust region is chosen to account for the point resolution and is adaptively updated to ensure feasible point distributions, see details in Sect. D.2.

To illustrate the optimization procedure, in Fig. 5.4 we show on the sphere the dynamics for a nodal distribution of polynomial degree 12 in the triangle. The initial approximation is depicted in Fig. 5.4(a), and the trajectory of the nodes is colored in terms of the iteration number. We observe that the restrictions on the maximum radius of the trust region computations force a smooth dynamics of the nodes. Interestingly, we appreciate a torsion-like movement of the interior nodes that allow the distribution to achieve the minimum of the basin. This phenomenon can be observed comparing Fig. 5.4(b) and Fig. 5.4(c), where we show the nodal set after 8 iterations and the final position, respectively. After 8 iterations, the interior node located at the bottom-center part of the region has not moved significantly but, at

some point, it moves to the left while the neighbor node at the top-left moves up and the neighbor node at the top-right moves down. These three nodes rotate clockwise in a synchronous way and, finally, the nodal set attains the minimum. We remark that this kind of dynamics is not specific to this polynomial degree or dimension. Actually, we have observed this pattern when optimizing high polynomial degrees distributions in the triangle and the tetrahedron, and we expect to find this coupling in higher dimensions with an even more complex entanglement.

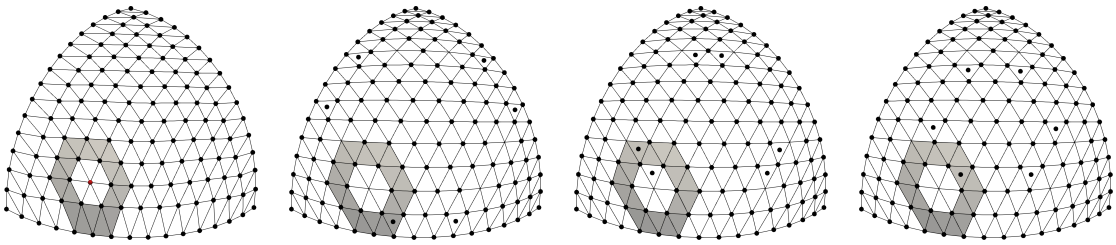
Unfortunately, the Chen-Babuska functional presents several local minima. For low polynomial degrees, Buss-Fillmore nodal distributions (Buss and Fillmore, 2001) are close to the global optimum, but for high polynomial degrees, the Buss-Fillmore distribution may not belong to the optimal convergence basin. Thus, finding the global minimum requires exploration approaches.

### 5.4.2 Nodal dislocations

To explore different local minima, we need a systematic manner to provide different initial nodal distributions. Ideally, these distributions must converge to different local minima, and if we explored all the local minima, we would obtain the global minimum. Herein, we only expect to explore a large number of local minima and thus, to obtain a quasi-optimal approximation of the global minimum. Next, we detail how to systematically generate the set of initial configurations.

First, we remark that each local minimum of the Chen-Babuska functional corresponds to a different point distribution which belongs to a different convergence basin. Unfortunately, since these points do not feature explicit connectivities, it is not easy to structurally identify different configurations. To heuristically distinguish point distributions, we compute the connectivity structure given by the Delaunay triangulation of the points on the spherical simplex, see Sect. 5.3. Herein, we consider that two nodal distributions are structurally different if the mesh connectivities are different. Thus, we explore different nodal configurations by exploring different mesh connectivities.

Moving one node, we can easily generate different mesh connectivities. If we fix all the nodes, we can obtain different configurations by moving a node to the centroid of some of the simplices of the Delaunay mesh. Specifically, let $\mathcal{E}$ be the set containing the mesh elements of the constrained Delaunay mesh generated from the nodal set $\boldsymbol{Z}$, and $\mathcal{I}(\boldsymbol{z}_i)$ the set of elements incident to node $\boldsymbol{z}_i$, see Sect. 5.3. We define the



(a) (b) (c) (d)

Figure 5.5: (a) Skeleton of the Delaunay mesh of a close to optimal nodal distribution of polynomial degree $p = 15$. The elements in the ring of the red node are colored in gray. (b)-(d) Nodal configurations after dislocating the interior red node to some of the elements in the ring.

elemental ring $R(\boldsymbol{z}_i)$ as the set of adjacent elements to the incident elements to node $\boldsymbol{z}_i$,

$$R(\boldsymbol{z}_i) := \left( \bigcup_{e \in \mathcal{I}(\boldsymbol{z}_i)} \bigcup_{\boldsymbol{z}_j \in e} \mathcal{I}(\boldsymbol{z}_j) \right) \setminus \mathcal{I}(\boldsymbol{z}_i) . \tag{5.3}$$

Then, we propose dislocating the node $\boldsymbol{z}_i$ to the projection onto the sphere of the centroid of each of the elements in $R(\boldsymbol{z}_i)$. Consequently, we are generating at most $|R(\boldsymbol{z}_i)|$ structurally different nodal distributions per nodal dislocation. In Fig. 5.5(a), we show a close to optimal nodal distribution of polynomial degree 15 and the skeleton of the Delaunay mesh, where we color in gray the elements in the ring of the red interior node. In Fig. 5.5(b)-Fig. 5.5(d), we show three of the fourteen nodal dislocations after relocating the red node to the centroid of the elements in its ring. Note that not only the red node is relocated, but also the nodes in its equivalence class.

This systematic exploration procedure has an energetic interpretation. This is so since, for a free node, the energy landscape of the Chen-Babuska functional is a proxy of the energy landscape of the Lebesgue constant. Consider the nodal distribution of polynomial degree $p = 15$ depicted in Fig. 5.6(a). Then, in Fig. 5.6(b), we plot the functional $\beta$ at the nodal distribution obtained as a result of relocating the red node from Fig. 5.6(a) to every position of the simplex while preserving symmetry. Comparing Fig. 5.3(b) and Fig. 5.6(b), we observe that the uphills of the energy landscapes of the Lebesgue constant $\Lambda$ and the Chen-Babuska $\beta$ are almost identical. Accordingly, we approximate the uphills of the energy landscape of the Chen-Babuska functional by the boundary of the elements incident the red node. Thus, by relocating

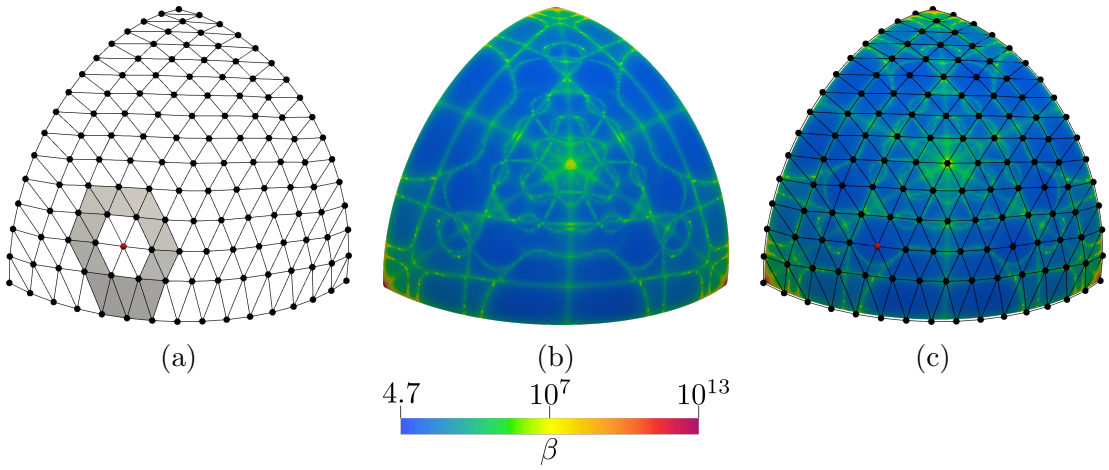



Figure 5.6: (a) Skeleton of the Delaunay mesh of a close to optimal nodal distribution and ring of the red node. (b) Value of the function $\beta$ evaluated at the nodal distribution resulting from placing the red node at every position of the spherical simplex preserving symmetry. (c) Superposition of the skeleton of the Delaunay mesh and Fig. 5.6(b).

a node onto the elements in its ring, we are enforcing a tunnel effect in such a way that the point crosses to the other side of the uphill energetic barrier. Furthermore, the centroid of each constrained simplex is an approximation of the point with highest influence of the surrounding Lagrange polynomials. Then, by moving an interpolation point on that highly influenced region, we are enforcing there a Lebesgue value of one, a value that is the smallest possible one. Finally, in Fig. 5.6(c) we merge Fig. 5.6(a) and Fig. 5.6(b) to illustrate that dislocating the red node to the centroid of the elements in its ring forces the tunnel effect and moves the node through the uphill barriers.

We remark that the energetic barriers on the functional also appear due to the fact that we are considering symmetric nodal configurations. For instance, a node that belongs to the median of the triangle cannot move beyond the barycenter by optimizing the Chen-Babuska functional. This is so because as this node approaches the barycenter, also do the nodes in its equivalence class and, consequently, the value of the functional increases. Thus, the proposed dislocation operation allows to explore distributions with a nodal disposition that would be difficult to obtain by means of smooth optimization.

Considering symmetric nodal distributions demands a precise description of the nodal dislocation operation. To dislocate a node, we relocate it to the centroid of an

element in the ring and project it onto the sphere. Unfortunately, this nodal dislocation operation is not well-posed for nodes with less than $d$ degrees of freedom. Recall that, as described in Sect. 2.1, the number of degrees of freedom of a representative is determined by the number of symmetry entities the node belongs to, and the intersection of these entities describes the domain manifold where the node can freely move. In particular, interior nodes which do not belong to any symmetry entity are free to move in the interior of the simplex. For those representatives with $d$ degrees of freedom, the projection onto the sphere of the centroid of an element belongs to the spherical simplex and the dislocation operation is well-defined. However, if we consider the case when the representative has less than $d$ degrees of freedom, for instance, a node on the median of the triangle with one degree of freedom, the projection onto the sphere of the centroid of an element does not belong, in general, to the spherical median. To circumvent this issue, we find the orthogonal projection of the centroid to the domain manifold on the sphere. More precisely, we dislocate the representative to the point on the spherical domain manifold such that the distance to the centroid is minimized. We find this point by projecting to the subspace that contains the vertices of the spherical manifold and passes through the origin, and then projecting again to the sphere. Finally, once the representative is relocated, the nodes in its equivalence class are repositioned according to the corresponding permutation to ensure symmetry is preserved.

By dislocating each of the nodal representatives to the elements in their ring, we expect to generate a set of structurally different point distributions. Heuristically, each of these nodal sets belongs to a different Chen-Babuska convergence basin and should be optimized to attain the local minimum. After optimizing these new configurations, we may repeat this process and generate a new set of initial approximations. We expect that some of these structurally different local minima improve the function value, and successive iterations of these steps converge to the global optimum. Since the number of local minima may be large, we need a criterion to determine which optima should be explored.

### 5.4.3 Exploration graph

We heuristically determine which local minima are worth exploring by classifying them in terms of their function values and the number of dislocations they have undergone. Given a local minimum of the Chen-Babuska functional $\beta$, we apply the

nodal dislocation technique to every representative of the nodal distribution and generate a family of nodal sets, see Sect. 5.4.2. These nodal configurations are used as initial approximations for the minimization procedure to obtain a local minimum of the function $\beta$ as detailed in Sect. 5.4.1. Abstractly, the operation that consists in dislocating a representative of an optimum and optimizing the resulting nodal distribution connects two (different) minima of the functional $\beta$. Therefore, we propose to represent local minima as vertices of a directed graph, and explore different local minima by visiting different vertices of the graph.

Specifically, we identify a local minimum of $\beta$ with a vertex of a directed graph, and build an edge from vertex $v$ to vertex $w$, $v \to w$, if two conditions hold. First, $w$ is the minimum attained when optimizing the functional $\beta$ with initial approximation given as a dislocation of a representative of $v$. Second, $\beta(w) < \beta(v)$. Note that, in contrast to standard graph search algorithms, the graph is unknown in advance since the vertices and edges of the graph are generated while we dislocate nodes and optimize distributions. Moreover, all the children of vertex $v$ correspond to local optima of $\beta$ obtained after an optimization of a nodal dislocation of $v$ and with a lower function value than $\beta(v)$. Consequently, the function value is reduced at each offspring and, in particular, the leaves of the graph correspond to nodal distributions with the smallest function values.

To determine the exploration order of the graph vertices, we sort them in terms of their function value and their position in the graph. Let us define the depth $d$ of a vertex as $d = d' + 1$, where $d'$ is the depth of its ancestor, and set the root of the tree to have depth 0. Next, we select which vertices to explore similar to how potentially optimal hyperrectangles are chosen in the DiRect algorithm (Jones et al., 1993). More precisely, we represent each unexplored vertex of the graph as a two-dimensional point with the first component given by its depth and the second component given by the function value at such minimum. Following, we consider the convex hull of this point cloud. Then, the points that compose the poly-segment of the lower boundary determine the next minima to be explored, see Fig. 5.7. With this choice we balance minima with low function values and unexplored optima. In particular, the minimum with the lowest function value will always be explored in the next iteration. We highlight that to obtain the lower boundary of the convex hull it is enough to consider the points with minimum value, which can be efficiently accomplished if, for each depth, we store the vertices in a priority queue sorted by

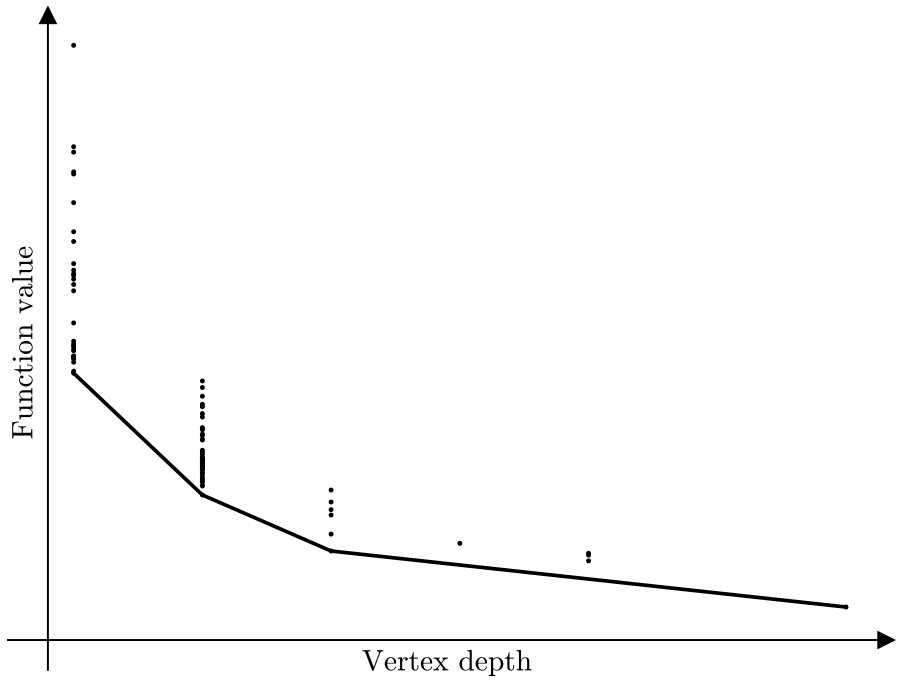



Figure 5.7: Two-dimensional representation of the vertices of the graph at a specific iteration of the algorithm. The dots conforming the lower boundary of the convex hull correspond to the vertices to be explored in the next iteration.

the function value.

### 5.4.4 The algorithm

In Algorithm 5.1, we detail the complete method for exploring local minima of the function $\beta$ based on nodal dislocations and smooth optimization. Fixed the dimension and the polynomial degree, the input data is the function to optimize $\beta$, and the initial approximation $\boldsymbol{Z}_0$. In our case, $\beta$ is the Chen-Babuska functional defined in Eq. (5.2), and the initial approximation is the Buss-Fillmore nodal distribution (Buss and Fillmore, 2001). First, in Line 2, we compute the corresponding degrees of freedom of the initial nodal distribution $\boldsymbol{y}_0$. Then, we initialize the graph structure and optimize the initial approximation, Lines 3 and 4. The optimized initial approximation is denoted as $\boldsymbol{y}_0^\star$ and corresponds to the root of the directed graph. Internally, the vertex structure stores the degrees of freedom of the optimized nodal set, the depth of the vertex, the function value at the minimum, and the parent and children information. The function `CreateVertex` creates an instance of this vertex

**Algorithm 5.1** Exploring local minima of a function.

**Input:** Function $\beta$, Initial approximation $\boldsymbol{Z}_0$
**Output:** Graph $\mathcal{G}$

1: **function** `ExploreFunctionMinima`($\beta$, $\boldsymbol{Z}_0$)
2: $\boldsymbol{y}_0 \leftarrow$ `GetVectorOfDOFs`($\boldsymbol{Z}_0$)
3: $\mathcal{G} \leftarrow$ `InitializeGraph`
4: $\boldsymbol{y}_0^\star \leftarrow$ `Optimize`($\beta$, $\boldsymbol{y}_0$)
5: `CreateVertex`($\mathcal{G}$, $\boldsymbol{y}_0^\star$, 0, $\beta(\boldsymbol{y}_0^\star)$)
6: **while** there are unexplored vertices **do**
7: $\{\boldsymbol{t}_1^\star, \ldots, \boldsymbol{t}_k^\star\} \leftarrow$ `GetDistributionsToExplore`($\mathcal{G}$)
8: $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l\} \leftarrow$ `GenerateNodalDislocations`($\{\boldsymbol{t}_1^\star, \ldots, \boldsymbol{t}_k^\star\}$)
9: $\{\boldsymbol{y}_1^\star, \ldots, \boldsymbol{y}_l^\star\} \leftarrow$ `Optimize`($\beta$, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l\}$)
10: `UpdateGraph`($\mathcal{G}$, $\{\boldsymbol{y}_1^\star, \ldots, \boldsymbol{y}_l^\star\}$)
11: `MarkAsExplored`($\{\boldsymbol{t}_1^\star, \ldots, \boldsymbol{t}_k^\star\}$)
12: **end while**
13: **return** $\mathcal{G}$
14: **end function**

structure and fills in the fields corresponding to the degrees of freedom, depth, and function values with the arguments provided. For the root vertex, the corresponding degrees of freedom are $\boldsymbol{y}_0^\star$, the depth is 0, and the function value is given by $\beta(\boldsymbol{y}_0^\star)$, Line 5. The root vertex has no parent nor children so far, yet this information is to be filled in later.

Following, we have the main loop of the algorithm. This loop is repeated until all the vertices of the graph have been explored, Line 6, and is composed of five steps. First, in Line 7, we get the distributions $\{\boldsymbol{t}_1^\star, \ldots, \boldsymbol{t}_k^\star\}$ to be explored in the current iteration. We choose these distributions in terms of their depth and function value, as described in Sect. 5.4.3. Second, in Line 8 and following the method detailed in Sect. 5.4.2, we generate the nodal sets $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l\}$ by dislocating each representative of each nodal distribution $\boldsymbol{t}_i^\star$, $i = 1, \ldots, k$. Third, in Line 9, we optimize each initial approximation $\boldsymbol{y}_j$ to a local minima $\boldsymbol{y}_j^\star$, $j = 1, \ldots l$. Fourth, we update the graph structure, Line 10. Lastly, in the fifth step, we mark the vertices $\{\boldsymbol{t}_1^\star, \ldots, \boldsymbol{t}_k^\star\}$ as explored to ensure they will not be explored again in the future, Line 11. The output of the algorithm is the graph structure.

The fourth step in Algorithm 5.1 corresponds to updating the graph structure. That is, we check whether the new found minimum already exists in the graph structure, create this new vertex if needed, and build an edge between the parent and the

**Algorithm 5.2** Updating the graph structure.

**Input:** Graph $\mathcal{G}$, Nodal sets $\{\boldsymbol{y}_1^\star, \ldots, \boldsymbol{y}_l^\star\}$

```
1: function UpdateGraph(G, {y_1*, ..., y_l*})
2:   for each y_j* in {y_1*, ..., y_l*} do
3:     if β(y_j*) < β(Parent(y_j*)) then
4:       if y_j* ∉ G then
5:         CreateVertex(G, y_j*, Depth(Parent(y_j*)) + 1, β(y_j*))
6:       end if
7:       BuildEdge(G, Parent(y_j*), y_j*)
8:     end if
9:   end for
10: end function
```

children. This process is detailed in Algorithm 5.2. Specifically, for each distribution $\boldsymbol{y}_j^\star$, we first check if the value of $\beta$ evaluated at the child is smaller than the value of $\beta$ evaluated at the parent, Line 3. If the child is better than the parent, we should create an edge between these vertices. However, we should first check if the child was already found previously. Graph vertices are stored in a hash table, with the hash value determined by a random number generated with a seed given by the first four significant digits of the function value at the minimum. Thus, to check if a vertex is in the graph, we simply look it up in this hash table. Now, if the nodal distribution was not already found, we create a new vertex calling the function `CreateVertex`, Line 5. Note that the depth of the child is one unit greater than the depth of its parent. Finally, we connect the parent with its (new) child with a directed edge, Line 7.

In Fig. 5.8, we show a representation of the graph for dimension 2 and polynomial degree 12, where vertices are labeled with their function value. We can observe that the values become smaller as we go down in the lineage. In particular, the leaves of the graph are those nodal distributions which are better than their ancestors and do not have any children improving the function value. We also stress that a child may have more than one parent, that is, a minimum of the functional $\beta$ might be found from the nodal dislocation of two different minima.

As regards the implementation, since each optimization run is independent, we distribute this operation in several CPUs. If the number of available CPUs is larger than the number of distributions to explore, we repeat the process in Line 7 until the desired load of minimization runs per CPU is achieved. We remark that this

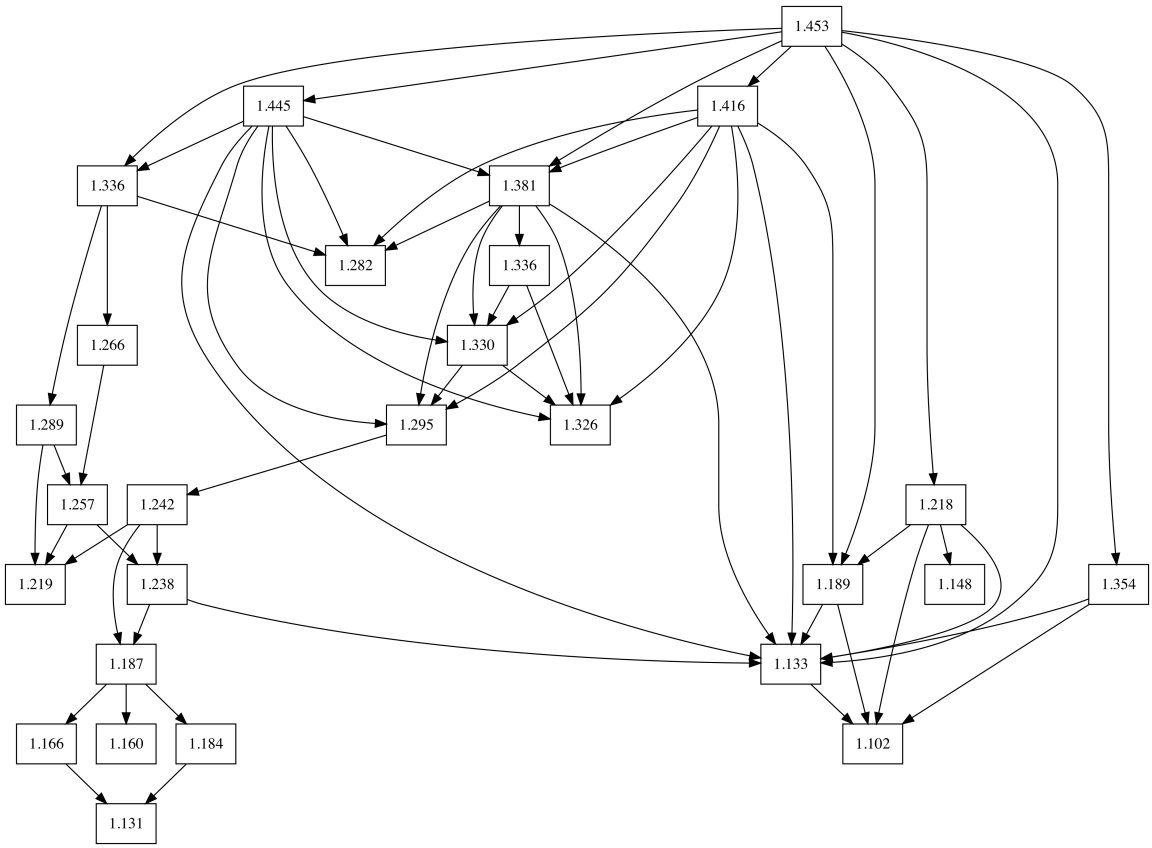



Figure 5.8: Graph of the exploration procedure for dimension 2 and polynomial degree 12. Each box corresponds to a local minima labeled with its function value.

exploration order allows the minima with lower function values to be explored first, and even though all the vertices of the graph will be eventually explored, this ordering is particularly useful when the computational resources are only available during a limited period of time, since we expect to obtain the best results within this time limitation.

## 5.5 Numerical results

In this section, we report the main properties of the nodal configurations found by the method presented in this work. All the algorithms are coded in Julia (Bezanson et al., 2017) and some parts feature a parallel implementation aiming to exploit the resources of the MareNostrum4 supercomputer located at the Barcelona Supercomputing Center. Even though the proposed methods do not present a dimensional

| Dislocations | Condition | # local minima | $\min \beta$ | $\min \Lambda$ |
|---|---|---|---|---|
| Ring | Descending | 2 | 1.106 | 5.918 |
| Ring | Non-descending | 144 | 1.106 | 5.918 |
| All | Descending | 2 | 1.106 | 5.918 |
| All | Non-descending | 221 | 1.106 | 5.918 |

Table 5.1: Number of local minima, minimum function value, and minimum Lebesgue constant found for the different versions of the method for polynomial degree 9 in the triangle, $d = 2$.

| | Ring and descending | | | All and descending | | |
|---|---|---|---|---|---|---|
| Degree | # local minima | $\min \beta$ | $\min \Lambda$ | # local minima | $\min \beta$ | $\min \Lambda$ |
| 10 | 5 | 1.027 | 7.085 | 9 | 1.027 | 7.085 |
| 11 | 6 | 1.069 | 7.266 | 16 | 1.069 | 7.266 |
| 12 | 27 | 1.102 | 8.669 | 57 | 1.102 | 8.669 |
| 13 | 72 | 1.094 | 8.877 | 267 | 1.094 | 8.877 |
| 14 | 439 | 1.088 | 8.988 | 840 | 1.088 | 8.988 |
| 15 | 1 242 | 1.112 | 10.306 | 2 443 | 1.112 | 10.306 |

Table 5.2: Number of local minima, minimum function value, and minimum Lebesgue constant found for the different versions of the method in terms of the polynomial degree in the triangle, $d = 2$.

limitation, the computational time is a practical barrier. Consequently, the reported values were obtained after using 2400 CPUs for at most 48 hours. We remark that in 2D and for low polynomial degrees in 3D and 4D, all the graph vertices were explored. In contrast, for high polynomial degrees in 3D and 4D, there are still graph vertices to explore once the time limitation is reached. Nevertheless, since graph vertices are explored by priority, see Sect. 5.4.3, we obtain nodal distributions with improved interpolation properties.

### 5.5.1 Method variations

In Sect. 5.4, we propose a method to explore different minima of a smooth functional. This heuristic method features local nodal dislocations, and it always explores the graph vertices in a decreasing fashion. In the following example, we illustrate the influence of these decisions on the optima found.

More precisely, in Algorithm 5.1 of Sect. 5.4, we propose to dislocate a node to

explore different mesh connectivities. The new locations are determined by the projection onto the sphere of the centroid of the elements in the ring around a given node, see Sect. 5.4.2. Alternatively, we can generate structurally different nodal sets by dislocating a node to the projection onto the sphere of the centroid of all the elements in the Delaunay mesh, not only those in the ring. With this alternative approach, we explore more mesh connectivities. Accordingly, we expect to find smaller local minima since we are enforcing crossing more energetic barriers, not only the ones with highest influence. Unfortunately, increasing the number of distributions to optimize increases the computational cost.

Regarding the graph exploration, we represent each function minima by a vertex of a directed graph and build an edge connecting a parent and its child if the function value decreases, see Sect. 5.4.3. As an alternative, we can consider the graph obtained when we do not impose the decrease condition. Under this criterion, we explore more possibilities since the graph has more vertices, and we may end up finding a better local minimum. On the downside, exploring more local minima is more computationally expensive.

In Table 5.1, we compare the four possible combinations arising from this discussion in the exploration of quasi-optimal nodal distributions of polynomial degree 9 in the triangle. We list the number of local minima found, the minimum function value, and the minimum Lebesgue constant. As expected, exploring worse children leads to an increase in the number of optima found, but these new optima do not feature better interpolation properties. Indeed, the same best local minimum and the same distribution with the lowest Lebesgue constant are found in all four cases.

To illustrate the influence of dislocating a node to the elements in the ring or to all mesh elements, we explore the minima for polynomial degrees from 10 to 15, see Table 5.2. We observe that, even though the number of local minima is larger when we dislocate a node to all mesh elements, the same best local minimum and the same distribution with the lowest Lebesgue constant are found with both methods. We believe that this is so because the initial approximation, the Buss-Fillmore distribution, is close to the quasi-optimal nodal set and local dislocations are enough to find the basin which contains the optimum. Furthermore, if such minimum is obtained after a single global nodal dislocation, we expect that it will also be reached after successive local nodal dislocations.

We conclude that none of the alternatives studied presents a clear advantage

with respect to the original method presented in Sect. 5.4 and they mainly increase the computational cost without an impact on the best nodal distributions found. Definitely, starting from a Buss-Fillmore distribution, it is sufficient to consider local nodal dislocations and children improving the function value.

### 5.5.2 Quasi-optimal Chen-Babuska points in 2D

The method presented in Sect. 5.4 explores different local minima of the Chen-Babuska functional. These minima correspond to nodal distributions with an associated function value $\beta$ and Lebesgue constant $\Lambda$. Since the Chen-Babuska functional is a proxy of the Lebesgue constant, the global optimum of Chen-Babuska may have a greater Lebesgue constant than another local minimum. For instance, for polynomial degree 9, we find two local minima, see Table 5.1. One minimum is at value $\beta = 1.106$ and has Lebesgue constant $\Lambda = 7.573$, while the other minimum is attained at value $\beta = 1.123$ and has Lebesgue constant $\Lambda = 5.918$. Therefore, the nodal configuration with the smallest function value may not correspond to the best interpolation set in terms of the Lebesgue constant. Furthermore, we are also interested in the condition number of the Vandermonde matrix and the interpolation error for several benchmark functions. Following, we study these quantities of interest for our nodal distributions in the triangle and compare the values against the nodal sets in the literature.

#### Chen-Babuska functional and Lebesgue constant

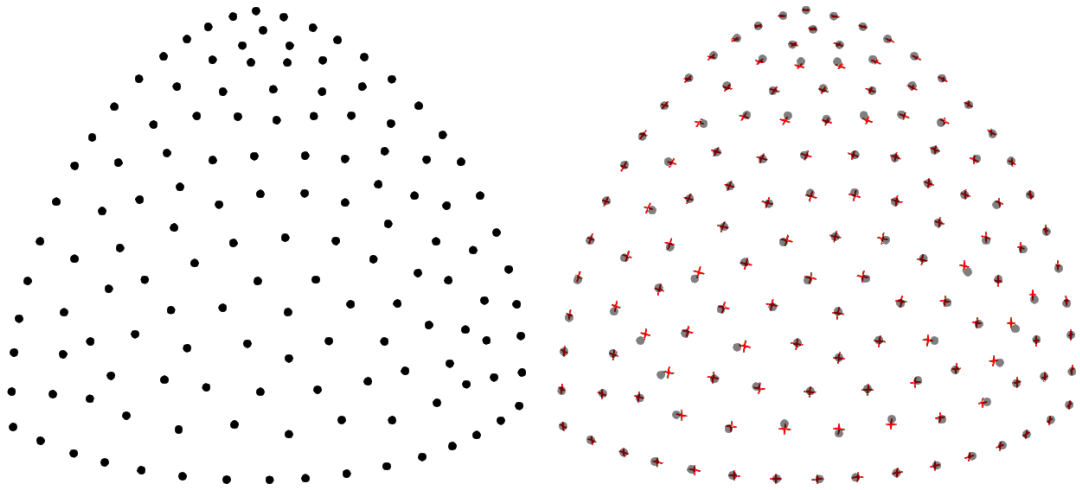
In Table 5.3, up to polynomial degree 20, we report the minimum function value $\beta$ and the minimum Lebesgue constant $\Lambda$ found with our method. We also list these values for the nodal sets computed in Chen and Babuška (1995), where the functional $\beta$ is optimized. We observe that up to polynomial degree 8 we obtain the same local minima for the functional $\beta$, yet we find lower function values for higher polynomial degrees. Similarly, up to polynomial degree 10, our values for the Lebesgue constant coincide with those from Chen and Babuška (1995), yet we find nodal distributions with smaller Lebesgue constant for higher polynomial degrees. Interestingly, some of these optimal nodal distributions present a structured configuration. Our quasi-optimal Lebesgue nodal sets of polynomial degree $p$, $p \leq 10$, present a lattice-like structure in the triangle and, hence, we believe that the initial approximation in the optimization procedure used in Chen and Babuška (1995) was the equispaced

| | Optimizing $\beta$ Chen and Babuška (1995) | | | Optimizing $\Lambda$ Roth (2005) | | Optimizing $\beta$ with our method | | |
|---|---|---|---|---|---|---|---|---|
| $p$ | $\beta$ | $\Lambda$ | $\Lambda^{1/p}$ | $\Lambda$ | $\Lambda^{1/p}$ | $\min \beta$ | $\min \Lambda$ | $\min \Lambda^{1/p}$ |
| 3 | 0.740 | 2.111 | 1.283 | 2.108 | 1.282 | 0.740 | 2.112 | 1.283 |
| 4 | 0.820 | 2.692 | 1.281 | 2.587 | 1.268 | 0.820 | 2.692 | 1.281 |
| 5 | 0.884 | 3.301 | 1.270 | 3.081 | 1.252 | 0.884 | 3.301 | 1.270 |
| 6 | 0.941 | 3.791 | 1.249 | 3.595 | 1.238 | 0.941 | 3.791 | 1.249 |
| 7 | 0.996 | 4.391 | 1.235 | 4.143 | 1.225 | 0.996 | 4.391 | 1.235 |
| 8 | 1.055 | 5.089 | 1.226 | 4.766 | 1.216 | 1.055 | 5.089 | 1.226 |
| 9 | 1.123 | 5.918 | 1.218 | 5.486 | 1.208 | 1.106 | 5.918 | 1.218 |
| 10 | 1.205 | 7.085 | 1.216 | 5.921 | 1.195 | 1.027 | 7.085 | 1.216 |
| 11 | 1.311 | 8.338 | 1.213 | 6.724 | 1.189 | 1.069 | 7.266 | 1.198 |
| 12 | 1.453 | 10.082 | 1.212 | 7.187 | 1.179 | 1.102 | 8.660 | 1.197 |
| 13 | 1.651 | 12.046 | 1.211 | 7.253 | 1.165 | 1.094 | 8.877 | 1.183 |
| 14 | | | | 7.706 | 1.157 | 1.088 | 8.988 | 1.170 |
| 15 | | | | 8.243 | 1.151 | 1.112 | 10.306 | 1.168 |
| 16 | | | | | | 1.134 | 10.970 | 1.161 |
| 17 | | | | | | 1.148 | 14.026 | 1.168 |
| 18 | | | | | | 1.138 | 12.344 | 1.150 |
| 19 | | | | | | 1.143 | 11.797 | 1.139 |
| 20 | | | | | | 1.162 | 12.861 | 1.136 |

Table 5.3: Minimum function value, Lebesgue constant and its convergence for our nodal sets in the triangle for polynomial degree $p$, $p = 3, \ldots, 20$. We also report the values from Chen and Babuška (1995) and Roth (2005). Blank spaces indicate not reported results.

distribution in the triangle. In contrast, for degrees $p \geq 11$, our distributions do not present a lattice-like pattern and we obtain smaller Lebesgue constants. These results agree with the belief that the optimal Lebesgue nodal sets feature a lattice-like structure until polynomial degree 9, yet there exist better Lebesgue configurations that do not exhibit this structure for higher polynomial degrees (Roth, 2005).

In Table 5.3, we also list the estimation of the Lebesgue constant for the quasi-optimal symmetrical nodal sets computed in Roth (2005). Our quasi-optimal nodal distributions feature larger Lebesgue constant than the ones found in Roth (2005). This fact is not surprising since our method explores local minima of the Chen-Babuska functional, which is a proxy of the Lebesgue constant, and the nodal sets found in Roth (2005) are computed optimizing the Lebesgue constant instead. Up to polynomial degree 9, the values suggest that these nodal sets feature a lattice-like



(a) (b)

Figure 5.9: Comparison of close to optimal nodal distributions of polynomial degree 15 in the triangle depicted on the sphere. (a) Black spheres indicate the points positions of the nodal distribution from Roth (2005). (b) Our distribution is depicted with red crosses, and the nodal set from Roth (2005) with semi-transparent black spheres.

structure, while for higher polynomial degrees it seems they exhibit an unstructured disposition. Unfortunately, the points coordinates of the nodal sets are not available and only a figure of the nodal distribution of polynomial degree $p = 15$ is shown, see Fig. 5.9(a). Remarkably, we have found a local minimum of the Chen-Babuska functional which is topologically close to this nodal distribution, see Fig. 5.9(b). This minimum has a Lebesgue constant of $\Lambda = 10.422$ and a Chen-Babuska value of $\beta = 1.112$. Thus, even though the Lebesgue constant of this nodal set is larger than the best value found, we believe that our local minima could be used as initial approximations for a method that directly optimizes the Lebesgue constant to obtain better point configurations.

It is also worth studying the series $\Lambda^{1/p}$ as the polynomial degree increases since its convergence to one is a sufficient condition to ensure uniform convergence, see Sect. 5.2. In Table 5.3, we show these values for our quasi-optimal nodal distributions and the nodal sets computed in Chen and Babuška (1995) and Roth (2005). We observe that the sequence $\Lambda^{1/p}$ is monotonically decreasing except between polynomial degrees 16 and 17, yet the three series have a general trend to decrease.

| | Condition number $\kappa$ | | | |
|---|---|---|---|---|
| $p$ | Equispaced | Warburton (2006) | Rapetti et al. (2012) | Our method |
| 3 | 5.828 | 5.903 | 5.903 | 5.906 |
| 4 | 7.599 | 6.777 | 7.064 | 6.915 |
| 5 | 10.167 | 7.845 | 8.242 | 8.129 |
| 6 | 14.658 | 9.591 | 9.651 | 9.639 |
| 7 | 22.239 | 11.160 | 11.642 | 11.170 |
| 8 | 35.627 | 13.886 | 14.599 | 13.869 |
| 9 | 59.949 | 16.896 | 17.184 | 16.847 |
| 10 | 104.164 | 21.670 | 22.059 | 21.376 |
| 11 | 186.545 | 27.401 | 31.751 | 18.965 |
| 12 | 344.977 | 36.132 | 24.137 | 24.264 |
| 13 | 626.693 | 47.219 | 31.071 | 31.882 |
| 14 | 1 198.655 | 63.679 | 46.113 | 27.075 |
| 15 | 2 194.382 | 85.692 | 30.931 | 32.565 |
| 16 | 4 284.058 | 117.712 | 44.424 | 35.659 |
| 17 | 7 879.557 | 166.716 | 50.798 | 35.043 |
| 18 | 15 597.334 | 247.913 | 55.253 | 41.240 |
| 19 | 28 798.839 | 344.617 | | 41.738 |
| 20 | 57 534.253 | 476.785 | | 42.792 |

Table 5.4: Condition number $\kappa$ of the Vandermonde matrix using different interpolative nodal sets in the triangle for polynomial degree $p$, $p = 3, \ldots, 20$. We compare the values for the equispaced distribution, the values from Warburton (2006) and Rapetti et al. (2012), and our nodal sets. Blank spaces indicate not reported results.

### Condition number of the Vandermonde matrix

To construct an interpolating polynomial, it is standard to evaluate the Lagrange interpolating basis by solving a linear system where the system matrix corresponds to the Vandermonde matrix in terms of an orthogonal basis. Thus, to indirectly measure how the nodal distribution affects the accuracy of an interpolation, we compare the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis (Koornwinder, 1975; Dubiner, 1991; Kirby, 2010). In Table 5.4, we report the condition number when evaluating this basis at the equispaced distribution and at our quasi-optimal nodal distribution. Since the nodal positions of the distribution in Roth (2005) are not available, we also list the condition number for the nodal sets in Warburton (2006) and Rapetti et al. (2012). We see that the values rapidly increase for the equispaced distribution. Nevertheless, when evaluating with

all the other nodal distributions, up to polynomial degree 15, we should expect to lose less than three digits of accuracy because all the condition numbers are smaller than 100.

### Interpolation error of benchmark functions

To further assess the interpolation properties of our nodal sets, we estimate the error of interpolating several benchmark functions. The first two benchmark functions, expressed in barycentric coordinates $\boldsymbol{\lambda} = (\lambda^i)_{i=1,\dots,d+1}$, with $\lambda^i \geq 0$, and $\sum_{i=1}^{d+1} \lambda^i = 1$, are

$$\begin{aligned} u_1(\boldsymbol{\lambda}) &= \left(\exp\left(-2\lambda^1\right) - 1\right) \prod_{i=2}^{d+1} \left(2\lambda^i\right), \\ u_2(\boldsymbol{\lambda}) &= \left(\cosh\left(-2\lambda^1\right) - 1\right) \prod_{i=2}^{d+1} \left(2\lambda^i\right). \end{aligned} \tag{5.4}$$

These functions are particularly relevant since they are devised to allow equispaced points to achieve uniform convergence. That is, they are the multi-dimensional version of a 1D analytic function that has the $n$-th derivative bounded by $Cn$, where $C$ is a constant. Hence, the upper bound of the interpolation error tends to zero as the polynomial degree increases (Davis, 1975). The third benchmark function is an extension of the Runge function to several dimensions, namely

$$u_3(\boldsymbol{\lambda}) = \frac{1}{1 + 2\alpha \left\| \boldsymbol{\lambda} - \frac{1}{d+1}\mathbf{1} \right\|_2^2}, \tag{5.5}$$

with $\alpha = 25$, and $\mathbf{1}$ the vector with all its components equal to one.

In Fig. 5.10, we plot the interpolation error $\|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i\|_\infty / \|u_i\|_\infty$, $i = 1, 2, 3$, in terms of the polynomial degree when interpolating these functions using equispaced points, warp-and-blend nodes (Warburton, 2006) and our quasi-optimal nodal distributions. To estimate these errors, we apply the adaptive sampling method proposed in Chapter 4 using as objective function $|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i|$. Indeed, the interpolations using either nodal sets converge to the functions $u_1$ and $u_2$, see Fig. 5.10(a) and (b). Up to degree 14, the three interpolation errors decrease as expected. From degree 15 to degree 20, the interpolation errors stop the previous decreasing trend because machine accuracy is reached when estimating the error. For the Runge function $u_3$, we observe in Fig. 5.10(c) that the interpolation error using equispaced points diverges, while with the warp-and-blend or our quasi-optimal nodal configurations the errors seem to converge.

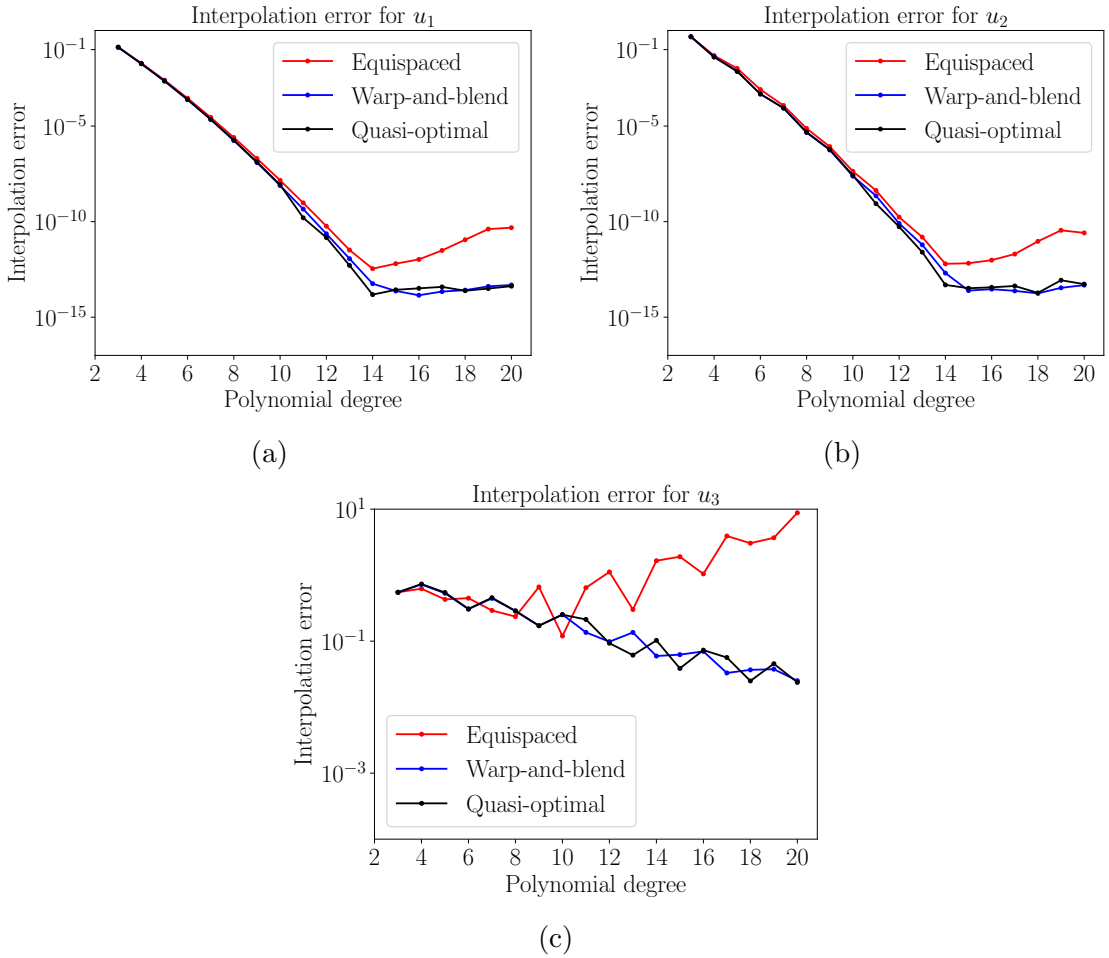



Figure 5.10: Interpolation error in the triangle for different polynomial degrees using equispaced (red), warp-and-blend (blue), and our quasi-optimal nodal distributions (black) for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

### 5.5.3 Quasi-optimal Chen-Babuska points in 3D

Analogously to the two-dimensional case, we repeat the same study for nodal distributions in the tetrahedron. We highlight that, up to polynomial degree 9, we explore all the graph vertices using the available computational resources in less than 48 hours. For higher polynomial degrees, there are still graph vertices to explore once the time limit is reached. In either case, since we are prioritizing which graph vertices should be explored first, see Sect. 5.4.3, we obtain nodal distributions with improved interpolation properties.

| | Optimizing $\beta$ Chen and Babuška (1996) | | | Explicit nodal sets Isaac (2020) | | Optimizing $\beta$ with our method | | |
|---|---|---|---|---|---|---|---|---|
| $p$ | $\beta$ | $\Lambda$ | $\Lambda^{1/p}$ | $\Lambda$ | $\Lambda^{1/p}$ | $\min \beta$ | $\min \Lambda$ | $\min \Lambda^{1/p}$ |
| 3 | 0.644 | 2.934 | 1.432 | 2.933 | 1.431 | 0.643 | 2.936 | 1.432 |
| 4 | 0.767 | 4.112 | 1.424 | 4.093 | 1.422 | 0.767 | 4.134 | 1.426 |
| 5 | 0.889 | 5.616 | 1.412 | 5.547 | 1.409 | 0.889 | 5.644 | 1.414 |
| 6 | 1.024 | 7.363 | 1.395 | 7.169 | 1.389 | 1.022 | 7.310 | 1.393 |
| 7 | 1.190 | 9.366 | 1.377 | 9.202 | 1.373 | 1.129 | 8.897 | 1.366 |
| 8 | 1.417 | 12.311 | 1.369 | 12.067 | 1.365 | 1.342 | 12.010 | 1.364 |
| 9 | 1.753 | 15.686 | 1.358 | 15.593 | 1.357 | 1.328 | 15.146 | 1.353 |
| 10 | | | | 20.623 | 1.353 | $1.381^\star$ | $15.611^\star$ | $1.316^\star$ |
| 11 | | | | 28.034 | 1.354 | $1.348^\star$ | $19.425^\star$ | $1.310^\star$ |
| 12 | | | | 38.649 | 1.356 | $1.423^\star$ | $23.569^\star$ | $1.301^\star$ |
| 13 | | | | 55.143 | 1.361 | $1.462^\star$ | $25.701^\star$ | $1.284^\star$ |
| 14 | | | | 81.037 | 1.369 | $1.514^\star$ | $29.539^\star$ | $1.274^\star$ |
| 15 | | | | 118.420 | 1.375 | $1.551^\star$ | $36.375^\star$ | $1.271^\star$ |

Table 5.5: Minimum function value, Lebesgue constant and its convergence for our nodal sets in the tetrahedron for polynomial degree $p$, $p = 3, \dots, 15$. We also report the values from Chen and Babuška (1996) and Isaac (2020). Blank spaces indicate not reported results, and values with $\star$ indicate unfinished exploration of the graph for the available computational resources.

**Chen-Babuska functional and Lebesgue constant**

In Table 5.5, we list our results up to polynomial degree 15. Specifically, we list the minimum function value $\beta$, the Lebesgue constant $\Lambda$ and the value $\Lambda^{1/p}$. We also report the results from Chen and Babuška (1996) since the same functional $\beta$ is optimized. Up to polynomial degree 5, we obtain the same minimum function values. Moreover, the distribution that corresponds to the minimum function value coincides with the distribution featuring the minimum Lebesgue constant. Nevertheless, we attribute the largest values reported here to a more precise estimation of the Lebesgue constant. For higher polynomial degrees, we find nodal distributions with smaller minimum function values and Lebesgue constants. We also compare our nodal configurations with the explicit nodal sets computed in Isaac (2020). We observe that, for $p \leq 6$, the Lebesgue constant for our nodes is slightly larger. However, for higher polynomial degrees, we obtain nodal sets with smaller Lebesgue constants, and as we increase the polynomial degree, the difference between the two values increases. Finally, we observe that the sequence $\Lambda^{1/p}$ has a general trend to decrease.

| | Condition number $\kappa$ | | |
|---|---|---|---|
| $p$ | Equispaced | Isaac (2020) | Our method |
| 3 | 10.248 | 10.505 | 10.537 |
| 4 | 15.392 | 15.817 | 15.878 |
| 5 | 20.245 | 18.766 | 18.763 |
| 6 | 32.997 | 25.851 | 25.803 |
| 7 | 52.301 | 37.339 | 35.490 |
| 8 | 91.944 | 55.905 | 54.849 |
| 9 | 162.216 | 86.315 | 82.432 |
| 10 | 300.740 | 137.967 | 79.217 |
| 11 | 558.909 | 224.111 | 76.962 |
| 12 | 1 064.722 | 371.800 | 101.923 |
| 13 | 2 027.690 | 621.826 | 122.222 |
| 14 | 3 917.567 | 1 052.205 | 142.834 |
| 15 | 7 560.901 | 1 785.689 | 167.841 |

Table 5.6: Condition number $\kappa$ of the Vandermonde matrix using different interpolative nodal sets in the tetrahedron for polynomial degree $p$, $p = 3, \ldots, 15$. We compare the values for the equispaced distribution, the values from Isaac (2020), and our nodal sets.

### Condition number of the Vandermonde matrix

In Table 5.6, we compare the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis evaluated at our nodal distributions, the nodes from Isaac (2020), and the equispaced nodal set. As in the two-dimensional case, we see that the values rapidly increase for the equispaced distribution and a similar trend is observed for the Isaac (2020) distribution. In contrast, when using our nodal distributions, the growing ratio is smaller and we should expect to lose less than four digits of accuracy.

### Interpolation error of benchmark functions

Regarding the interpolation error, we use the same functions $u_i$, $i = 1, \ldots, 3$, stated in Eq. (5.4) and Eq. (5.5). In Fig. 5.11, we plot $\|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i\|_\infty / \|u_i\|_\infty$ in terms of the polynomial degree when using equispaced points, the nodal sets from Isaac (2020) and our quasi-optimal nodal distributions. Indeed, the interpolations using either nodal sets converge to the functions $u_1$ and $u_2$, see Fig. 5.11(a) and (b). Up to degree 14, the three interpolation errors decrease as expected, but for polynomial degree 15

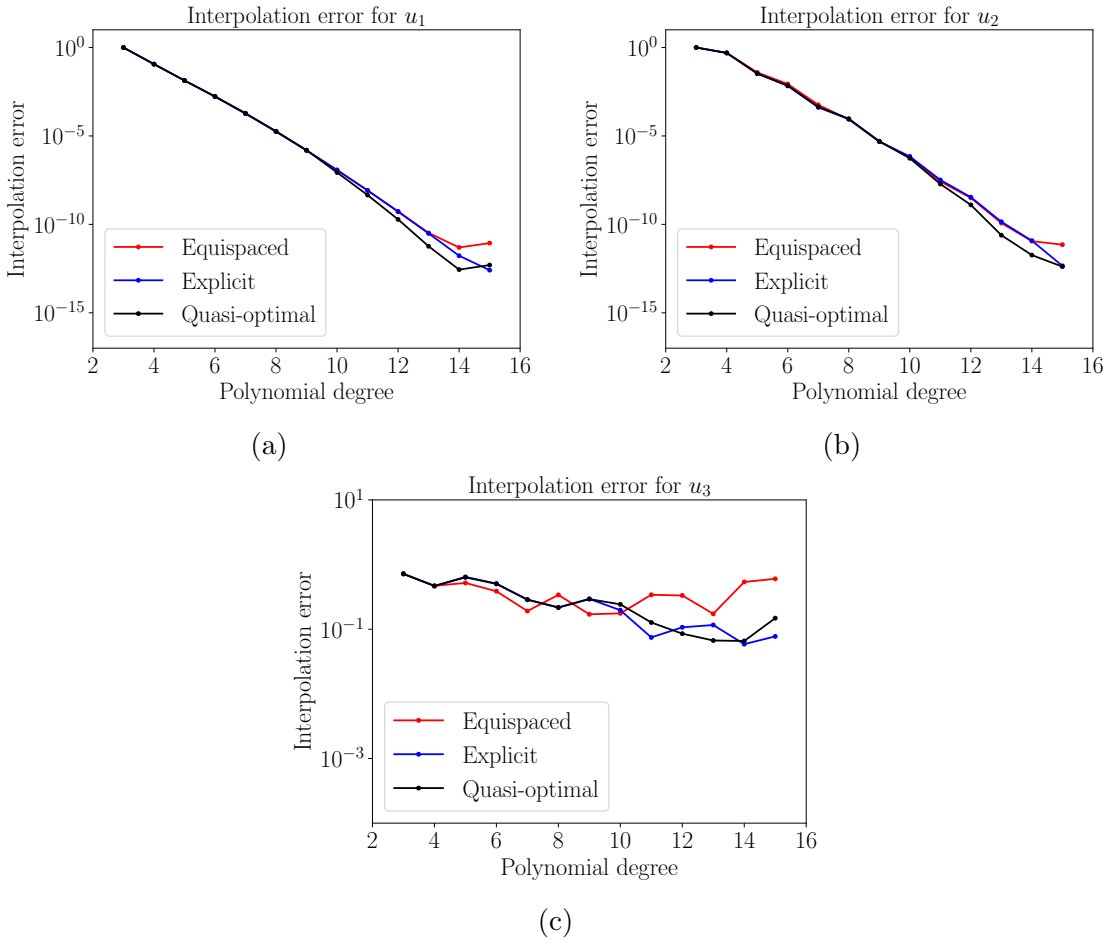



Figure 5.11: Interpolation error in the tetrahedron for different polynomial degrees using equispaced (red), the explicit nodal sets from Isaac (2020) (blue), and our quasi-optimal nodal distributions (black) for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

we reach the machine accuracy in the error estimation and the decreasing trend is lost. For the Runge function $u_3$, we observe in Fig. 5.11(c) that the interpolation error using equispaced points diverges. When using the explicit nodal sets from Isaac (2020) or our quasi-optimal nodal configurations, the errors are smaller and tend to be reduced as the polynomial degree increases.

### 5.5.4 Quasi-optimal Chen-Babuska points in 4D

For dimension $d = 4$, we are able to compute quasi-optimal nodal distributions up to moderate polynomial degree. Up to degree 8, we are able to explore all the graph vertices using the available computational resources, while up to degree 10, our im-

| | Equispaced | | Explicit nodal sets from Isaac (2020) | | Optimizing $\beta$ with our method | | |
|---|---|---|---|---|---|---|---|
| $p$ | $\Lambda$ | $\Lambda^{1/p}$ | $\Lambda$ | $\Lambda^{1/p}$ | $\min \beta$ | $\min \Lambda$ | $\min \Lambda^{1/p}$ |
| 3 | 3.880 | 1.571 | 4.200 | 1.613 | 0.565 | 4.169 | 1.609 |
| 4 | 6.244 | 1.581 | 6.147 | 1.575 | 0.719 | 6.130 | 1.573 |
| 5 | 10.918 | 1.613 | 8.839 | 1.546 | 0.898 | 8.865 | 1.547 |
| 6 | 19.224 | 1.637 | 12.502 | 1.523 | 1.120 | 12.532 | 1.524 |
| 7 | 34.085 | 1.656 | 17.231 | 1.502 | 1.354 | 17.085 | 1.500 |
| 8 | 60.859 | 1.671 | 23.366 | 1.483 | 1.652 | 22.310 | 1.474 |
| 9 | 109.427 | 1.685 | 32.646 | 1.473 | $1.739^\star$ | $31.328^\star$ | $1.472^\star$ |
| 10 | 198.083 | 1.697 | 45.768 | 1.466 | $1.913^\star$ | $40.873^\star$ | $1.449^\star$ |

Table 5.7: Minimum function value, Lebesgue constant and its convergence for our nodal sets in the pentatope for polynomial degree $p$, $p = 3, \dots, 10$. Values with $\star$ indicate unfinished exploration of the graph for the available computational resources. We also report the values of the equispaced distribution and the nodal sets from Isaac (2020).

plementation permits exploring the graph up to a certain depth only. For higher polynomial degrees, we compute all the dislocations of the nodal representatives of the initial approximation, but not all the optimization runs converge before the time limit of 48 hours is reached. Accordingly, we only analyze nodal distributions up to polynomial degree 10.

### Chen-Babuska functional and Lebesgue constant

In Table 5.7, we report the minimum function value $\beta$ and the minimum Lebesgue constant $\Lambda$ found with our method, as well as the values $\Lambda^{1/p}$, for $p = 3, \dots, 10$. We also list the Lebesgue constant of the explicit nodal distributions in the pentatope computed in Isaac (2020). These explicit nodal sets preserve the lattice-like structure, while our method explores structurally different configurations. We believe that the similar values of the Lebesgue constant might be due to the fact that quasi-optimal interpolation points present a grid structure for up to moderate polynomial degree. Indeed, we see a similar behavior in the triangle and the tetrahedron, see for instance Table 5.5, when it is not until polynomial degree 10 that we appreciate a significant difference between the Lebesgue constant of our nodal distribution and the explicit nodal set from Isaac (2020). Even so, our values correspond to minima of the Chen-Babuska functional, so there is still room for improvement if these configurations are

| | Condition number $\kappa$ | | |
|---|---|---|---|
| $p$ | Equispaced | Isaac (2020) | Our method |
| 3 | 16.341 | 17.328 | 17.222 |
| 4 | 30.487 | 32.801 | 32.365 |
| 5 | 43.949 | 44.850 | 43.593 |
| 6 | 73.139 | 65.410 | 63.335 |
| 7 | 125.524 | 108.158 | 101.359 |
| 8 | 229.769 | 194.218 | 144.363 |
| 9 | 428.923 | 358.026 | 239.176 |
| 10 | 823.800 | 681.022 | 415.212 |

Table 5.8: Condition number of the Vandermonde matrix using different interpolative nodal sets in the pentatope for polynomial degree $p$, $p = 3, \ldots, 10$. We compare the values for the equispaced distribution, the values from Isaac (2020), and our nodal sets.

used as initial approximations for optimizing the Lebesgue constant.

### Condition number of the Vandermonde matrix

In Table 5.8, we compare the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis evaluated at our nodal distributions, the nodes from Isaac (2020), and the equispaced nodal set. Although our nodal configurations feature slightly smaller values, the three nodal families present a similar growing trend, and we should expect to lose less than four digits of accuracy.

### Interpolation error of benchmark functions

Finally, in Fig. 5.12, we plot the interpolation errors of the functions $u_i$, $i = 1, \ldots, 3$, see Eq. (5.4) and Eq. (5.5), in terms of the polynomial degree when using equispaced points, the nodal sets from Isaac (2020) and our quasi-optimal nodal distributions. Even though we only have used polynomials up to degree 10, we observe a clear trend to decrease in the interpolation error of the functions $u_1$ and $u_2$, with the three nodal sets providing similar results, see Fig. 5.12(a) and (b). In the case of the Runge function $u_3$, see Fig. 5.12(c), for the three nodal families the curves are not monotone and the errors are large.

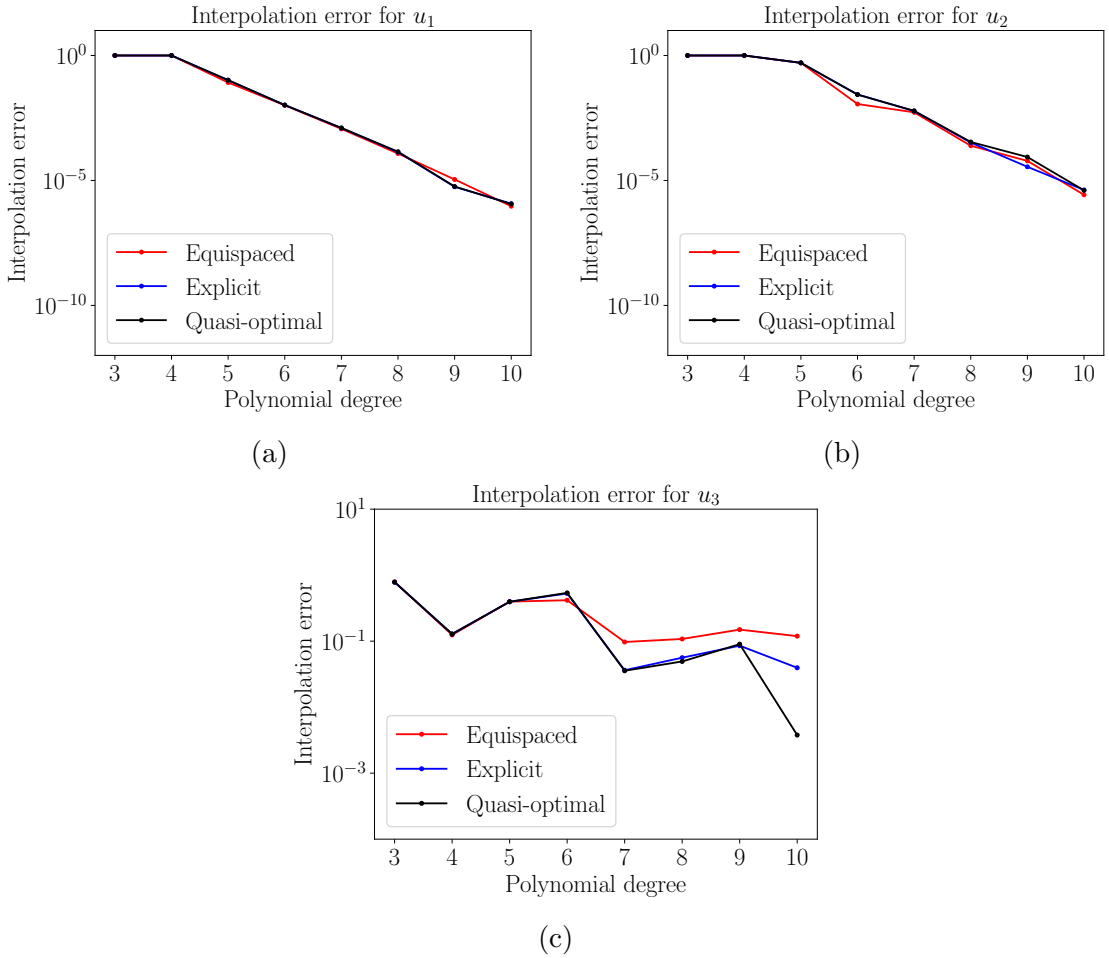



Figure 5.12: Interpolation error in the pentatope for different polynomial degrees using equispaced (red), the explicit nodal sets from Isaac (2020) (blue), and our quasi-optimal nodal distributions (black) for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

### 5.5.5 Quasi-optimal Fekete points in 2D

As detailed in Sect. 5.1, many proxies to the Lebesgue constant have been considered in the literature. In this work, we use the Chen-Babuska functional to optimize dislocated nodal configurations but any functional can be minimized. Next, we explore the local maximum of the Fekete functional $\mathcal{F}$, which aims to maximize the logarithm of the determinant of the Vandermonde matrix.

In Table 5.9, for polynomial degree $p$, $p = 3, \ldots, 20$, we list the maximum function value, and the minimum Lebesgue constant $\Lambda$ and $\Lambda^{1/p}$ of the quasi-optimal nodal distributions in the triangle obtained by optimizing the Fekete functional. We compare these values with the ones reported in Taylor et al. (2000) since the same

| | Optimizing $\mathcal{F}$ Taylor et al. (2000) | | Optimizing $\Lambda$ Roth (2005) | | Optimizing $\mathcal{F}$ with our method | | |
|---|---|---|---|---|---|---|---|
| $p$ | $\Lambda$ | $\Lambda^{1/p}$ | $\Lambda$ | $\Lambda^{1/p}$ | $\max \mathcal{F}$ | $\min \Lambda$ | $\min \Lambda^{1/p}$ |
| 3 | | | 2.108 | 1.282 | 12.727 | 2.112 | 1.283 |
| 4 | | | 2.587 | 1.268 | 21.801 | 2.729 | 1.285 |
| 5 | | | 3.081 | 1.252 | 33.579 | 3.611 | 1.293 |
| 6 | 4.17 | 1.269 | 3.595 | 1.238 | 48.196 | 4.171 | 1.269 |
| 7 | 4.91 | 1.255 | 4.143 | 1.225 | 65.800 | 4.929 | 1.256 |
| 8 | 5.90 | 1.248 | 4.766 | 1.216 | 86.375 | 5.905 | 1.249 |
| 9 | 6.80 | 1.237 | 5.486 | 1.208 | 110.339 | 6.803 | 1.237 |
| 10 | 7.75 | 1.227 | 5.921 | 1.195 | 138.508 | 7.879 | 1.229 |
| 11 | 7.89 | 1.207 | 6.724 | 1.189 | 169.234 | 7.907 | 1.207 |
| 12 | 8.03 | 1.190 | 7.187 | 1.179 | 203.538 | 9.421 | 1.206 |
| 13 | 9.21 | 1.186 | 7.253 | 1.165 | 241.805 | 9.279 | 1.187 |
| 14 | 9.72 | 1.176 | 7.706 | 1.157 | 283.855 | 9.959 | 1.178 |
| 15 | 9.97 | 1.166 | 8.243 | 1.151 | 329.275 | 10.021 | 1.166 |
| 16 | 12.1 | 1.169 | | | 378.644 | 10.689 | 1.160 |
| 17 | 13.3 | 1.164 | | | 431.601 | 11.534 | 1.155 |
| 18 | 13.5 | 1.156 | | | 488.695 | 12.377 | 1.150 |
| 19 | | | | | 550.251 | 12.634 | 1.143 |
| 20 | | | | | 615.259 | 13.863 | 1.140 |

Table 5.9: Maximum function value, Lebesgue constant and its convergence for our nodal sets in the triangle for polynomial degree $p$, $p = 3, \ldots, 20$. We also report the values from Taylor et al. (2000) and Roth (2005). Blank spaces indicate not reported results.

functional $\mathcal{F}$ is optimized. We observe that up to polynomial degree 15, the values of the Lebesgue constant are similar, but we obtain nodal configurations featuring smaller Lebesgue constant for higher polynomial degrees. We also report the values from Roth (2005). Since the Fekete functional is a proxy of the Lebesgue constant, our nodal sets present larger values.

Next, we compare the values in Table 5.3 and Table 5.9 to determine which proxy provides better results. We can see that the Lebesgue constant of the nodal distributions that correspond to the local minima of the Chen-Babuska functional are smaller than the ones found when exploring the Fekete minima. It is also worth noting that the Chen-Babuska functional aims to minimize the vectorial $\mathcal{L}_2$-norm of the Lagrange polynomials which can be understood as the physical energy of the system and its value is of order one. In contrast, the Fekete functional in the one-

dimensional segment is related to the potential energy of electric charges (Hesthaven, 1998), but in higher dimensions this physical interpretation is not clear. Furthermore, the determinant of the Vandermonde matrix can be very large. Indeed, the function values $\mathcal{F}$ reported in Table 5.9 correspond to the logarithm of the determinant and, therefore, the actual value of the determinant for polynomial degree 20 is of order $10^{267}$. Thus, agreeing with our experience, optimizing the Fekete functional is prone to numerical overflows. Hence we favor the Chen-Babuska functional as our proxy of the Lebesgue constant.

## 5.6 Concluding remarks

We have proposed a specific-purpose deterministic optimization method to obtain fair point distributions for interpolation on the high-dimensional simplex using the point coordinates as design variables. To do so, we have devised a method to explore the multiple local minima of a twice-differentiable proxy of the Lebesgue constant. The optimized quasi-optimal nodal distributions do not feature a grid-like structure and cluster interior nodes towards the simplex boundary. We use an advantageous representation of the nodal distribution on the orthant of the $(d+1)$-sphere. On this surface, the euclidean Voronoi diagram of the nodes approximates the influence zone of the Lagrange interpolating polynomials. The dual of this diagram, the Delaunay triangulation, provides the nodal distribution with a connectivity structure. Furthermore, for a free node, the boundary of the union of the elements incident to such node determines a proxy of the uphills of the energy landscape of the Lebesgue constant. To explore structurally different point configurations, we heuristically enforce a tunnel effect by relocating one node to the centroid of the elements across the uphills. Each of these structurally different point configurations determines an initial approximation that is optimized using Newton's method equipped with a trust-region globalization taking into account the point resolution. The optimized nodal sets are represented as nodes of a tree sorted in decreasing order of the proxy objective function value. To fully exploit the available computational resources, the tree nodes are explored in such a way that nodes with the smallest values are explored first.

The results show that the proposed local nodal dislocation technique and the always-decreasing graph exploration approach are well-suited heuristics to obtain nodal distributions with good interpolation properties using the available compu-

tational resources. Furthermore, in 2D, our point distributions feature interpolation properties comparable to those nodal sets obtained by optimizing directly the Lebesgue constant. Up to 4D, we find nodal configurations of moderate polynomial degree which are the best-known local minima of the Chen-Babuska functional and present the lowest Lebesgue constant up to date.

In perspective, we expect these nodal distributions to be of practical interest for several communities. In particular, our nodal sets might be used to interpolate curved geometry and perform space-time simulation using nodal high-order unstructured methods. Additionally, we believe that methods optimizing the Lebesgue constant might benefit from using our nodal distributions as initial approximations for the optimization procedure.

# Chapter 6

# Computing nodal distributions with quasi-optimal Lebesgue constant

## 6.1 Introduction

In 4D space-time discretizations on complex geometry, the interest in point distributions featuring quasi-optimal interpolation error for unknown functions is twofold. First, they are accepted as a proxy of fair nodal distributions for high-order unstructured methods such as the discontinuous (Hesthaven and Warburton, 2007) and continuous Galerkin (Karniadakis and Sherwin, 2005) methods. Second, they might allow representing (Jiménez-Ramos et al., 2020) and modeling (Jiménez-Ramos et al., 2022) complex curved geometry preserving the simulation intent, see Chapter 3. Since complex geometries can be discretized using automatic simplicial mesh generators (Peraire et al., 1987; Löhner and Parikh, 1988; George et al., 1990; Shewchuk, 2005; Si, 2015; Barber et al., 1996; The CGAL Project, 2023), we focus on obtaining quasi-optimal point distributions on the simplex.

To find such quasi-optimal nodal distributions, it is standard to minimize the infinity norm of the interpolation operator, known as Lebesgue constant. Nodal distributions with optimal Lebesgue constant are conjectured to fulfill two properties: they have interior nodes clustered towards the boundary and do not feature a grid-like structure (Taylor et al., 2000; Roth, 2005). The first property is exploited by explicit

and semi-explicit methods to generate sub-optimal Lebesgue nodal configurations. Specifically, these approaches distort a grid of points and enforce clustering the interior nodes towards the boundary based on heuristics. In particular, semi-explicit methods optimize the Lebesgue constant in terms of a single blending parameter which controls how the prescribed nodes on the edges are incorporated into the interior of the simplex. These explicit or semi-explicit nodal distributions have been computed in the triangle (Blyth and Pozrikidis, 2005; Warburton, 2006), the tetrahedron (Luo and Pozrikidis, 2006; Warburton, 2006), and the $d$-simplex (Isaac, 2020) and feature good interpolation properties for low polynomial degrees. However, since the initial approximation is given by a grid of points, this grid-like structure is preserved, so they underperform as the polynomial degree increases.

Alternatively, some implicit methods using the point coordinates as design variables are capable of breaking the grid-like structure and finding the appropriate disposition of the boundary and interior nodes. Unfortunately, these methods have to deal with the non-differentiability of the objective function. Specifically, the optimization of the Lebesgue constant addresses the minimization with respect to the node coordinates of the maximum over the simplex points of the Lebesgue function. Accordingly, with respect to the node coordinates, the Lebesgue constant is not differentiable not only because the Lebesgue function is not everywhere differentiable but mainly because the involved maximum is not differentiable.

In 1D, the non-differentiable points of the Lebesgue function coincide with the coordinates of the interpolation nodes. This particularity of the 1D case can be exploited to numerically compute the optimal Lebesgue nodes by means of a sequential linear programming approach (Angelos et al., 1989). In higher dimensions, besides the interpolation nodes, there are more regions where the Lebesgue function is not differentiable. Nevertheless, for any spatial point where none of the Lagrange interpolating polynomials vanish, the derivatives of the Lebesgue function with respect to the node coordinates are well-defined.

The Lebesgue constant, which depends exclusively on the nodal coordinates, is not differentiable, yet the derivatives can be approximated using finite differences. Using these approximations, it is possible to optimize the point coordinates by means of steepest-descent or damped Newton's method (Heinrichs, 2005), or by solving a quadratic problem in a sequential quadratic programming approach (Briani et al., 2012; Rapetti et al., 2012). These local optimization methods find a minimum close

to the given initial approximation, so if the initial condition features a grid-like structure, the optimized nodal distribution preserves such structure. Thus, these methods neither break the grid-like structure of the initial approximation nor are consistent with the smoothness of the Lebesgue function.

To avoid the non-differentiability of the Lebesgue constant, there exist zeroth-order optimization methods using only the values of the objective function. The key point of these zeroth-order methods is that they are capable of changing the structure of the nodal distribution by randomly perturbing a node. Thus, these methods can be applied to break the grid-like structure of an initial approximation and further improve the Lebesgue constant (Roth, 2005; Rapetti et al., 2012). Unfortunately, these methods are not deterministic and suffer the curse of dimensionality. As the dimension and the polynomial degree increases, the search space becomes larger and zeroth-order methods fail to exhaustively explore regions with potentially optimal minima.

Accordingly, the goal of this chapter is to obtain interpolative nodal distributions on the high-dimensional simplex by optimizing the Lebesgue constant using the node coordinates as design variables, yet consistently exploiting first-order derivatives of the Lebesgue function. We expect that the obtained nodal distributions feature quasi-optimal interpolation properties and provide the required accuracy for the aforementioned applications in complex geometry.

To meet our goal, the main contribution of this chapter is a specific-purpose deterministic method to optimize the Lebesgue constant of a given nodal distribution in the $d$-dimensional simplex using the point coordinates as design variables. In a sequential linear programming approach, we sample at relevant spatial points and compute a candidate descent direction using the first-order derivatives of the Lebesgue function with respect to the node coordinates. The derivatives are computed analytically and the Lebesgue constant is not differentiated in any case. The method finds an optimum close to the given initial approximation. Thus, it is well-suited to further improve the Lebesgue constant of nodal distributions with already good interpolation properties. Remarkably, the nodes are represented on the orthant of the $(d+1)$-sphere, a representation that is exploited to ensure smooth dynamics during the optimization procedure. The method is validated by finding the optimal Lebesgue nodes in 1D. Furthermore, using the nodal distributions from Chapter 5 as initial approximations, we reproduce the literature results in 2D, and we find and

report the nodal distributions with the lowest Lebesgue constant up to date in 3D and 4D.

The rest of the chapter is organized as follows. First, in Sect. 6.2, we recall some preliminaries regarding symmetric nodal distributions, the Lebesgue constant, and the optimization method to compute optimal Lebesgue configurations in the interval (Angelos et al., 1989). Then, in Sect. 6.3, we present our constrained optimization method to minimize the Lebesgue constant of a given nodal distribution in the high-dimensional simplex. In Sect. 6.4, we report and discuss the obtained results, and finally, in Sect. 6.5, we finish with some concluding remarks.

## 6.2 Preliminaries

Next, we recall some notation and definitions introduced in Chapter 2 used throughout this chapter regarding the parameterization of symmetric nodal distributions in the simplex, Sect. 6.2.1, and the Lebesgue constant, Sect. 6.2.2. Then, in Sect. 6.2.3, we describe the algorithm from Angelos et al. (1989) to compute the optimal Lebesgue nodes in the interval.

### 6.2.1 Parameterization of symmetric nodal distributions

For a given polynomial degree $p$, we denote by $\boldsymbol{Z} = \{\boldsymbol{z}_j\}_{j=1,\dots,N_{p,d}}$ a set of interpolation nodes in a simplex $K^d \subset \mathbb{R}^d$. Specifically, we consider symmetric nodal distributions and thus, the coordinates of a subset of nodes are enough to describe the whole nodal set, see details in Sect. 2.1. More precisely, only $n$ degrees of freedom encoded in a vector $\boldsymbol{y} \in \mathbb{R}^n$ are the actual design variables. The $n_i$ degrees of freedom of representative node $i$ are encoded in the vector $\boldsymbol{y}_i = (y_i^1, \dots, y_i^{n_i})$. As detailed in Appendix C, these values should satisfy

$$\begin{aligned} &0 < y_i^j < 1, \quad \text{for } j = 1, \dots, n_i, \\ &\sum_{j=1}^{n_i} y_i^j < 1, \end{aligned} \tag{6.1}$$

to describe a point inside its associated $n_i$-simplex. We recall that $\boldsymbol{\sigma}$, see Eq. (C.2), is the function mapping the vector of degrees of freedom to the whole nodal distribution,

$$\begin{array}{rccc} \boldsymbol{\sigma}\colon & \mathbb{R}^n & \to & \mathbb{R}^d \times \overset{N_{p,d}}{\cdots} \times \mathbb{R}^d \\ & \boldsymbol{y} & \mapsto & \boldsymbol{\sigma}(\boldsymbol{y}) = \boldsymbol{Z} \end{array}.$$

### 6.2.2 The Lebesgue constant

The Lebesgue constant $\Lambda$ corresponds to the infinity norm of the interpolation operator $\mathcal{I}_{\boldsymbol{Z}}$ at the nodal set $\boldsymbol{Z}$ and appears in the upper bound of the interpolation error of a function $f$,

$$\left\| f - \mathcal{I}_{\boldsymbol{Z}} f \right\|_\infty \leq (1 + \Lambda) \left\| f^\star - f \right\|_\infty ,$$

where $f^\star$ denotes the best approximating polynomial. Accordingly, to attain convergence of the interpolator $\mathcal{I}_{\boldsymbol{Z}} f$ to the target function $f$, $\Lambda$ should grow slower in $p$ than $\left\| f^\star - f \right\|_\infty$ dies away. In particular, uniform convergence is attained if

$$\lim_{p \to +\infty} \Lambda^{1/p} = 1.$$

Remarkably, the Lebesgue constant can be expressed as

$$\Lambda \left( \boldsymbol{Z} \right) = \max_{\boldsymbol{x} \in K^d} \sum_{i=1}^{N_{p,d}} \left| \phi_i \left( \boldsymbol{x}; \boldsymbol{Z} \right) \right| , \tag{6.2}$$

where $\phi_j$ is the Lagrange interpolating polynomial associated to the node $\boldsymbol{z}_j$, $j = 1, \ldots, N_{p,d}$.

### 6.2.3 Optimization of the Lebesgue constant in the interval

The position of the optimal Lebesgue nodes in the interval is known algebraically only for polynomial degree $p \leq 3$ (Rack, 1984), yet it is possible to compute numerically the optimal Lebesgue nodal distribution for arbitrary degree $p$ (Angelos et al., 1989). The method reduces the Lebesgue constant of a given nodal distribution by means of a sequential linear programming approach. At each iteration, the information gathered from the current local maxima of the Lebesgue function is used to compute a nodal displacement which leads to a new nodal distribution with an improved Lebesgue constant.

Consider a nodal distribution $\boldsymbol{Z} = \{z_i\}_{i=1,\ldots,N_p}$ of polynomial degree $p$, with $z_1 = -1$, $z_{N_p} = 1$, and $z_i < z_j$, for $i < j$, and let $\Omega_k$ be the subinterval $[z_k, z_{k+1}]$, for $k = 1, \ldots, m = N_p - 1$. Note that the Lebesgue function has a single maximum on each of these intervals (Luttmann and Rivlin, 1965). We denote by $\tau_k \in \Omega_k$ the point where the maximum is attained, and by $\lambda_k := L \left( \tau_k; \boldsymbol{Z} \right)$ the value of such maximum. To preserve the symmetry, it is enough to consider the nodes in half the domain.

Specifically, the vector of degrees of freedom $\boldsymbol{y} \in \mathbb{R}^n$, $n = \lfloor (N_p - 1)/2 \rfloor$, encodes the position of the points with positive coordinates.

Then, to compute a nodal displacement $\boldsymbol{v} \in \mathbb{R}^n$, the following linear problem is solved,

$$\begin{aligned} \min_{w \in \mathbb{R}, \boldsymbol{v} \in \mathbb{R}^n} \quad & w \\ \text{s.t.} \quad & L\left(\tau_k; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) + \nabla_{\boldsymbol{y}} L\left(\tau_k; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \cdot \boldsymbol{v} \leq w, \quad \text{for } k = 1, \ldots, m \\ & \left(y_{i+1} + v_{i+1}\right) - \left(y_i + v_i\right) \geq \delta, \quad \text{for } i = 1, \ldots, n \\ & -b_i \leq v_i \leq b_i, \quad \text{for } i = 1, \ldots, n. \end{aligned} \tag{6.3}$$

where the term $\nabla_{\boldsymbol{y}} L\left(\tau_k; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \in \mathbb{R}^n$ denotes the derivatives of $L\left(x; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right)$ with respect to the degrees of freedom $\boldsymbol{y}$ evaluated at $x = \tau_k$, $v_i$ denotes the displacement associated to the $i$-th degree of freedom, and $\delta$ is a small parameter.

The first set of constraints corresponds to a first-order approximation of the function $L\left(x; \boldsymbol{Z}\right)$ with respect to $\boldsymbol{Z}$ around the current position of the nodes. Taking into account symmetry, these constraints read as

$$L\left(\tau_k; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) + \nabla_{\boldsymbol{y}} L\left(\tau_k; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \cdot \boldsymbol{v} \leq w, \quad \text{for } k = 1, \ldots, m. \tag{6.4}$$

Variable $w$ serves as an upper bound of the value of the Lebesgue function at the maxima. Thus, the combination of the minimization of $w$ and constraints Eq. (6.4) aims to find the nodal displacement $\boldsymbol{v}$ which leads to a new nodal distribution with the lowest value of the Lebesgue constant as possible. The second set of constraints

$$\left(y_{i+1} + v_{i+1}\right) - \left(y_i + v_i\right) \geq \delta, \tag{6.5}$$

enforces the nodes to preserve their current order and avoids swapping. Finally, the third set of constraints ensures each of the components of the displacement vector $\boldsymbol{v}$ is small,

$$|v_i| \leq b_i. \tag{6.6}$$

These constraints are needed because Eq. (6.4) is a linearization around $\boldsymbol{Z}$. We remark that these constraints are linear for the one-dimensional case,

$$|v_i| \leq b_i \Leftrightarrow -b_i \leq v_i \leq b_i.$$

The solution to this linear problem gives a candidate descent direction $\boldsymbol{v}$, but the nodal distribution $\boldsymbol{Z}_{\text{new}} = \boldsymbol{\sigma}\left(\boldsymbol{y} + \boldsymbol{v}\right)$ may have larger Lebesgue constant. This

**Algorithm 6.1** Computing the optimal Lebesgue nodal distribution in the interval.

**Input:** Initial nodal distribution $\boldsymbol{Z}_0$
**Output:** Optimized nodal distribution $\boldsymbol{Z}$

```
1: function OptimizeLebesgue1D(Z_0)
2:   y ← GetVectorOfDOFs(Z_0)
3:   {λ_k}, T ← ComputeMaximaEachInterval(σ(y))
4:   {b_i} ← InitializeBounds(Z_0)
5:   while min_k λ_k ≠ max_k λ_k do
6:     v ← ComputeDescentDirection(σ(y), T, {b_i})
7:     α ← BacktrackingLineSearch(σ(y), v)
8:     {b_i} ← AdjustBounds({b_i})
9:     y ← y + αv
10:    {λ_k}, T ← ComputeMaximaEachInterval(σ(y))
11:  end while
12:  Z ← σ(y)
13:  return Z
14: end function
```

is so because direction $\boldsymbol{v}$ is computed using the local maxima as sample points of the Lebesgue function, that is, only partial information is being used. Thus, even though $L(\tau_k; \boldsymbol{Z}_{\text{new}}) < L(\tau_k; \boldsymbol{Z})$ for all $k$, we may find a greater value of the Lebesgue function at another point, usually near a local maximum, and consequently, the actual Lebesgue constant of $\boldsymbol{Z}_{\text{new}}$ may be larger than that of $\boldsymbol{Z}$. This issue is circumvented by means of a backtracking line search over the restriction set. Specifically, a value $\alpha \in [0, 1]$ is found such that the Lebesgue constant of $\boldsymbol{\sigma}(\boldsymbol{y} + \alpha\boldsymbol{v})$ improves the current value. This process is iterated until convergence is achieved.

The whole procedure is described in Algorithm 6.1. The input is a nodal distribution $\boldsymbol{Z}_0$. First, in Line 2, we compute the degrees of freedom $\boldsymbol{y}$ that define $\boldsymbol{Z}_0$. Following, in Line 3, all the local maxima of the Lebesgue function in the interval are computed. Two sets are the output of function `ComputeMaximaEachInterval`: the maxima values $\{\lambda_k\}_{k=1,\dots,m}$, and the coordinates of the points where these maxima are attained $\mathrm{T} = \{\tau_k\}_{k=1,\dots,m}$. Then, in Line 4, the bounds $\{b_i\}_{i=1,\dots,N_p}$ are initialized. Next, there is the main loop of the algorithm. In Line 6, a candidate descent direction $\boldsymbol{v}$ is computed by solving the linear problem stated in Eq. (6.3), and, in Line 7, it takes place the backtracking line search. Finally, we adjust the bounds, update the nodal positions and compute the new maxima, see Lines 8-10. This process is repeated until all the maxima are equal, see Line 5. This is so because it

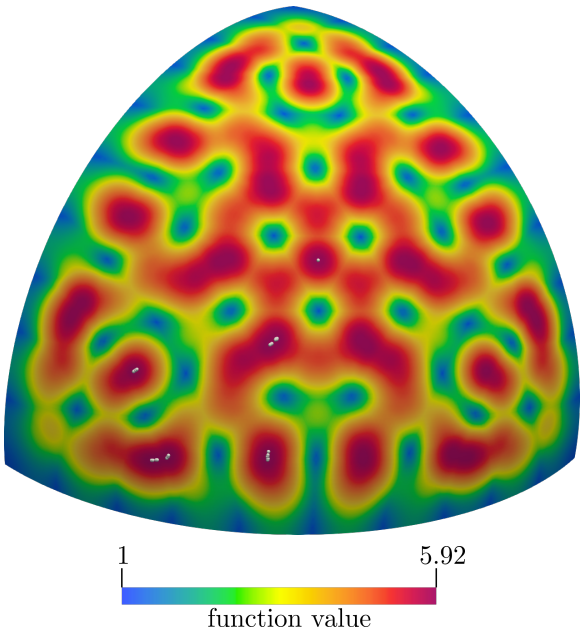



Figure 6.1: Lebesgue function of our quasi-optimal nodal distribution of polynomial degree $p = 10$ in the triangle and the 83 maxima in the sextant (white points) found during the optimization process.

was conjectured in Bernstein (1931) and later proved in De Boor and Pinkus (1978); Kilgore (1977, 1978) that the maxima in each subinterval attain the same value in the optimal Lebesgue nodal configuration. Thus, the stopping criterion ensures that the obtained nodal distribution corresponds to the optimal Lebesgue configuration.

## 6.3 Optimizing the Lebesgue constant

In this section, we present a local method to optimize the Lebesgue constant of an initial nodal distribution in the $d$-dimensional simplex. Our method is inspired by the one-dimensional algorithm (Angelos et al., 1989) detailed in Sect. 6.2.3 and the structure is analogous, but it features two remarkable differences. First, in higher dimensions, there is not an optimal condition to ensure that the global optimum has been found. To circumvent this issue, the stopping criterion takes into account the relative error in the nodal displacement. Second, in a higher dimensional simplex, it is not known what are the regions where the Lebesgue function has the maxima and, therefore, we cannot define regions containing one single maximum. Instead, we store in a set T the global maximum at each iteration. Thus, this set T contains points that

**Algorithm 6.2** Optimizing the Lebesgue constant of a nodal distribution in the simplex.

**Input:** Initial nodal distribution $\boldsymbol{Z}_0$
**Output:** Optimized nodal distribution $\boldsymbol{Z}$

1: **function** `OpimitizeLebesgue`($\boldsymbol{Z}_0$)
2: $\boldsymbol{y} \leftarrow$ `GetVectorOfDOFs`($\boldsymbol{Z}_0$)
3: $\boldsymbol{\tau}$, $\Lambda \leftarrow$ `ComputeLebesgueConstant`($\boldsymbol{\sigma}(\boldsymbol{y})$)
4: $\mathrm{T} \leftarrow \{\boldsymbol{\tau}\}$
5: $b \leftarrow$ `InitializeBounds`($\boldsymbol{Z}_0$)
6: **while** $((\alpha < 1)$ or $(b > \varepsilon))$ and $(\alpha > \varepsilon_\alpha)$ **do**
7: $\boldsymbol{v}$, optimal, numLPSolved $\leftarrow$ `ComputeDescentDirection`($\boldsymbol{y}$, T, $b$)
8: **if** not optimal **then**
9: $\boldsymbol{Z} \leftarrow \boldsymbol{\sigma}(\boldsymbol{y})$
10: **return** $\boldsymbol{Z}$
11: **end if**
12: $\alpha$, $\Lambda_{\text{new}}$, $\mathrm{T} \leftarrow$ `BacktrackingLineSearch`($\boldsymbol{y}$, $\Lambda$, T, $\boldsymbol{v}$)
13: $b \leftarrow$ `AdjustBounds`($\alpha$, numLPSolved, $b$)
14: $\boldsymbol{y} \leftarrow \boldsymbol{y} + \alpha\boldsymbol{v}$
15: $\Lambda \leftarrow \Lambda_{\text{new}}$
16: **end while**
17: $\boldsymbol{Z} \leftarrow \boldsymbol{\sigma}(\boldsymbol{y})$
18: **return** $\boldsymbol{Z}$
19: **end function**

have been relevant in the past and might be close to a global maximum in the future. Consequently, these points approximate the function peaks and allow us to compute an accurate descent direction and perform fewer iterations in the backtracking line search procedure.

To illustrate this, in Fig. 6.1, we show the Lebesgue function of our quasi-optimal nodal distribution of polynomial degree 10 in the triangle, and the set of global maxima appearing during the optimization procedure to be described next. We observe that the points are on those regions where the Lebesgue function attains high values and indeed, they are a good approximation of the local maxima of the function.

### 6.3.1 Outline of the algorithm

Despite these differences with respect to the one-dimensional problem, the structure of our algorithm is analogous, see Algorithm 6.2. Given a nodal distribution $\boldsymbol{Z}_0$, the first step is to compute the corresponding degrees of freedom $\boldsymbol{y}$, Line 2. Next, in Line

3, we compute the Lebesgue constant using the adaptive sampling method proposed in Chapter 4. The output is the maximum point $\boldsymbol{\tau}$ and the value of the Lebesgue constant $\Lambda$. Following, in Line 4, we initialize the set T with the maximum $\boldsymbol{\tau}$ of the current nodal distribution and, in Line 5, we initialize the bounds to be used in the computation of the descent direction. Herein, we use a single bound $b \in \mathbb{R}$ for all the degrees of freedom.

Then, we enter the main loop of the algorithm. In Line 7, we compute the descent direction $\boldsymbol{v}$ and, in Line 12, we compute the step length $\alpha$ such that the nodal distribution represented by $\boldsymbol{y} + \alpha\boldsymbol{v}$ improves the Lebesgue constant. Finally, we adjust the bounds, Line 13, and update the variables, Lines 14 and 15. This process is repeated until the maximum bound is less than a prescribed tolerance $\varepsilon = 10^{-3}\frac{1}{64p}$, see Line 6. As an additional safety stopping criterion, we decide to stop the optimization if $\alpha < \varepsilon_\alpha = 10^{-6}$. Following, we precisely describe the routines `ComputeDescentDirection`, `BacktrackingLineSearch`, and `AdjustBounds`.

### 6.3.2 Computing the descent direction

To compute a descent direction $\boldsymbol{v}$, we solve a linear problem similar to the one proposed for the interval in Angelos et al. (1989). As in the one-dimensional case, we seek a minimum for the bound $w$ subject to a set of constraints. Eq. (6.4) extends straight-forwardly to higher dimensions as

$$L\left(\boldsymbol{\tau};\boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) + \nabla_{\boldsymbol{y}} L\left(\boldsymbol{\tau};\boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right)\cdot\boldsymbol{v} \leq w, \quad \text{for } \boldsymbol{\tau} \in \mathrm{T}. \tag{6.7}$$

Note that the function $L\left(\boldsymbol{\tau};\boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right)$ is not differentiable with respect to the degrees of freedom if any of the Lagrange interpolating polynomials vanishes at $\boldsymbol{\tau}$, see Eq. (6.2). To address this issue, we consider a regularized absolute value function $|x|_{\delta_{\mathrm{Abs}}} = \sqrt{x^2 + \delta_{\mathrm{Abs}}^2}$, with $\delta_{\mathrm{Abs}} = 10^{-16}$. This value has no influence when $\phi_i\left(\boldsymbol{\tau};\boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \neq 0$, yet it allows computing the derivatives when $\boldsymbol{\tau}$ is at a non-differentiable location.

The first-order approximation in Eq. (6.7) is only valid for small values of $\|\boldsymbol{v}\|$. Accordingly, we impose the following set of linear constraints,

$$-b \leq v_i \leq b, \quad \text{for } i = 1, \ldots, n,$$

where $v_i$ denotes the $i$-th component of vector $\boldsymbol{v}$.

In the interval, the nodes should preserve the order of the coordinates, see Eq. (6.5). In contrast, in higher dimensions the nodes should freely move and no restriction

regarding the order is needed. Nevertheless, the new degrees of freedom should determine a valid nodal distribution, see Eq. (6.1). Thus, for each node representative $i$, we impose the following constraints

$$\begin{aligned} &0 < y_i^j + v_i^j < 1, \quad \text{for } j = 1, \dots, n_i, \\ &\sum_{j=1}^{n_i} y_i^j + v_i^j < 1, \end{aligned}$$

where $v_i^j$ denotes the displacement associated with the $j$-th components of the vector of degrees of freedom encoding the $i$-th node representative. We remark that two nodes will avoid collapsing since otherwise the Lebesgue function would increase in-between, and, in that case, the upper bound $w$ would increase and the displacement $\boldsymbol{v}$ would not be a descent direction.

In conclusion, the linear problem to compute a descent direction reads as

$$\begin{aligned} \min_{w \in \mathbb{R}, \boldsymbol{v} \in \mathbb{R}^n} \quad & w \\ \text{s.t.} \quad & L\left(\boldsymbol{\tau}; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) + \nabla_{\boldsymbol{y}} L\left(\boldsymbol{\tau}; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \cdot \boldsymbol{v} \leq w, \quad \text{for } \boldsymbol{\tau} \in \mathrm{T}, \\ & -b \leq v_i \leq b, \quad \text{for } i = 1, \dots, n, \\ & 0 < y_i^j + v_i^j < 1, \quad \text{for } j = 1, \dots, n_i, \quad \text{for } i = 1, \dots, m, \\ & \sum_{j=1}^{n_i} y_i^j + v_i^j < 1, \quad \text{for } i = 1, \dots, m. \end{aligned} \tag{6.8}$$

The routine `ComputeDescentDirection` computes nodal displacements, see details in Algorithm 6.3. Given the degrees of freedom describing a nodal distribution $\boldsymbol{y}$, the set of maxima T, and the bounds $b$, in Line 2, we compute the nodal displacement $\boldsymbol{v}$ solving the linear problem in Eq. (6.8). To this end, we use the simplex method implemented in the GLPK solver (GLPK) through the JuMP interface (Dunning et al., 2017). In Line 3, we initialize the counter of solved linear problems in the current iteration. Note that the function `SolveLP` returns the computed nodal displacement $\boldsymbol{v}$ and an additional boolean variable telling whether the solution is optimal. If the solution is not optimal, we halve the bounds $b$, Line 5, try to solve the linear problem again, Line 6, and increase by one the counter of linear problems solved, Line 7. This process is repeated until we find an optimal solution to the linear problem or the bounds are below the threshold $\varepsilon$. In either case, in Line 9, the routine returns the computed displacement $\boldsymbol{v}$, the boolean variable `optimal`, and the number of linear problems solved `numLPSolved`. We highlight that the main algorithm

**Algorithm 6.3** Computing the descent direction.

**Input:** Degrees of freedom $\boldsymbol{y}$, Set of maxima T, Bounds $b$

**Output:** Nodal displacement $\boldsymbol{v}$, Boolean variable `optimal`, Number of LP solved `numLPSolved`

```
1: function ComputeDescentDirection(y, T, b)
2:   v, optimal ← SolveLP(y, T, b)
3:   numLPSolved ← 1
4:   while not optimal and (b > ε) do
5:     b ← b/2
6:     v, optimal ← SolveLP(y, T, b)
7:     numLPSolved ← numLPSolved+ 1
8:   end while
9:   return v, optimal, numLPSolved
10: end function
```

described in Algorithm 6.2 is responsible for stopping the optimization procedure if the solution is not optimal. Nonetheless, even when the displacement $\boldsymbol{v}$ is optimal, we should ensure that the Lebesgue constant is reduced.

### 6.3.3 Backtracking line search

Although $\boldsymbol{v}$ is the optimal solution to the linear problem, the nodal set described by $\boldsymbol{y}+\boldsymbol{v}$ may worsen the Lebesgue constant since the maximum of the nodal distribution given by $\boldsymbol{y}+\boldsymbol{v}$ may not belong to the set T. Thus, we perform a backtracking line search to find a value of $\alpha \in [0,1]$ such that the new position of the nodes, $\boldsymbol{y}+\alpha\boldsymbol{v}$, leads to a decrease in the Lebesgue constant. We remark that this is the most time-consuming part of the algorithm since we should estimate the Lebesgue constant at each reduction step.

The complete backtracking line search algorithm is described in Algorithm 6.4. Given the current degrees of freedom $\boldsymbol{y}$, the Lebesgue constant $\Lambda$ of $\boldsymbol{\sigma}(\boldsymbol{y})$, the list of maxima T, and the computed nodal displacement $\boldsymbol{v}$, first, in Line 2, we initialize the value of $\alpha$ to 1 and, in Line 3, we compute the maximum point $\boldsymbol{\tau}_\alpha$ and the value of the Lebesgue constant $\Lambda_\alpha$ of the nodal distribution $\boldsymbol{\sigma}(\boldsymbol{y}+\alpha\boldsymbol{v})$. Then, we add the point $\boldsymbol{\tau}_\alpha$ to the set of maxima T, Line 4. Thus, in the following iterations, new and relevant sample points are used in the computation of the descent direction $\boldsymbol{v}$. Now, if $\Lambda_\alpha \geq \Lambda$, we halve the value of $\alpha$, Line 9, compute again $\boldsymbol{\tau}_\alpha$ and $\Lambda_\alpha$, Line 10, and append the maxima to the list, Line 11. This process is repeated until $\Lambda_\alpha < \Lambda$ or the value of

**Algorithm 6.4** Computing a value of $\alpha$ such that $\boldsymbol{y} + \alpha\boldsymbol{v}$ improves the Lebesgue constant.

**Input:** Degrees of freedom $\boldsymbol{y}$, Lebesgue constant $\Lambda$, List of maxima T, Nodal displacement $\boldsymbol{v}$

**Output:** Alpha value $\alpha$, New Lebesgue constant $\Lambda_\alpha$, Updated list of maxima T

1: **function** `BacktrackingLineSearch`($\boldsymbol{y}$, $\Lambda$, T, $\boldsymbol{v}$)
2: $\alpha \leftarrow 1$
3: $\boldsymbol{\tau}_\alpha$, $\Lambda_\alpha \leftarrow$ `ComputeLebesgueConstant`($\boldsymbol{\sigma}(\boldsymbol{y} + \alpha\boldsymbol{v})$)
4: $\mathrm{T} \leftarrow \mathrm{T} \cup \{\boldsymbol{\tau}_\alpha\}$
5: **while** $\Lambda_\alpha \geq \Lambda$ **do**
6: **if** $\alpha < \varepsilon_\alpha$ **then**
7: **return** 0, $\Lambda$, T
8: **end if**
9: $\alpha \leftarrow \alpha/2$
10: $\boldsymbol{\tau}_\alpha$, $\Lambda_\alpha \leftarrow$ `ComputeLebesgueConstant`($\boldsymbol{\sigma}(\boldsymbol{y} + \alpha\boldsymbol{v})$)
11: $\mathrm{T} \leftarrow \mathrm{T} \cup \{\boldsymbol{\tau}_\alpha\}$
12: **end while**
13: **return** $\alpha$, $\Lambda_\alpha$, T
14: **end function**

$\alpha$ is less than a prescribed tolerance $\varepsilon_\alpha = 10^{-6}$, Lines 5 and 6. Note that we do not impose a sufficient decrease condition as it is standard with high-order optimization methods. This is so because the Lebesgue constant is not differentiable. Thus, as soon as the Lebesgue constant is reduced, the value of $\alpha$ is accepted. Finally, in Line 13, we return the step length $\alpha$, the Lebesgue constant of the new nodal distribution $\boldsymbol{\sigma}(\boldsymbol{y} + \alpha\boldsymbol{v})$, and the updated list of maxima points T.

### 6.3.4 Adjusting the bounds

The value $b \in \mathbb{R}$ bounds the allowed nodal displacement in the linear problem, see Eq. (6.8). We determine the initial value for these bounds in terms of the polynomial degree $p$ of the nodal distribution to optimize. More precisely, in Line 5 from Algorithm 6.2, we set $b = \frac{\pi}{2}\frac{1}{64p}$, that is, we limit the nodal displacement by a fraction of the spherical simplex edge length taking into account the point resolution in terms of the polynomial degree $p$. This is so because points having a close to optimal nodal distribution are conjectured to follow an equispaced arrangement on the sphere, see Sect. 5.3.

Aiming to compute a nodal displacement $\boldsymbol{v}$ with the correct magnitude and reduce

**Algorithm 6.5** Adjusting the bounds of the degrees of freedom.

**Input:** Alpha $\alpha$, numLPSolved, Bounds $b$
**Output:**

```
1: function AdjustBounds(α, numLPSolved, b)
2:   if α < 1 then
3:     b ← b/2
4:   else if numLPSolved is 1 then
5:     b ← min(2b, π/2 · 1/(64p))
6:   end if
7: end function
```

the backtracking line search iterations, the value of the bounds is adjusted at each optimization step, see Line 13 of Algorithm 6.2. The routine `AdjustBounds` is detailed in Algorithm 6.5. The inputs are the previously computed value of $\alpha$, the counter of linear problems solved, and the set of bounds to update in-place. Then, in Line 2, we check if $\alpha < 1$. If $\alpha$ is less than one, the nodal displacement computed in the current iteration does not have the correct magnitude, and it had to be shortened. Thus, in Line 3, we halve the bounds. In the next iteration, we expect a shorter nodal displacement but with the appropriate magnitude. In contrast, if $\alpha = 1$, the linear problem is solved once. That is, the nodal displacement has the correct magnitude, and we may allow larger displacements in the next iteration. Thus, in Line 5, we double the bounds, yet they never exceed $\frac{\pi}{2}\frac{1}{64p}$.

## 6.4 Results

The Lebesgue constant has multiple minima and we require exploration approaches to find the global minimum. Conveniently, some local minima of the Chen-Babuska functional, a proxy of the Lebesgue constant, have been already explored in Chapter 5. These local minima feature good interpolation properties and thus, they are suitable candidate initial approximations to be further optimized with our method.

All of these candidates are optimized with the method proposed in Sect. 6.3. Our quasi-optimal Lebesgue distribution is obtained after optimizing a local minimum with good Chen-Babuska and Lebesgue constant values. However, optimizing only the best local minimum in terms of the Chen-Babuska functional does not lead to the best Lebesgue configuration, and neither does the local Chen-Babuska minimum with the smallest Lebesgue constant lead to the best Lebesgue configuration. Thus, we

| $p$ | $\Lambda$ | $\Lambda^{1/p}$ | $\max_k \lambda_k - \min_k \lambda_k$ |
|---|---|---|---|
| 3 | 1.4229 | 1.1248 | $8.171 \cdot 10^{-14}$ |
| 4 | 1.5595 | 1.1175 | $2.635 \cdot 10^{-12}$ |
| 5 | 1.6722 | 1.1083 | $4.645 \cdot 10^{-13}$ |
| 6 | 1.7681 | 1.0996 | $3.935 \cdot 10^{-12}$ |
| 7 | 1.8516 | 1.0920 | $7.853 \cdot 10^{-12}$ |
| 8 | 1.9255 | 1.0853 | $1.193 \cdot 10^{-11}$ |
| 9 | 1.9917 | 1.0796 | $8.483 \cdot 10^{-12}$ |
| 10 | 2.0517 | 1.0745 | $7.820 \cdot 10^{-12}$ |
| 11 | 2.1066 | 1.0701 | $9.444 \cdot 10^{-10}$ |
| 12 | 2.1571 | 1.0662 | $1.484 \cdot 10^{-10}$ |
| 13 | 2.2040 | 1.0627 | $1.239 \cdot 10^{-9}$ |
| 14 | 2.2476 | 1.0596 | $1.211 \cdot 10^{-8}$ |
| 15 | 2.2884 | 1.0567 | $2.103 \cdot 10^{-8}$ |

Table 6.1: Properties of the optimal Lebesgue distribution of polynomial degree $p$ in the interval, $p = 3, \ldots, 15$. We report the Lebesgue constant $\Lambda$, the series $\Lambda^{1/p}$, and the difference between the maximum and minima maxima values in each subinterval.

optimize all the minima of the Chen-Babuska functional found in Chapter 5, and the nodal distribution with the smallest Lebesgue constant is our quasi-optimal Lebesgue interpolation nodal set.

To assess the interpolation properties of the nodal distributions found with the proposed method, we study the value of the Lebesgue constant $\Lambda$, the convergence of the sequence $\Lambda^{1/p}$ as we increase the polynomial degree $p$, and the interpolation error for some benchmark functions. To validate the proposed method, we first find the optimal Lebesgue nodes in the interval, see Sect. 6.4.1. Next, in Sect. 6.4.2-Sect. 6.4.4, we analyze the optimized nodal sets for several polynomial degrees in the triangle, the tetrahedron, and the pentatope, respectively. Finally, in Sect. 6.4.5, we revisit the example in Sect. 3.8.1 to interpolate a geometry representation given by a subdivision limit model.

### 6.4.1 Optimal Lebesgue nodes in the interval

To validate our method, we optimize the position of the equispaced nodal distribution in the interval. Remarkably, in 1D, Lebesgue nodes satisfy $\max_k \lambda_k = \min_k \lambda_k$, where $\lambda_k$ denotes the maximum of the Lebesgue function at the interval $[z_k, z_{k+1}]$, see De Boor and Pinkus (1978); Kilgore (1977, 1978). In Table 6.1, we report the

Lebesgue constant $\Lambda$, the value $\Lambda^{1/p}$ and the difference $\max_k \lambda_k - \min_k \lambda_k$ for the optimized nodal distributions of polynomial degree $p$, $p = 3, \ldots, 15$. We observe that the difference between maxima values at each subinterval is approximately zero, that is, all the maxima attain the same value $\Lambda$ and, therefore, we find the optimal Lebesgue nodal distributions. Moreover, there is numerical evidence that the sequence $\Lambda^{1/p}$ tends to 1, which ensures that the interpolant converges uniformly to the interpolated function, see Sect. 6.2.

### 6.4.2 Quasi-optimal interpolation points in 2D

The Chen-Babuska minima computed with the exploration method proposed in Chapter 5 are excellent initial approximations for the Lebesgue optimization method described in Sect. 6.3. Following, we optimize all of these nodal distributions and analyze the interpolation properties of the optimized nodal sets.

#### Lebesgue constant

In Table 6.2, we report the minimum Lebesgue constant $\Lambda$ found after optimizing each Chen-Babuska minima. We also list the Lebesgue constant of the initial approximation, that is, the Lebesgue constant of the nodal distribution that leads to the minimum Lebesgue constant. We observe that the difference between the Lebesgue constant of the initial approximation and the optimized nodal set is almost irrelevant up to polynomial degree 9, yet the improvement becomes more significant as we increase the polynomial degree. We also show the values of the sequence $\Lambda^{1/p}$ and observe that it has a decreasing tendency to 1. Thus, there is numerical evidence that, using these nodal distributions, the interpolative polynomial might converge uniformly to the function being interpolated.

Finally, we measure the distance between the initial approximation $\boldsymbol{Z}^0 = \{\boldsymbol{z}_i^0\}_{i=1,\ldots,N_{p,d}}$ and the optimized nodal configuration $\boldsymbol{Z}^\star = \{\boldsymbol{z}_i^\star\}_{i=1,\ldots,N_{p,d}}$ defined as the maximum distance on the sphere between the position of the $i$-th node before and after the optimization,

$$d\left(\boldsymbol{Z}^0, \boldsymbol{Z}^\star\right) := \frac{\pi}{2p} \max_{i=1,\ldots,N_{p,d}} d_{\mathcal{S}^d}\left(\boldsymbol{z}_i^0, \boldsymbol{z}_i^\star\right).$$

We remark that this value is normalized to account for the spherical simplex edge length and the polynomial degree. We observe that since the method is local, small

| $p$ | Initial $\Lambda$ | Optimized $\Lambda$ | $\Lambda^{1/p}$ | $d(\boldsymbol{Z}^0, \boldsymbol{Z}^\star)$ |
|---|---|---|---|---|
| 3 | 2.111 | 2.108 | 1.282 | $5.904 \cdot 10^{-3}$ |
| 4 | 2.692 | 2.587 | 1.268 | $1.336 \cdot 10^{-2}$ |
| 5 | 3.301 | 3.081 | 1.252 | $1.432 \cdot 10^{-2}$ |
| 6 | 3.791 | 3.595 | 1.238 | $8.477 \cdot 10^{-3}$ |
| 7 | 4.391 | 4.143 | 1.225 | $4.754 \cdot 10^{-3}$ |
| 8 | 5.089 | 4.766 | 1.216 | $5.112 \cdot 10^{-3}$ |
| 9 | 5.918 | 5.486 | 1.208 | $3.879 \cdot 10^{-3}$ |
| 10 | 7.158 | 5.921 | 1.195 | $4.417 \cdot 10^{-3}$ |
| 11 | 8.010 | 6.720 | 1.189 | $4.675 \cdot 10^{-3}$ |
| 12 | 8.972 | 7.187 | 1.179 | $3.916 \cdot 10^{-3}$ |
| 13 | 9.591 | 7.253 | 1.165 | $2.303 \cdot 10^{-3}$ |
| 14 | 8.988 | 7.705 | 1.157 | $1.691 \cdot 10^{-3}$ |
| 15 | 10.422 | 8.242 | 1.151 | $2.621 \cdot 10^{-3}$ |
| 16 | 12.438 | 8.729 | 1.145 | $2.909 \cdot 10^{-3}$ |
| 17 | 14.026 | 9.449 | 1.141 | $2.147 \cdot 10^{-3}$ |
| 18 | 14.241 | 9.576 | 1.134 | $2.240 \cdot 10^{-3}$ |
| 19 | 13.242 | 9.802 | 1.128 | $1.116 \cdot 10^{-3}$ |
| 20 | 13.093 | 10.414 | 1.124 | $1.721 \cdot 10^{-3}$ |

Table 6.2: Minimum Lebesgue constant of the optimized nodal distribution in the triangle. For each polynomial degree $p$, $p = 3, \ldots, 20$, we report the Lebesgue constant before and after the optimization, the value $\Lambda^{1/p}$, and the distance between the initial approximation and the optimized distribution.

displacements are undergone, yet the Lebesgue constant improves significantly. The small distances suggest that the optimized nodal distribution is in the same convergence basin as the initial approximation, and, consequently, the optimized nodal distributions preserve the point structure on the spherical simplex, see Sect. 5.3. Thus, to find quasi-optimal Lebesgue minima, we have to consider structurally different nodal distributions as initial approximations, and this is precisely the case of the local minima of the Chen-Babuska functional computed in Chapter 5.

In Table 6.3, we compare our results with those from the literature. Specifically, we compare the nodal sets from Roth (2005) obtained using a genetic algorithm, the nodal sets found by optimizing a proxy of the Lebesgue constant using the approach proposed in Chapter 5, and the nodal sets found by optimizing the Lebesgue constant with the deterministic method proposed in Sect. 6.3. On the one hand, we observe that the quasi-optimal interpolation nodal distributions have smaller Lebesgue con-

| | Lebesgue constant $\Lambda$ | | |
|---|---|---|---|
| $p$ | Roth (2005) | Our method from Chapter 5 | Our method from Sect. 6.3 |
| 3 | 2.108 | 2.112 | 2.108 |
| 4 | 2.587 | 2.692 | 2.587 |
| 5 | 3.081 | 3.301 | 3.081 |
| 6 | 3.595 | 3.791 | 3.595 |
| 7 | 4.143 | 4.391 | 4.143 |
| 8 | 4.766 | 5.089 | 4.766 |
| 9 | 5.486 | 5.918 | 5.486 |
| 10 | 5.921 | 7.085 | 5.921 |
| 11 | 6.724 | 7.266 | 6.720 |
| 12 | 7.187 | 8.660 | 7.187 |
| 13 | 7.253 | 8.877 | 7.253 |
| 14 | 7.706 | 8.988 | 7.705 |
| 15 | 8.243 | 10.306 | 8.242 |
| 16 | | 10.970 | 8.729 |
| 17 | | 14.026 | 9.449 |
| 18 | | 12.344 | 9.576 |
| 19 | | 11.797 | 9.802 |
| 20 | | 12.861 | 10.414 |

Table 6.3: Lebesgue constant $\Lambda$ of different nodal distribution of polynomial degree $p$ in the triangle, $p = 3, \ldots, 20$. Blank spaces indicate not reported results.

stant than the nodal distributions found in Chapter 5. Indeed, given a nodal set, the method proposed in Sect. 6.3 optimizes the Lebesgue constant. On the other hand, our quasi-optimal interpolation nodal distributions feature the same Lebesgue constant as for the nodal sets obtained using a genetic algorithm (Roth, 2005). Thus, our quasi-optimal interpolation nodes coincide with the nodal distributions with the smallest Lebesgue constant up to date.

### Condition number of the Vandermonde matrix

To further assess the interpolation properties of our quasi-optimal nodal sets, we report the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis (Koornwinder, 1975; Dubiner, 1991; Kirby, 2010). In Table 6.4, we compare the condition number for the equispaced distribution, for the nodal sets in Warburton (2006) and Rapetti et al. (2012), and for our quasi-optimal nodal sets. These values indicate that, using double-precision calculations, we should

| | Condition number $\kappa$ | | | |
|---|---|---|---|---|
| $p$ | Equispaced | Warburton (2006) | Rapetti et al. (2012) | Our method |
| 3 | 5.828 | 5.903 | 5.903 | 5.928 |
| 4 | 7.599 | 6.777 | 7.064 | 7.105 |
| 5 | 10.167 | 7.845 | 8.242 | 8.252 |
| 6 | 14.658 | 9.591 | 9.651 | 9.650 |
| 7 | 22.239 | 11.160 | 11.642 | 11.882 |
| 8 | 35.627 | 13.886 | 14.599 | 14.520 |
| 9 | 59.949 | 16.896 | 17.184 | 17.238 |
| 10 | 104.164 | 21.670 | 22.059 | 17.346 |
| 11 | 186.545 | 27.401 | 31.751 | 23.324 |
| 12 | 344.977 | 36.132 | 24.137 | 25.466 |
| 13 | 626.693 | 47.219 | 31.071 | 25.524 |
| 14 | 1 198.655 | 63.679 | 46.113 | 28.403 |
| 15 | 2 194.382 | 85.692 | 30.931 | 30.820 |
| 16 | 4 284.058 | 117.712 | 44.424 | 37.476 |
| 17 | 7 879.557 | 166.716 | 50.798 | 35.678 |
| 18 | 15 597.334 | 247.913 | 55.253 | 41.688 |
| 19 | 28 798.839 | 344.617 | | 42.670 |
| 20 | 57 534.253 | 476.785 | | 45.387 |

Table 6.4: Condition number $\kappa$ of the Vandermonde matrix using different interpolative nodal sets in the triangle for polynomial degree $p$, $p = 3, \ldots, 20$. Blank spaces indicate not reported results.

expect to lose less than three digits of accuracy because all the condition numbers are smaller than 100.

### Interpolation error of benchmark functions

Moreover, we estimate the error of interpolating the following benchmark functions introduced in Chapter 5,

$$\begin{aligned} u_1(\boldsymbol{\lambda}) &= \left(\exp\left(-2\lambda^1\right) - 1\right) \prod_{i=2}^{d+1} \left(2\lambda^i\right), \\ u_2(\boldsymbol{\lambda}) &= \left(\cosh\left(-2\lambda^1\right) - 1\right) \prod_{i=2}^{d+1} \left(2\lambda^i\right), \\ u_3(\boldsymbol{\lambda}) &= \frac{1}{1 + 2\alpha \left\|\boldsymbol{\lambda} - \frac{1}{d+1}\mathbf{1}\right\|_2^2}, \end{aligned} \tag{6.9}$$

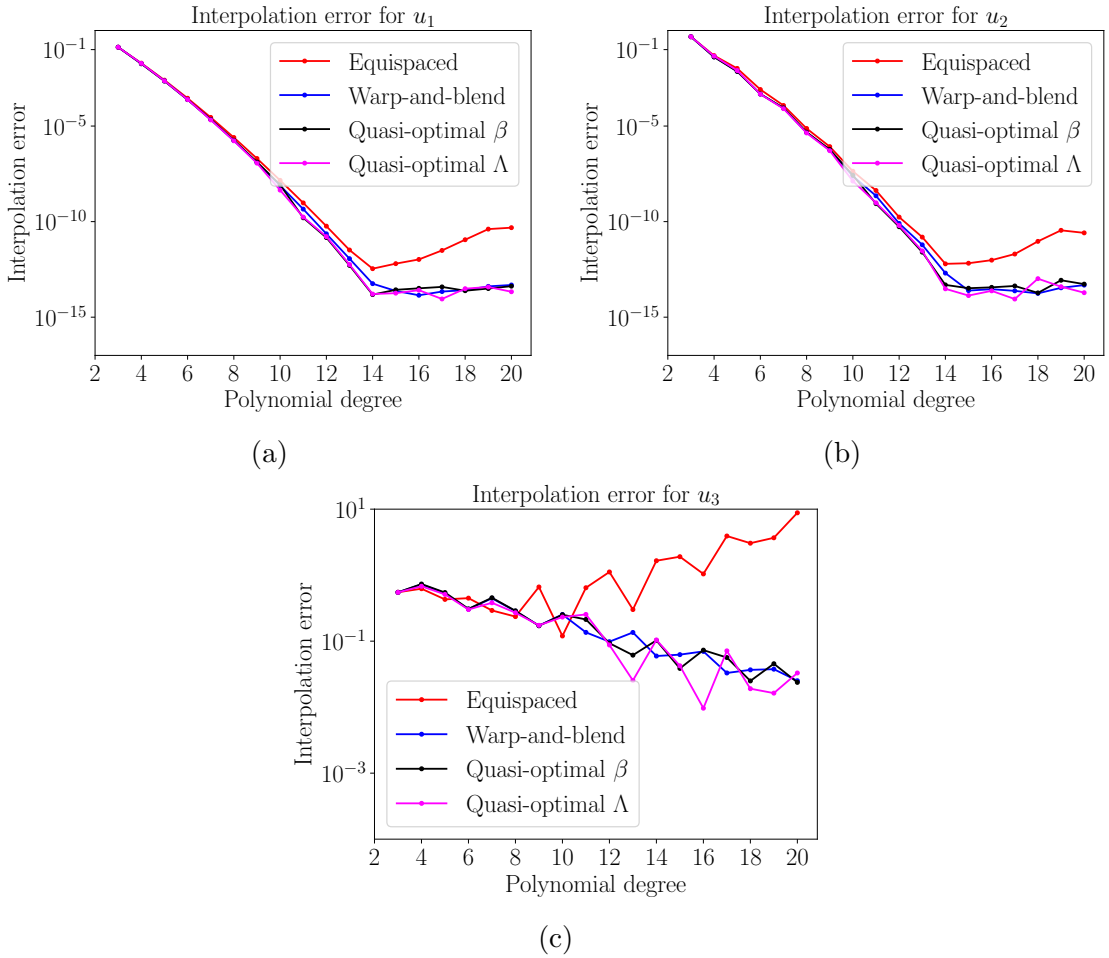



Figure 6.2: Interpolation error in the triangle for different polynomial degrees using equispaced (red), warp-and-blend (blue), our $\beta$ quasi-optimal from Chapter 5 (black), and our Lebesgue quasi-optimal (magenta) nodal distributions for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

with $\alpha = 25$. Functions $u_1$ and $u_2$ are particularly relevant since they are devised to allow equispaced points to achieve uniform convergence, and the third benchmark function $u_3$ is an extension of the Runge function to several dimensions.

The relative interpolation error $\|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i\|_\infty / \|u_i\|_\infty$, $i = 1, 2, 3$, in terms of the polynomial degree is plotted in Fig. 6.2. We also show the error for the equispaced distribution, the nodal sets from Warburton (2006) and the nodal sets computed in Chapter 5 optimizing the Chen-Babuska functional. The interpolations using either nodal sets converge to the functions $u_1$ and $u_2$, see Fig. 6.2(a) and (b). For polynomial degrees $p \geq 15$, the interpolation errors stop the decreasing trend because machine

| $p$ | Initial $\Lambda$ | Optimized $\Lambda$ | $\Lambda^{1/p}$ |
|---|---|---|---|
| 3 | 2.936 | 2.930 | 1.431 |
| 4 | 4.134 | 4.005 | 1.415 |
| 5 | 5.643 | 5.258 | 1.394 |
| 6 | 7.310 | 6.759 | 1.375 |
| 7 | 9.207 | 8.252 | 1.352 |
| 8 | 13.910 | 10.594 | 1.343 |
| 9 | 15.514 | 13.693 | 1.337 |
| 10 | 15.611 | 14.273 | 1.305 |
| 11 | 21.393 | 17.032 | 1.294 |
| 12 | 26.468 | 19.696 | 1.282 |
| 13 | 27.857 | 21.948 | 1.268 |
| 14 | 33.760 | 24.168 | 1.255 |
| 15 | 39.538 | 27.870 | 1.248 |

Table 6.5: Minimum Lebesgue constant of the optimized nodal distribution in the tetrahedron. For each polynomial degree $p$, $p = 3, \dots, 15$, we report the Lebesgue constant before and after the optimization and the value $\Lambda^{1/p}$.

accuracy is reached when estimating the error. For the Runge function $u_3$, we observe in Fig. 6.2(c) that the interpolation error using equispaced points diverges, while with the warp-and-blend or our quasi-optimal nodal configurations the errors seem to be reduced.

### 6.4.3 Quasi-optimal interpolation points in 3D

Analogously to the two-dimensional case, we repeat the same study for nodal distributions in the tetrahedron.

#### Lebesgue constant

In Table 6.5, we report the minimum Lebesgue constant $\Lambda$ found after optimizing each Chen-Babuska minimum. We also list the Lebesgue constant of the initial approximation, that is, the Lebesgue constant of the nodal distribution that leads to the minimum Lebesgue constant. We observe that the improvement of the optimized distributions becomes more apparent as we increase the polynomial degree. Furthermore, we report the values of the sequence $\Lambda^{1/p}$ and observe that it has a decreasing tendency to 1.

| $p$ | Lebesgue constant $\Lambda$ | | | |
|---|---|---|---|---|
| | Chen and Babuška (1996) | Isaac (2020) | Our method from Chapter 5 | Our method from Sect. 6.3 |
| 3 | 2.934 | 2.933 | 2.936 | 2.930 |
| 4 | 4.112 | 4.093 | 4.134 | 4.005 |
| 5 | 5.616 | 5.547 | 5.644 | 5.258 |
| 6 | 7.363 | 7.169 | 7.310 | 6.759 |
| 7 | 9.366 | 9.202 | 8.897 | 8.252 |
| 8 | 12.311 | 12.067 | 12.010 | 10.594 |
| 9 | 15.686 | 15.593 | 15.146 | 13.693 |
| 10 | | 20.623 | 15.611 | 14.273 |
| 11 | | 28.034 | 19.425 | 17.032 |
| 12 | | 38.649 | 23.569 | 19.696 |
| 13 | | 55.143 | 25.701 | 21.948 |
| 14 | | 81.037 | 29.539 | 24.168 |
| 15 | | 118.420 | 36.375 | 27.870 |

Table 6.6: Lebesgue constant $\Lambda$ of different nodal distribution of polynomial degree $p$ in the tetrahedron, $p = 3, \ldots, 15$. Blank spaces indicate not reported results.

In Table 6.6, we compare our results with those from the literature. Specifically, we compare the nodal sets from Chen and Babuška (1996) obtained optimizing a proxy of the Lebesgue constant, the explicit nodal sets from Isaac (2020), the nodal sets found by optimizing a proxy of the Lebesgue constant using the approach proposed in Chapter 5, and the nodal sets found by optimizing the Lebesgue constant with the method proposed in Sect. 6.3. We observe that, for small polynomial degrees, all the nodal distributions feature similar Lebesgue constant. In contrast, for high polynomial degrees, our Lebesgue quasi-optimal nodal distributions have significantly smaller values. In particular, for polynomial degree $p = 15$, we improve an order of magnitude the value from Isaac (2020). In conclusion, our quasi-optimal interpolation nodes have the smallest Lebesgue constant up to date.

### Condition number of the Vandermonde matrix

To further assess the interpolation properties of our quasi-optimal nodal sets, we report the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis (Koornwinder, 1975; Dubiner, 1991; Kirby, 2010). In Table 6.7, we compare the condition number for the equispaced distribution, the explicit

| $p$ | Condition number $\kappa$ | | | |
|---|---|---|---|---|
| | Equispaced | Isaac (2020) | Our method from Chapter 5 | Our method from Sect. 6.3 |
| 3 | 10.248 | 10.505 | 10.537 | 10.447 |
| 4 | 15.392 | 15.817 | 15.878 | 15.640 |
| 5 | 20.245 | 18.766 | 18.763 | 18.908 |
| 6 | 32.997 | 25.851 | 25.803 | 26.318 |
| 7 | 52.301 | 37.339 | 35.490 | 35.277 |
| 8 | 91.944 | 55.905 | 54.849 | 51.521 |
| 9 | 162.216 | 86.315 | 82.432 | 76.600 |
| 10 | 300.740 | 137.967 | 79.217 | 77.593 |
| 11 | 558.909 | 224.111 | 76.962 | 82.866 |
| 12 | 1 064.722 | 371.800 | 101.923 | 101.617 |
| 13 | 2 027.690 | 621.826 | 122.222 | 131.180 |
| 14 | 3 917.567 | 1 052.205 | 142.834 | 141.867 |
| 15 | 7 560.901 | 1 785.689 | 167.841 | 156.978 |

Table 6.7: Condition number $\kappa$ of the Vandermonde matrix using different interpolative nodal sets in the tetrahedron for polynomial degree $p$, $p = 3, \dots, 15$.

nodal sets in Isaac (2020), the nodal sets found in Chapter 5, and the quasi-optimal Lebesgue nodal sets found with the method proposed in Sect. 6.3. For high polynomial degrees, we observe a difference of two orders of magnitude between the values obtained with our method and the explicit nodal distributions. Moreover, using our nodes, all the condition numbers are smaller than 100 and, thus, we should expect to lose less than three digits of accuracy using double-precision calculations.

### Interpolation error of benchmark functions

Finally, we estimate the error of interpolating the benchmark functions in Eq. (6.9). The relative interpolation error $\|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i\|_\infty / \|u_i\|_\infty$, $i = 1, 2, 3$, in terms of the polynomial degree is plotted in Fig. 6.3. We also show the error for the equispaced distribution, the nodal sets from Isaac (2020) and the nodal sets computed in Chapter 5 optimizing the Chen-Babuska functional. For all the considered nodal distributions, the interpolation error for functions $u_1$ and $u_2$ decreases as the polynomial degree increases, see Fig. 6.3(a) and (b). For the Runge function $u_3$, we observe in Fig. 6.3(c) that the interpolation error using equispaced points diverges, while with the other nodal configurations the errors seem to be reduced.

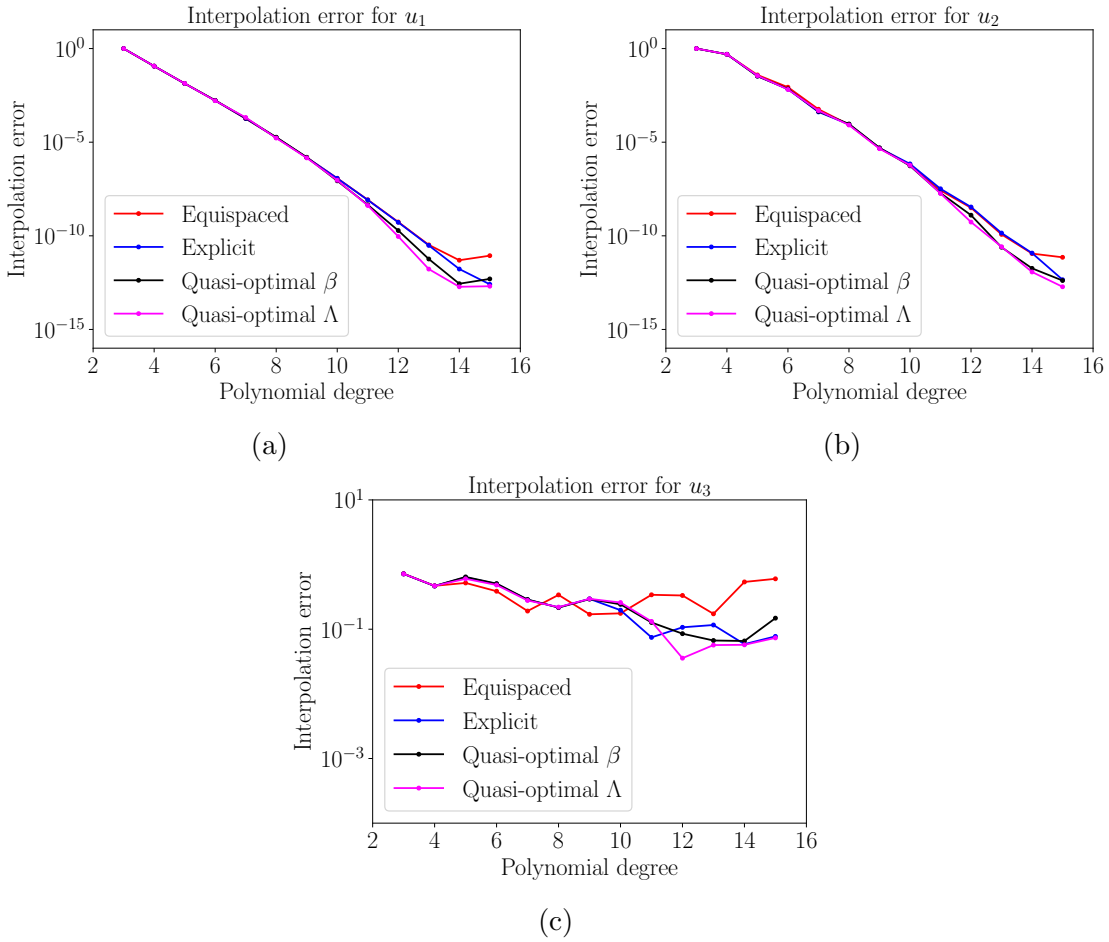



Figure 6.3: Interpolation error in the tetrahedron for different polynomial degrees using equispaced (red), explicit (Isaac, 2020) (blue), our $\beta$ quasi-optimal from Chapter 5 (black), and our Lebesgue quasi-optimal (magenta) nodal distributions for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

### 6.4.4 Quasi-optimal interpolation points in 4D

In this section, we repeat the same study for nodal distributions in the pentatope.

#### Lebesgue constant

In Table 6.8, we report the minimum Lebesgue constant $\Lambda$ found after optimizing each Chen-Babuska minimum. Comparing the values of the Lebesgue constant before and after the optimization procedure, we observe a further improvement for high polynomial degrees. Finally, we report the values of the sequence $\Lambda^{1/p}$ and observe that it has a decreasing tendency to 1.

| $p$ | Initial $\Lambda$ | Optimized $\Lambda$ | $\Lambda^{1/p}$ |
|---|---|---|---|
| 3 | 4.169 | 3.760 | 1.555 |
| 4 | 6.130 | 5.964 | 1.563 |
| 5 | 8.865 | 8.488 | 1.534 |
| 6 | 12.532 | 11.683 | 1.506 |
| 7 | 17.568 | 15.629 | 1.481 |
| 8 | 22.310 | 21.053 | 1.464 |
| 9 | 31.328 | 27.500 | 1.445 |
| 10 | 46.505 | 35.668 | 1.430 |

Table 6.8: Minimum Lebesgue constant of the optimized nodal distribution in the pentatope. For each polynomial degree $p$, $p = 3, \dots, 10$, we report the Lebesgue constant before and after the optimization and the value $\Lambda^{1/p}$.

| | Lebesgue constant $\Lambda$ | | | |
|---|---|---|---|---|
| $p$ | Equispaced | Isaac (2020) | Our method from Chapter 5 | Our method from Sect. 6.3 |
| 3 | 3.880 | 4.200 | 4.169 | 3.760 |
| 4 | 6.244 | 6.147 | 6.130 | 5.964 |
| 5 | 10.918 | 8.839 | 8.865 | 8.488 |
| 6 | 19.224 | 12.502 | 12.532 | 11.683 |
| 7 | 34.085 | 17.231 | 17.085 | 15.629 |
| 8 | 60.859 | 23.366 | 22.310 | 21.053 |
| 9 | 109.427 | 32.646 | 31.328 | 27.500 |
| 10 | 198.083 | 45.768 | 40.873 | 35.668 |

Table 6.9: Lebesgue constant $\Lambda$ of different nodal distribution of polynomial degree $p$ in the pentatope, $p = 3, \dots, 10$.

In Table 6.9, we compare our results with those from the literature. Specifically, we compare the explicit nodal sets from Isaac (2020), the nodal sets found by optimizing a proxy of the Lebesgue constant using the approach proposed in Chapter 5, and the nodal sets found by optimizing the Lebesgue constant with the method proposed in Sect. 6.3. For small polynomial degrees, all the nodal distributions possess similar Lebesgue constant, yet for high polynomial degrees, our Lebesgue quasi-optimal nodal distributions present significantly smaller values. We highlight that, for polynomial degrees 9 and 10, the method presented in Chapter 5 is not able to explore all the graph vertices within the time limitation that the computational resources are available. Hence, there may exist nodal distributions with even smaller Lebesgue

| $p$ | Condition number $\kappa$ | | | |
|---|---|---|---|---|
| | Equispaced | Isaac (2020) | Our method from Chapter 5 | Our method from Sect. 6.3 |
| 3 | 16.341 | 17.328 | 17.222 | 16.055 |
| 4 | 30.487 | 32.801 | 32.365 | 30.786 |
| 5 | 43.949 | 44.850 | 43.593 | 43.092 |
| 6 | 73.139 | 65.410 | 63.335 | 62.355 |
| 7 | 125.524 | 108.158 | 101.359 | 97.771 |
| 8 | 229.769 | 194.218 | 144.363 | 164.940 |
| 9 | 428.923 | 358.026 | 239.176 | 236.985 |
| 10 | 823.800 | 681.022 | 415.212 | 261.944 |

Table 6.10: Condition number $\kappa$ of the Vandermonde matrix using different interpolative nodal sets in the pentatope for polynomial degree $p$, $p = 3, \ldots, 10$.

constant. Even so, our nodal sets feature the smallest Lebesgue constant up to date.

### Condition number of the Vandermonde matrix

Following, we analyze the condition number of the Vandermonde matrix using the Koornwinder-Dubiner orthogonal basis (Koornwinder, 1975; Dubiner, 1991; Kirby, 2010). In Table 6.10, we compare the condition number for the equispaced distribution, the explicit nodal sets in Isaac (2020), the nodal sets found in Chapter 5, and the quasi-optimal Lebesgue nodal sets found with the method proposed in Sect. 6.3. Since all the values are below 1000, we should expect to lose less than four digits of accuracy. Nonetheless, the values using our quasi-optimal Lebesgue distributions seem to grow slower than when using any other nodal set.

### Interpolation error of benchmark functions

Lastly, we estimate the error of interpolating the benchmark functions in Eq. (6.9). The relative interpolation error $\|u_i - \mathcal{I}_{\boldsymbol{Z}} u_i\|_\infty / \|u_i\|_\infty$, $i = 1, 2, 3$, in terms of the polynomial degree using different nodal distributions is plotted in Fig. 6.4. We show the error for the equispaced distribution, the nodal sets from Isaac (2020), the nodal sets computed in Chapter 5 optimizing the Chen-Babuska functional, and our quasi-optimal Lebesgue nodal sets. For all the considered nodal distributions, the interpolation error for functions $u_1$ and $u_2$ decreases as the polynomial degree increases, see

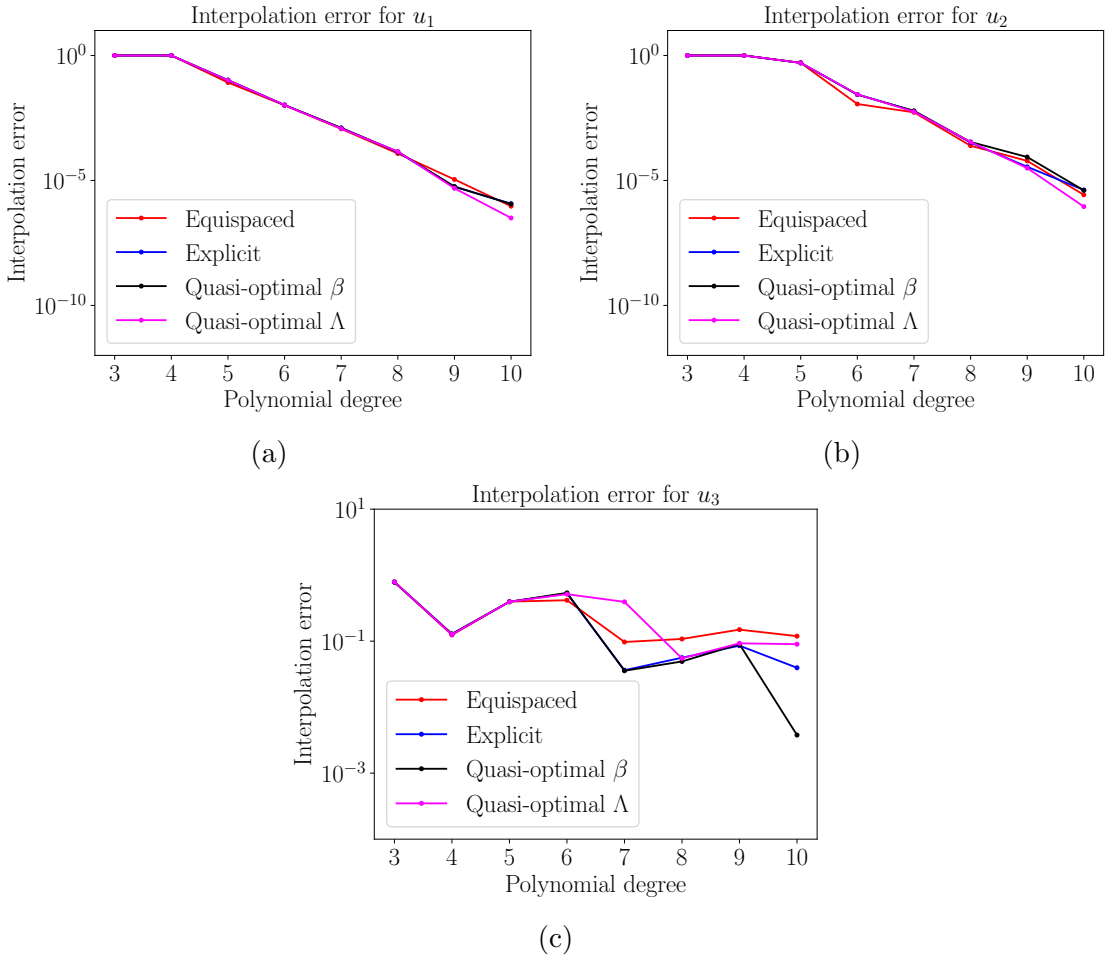



Figure 6.4: Interpolation error in the pentatope for different polynomial degrees using equispaced (red), explicit (Isaac, 2020) (blue), our $\beta$ quasi-optimal from Chapter 5 (black), and our Lebesgue quasi-optimal (magenta) nodal distributions for functions: (a) $u_1$; (b) $u_2$; and (c) $u_3$.

Fig. 6.4(a) and (b). For the Runge function $u_3$, we observe in Fig. 6.4(c) that for the four nodal families the curves are not monotone and the errors are large.

### 6.4.5 Quasi-optimal geometry interpolation

In this example, we use our quasi-optimal nodal distributions to interpolate a subdivision limit model, see Chapter 3. We consider a discretization of a sphere of unit radius with linear triangular elements. This linear mesh model determines a $\mathcal{C}^1$-continuous subdivision limit model interpolating the given data points. To interpolate this curved limit model, we use the method proposed in Chapter 3 to generate piece-

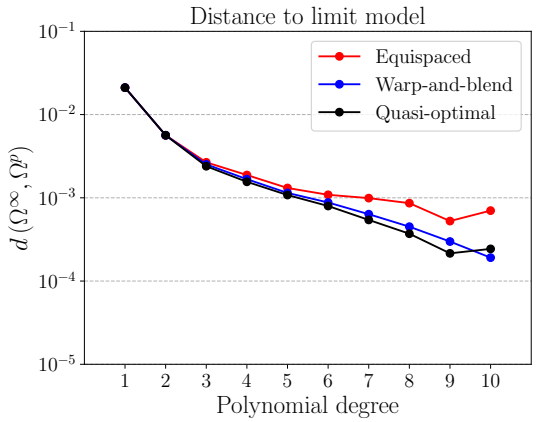



Figure 6.5: Convergence of the distance to the limit model for the sphere meshes of polynomial degree $p$, $p = 1, \ldots, 10$, with equispaced (red), explicit (blue) and quasi-optimal Lebesgue (black) nodal distributions.

wise polynomial representations of polynomial degree $p$. As described in Sect. 3.6.3, we approximate the distance between the piecewise polynomial model $\Omega^p$ and the limit model $\Omega^\infty$ with a fine sampling of $N_d$ points on each facet $f$,

$$d\left(\Omega^\infty, \Omega^p\right) = \max_{f=1,\ldots,n_f} \max_{j=1,\ldots,N_d} \left\| \phi_f^\infty\left(\boldsymbol{\lambda}_j\right) - \phi_f^p\left(\boldsymbol{\lambda}_j\right) \right\|_2 .$$

In Fig. 6.5, we plot the distance between the piecewise polynomial representation and the limit model in terms of the polynomial degree for different nodal distributions. We observe that the explicit nodal distribution from Warburton (2006), in blue, and our quasi-optimal nodal sets, in black, provide very similar results. Accordingly, the explicit nodal distributions (Warburton, 2006) are well-suited to interpolate 2D geometry for up to moderate polynomial degree. Nonetheless, it is not until we compare the explicit nodal distributions and our quasi-optimal Lebesgue nodal sets that we can assess the suitability of explicit nodes. However, for high polynomial degrees, the condition number of the Vandermonde matrix using our quasi-optimal nodal distributions is approximately one order of magnitude smaller than when using these explicit nodes, see Table 6.4. Hence, we advocate for using our quasi-optimal interpolation nodal distributions to represent 2D geometry since they are closer to the interpolated geometry, have a smaller Lebesgue constant, and the condition number of the Vandermonde matrix is much smaller.

## 6.5 Concluding remarks

We have proposed a specific-purpose deterministic method to optimize the Lebesgue constant of a given nodal distribution in the $d$-dimensional simplex using the point coordinates as design variables. The method preserves the structure of the initial approximation and consistently exploits first-order derivatives of the Lebesgue function. In a sequential linear programming approach, at each iteration, we compute a candidate descent direction using the first-order derivatives of the Lebesgue function with respect to the node coordinates evaluated at maximum points in previous iterations. The derivatives are computed analytically and the Lebesgue constant is not differentiated in any case. We consider an advantageous representation of the nodes on the orthant of the $(d+1)$-sphere to ensure smooth dynamics during the optimization procedure.

The results show that the computed nodal distributions present quasi-optimal Lebesgue constants. In 1D, the method computes the optimal Lebesgue nodes. In 2D, we reproduce the literature results up to polynomial degree 15, and improve the current values up to polynomial degree 20. In 3D and 4D and up to polynomial degree 15 and 10, respectively, we find and report the nodal distributions with the lowest Lebesgue constant up to date.

There is numerical evidence that our quasi-optimal Lebesgue nodes are suitable for interpolation. According to Bloom et al. (1992); Roth (2005), we must check that $\Lambda^{1/p}$ converges to one to ensure uniform convergence of the interpolator to the target function. Using our quasi-optimal nodal distributions, for increasing degrees, we have observed that $\Lambda^{1/p}$ seems to converge to one. Hence, we have numerically checked their interpolation suitability.

The results suggest that explicit nodal distributions might be well-suited to interpolate geometry for low polynomial degrees. Nevertheless, it is not until we compare explicit and our quasi-optimal Lebesgue nodal distributions that we can draw such conclusions. However, using the Koornwinder-Dubiner orthogonal basis, the condition number of the Vandermonde matrix using our nodal set is reduced by orders of magnitude in comparison to using explicit nodal configurations. Thus, we advocate for using our quasi-optimal Lebesgue nodal sets.

In conclusion, in combination with the method proposed in Chapter 5, we have devised a two-stage method to compute quasi-optimal Lebesgue nodal distributions. First, we explore the local minima of a proxy of the Lebesgue constant with the

method described in Chapter 5. These nodal distributions are good candidates for optimal Lebesgue configurations since they do not feature a grid-like structure and cluster the nodes towards the boundary. Second, we deterministically optimize the Lebesgue constant of a nodal distribution by consistently exploiting the first-order derivatives of the Lebesgue function with respect to the node coordinates. The results are nodal distributions for high polynomial degrees up to 4D with no grid-like structure and with the lowest Lebesgue constant up to date. In perspective, we expect these quasi-optimal Lebesgue nodal distributions to be useful for high-order unstructured methods and for representing and modeling complex curved geometry.

# Chapter 7

# Computing interpolation-aware numerical quadratures

## 7.1 Introduction

The solution of linear problems with the mass matrix as the system matrix is a key capability in computational methods. These linear problems appear in many problems such as in explicit time integration and $\mathcal{L}_2$ projections to approximate functions. To accelerate the solution of these problems, it is possible to collocate the interpolation points with the quadrature points. Thus, the resulting mass matrix is diagonal, a matrix structure that leads to linear systems that can be solved efficiently. The collocation of the interpolation points with the quadrature points has many applications. For instance, it is a key aspect of the spectral element method (Patera, 1984), it leads to evaluating the functions only on the interpolation points (Hesthaven and Warburton, 2007; Kopriva and Gassner, 2010), it improves the numerical stability for specific finite element methods (Jameson et al., 2012; Williams and Jameson, 2013; Williams, 2013; Witherden and Vincent, 2014), and it is also of interest in space-time discretizations (Williams et al., 2020).

When the computational domain is discretized using tensor-product-like shapes such as quadrilaterals or hexahedra, it is standard to use Legendre-Gauss-Lobatto points since they have both good interpolation and integration properties. However, to deal with complex geometry, simplicial unstructured meshes have demonstrated

to be better suited than tensor-product-like shapes. Thus, for time-marching and space-time discretizations, it is of major interest finding nodal distribution on the simplex featuring good properties regarding not only function interpolation but also function integration (Williams et al., 2020).

Bearing in mind the previous applications, we are interested in collocating interpolation points with integration points up to the four-dimensional simplex. To do so, we require them to satisfy three properties. First, in order to be invariant of the element orientation, we require symmetric nodal distributions. Accordingly, the integration weights should also be symmetric. Second, we require closed sets with a simplicial number of points. This is so because we want to facilitate the implementation of conformal methods with a unisolvent nodal distribution for interpolation. Third, we require positive integration weights to guarantee that the integral of positive functions is positive.

There are several numerical quadratures for the simplex, but they are not ready for 4D symmetric numerical integration with positive weights and a simplicial number of closed points. Symmetric numerical quadratures have been extensively studied in the triangle (Lyness and Jespersen, 1975; Zhang et al., 2009; Wandzura and Xiao, 2003; Taylor et al., 2007; Williams et al., 2014; Witherden and Vincent, 2015), the tetrahedron (Keast, 1986; Zhang et al., 2009; Shunn and Ham, 2012; Williams et al., 2014; Witherden and Vincent, 2015), and the higher dimensional simplex (Hammer et al., 1956; Hammer and Stroud, 1956; Silvester, 1970; Grundmann and Möller, 1978; Dan and Wang, 2009; Xiao and Gimbutas, 2010; Williams et al., 2020). Unfortunately, some of these symmetric numerical quadratures in the four-dimensional simplex either do not feature a simplicial number of points (Hammer et al., 1956; Hammer and Stroud, 1956; Grundmann and Möller, 1978; Dan and Wang, 2009; Xiao and Gimbutas, 2010), or do feature a simplicial number of points but have negative weights (Silvester, 1970) or open points (Williams et al., 2020).

Alternatively, there exist other approaches specifically devised to exploit the advantages of the spectral element method, but they do not meet all the aforementioned requirements. Many of these approaches consider Fekete nodes for interpolation. Fekete nodes are a set of points with good interpolation properties since they aim to maximize the determinant of the Vandermonde matrix. Thus, Fekete nodes might be used as an interpolation nodal set in combination with another set of points for integration. If Fekete nodes are also used as integration points, the mass matrix is

diagonal but the weights might be negative (Taylor and Wingate, 1999, 2000). Alternatively, two different sets of points might be used — Fekete points for interpolation and Gauss points for integration. Then, if the function values at the interpolation nodes are known, the function values at the integration points can be obtained through matrix multiplication (Pasquetti and Rapetti, 2006, 2013). In this case, the mass matrix is no longer diagonal. A different approach is presented in Giraldo and Taylor (2006). Therein, the polynomial space used for interpolation is enriched with additional interior modes vanishing at the boundary. These extra degrees of freedom are used to attain high-order quadratures. However, the number of points is larger than the target simplicial number.

Summarizing, symmetric numerical integration with positive weights and a simplicial number of closed points in the four-dimensional simplex has not been considered. Accordingly, the goal of this chapter is to optimize the collocation of the interpolation and integration points subject to fulfilling the previous requirements. A nodal distribution featuring good interpolation and integration properties might be used both for interpolation and integration purposes in high-order methods. Hence, it will enable accomplishing not only the computational efficiency of dealing with a diagonal mass matrix, but also the reduced approximation error of a high-order quadrature.

To meet our goal, the main contribution of this chapter is to formulate and minimize the interpolation error of a symmetric and closed nodal distribution subject to integrating exactly high-order polynomials with positive integration weights. The design variables are the weights and coordinates of the nodal representatives. The interpolation error is approximated with the twice-differentiable vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials. Because we consider symmetric distributions, the constraints only impose exact integration of high-order symmetry-invariant polynomials. Solving the constrained problem with standard optimization methods, we obtain preliminary results in the two-, three-, and four-dimensional simplex.

The rest of the chapter is organized as follows. First, in Sect. 7.2, we present some preliminaries. Then, in Sect. 7.3, we propose our constrained minimization problem. Next, in Sect. 7.4, we report preliminary results and, finally, in Sect. 7.5, we present some concluding remarks.

## 7.2 Preliminaries

Following, we recall some notation and definitions introduced in Chapter 2 used throughout this chapter regarding the parameterization of symmetric nodal distributions in the simplex, Sect. 7.2.1, and the Chen-Babuska proxy of the Lebesgue constant, Sect. 7.2.2. Then, in Sect. 7.2.3, we give basic results about invariant theory to compute symmetry-invariant polynomials in the simplex.

First, let us define the standard simplex $K_S^d \subset \mathbb{R}^d$ given by the set of points

$$K_S^d = \{\boldsymbol{x} \in \mathbb{R}^d \colon \sum_{i=1}^{d} x_i = 1,\ x_i \geq 0\ \forall i = 1, \ldots, d\}. \tag{7.1}$$

This set describes the domain of barycentric coordinates of points in a $(d-1)$-dimensional simplex. We also introduce the set $\mathcal{S}_d$, the group of permutations of the set $\{1, \ldots, d\}$. We note that the standard simplex $K_S^d$ is invariant under any $\rho \in \mathcal{S}_d$, $\rho\left(K_S^d\right) = K_S^d$.

### 7.2.1 Symmetric point distribution

In this section, we recall the notation used to describe a symmetric point distribution $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1,\ldots,N_{p,d}}$ in the simplex, see details in Sect. 2.1. The coordinates of a subset of nodes are enough to describe the whole nodal set. Specifically, the design variables are the $n$ degrees of freedom encoded in a vector $\boldsymbol{y} \in \mathbb{R}^n$ describing the position of the $m$ nodal representatives. The $n_i$ degrees of freedom of representative node $i$ are encoded in the vector $\boldsymbol{y}_i = (y_i^1, \ldots, y_i^{n_i})$, and these values should satisfy

$$\begin{aligned} &0 < y_i^j < 1, \text{ for } j = 1, \ldots, n_i, \\ &\sum_{j=1}^{n_i} y_i^j < 1, \end{aligned}$$

to describe a valid symmetric point in the simplex.

The barycentric coordinates $\boldsymbol{\lambda}_i$ of the $i$-th node representative are obtained via the mapping $\sigma_1$, $\sigma_1(\boldsymbol{y}_i) = \boldsymbol{\lambda}_i$, and the full nodal distribution $\boldsymbol{Z}$ is the image by mapping $\boldsymbol{\sigma}$ of the whole vector of degrees of freedom, $\boldsymbol{Z} = \boldsymbol{\sigma}(\boldsymbol{y})$. The number of elements in the equivalence class $[\boldsymbol{\lambda}_i]$ is called the multiplicity of the node and is denoted as $\mu_i$.

### 7.2.2 Chen-Babuska functional

A proxy of the Lebesgue constant is given by the square of the vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials (Chen and Babuška, 1995),

$$\beta(\boldsymbol{y}) := \frac{1}{\mathrm{vol}(K^d)} \int_{K^d} \sum_{i=1}^{N_{p,d}} \phi_i^2(\boldsymbol{x}; \boldsymbol{\sigma}(\boldsymbol{y})) \, \mathrm{d}\boldsymbol{x}.$$

Remarkably, there exist analytical expressions for evaluating the functional value and the first and second derivatives with no need for evaluating integrals, see Appendix D. Accordingly, this functional can be optimized using second-order optimization methods.

### 7.2.3 Symmetric polynomials

To compute a symmetric numerical quadrature, one approach consists in finding integration points and weights such that the $\binom{d+q}{d}$ elements of a basis of polynomial degree $q$ are integrated exactly. Remarkably, not all these constraints are needed for computing symmetric quadratures, only the generators of the ring of symmetric polynomials have to be considered (Maeztu and Sáinz de La Maza, 1995; Wandzura and Xiao, 2003; Chuluunbaatar et al., 2022). Following, we detail how to generate a basis of the ring of symmetric polynomials and preliminary results about integration in the simplex.

Let $p \in \mathbb{R}[\boldsymbol{x}] = \mathbb{R}[x_1, \ldots, x_d]$ be a polynomial defined in the $d$-dimensional simplex $K^d$. By an affine change of variable, we express $p$ in barycentric coordinates as $p(\boldsymbol{\lambda}) = p(\lambda_1, \ldots, \lambda_{d+1})$, with $\sum_{i=1}^{d+1} \lambda_i = 1$. We say $p \in \mathbb{R}[\boldsymbol{\lambda}]$ is symmetry invariant if it is invariant under every permutation $\rho \in \mathcal{S}_{d+1}$ of the variables. As an example, the $k$-th power sum $s_k(\boldsymbol{\lambda}) := \lambda_1^k + \ldots + \lambda_{d+1}^k$ is a symmetry invariant polynomial. Note that $s_1(\boldsymbol{\lambda}) = 1$. The key theoretical results used in our formulation are summarized in the following theorems.

**Theorem 7.1.** *(Sturmfels, 2008) The ring of symmetry invariant polynomials* $\mathbb{C}[\boldsymbol{\lambda}]^{\mathcal{S}_{d+1}}$ *is generated by the first* $d+1$ *power sums,*

$$\mathbb{C}[\boldsymbol{\lambda}]^{\mathcal{S}_{d+1}} = \mathbb{C}[s_1, \ldots, s_{d+1}].$$

Consequently, a symmetry invariant polynomial $p(\boldsymbol{\lambda})$ of degree $q$ can be expressed as a linear combination of products of power sums. More precisely, let

$\boldsymbol{\alpha} = (\alpha_2, \ldots, \alpha_{d+1})$, $\alpha_i \in \mathbb{N}$, and denote by $\phi_{\boldsymbol{\alpha}} = s_2^{\alpha_2} \cdots s_{d+1}^{\alpha_{d+1}}$. The set

$$\mathcal{I} = \{\phi_{\boldsymbol{\alpha}} : \alpha_i \in \mathbb{N}, 2\alpha_2 + \ldots + (d+1)\,\alpha_{d+1} \leq q\}$$

constitutes a basis of the set of symmetry invariant polynomials up to degree $q$.

**Theorem 7.2.** *(Wandzura and Xiao, 2003) For any function $f(\boldsymbol{\lambda})$ defined in $K_S^{d+1}$,*

$$\int_{K_S^{d+1}} f(\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{\lambda} = \int_{K_S^{d+1}} \bar{f}(\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{\lambda},$$

*where $\bar{f}$ denotes the Reynolds average defined as*

$$\bar{f}(\boldsymbol{\lambda}) = \frac{1}{|\mathcal{S}_{d+1}|} \sum_{\rho \in \mathcal{S}_{d+1}} f(\rho(\boldsymbol{\lambda})).$$

**Theorem 7.3.** *(Wandzura and Xiao, 2003) The Reynolds average $\bar{f}(\boldsymbol{\lambda})$ is symmetry invariant.*

Combining these three results, the integral of a polynomial of up to degree $q$ can be computed as a linear combination of the integrals of the elements in $\mathcal{I}$. Thus, to find a symmetric numerical quadrature such that it exactly integrates polynomials of degree $q$, we need not impose the exact integration for all the $\binom{d+q}{d}$ elements of a basis of the polynomial space $\mathcal{P}_p^d$, yet we only need to consider exact integration for the elements in $\mathcal{I} \subseteq \mathcal{P}_p^d$.

We analytically compute the integrals of the elements in $\mathcal{I}$. Indeed, the integral of a polynomial $\phi_{\boldsymbol{\alpha}} \in \mathcal{I}$ reduces to a sum of integrals of the form

$$\int_{K_S^{d+1}} \lambda_1^{\bar{\alpha}_1} \cdots \lambda_{d+1}^{\bar{\alpha}_{d+1}}\,\mathrm{d}\boldsymbol{\lambda}, \tag{7.2}$$

with $\sum_{i=1}^{d+1} \lambda_i = 1$. The exponents $\bar{\alpha}_i$ are linear combinations of the exponents $\alpha_j$ defining the polynomial $\phi_{\boldsymbol{\alpha}}$. Remarkably, the integral Eq. (7.2) can be analytically computed using the following result.

**Theorem 7.4.** *(Grundmann and Möller, 1978) Let $K_S^{d+1}$ be the standard simplex defined in Eq. (7.1), and $\boldsymbol{\lambda} = \left(\lambda_1, \ldots, \lambda_d, 1 - \sum_{i=1}^d \lambda_i\right)$. Then,*

$$\int_{K_S^{d+1}} \lambda_1^{\bar{\alpha}_1} \cdots \lambda_d^{\bar{\alpha}_d} \left(1 - \sum_{i=1}^{d} \lambda_i\right)^{\bar{\alpha}_{d+1}} \mathrm{d}\boldsymbol{\lambda} = \frac{\bar{\alpha}_1! \cdots \bar{\alpha}_d!\,\bar{\alpha}_{d+1}!}{(d + \bar{\alpha}_1 + \cdots + \bar{\alpha}_d + \bar{\alpha}_{d+1})!}.$$

In conclusion, to compute the integral of $\phi_{\boldsymbol{\alpha}}$, it is enough to evaluate an analytical expression only in terms of the exponents $\alpha_2, \ldots, \alpha_{d+1}$. Furthermore, we highlight that the $r$-th power $\phi_{\boldsymbol{\alpha}}^r$ is equal to $\phi_{r\boldsymbol{\alpha}}$, $\phi_{\boldsymbol{\alpha}}^r = \phi_{r\boldsymbol{\alpha}}$. In particular, computing the square of the $\mathcal{L}_2$-norm of $\phi_{\boldsymbol{\alpha}}$ translates into computing the integral of $\phi_{2\boldsymbol{\alpha}}$.

## 7.3 Interpolation-aware numerical quadratures

Following, we propose a method to minimize the interpolation error of a symmetric and closed nodal distribution subject to integrating exactly high-order polynomials with positive integration weights. In Sect. 7.3.1, we describe our formulation of the constrained optimization problem, and in Sect. 7.3.2 we detail some aspects regarding the numerical implementation.

### 7.3.1 Constrained optimization formulation

To compute a numerical quadrature with points $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_i\}_{i=1,\dots,N_{p,d}}$ and weights $\boldsymbol{W} = \{w_i\}_{i=1,\dots,N_{p,d}}$, it is standard to solve a constrained non-linear optimization problem using the point coordinates and weights as design variables. Since we consider close and symmetric quadratures with a simplicial number of points, the set of points $\boldsymbol{\xi}$ is parametrized by a vector of degrees of freedom $\boldsymbol{y}$ as described in Sect. 7.2.1. Moreover, points in the same equivalence class have the same weight. Thus, only the subset of the weights corresponding to the point representatives has to be considered. Therefore, the actual variables are $\boldsymbol{y} \in \mathbb{R}^n$ and $\tilde{\boldsymbol{w}} \in \mathbb{R}^m$.

We distinguish three kinds of constraints to be satisfied. First, the degrees of freedom should determine a valid point distribution, see Sect. 7.2.1. For the $i$-th nodal representative with $n_i$ degrees of freedom, $i = 1, \dots, m$, the conditions are

$$\begin{aligned} &0 < y_i^j < 1, \text{ for } j = 1, \dots, n_i, \\ &\sum_{j=1}^{n_i} y_i^j < 1. \end{aligned}$$

Second, the integration weights should be positive to ensure that the numerical integration of a positive function is positive. To ease the problem resolution, we reduce the number of constraints by introducing the change of variables $\tilde{w}_i = e^{s_i}$, for $i = 1, \dots, m$. Now, variables $\boldsymbol{s}$ are unconstrained, yet $\tilde{\boldsymbol{w}}$ always remain positive.

Third, the quadrature should integrate exactly polynomials up to degree $q$. As discussed in Sect. 7.2.3, we only need to consider the elements in the set of symmetry invariant polynomials $\mathcal{I}$. Specifically, to exactly integrate polynomials of degree $q$, for each element $\phi_{\boldsymbol{\alpha}} \in \mathcal{I}$, we impose

$$\sum_{i=1}^{m} \phi_{\boldsymbol{\alpha}} \left(\sigma_1 \left(\boldsymbol{y}_i\right)\right) e^{s_i} \mu_i = \int_{K_S^{d+1}} \phi_{\boldsymbol{\alpha}} \left(\boldsymbol{\lambda}\right) \mathrm{d}\boldsymbol{\lambda},$$

where $\sigma_1$ maps the degrees of freedom onto the barycentric coordinates, and $\mu_i$ is the multiplicity of the $i$-th point representative, see Sect. 7.2.1. We analytically compute the integral on the right-hand side of the equation, see details in Sect. 7.2.3.

Finally, the objective function of the problem assesses the interpolation properties of the point distribution. To do so, we consider a proxy of the Lebesgue constant, namely the twice-differentiable Chen-Babuska functional,

$$\beta\left(\boldsymbol{y}\right) = \frac{1}{\mathrm{vol}\left(K^d\right)} \int_{K^d} \sum_{i=1}^{N_{p,d}} \phi_i^2\left(\boldsymbol{x}; \boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right) \mathrm{d}\boldsymbol{x}.$$

In conclusion, the constrained nonlinear problem reads as

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^n,\, \boldsymbol{s}\in\mathbb{R}^m} \quad & \beta\left(\boldsymbol{y}\right) \\ \text{s.t.} \quad & 0 \leq \boldsymbol{y}_i \leq 1, \quad \text{for } i = 1, \ldots, m, \\ & \sum_{j=1}^{n_i} y_i^j < 1, \quad \text{for } i = 1, \ldots, m, \\ & \sum_{i=1}^{m} \phi_{\boldsymbol{\alpha}}\left(\sigma_1\left(\boldsymbol{y}_i\right)\right) e^{s_i} \mu_i = \int_{K_S^{d+1}} \phi_{\boldsymbol{\alpha}}\left(\boldsymbol{\lambda}\right) \mathrm{d}\boldsymbol{\lambda}, \quad \text{for } \phi_{\boldsymbol{\alpha}} \in \mathcal{I}. \end{aligned} \tag{7.3}$$

### 7.3.2 Numerical implementation

The constrained nonlinear problem in Eq. (7.3) is solved by means of a trust-region interior point method (Byrd et al., 1999) implemented in the SciPy library (Jones et al., 2001). This interior point algorithm writes the problem in a lagrangian form and solves a sequence of equality-constrained barrier problems. Each of these sub-problems, in turn, is solved using a trust-region sequential quadratic programming method (Lalee et al., 1998; Nocedal and Wright, 2006).

To ensure smooth point dynamics, we would like to provide the algorithm with a maximum trust-region radius in terms of the edge-length of the spherical simplex and the polynomial degree $p$. Unfortunately, this feature is not supported in the current implementation. Alternatively, we solve a sequence of problems within a tighter domain, and we continue the solution between iterations.

More precisely, consider an initial approximation of the integration points $\boldsymbol{y}^{(0)}$, and denote by $\varepsilon$ the maximum allowed trust-region radius. We set $\varepsilon$ to a value accounting for the edge-length of the spherical simplex and point resolution in terms of the polynomial degree, $\varepsilon = \frac{\pi}{2}\frac{1}{2p}$. Then, to enforce smooth dynamics, we modify

Eq. (7.3) and replace the constraints $0 \leq \boldsymbol{y}_i \leq 1$ with

$$\max\left(0, \boldsymbol{y}_i^{(0)} - \varepsilon\right) \leq \boldsymbol{y}_i \leq \min\left(1, \boldsymbol{y}_i^{(0)} + \varepsilon\right).$$

Thus, the degrees of freedom describing the point coordinates can only move in an interval of length $2\varepsilon$ with respect to the initial approximation. Next, we solve the constrained nonlinear problem and obtain a solution $\boldsymbol{y}^{(1)}$. This solution becomes the initial approximation for a new subproblem with an updated set of bounds. At stage $k$ of the algorithm, the problem to be solved reads as

$$\begin{aligned}
\min_{\boldsymbol{y}\in\mathbb{R}^n,\, \boldsymbol{s}\in\mathbb{R}^m} \quad & \beta\left(\boldsymbol{y}\right) \\
\text{s.t.} \quad & \max\left(0, \boldsymbol{y}_i^{(k-1)} - \varepsilon\right) \leq \boldsymbol{y}_i \leq \min\left(1, \boldsymbol{y}_i^{(k-1)} + \varepsilon\right), \quad \text{for } i = 1, \dots, m, \\
& \sum_{j=1}^{n_i} y_i^j < 1, \quad \text{for } i = 1, \dots, m, \\
& \sum_{i=1}^{m} \phi_{\boldsymbol{\alpha}}\left(\sigma_1\left(\boldsymbol{y}_i\right)\right) e^{s_i} \mu_i = \int_{K_S^{d+1}} \phi_{\boldsymbol{\alpha}}\left(\boldsymbol{\lambda}\right) \mathrm{d}\boldsymbol{\lambda}, \quad \text{for } \phi_{\boldsymbol{\alpha}} \in \mathcal{I}.
\end{aligned} \tag{7.4}$$

We expect that the sequence of solutions to these problems with tighter constraints smoothly converges to a feasible solution to the original problem Eq. (7.3).

The complete procedure is described in Algorithm 7.1. Given a symmetric nodal configuration $\boldsymbol{\Xi}_0$, in Line 2 we compute the corresponding degrees of freedom $\boldsymbol{y}^{(0)}$. Next, in Line 3, we set the parameter $\varepsilon$, and, in Line 4, we compute the basis of the ring of symmetric polynomials $\mathcal{I}$. This basis is normalized such that the elements $\phi_{\boldsymbol{\alpha}} \in \mathcal{I}$ have unitary $\mathcal{L}_2$-norm. Then, in Line 5, we compute an initial approximation for the weights. To do so, we compute the positive weight values minimizing the $\mathcal{L}_2$-norm of the integration constraints, namely

$$\begin{aligned}
\min_{\tilde{\boldsymbol{w}}\in\mathbb{R}^m} \quad & \sum_{\phi_{\boldsymbol{\alpha}}\in\mathcal{I}} \left( \sum_{i=1}^{m} \phi_{\boldsymbol{\alpha}}\left(\sigma_1\left(\boldsymbol{y}_i\right)\right) \tilde{w}_i \mu_i - \int_{K_S^{d+1}} \phi_{\boldsymbol{\alpha}}\left(\boldsymbol{\lambda}\right) \mathrm{d}\boldsymbol{\lambda} \right)^2 \\
\text{s.t.} \quad & \tilde{w}_i \geq 0, \quad \text{for } i = 1, \dots, m
\end{aligned} \tag{7.5}$$

Following, we enter the main loop of the algorithm. At an intermediate iteration $k$, we have vectors $\boldsymbol{y}^{(k)}$ and $\boldsymbol{s}^{(k)}$ describing the current position of the nodes and the weights values, respectively. Then, in Line 9, we solve the problem in Eq. (7.4). This function call returns the solution found by the solver, $\boldsymbol{y}^{(k+1)}$ and $\boldsymbol{s}^{(k+1)}$, and a boolean variable with information about the solution found. Specifically, `optSol` is true in

**Algorithm 7.1** Computing interpolation-aware numerical quadratures.

**Input:** Nodal distribution $\boldsymbol{\Xi}_0$, Function to optimize $\beta$, Integration degree $q$
**Output:** Nodal distribution $\boldsymbol{\Xi}$, Integration weights $\boldsymbol{w}$

1: **function** `ComputeIntegrationRule`($\boldsymbol{\Xi}_0$, $\beta$, $q$)
2: $\boldsymbol{y}^{(0)} \leftarrow$ `GetVectorOfDOFs`($\boldsymbol{\Xi}_0$)
3: $\varepsilon = \frac{\pi}{2}\frac{1}{2p}$
4: $\mathcal{I} =$ `ComputeSymmetricPolynomialBasis`($d$, $q$)
5: $\tilde{\boldsymbol{w}}^{(0)} \leftarrow$ `OptimizeWeights`($\boldsymbol{y}^{(0)}$, $\mathcal{I}$)
6: $\boldsymbol{s}^{(0)} \leftarrow \log\left(\tilde{\boldsymbol{w}}^{(0)}\right)$
7: $k \leftarrow 1$
8: **while** not converged and ($k < k_{\max}$) **do**
9: $\boldsymbol{y}^{(k+1)}$, $\boldsymbol{s}^{(k+1)}$, optSol $\leftarrow$ `TrustRegionConstrained`($\boldsymbol{y}^{(k)}$, $\boldsymbol{s}^{(k)}$, $\beta$, $\mathcal{I}$, $\varepsilon$)
10: **if** optSol and $\left(\left\|\boldsymbol{y}^{(k+1)} - \boldsymbol{y}^{(k)}\right\|_\infty < \delta\right)$ and $\left(\left\|\boldsymbol{s}^{(k+1)} - \boldsymbol{s}^{(k)}\right\|_\infty < \delta\right)$ **then**
11: converged $\leftarrow$ True
12: **end if**
13: $k \leftarrow k + 1$
14: **end while**
15: $\tilde{\boldsymbol{w}}^{(k)} \leftarrow e^{\boldsymbol{s}^{(k)}}$
16: $\boldsymbol{\Xi}$, $\boldsymbol{W} \leftarrow$ `Symmetrize`($\boldsymbol{y}^{(k)}$, $\tilde{\boldsymbol{w}}^{(k)}$)
17: **return** $\boldsymbol{\Xi}$, $\boldsymbol{W}$
18: **end function**

two cases. First, `optSol` is true if the solver finds an optimal solution, that is, both the infinity norm of the Lagrangian gradient and the constraint violation are smaller than $\varepsilon_{\text{Lag}} = 10^{-8}$. Second, `optSol` is true if the trust-region radius used by the solver is below $r_{\min} = 10^{-8}$. Next, we check if the sequence of solutions has converged, Line 10. Explicitly, we check if `optSol` is true and if the infinity norm of the difference between iterates is less than a prescribed value $\delta = 10^{-16}$. This process is repeated until the algorithm converges or the maximum number of iterations $k_{\max} = 100$ is reached, Line 8. Finally, in Line 15, we undo the change of variables for the weights and, in Line 16, we recover the full symmetric numerical quadrature from the vectors of degrees of freedom.

We highlight that the number of design variables determines an upper bound for the integration accuracy. In particular, the polynomial degree $q$ is chosen such that the number of elements in $\mathcal{I}$ does not exceed the number of variables $n + m$. However, in some cases, the problem in Eq. (7.3) may not have a feasible solution when $|\mathcal{I}| = n+m$. For instance, the numerical quadrature that corresponds to a point distribution of polynomial degree three has four degrees of freedom, $n = 1, m = 3$,

and the number of elements in $\mathcal{I}$ for $q = 5$ is also four. Therefore, the problem has four variables and four nonlinear equality constraints. Unfortunately, the solution to this system of nonlinear equations is not feasible because one weight is negative. Thus, we can only integrate exactly polynomials up to degree $q = 4$. For the general case, there exist precise relations between the number of degrees of freedom and the number of constraints to ensure exact quadratures up to a certain polynomial degree (Maeztu and Sáinz de La Maza, 1995; Chuluunbaatar et al., 2022). Herein, we first compute the number of variables and choose the highest possible value for $q$ such that $|\mathcal{I}| \leq n + m$. Then, we try to solve the problem in Eq. (7.3) for such $q$. If the solver fails to find a solution before the maximum number of iterations permitted is reached, we understand that there is no feasible solution for such $q$, and try to solve for $q - 1$. This process is repeated until the solver finds a feasible solution.

## 7.4 Results

Next, we report preliminary results for our numerical quadratures computed using the method proposed in Sect. 7.3. Ideally, the initial approximation for the optimization problem should be as close as possible to the optimal feasible solution. Conveniently, the nodal distributions computed in Chapter 5 are local minima of the Chen-Babuska functional. Hence, the initial approximation is given by the point coordinates of the nodal distribution with the lowest Chen-Babuska value. The initial value for the integration weights is obtained by solving a quadratic problem with linear constraints, see Eq. (7.5).

### 7.4.1 Interpolation-aware numerical quadratures in 2D

Following, we study the integration properties of the optimized nodal sets in the triangle. First, we analyze the maximum integration order of the quasi-optimal Lebesgue nodal distributions found in Chapter 6. To compute a set of integration weights given the nodes coordinates, it is standard to compute the $i$-th integration weight $\tilde{w}_i$ as the integral of the Lagrange interpolating polynomial $\phi_i$. However, this approach might yield negative weights. To enforce obtaining positive integration weights, we solve the quadratic problem with linear constraints described in Eq. (7.5). In Table 7.1, in the triangle and for polynomial degree $p$, $p = 3, \ldots, 15$, we report the highest polynomial degree $q$ we are able to integrate exactly using the quasi-optimal Lebesgue nodal dis-

| $p$ | Optimizing weights | | Optimizing nodes and weights | |
|---|---|---|---|---|
| | Positive | Arbitrary signed | Positive | Arbitrary signed |
| 3 | 3 | 3 | 3 | 4 |
| 4 | 3 | 4 | 5 | 5 |
| 5 | 5 | 5 | 6 | 7 |
| 6 | 5 | 6 | 8 | 8 |
| 7 | 7 | 7 | 10 | 10 |
| 8 | 7 | 8 | 11 | 11 |
| 9 | 7 | 9 | 13 | 13 |
| 10 | 7 | 10 | 15 | 15 |
| 11 | 7 | 11 | 15 | 15 |
| 12 | 7 | 12 | 16 | 18 |
| 13 | 7 | 13 | 18 | 18 |
| 14 | 7 | 14 | 18 | 21 |
| 15 | 7 | 15 | 18 | 21 |

Table 7.1: In the triangle and for nodal distributions of polynomial degree $p$, $p = 3, \dots, 15$, the maximum polynomial degree $q$ we are able to exactly integrate using only (positive) weights or the nodes and (positive) weights as design variables.

tribution from Chapter 6, and either the positive integration weights obtained solving the problem in Eq. (7.5) or the arbitrary signed set of integration weights computed by integrating the Lagrange interpolating polynomials. We observe that using positive weights we are able to integrate exactly polynomials of smaller degrees than with arbitrary signed weights. We should expect this behavior because when we enforce positivity of the integration weights we are restricting to the positive region of the search space and, therefore, we are reducing the set of feasible solutions.

To simultaneously account for the interpolation and integration properties of a nodal set, we optimize the nodal coordinates and integration weights by solving the constrained non-linear problem presented in Sect. 7.3. We recall that to enforce positive integration weights, we introduce the change of variables $\tilde{\boldsymbol{w}} = e^{\boldsymbol{s}}$, but it is possible to consider negative integration weights working directly with variables $\tilde{\boldsymbol{w}}$ as design variables. In Table 7.1, we also report the highest polynomial degree $q$ we are able to integrate exactly using these optimized nodes and (positive) weights. As before, we are able to integrate polynomials of higher degrees if we consider arbitrary signed integration weights. Nevertheless, we favor positive weights to ensure we obtain positive values when integrating positive functions. Comparing the maximum

| $p$ | Lebesgue constant $\Lambda$ | |
|---|---|---|
| | Nodal distribution from Chapter 6 | Nodal distribution for interpolation and integration |
| 3 | 2.108 | 2.111 |
| 4 | 2.587 | 2.837 |
| 5 | 3.081 | 3.921 |
| 6 | 3.595 | 4.589 |
| 7 | 4.143 | 5.690 |
| 8 | 4.766 | 6.781 |
| 9 | 5.486 | 8.122 |
| 10 | 5.921 | 38.798 |
| 11 | 6.720 | 10.185 |
| 12 | 7.187 | 10.872 |
| 13 | 7.253 | 9.321 |
| 14 | 7.705 | 9.577 |
| 15 | 8.242 | 10.532 |

Table 7.2: In the triangle and for nodal distributions of polynomial degree $p$, $p = 3, \ldots, 15$, the Lebesgue constant $\Lambda$ of the quasi-optimal Lebesgue nodal distributions from Chapter 6 and the nodal distribution suitable for interpolation and integration.

integration order against those obtained when only the weights are optimized, we observe that considering the nodes position as degrees of freedom presents a clear advantage.

We highlight that the same integration order is achieved for polynomial degrees $p = 13, 14$, and 15. We believe this is due to numerical inaccuracies in the computation of the elements in the ring of symmetric polynomials. We expect that a more numerical stable basis will lead to higher integration orders. Moreover, considering the integration weights as dependent variables of the nodes (Taylor et al., 2007) reduces the number of degrees of freedom and may improve the obtained results. Nonetheless, using a nodal distribution of polynomial degree $p = 10$, we are able to exactly integrate polynomials up to degree $q = 15$.

Next, we analyze the interpolation properties of the optimized nodal sets. In Table 7.2, we compare the Lebesgue constant of the quasi-optimal Lebesgue nodal distributions from Chapter 6 and the optimized nodal set suitable for interpolation and integration using the points and weights as design variables. We observe that the Lebesgue constant of the nodal distributions accounting for interpolation and integration is larger than for the quasi-optimal Lebesgue nodal sets. Nevertheless,

| | Method from Sect. 7.3 | | Method from Chapter 6 |
|---|---|---|---|
| $p$ | $\max q$ | $\Lambda$ | $\Lambda$ |
| 3 | 3 | 2.936 | 2.930 |
| 4 | 4 | 4.576 | 4.005 |
| 5 | 5 | 6.355 | 5.258 |
| 6 | 7 | 11.215 | 6.759 |
| 7 | 8 | 14.143 | 8.252 |
| 8 | 10 | 24.687 | 10.594 |
| 9 | 10 | 20.670 | 13.693 |
| 10 | 10 | 18.736 | 14.273 |

Table 7.3: In the tetrahedron and for nodal distributions of polynomial degree $p$, $p = 3, \dots, 10$, the maximum polynomial degree $q$ we are capable of integrating exactly, and the Lebesgue constant $\Lambda$ of the nodal distribution suitable for interpolation and integration and of the quasi-optimal Lebesgue nodal distributions from Chapter 6.

because the Lebesgue constants are still reasonable, we conclude that our nodal distributions possess good interpolation properties and are capable of exactly integrating polynomials up to moderate degree.

### 7.4.2 Interpolation-aware numerical quadratures in 3D

Analogously, we study the integration and interpolation properties of the nodal distributions in the tetrahedron obtained using the method in Sect. 7.3. In Table 7.3, we report the maximum polynomial degree $q$ we are able to integrate exactly and the Lebesgue constant. We also show the Lebesgue constant of the quasi-optimal Lebesgue nodal distributions from Chapter 6. We observe that for low polynomial degrees, the maximum integration order $q$ coincides with $p$. For low polynomial degrees, almost all the nodes in the tetrahedron are on the element boundary and, thus, there are not enough degrees of freedom to achieve exact high-order integration. In contrast, as we increase the polynomial degree, interior nodes appear and these degrees of freedom allow achieving higher integration order. For polynomial degree $p = 9$ and 10, there is no improvement in the value of $q$. Maybe, a more numerically stable polynomial basis will help to attain higher-order quadratures. Regarding the interpolation properties, we observe that the nodal distributions accounting for exact integration of polynomials present larger Lebesgue constant, yet they still have good interpolation properties.

| $p$ | Method from Sect. 7.3 $\max q$ | $\Lambda$ | Method from Chapter 6 $\Lambda$ |
|---|---|---|---|
| 3 | 2 | 4.169 | 3.760 |
| 4 | 3 | 6.130 | 5.964 |
| 5 | 4 | 9.322 | 8.488 |
| 6 | 5 | 12.867 | 11.683 |
| 7 | 7 | 27.567 | 15.629 |
| 8 | 7 | 30.062 | 21.053 |
| 9 | 9 | 66.209 | 27.500 |
| 10 | 11 | 101.511 | 35.668 |

Table 7.4: In the pentatope and for nodal distributions of polynomial degree $p$, $p = 3, \dots, 10$, the maximum polynomial degree $q$ we are capable of integrating exactly, and the Lebesgue constant $\Lambda$ of the nodal distribution suitable for interpolation and integration and of the quasi-optimal Lebesgue nodal distributions from Chapter 6.

### 7.4.3 Interpolation-aware numerical quadratures in 4D

We repeat a similar study for the nodal distributions in the pentatope, see Table 7.4. Unfortunately, we do not attain an integration degree that is larger than the interpolation degree. We may achieve higher integration orders expressing the weights in terms of the nodal coordinates, using a more numerical stable polynomial basis, or considering a different initial approximation. Even exploration approaches similar to the ones proposed in Chapter 5 might be useful to enforce a tunnel effect through the problem constraints.

## 7.5 Concluding remarks

We have proposed a nonlinear constrained optimization problem to minimize the interpolation error of a symmetric and closed nodal distribution with a simplicial number of points subject to integrating exactly high-order polynomials with positive integration weights. The design variables are the weights and coordinates of the nodal representatives. The interpolation error is approximated with the twice-differentiable vectorial $\mathcal{L}_2$-norm of the Lagrange interpolating polynomials. Because we consider symmetric distributions, the constraints only impose exact integration of high-order symmetry-invariant polynomials.

The preliminary results in the two- and three-dimensional simplex show that the

obtained nodal distributions feature low Lebesgue constant and are able to integrate exactly polynomials of higher degree than the nodal degree. In the 4D case, integration degrees higher than the nodal degree have not been attained. In the near future, we will improve the current numerical solver to either understand this issue or obtain higher integration order.

In conclusion, to obtain diagonal mass matrices, we have proposed a method to compute an interpolation-aware numerical quadrature featuring a closed and symmetric point set with a simplicial number of points, and positive weights. In 2D and 3D, the preliminary results show that high-order integration can be attained with low effect on the interpolation error. In perspective, for complex geometry, we expect that our collocation of the interpolation points with the integration points will be used to improve the efficiency and stability of the spectral element method and other unstructured high-order methods.

# Chapter 8

# Conclusions and future work

In this thesis, we have enabled high-order numerical interpolation and integration on complex geometry suitable for simulation up to four dimensions. To this end, we have fulfilled the following objectives: to nodally represent boundary geometry for flow simulation (Jiménez-Ramos et al., 2020); to model flow simulation intent with a nodal boundary representation, Chapter 3 (Jiménez-Ramos et al., 2022); to estimate the interpolation error of a given point distribution, Chapter 4 Jiménez-Ramos et al. (2023d); to obtain candidate point distributions suitable for interpolation, Chapter 5; to obtain point distributions with quasi-optimal interpolation error, Chapter 6; and to obtain integration points also suitable for interpolation, Chapter 7. To enable high-order numerical interpolation and integration on complex geometries, we have used mathematical formulations and derivations, optimization methods, heuristics, computer implementations, runtime checks, and verification approaches.

There are two central findings to obtain the interpolation and integration results on the $d$-dimensional simplex. First, to propose the heuristics and methods to obtain the results, we have relied on the claim that nodal distributions that feature optimal interpolation accuracy are equidistributed in the $(d+1)$-sphere orthant. Since we have exploited this claim in several of the proposed heuristics, it is crucial to obtain our quasi-optimal nodal distribution results for numerical interpolation and integration. Second, to efficiently explore nodal distributions that are local optima and do not feature a lattice-like structure, we have emulated a tunnel effect in the energy landscape of a free node on the $(d+1)$-sphere orthant. We have proposed to proxy the uphills of the energy landscape of the interpolation error for a free node with

the faces of a Delaunay tessellation of the nodal distribution on the $(d+1)$-sphere orthant. Using this proxy, we have been able to emulate a tunnel effect that heuristically enforces a node to go to the other side of an uphill energy landscape by simply relocating the node on the centroid of the element on the other side of the face. Using deterministic continuous optimization without this tunnel effect we could not recover the non lattice-like structures in 2D of the best Lebesgue nodal distributions reported in the literature. Hence, the tunnel effect emulation is crucial to verify our 2D results and obtain our 3D and 4D results for numerical interpolation and integration.

The work carried out in this thesis leaves open some research activities that should be performed in the near future. First, it would be interesting to explore an extension to 4D of the proposed curved geometry modeling. Since the boundary of a 4D simplicial mesh is composed of tetrahedra, we should devise a tetrahedral subdivision method featuring $\mathcal{C}^1$-continuity. Second, regarding the combinatorics for the number of points in each sub-simplex of the symmetry simplex, we have considered only symmetric distributions that have the same combinatorics as the equidistributed nodal distribution. Even though these seem to be the only combinations giving rise to unisolvent nodal distributions in 2D for low polynomial degrees, it might not be the case for higher degrees or higher-dimensional simplices. Accordingly, we would like to study other symmetry combinatorics for the nodal distributions and compare their interpolation and integration performance with the nodal sets found in this thesis. Third, we have obtained preliminary results regarding nodal sets suitable for interpolation and integration, but the implementation could be improved if we considered a more numerically stable polynomial basis to enforce exact integration. Moreover, it is possible to consider the integration weights as dependent variables. Consequently, the number of design variables would be reduced and higher integration order may be achieved.

Nevertheless, we have contributed to enabling 4D high-order space-time discretizations for 3D unsteady high-fidelity simulation. Specifically, we have proposed methods to model and represent the simulation intent, to obtain nodal distributions with quasi-optimal interpolation, and to determine weights and integration points that are also suitable for interpolation.

In perspective, our high-order geometry modeling and representation methods, numerical interpolation, and integration on complex geometry will be key ingredients in 4D high-order space-time discretizations. In these discretizations, the boundary

representation might be represented as a piecewise polynomial boundary mesh preserving the simulation intent. Meanwhile, the computation nodes might be determined by our quasi-optimal interpolation distributions. Finally, in applications such as the spectral element method, our closed numerical quadratures might provide the node location.

# Appendix A

# Non-interpolative to interpolative

Given a control mesh, the schemes in Sect. 3.4.1 and Sect. 3.4.2 for mesh subdivision generate a hierarchy of subdivided meshes all of them tending to the same limit model (curve or surface). These schemes do not preserve the position of the initial vertices of the mesh. However, a new control mesh can be computed so that the limit model contains the nodes of the initial mesh (Persson et al., 2006; Jiménez-Ramos et al., 2020). That is, new points can be found such that the limit curves and surfaces interpolate the given data points.

Given a polygon, the curve subdivision scheme presented in Lane and Riesenfeld (1980) generates a sequence of refined polygons, all of them converging to the same cubic $\mathcal{C}^2$-continuous curve. We remark that the initial control mesh determines the limiting curve, so all the refined polygons converge to the same curve. The expression to map a node of the polygon onto the limit curve becomes linear when applied to the nodes of the polygon. Specifically, denote by $\boldsymbol{x}_i^l$ the position of node $i$ at refinement level $l$, and by $\boldsymbol{x}_{i-1}^l$ and $\boldsymbol{x}_{i+1}^l$ its neighbor nodes. Then, the position of node $\boldsymbol{x}_i^l$ on the limit curve, $\boldsymbol{x}_i^{l,\infty}$, is determined by the following linear expression

$$\boldsymbol{x}_i^{l,\infty} = \frac{1}{6}\left(\boldsymbol{x}_{i-1}^l + 4\boldsymbol{x}_i^l + \boldsymbol{x}_{i+1}^l\right). \tag{A.1}$$

In the case of surfaces, Loop's subdivision scheme (Loop, 1987) defines a hierarchy of control meshes, all of them converging to the same limit surface. Analogously to the curve case, we can compute the limiting location for the nodes at any refinement level. In particular, given a linear mesh, the limit position for the node $v$ at any

refinement level $l$, denoted by $\boldsymbol{x}_v^{l,\infty}$, is given by the following linear expression

$$\boldsymbol{x}_v^{l,\infty} = (1 - k\chi_k)\, \boldsymbol{x}_v^l + \chi_k \sum_{i=1}^{k} \boldsymbol{x}_i^l, \tag{A.2}$$

where $\{\boldsymbol{x}_i^l\}_{i=1,\dots,k}$ are the positions of the $k$ neighbor nodes of $v$ at level $l$, and where the weights are computed as

$$\chi_k = \frac{1}{k + \frac{3}{8\omega_k}},$$

with $\omega_k = \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{k}\right)\right)^2\right)$.

To ensure that the limit model contains the nodes of the initial mesh, we compute a new control mesh. Specifically, let $\boldsymbol{L}$ be the operator that maps the nodes of the initial mesh, $\boldsymbol{X}^0$, onto their limit position. Row $i$ of matrix $\boldsymbol{X}^0$ corresponds to the position of node $i$ of the boundary mesh $\boldsymbol{x}_i^0$. Then, we compute a new control mesh, with nodes position denoted by $\boldsymbol{X}^C$, such that

$$\boldsymbol{L}\boldsymbol{X}^C = \boldsymbol{X}^0,$$

where the $i$-th row of matrix $\boldsymbol{X}^C$ corresponds to the unknown position of node $i$ of the control mesh, $\boldsymbol{x}_i^C$. In the case of the subdivision schemes considered in this thesis, $\boldsymbol{L}$ is a linear application with rows given by the coefficients in Eq. (A.1) or Eq. (A.2), for curve and surface points, respectively. This matrix is sparse and the solution of the linear system can be computed using a sparse direct solver. In our Python implementation, we call the sparse direct solver of the SuperLU library (Demmel et al., 1999; Li et al., 1999) through the Python SciPy package (Jones et al., 2001). Recall that this operation is performed on the boundary of a tetrahedral mesh. Therefore, the dimension of the linear system is of the order of the number of boundary nodes and not of the order of nodes of the volume mesh.

# Appendix B

# Accommodate the curvature of the boundary

In this appendix, we detail the computation TFI-based approach proposed in Chapter 3 to accommodate the curving of the boundary surface mesh into the interior of the volume mesh in the mesh approximation process of the limit model, see Sect. 3.7.

First, consider an edge of a high-order boundary element with one of its endpoints on the curved surface. The new location of the edge nodes is given by the linear isoparametric mapping between the one-dimensional reference domain and the physical edge, denoted as $\phi^1$. Specifically, the new position of the $k$-th node of the edge, $\boldsymbol{x}_k$, for $k = 0, \ldots, q$, is determined as

$$\boldsymbol{x}_k = \phi^1(\boldsymbol{\xi}_k) = \sum_{l=0}^{1} \boldsymbol{x}_{v_l} N_{v_l}(\boldsymbol{\xi}_k)$$

where $\boldsymbol{\xi}_k$ is the position of the $k$-th interpolation node of the reference domain, $\boldsymbol{x}_{v_l}$ is the position of the $l$-th endpoint of the edge, and $N_{v_l}$ is the linear nodal shape function of the interval associated with the $l$-th endpoint. In Fig. 3.7(c), we illustrate the relocation of the nodes of the edges in a triangle when one of its edges (in bold) is on the boundary.

Now, we are interested in relocating the nodes on the interior of a face of a boundary tetrahedron. On the one hand, if such face belongs to the boundary, its nodes have been already relocated onto the surrogate geometry, see bold edge in Fig. 3.7(c). On the other hand, if such face does not belong to the boundary, its

edge nodes have been modified using the transfinite interpolation for edges explained above, see non-bold edges in Fig. 3.7(c). For the latter, we apply the transfinite interpolation for faces, that is, we accommodate the deformation of the curving of the edges to the interior of the triangular faces.

We denote by $f_i$ the edge of the triangle opposite to vertex $i$, $i = 1, 2, 3$. Let us denote as $\boldsymbol{x}_{f_i,k}$ the coordinates of the $k$-th node of the edge $f_i$ in the physical element, $k = 0, \ldots, q$. These nodes are fixed now, since as previously detailed their location has already been computed. Therefore, each curved or relocated edge can be parametrized using the restriction to such edge of the two-dimensional isoparametric mapping of degree $q$, $\phi^q$, as:

$$\phi_{f_i}(\boldsymbol{\xi}) := \phi^q|_{f_i}(\boldsymbol{\xi}) = \sum_{k=0}^{q} \boldsymbol{x}_{f_i,k} N_{f_i,k}(\boldsymbol{\xi}) \tag{B.1}$$

where $N_{f_i,k}(\xi)$ is the high-order nodal shape function of the triangle associated to the $k$-th node of the edge $f_i$. In particular, the boundary of the triangle is fixed and parameterized by the three mappings $\{\phi_{f_i}\}_{i=1,2,3}$.

Consider a point $\boldsymbol{x}$ in the physical triangle, to which we want to compute its displaced position in terms of the location of the boundary edge nodes. Denote by $\boldsymbol{\xi}$ the position of the point in the reference triangle expressed in cartesian coordinates such that $\phi^q(\boldsymbol{\xi}) = \boldsymbol{x}$. Now, denote by $\boldsymbol{\lambda}$ the same point in the reference domain expressed in barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$, $\sum_{i=1}^{3} \lambda_i = 1$. Following Perronnet (1998), we compute the projection of the point to the edges. A point on an edge can be parametrized as a function of the barycentric coordinates of the two vertices of the triangle defining the edge. Therefore, two different projections ($\boldsymbol{\lambda}^j_{f_*}$ for $j \in f_*$) are computed for each one of the three edges of the triangle (rows, $\boldsymbol{\lambda}^*_{f_i}$ for $i = 1, 2, 3$):

$$f_1 = (2,3): \begin{cases} \boldsymbol{\lambda}^2_{f_1} = (0, 1-\lambda_3, \lambda_3), \\ \boldsymbol{\lambda}^3_{f_1} = (0, \lambda_2, 1-\lambda_2), \end{cases}$$

$$f_2 = (1,3): \begin{cases} \boldsymbol{\lambda}^1_{f_2} = (1-\lambda_3, 0, \lambda_3), \\ \boldsymbol{\lambda}^3_{f_2} = (\lambda_1, 0, 1-\lambda_1), \end{cases}$$

$$f_3 = (1,2): \begin{cases} \boldsymbol{\lambda}^1_{f_3} = (1-\lambda_2, \lambda_2, 0), \\ \boldsymbol{\lambda}^2_{f_3} = (\lambda_1, 1-\lambda_1, 0). \end{cases}$$

Note that $\boldsymbol{\lambda}^j_{f_i}$ belongs to edge $f_i$ and has the $j$-th component expressed in terms of the other components. Then, we express these six projections of the point at the

edges, computed in barycentric coordinates, back in the reference coordinates $\boldsymbol{\xi}$. As previously remarked, we denote the change from barycentric coordinates of a point $\boldsymbol{\lambda}_{f_i}^j$ to reference coordinates as $\boldsymbol{\xi}_{f_i}^j$, for $i = 1, 2, 3$. Since these points are on the edges of the triangle, they can be mapped onto the physical triangle through the mappings $\phi_{f_i}$ of the edges, $i = 1, 2, 3$, presented in Eq. (B.1) as:

$$\boldsymbol{x}_{f_i}^j := \phi_{f_i}\left(\boldsymbol{\xi}_{f_i}^j\right)$$

We highlight that given a point $\boldsymbol{x}$, $\boldsymbol{x}_{f_i}^j$ corresponds to the coordinates on the physical element of the projection $j$ of the point to the edge $f_i$, $i = 1, 2, 3$.

Finally, the new position of point $\boldsymbol{x}$ in the physical triangle, denoted as $\hat{\boldsymbol{x}}$, is given in Perronnet (1998) as:

$$\begin{aligned}\hat{\boldsymbol{x}} = {} & \lambda_1\left(\boldsymbol{x}_{f_2}^1 + \boldsymbol{x}_{f_3}^1 - \boldsymbol{x}_{v_1}\right) + \lambda_2\left(\boldsymbol{x}_{f_1}^2 + \boldsymbol{x}_{f_3}^2 - \boldsymbol{x}_{v_2}\right) \\ & + \lambda_3\left(\boldsymbol{x}_{f_1}^3 + \boldsymbol{x}_{f_2}^3 - \boldsymbol{x}_{v_3}\right)\end{aligned}$$

where $\boldsymbol{x}_{v_j}$ is the position of the $j$-th vertex of the physical triangle. We highlight that the transfinite interpolation for triangles can be expressed as a function of the isoparametric mapping of the edges and the location of the vertices of the triangle.

In order to relocate the nodes in the interior of the high-order physical faces, the steps detailed above are applied to the interpolation nodes in the interior of the high-order reference triangle, see Fig. 3.7(d). This procedure is repeated for all the faces with a boundary node or edge.

Lastly, we follow an analogous approach to modify the position of the nodes in the interior of the boundary tetrahedra. The boundary of a tetrahedron is composed of four faces and six edges. These faces and edges have already been curved with the procedures detailed above. Therefore, we relocate the interior nodes according to the curved boundary already accommodated to the edges and faces. Similarly to the triangle case, the transfinite interpolation for tetrahedra can be expressed as a function of the isoparametric mapping of the edges, the isoparametric mapping of the faces, and the location of the vertices of the tetrahedron.

Denote by $T$ the set of vertices of a tetrahedron. We define the entity $f_{i_1,\ldots,i_k}$ as the entity of dimension $d-k$, $d = 3$, with vertices given by the nodes $T \setminus \{i_1, \ldots, i_k\}$. Note that the face opposite to node $i$ is denoted by $f_i$, and the edge shared by the faces $f_i$ and $f_k$ is $f_{ik}$. Given the three-dimensional isoparametric mapping of degree $q$, $\phi^q$, analogously to Eq. (B.1), we denote the restriction to the face $f_i$ as $\phi_{f_i}$, and the restriction to the edge $f_{ik}$ as $\phi_{f_{ik}}$.

Similarly to the two-dimensional case, consider a point $\boldsymbol{x}$ in the physical tetrahedron, and denote by $\boldsymbol{\xi}$ and $\boldsymbol{\lambda}$ its preimage in the reference tetrahedron expressed in cartesian and barycentric coordinates, respectively. Now, denote by $\boldsymbol{\lambda}^j_{f_i}$ the projection of the point to the face $f_i$ that has the $j$-th component expressed in terms of the other components. $\boldsymbol{\lambda}^j_{f_{ik}}$ denotes the projection of the point to the edge $f_{ik}$ that has the $j$-th component expressed in terms of the others. These projections in the reference domain expressed in cartesian coordinates are denoted by $\boldsymbol{\xi}^j_{f_i}$ and $\boldsymbol{\xi}^j_{f_{ik}}$, respectively.

Since these points are on the faces and edges of the tetrahedron, they can be mapped onto the physical element through the mappings $\phi_{f_i}$ on the faces and $\phi_{f_{ik}}$ on the edges as:

$$\boldsymbol{x}^j_{f_i} := \phi_{f_i}\left(\boldsymbol{\xi}^j_{f_i}\right), \quad \boldsymbol{x}^j_{f_{ik}} := \phi_{f_{ik}}\left(\boldsymbol{\xi}^j_{f_{ik}}\right)$$

Finally, the new position of point $\boldsymbol{x}$ in the physical triangle, denoted as $\hat{\boldsymbol{x}}$, is given in Perronnet (1998) as:

$$\begin{aligned}
\hat{\boldsymbol{x}} = {} & \lambda_1\left(\boldsymbol{x}^1_{f_2} + \boldsymbol{x}^1_{f_3} + \boldsymbol{x}^1_{f_4} - \boldsymbol{x}^1_{f_{23}} - \boldsymbol{x}^1_{f_{24}} - \boldsymbol{x}^1_{f_{34}} + \boldsymbol{x}_{v_1}\right) \\
& + \lambda_2\left(\boldsymbol{x}^2_{f_1} + \boldsymbol{x}^2_{f_3} + \boldsymbol{x}^2_{f_4} - \boldsymbol{x}^2_{f_{13}} - \boldsymbol{x}^2_{f_{14}} - \boldsymbol{x}^2_{f_{34}} + \boldsymbol{x}_{v_2}\right) \\
& + \lambda_3\left(\boldsymbol{x}^3_{f_1} + \boldsymbol{x}^3_{f_2} + \boldsymbol{x}^3_{f_4} - \boldsymbol{x}^3_{f_{12}} - \boldsymbol{x}^3_{f_{14}} - \boldsymbol{x}^3_{f_{24}} + \boldsymbol{x}_{v_3}\right) \\
& + \lambda_4\left(\boldsymbol{x}^4_{f_1} + \boldsymbol{x}^4_{f_2} + \boldsymbol{x}^4_{f_3} - \boldsymbol{x}^4_{f_{12}} - \boldsymbol{x}^4_{f_{13}} - \boldsymbol{x}^4_{f_{23}} + \boldsymbol{x}_{v_4}\right)
\end{aligned}$$

# Appendix C

# Parameterization of a symmetric high-order nodal distribution in the simplex

---

A $d$-simplex $K^d \subset \mathbb{R}^d$ of polynomial degree $p$ is composed of $N_{p,d} := \binom{d+p}{d}$ nodes defining an interpolative nodal distribution $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_{p,d}}\}$. In this work, we consider symmetric nodal distributions on the simplex. We favor symmetric over non-symmetric nodal sets because we want elements that are label-invariant and orientation free. As a consequence, since two adjacent simplices in a mesh intersect along an entity of dimension $k$, for $k = 1, \ldots, d-1$, the nodal distribution of both elements coincides along their intersection.

The choice of the symmetry pattern used herein is determined by two aspects. First, on the entities of dimension $k$, we impose that the number of nodes on the entity matches the number of nodes of the $k$-dimensional simplex, for $k = 1, \ldots, d-1$. Therefore, the restriction of the nodal basis on an entity coincides with the nodal high-order basis of the lower-dimensional simplex. Second, the nodal set should be unisolvent, that is, there should exist a unique polynomial of degree $p$ defined by the function value at the nodes. Remarkably, a nodal set is unisolvent if, and only if, its Vandermonde matrix is non-singular. The symmetry pattern that coincides with the equispaced distribution seems to be the only one that satisfies these two conditions (Marchildon and Zingg, 2022). Furthermore, this symmetry pattern allows

us to determine in a systematic manner the subset of the nodes describing the whole symmetric nodal distribution.

## C.1 Representatives of a symmetric nodal distribution

Next, we detail our choice of the nodal representatives describing a symmetric high-order nodal distribution. Given a symmetric nodal distribution $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1,\dots,N_{p,d}}$, we denote by $\boldsymbol{\lambda}_i = \left(\lambda_i^1, \dots, \lambda_i^{d+1}\right)$ the barycentric coordinates of node $i$. Then, given a permutation $\rho$ in the set of permutations $\mathcal{S}_{d+1}$, we define as

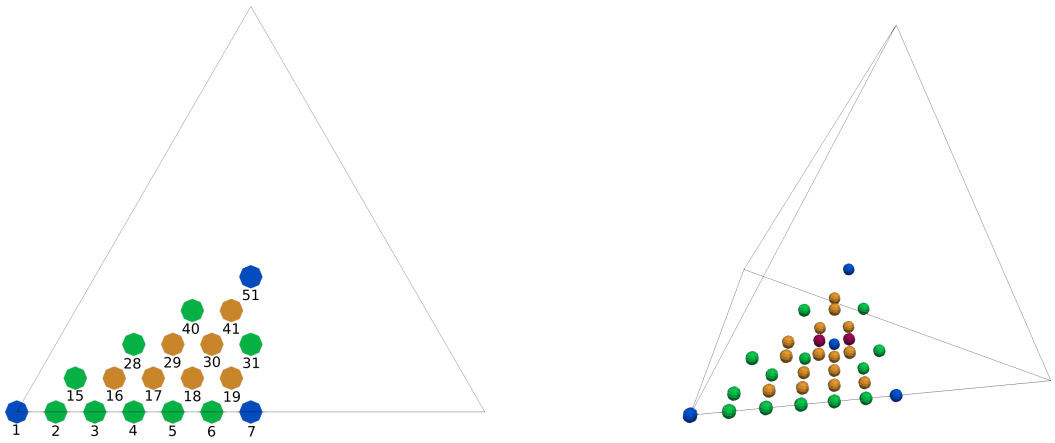
$$\begin{array}{rccc} P[\rho]: & \mathbb{R}^{d+1} & \longrightarrow & \mathbb{R}^{d+1} \\ & \boldsymbol{\lambda} = \left(\lambda^1, \dots, \lambda^{d+1}\right) & \longmapsto & P[\rho](\boldsymbol{\lambda}) = \left(\lambda^{\rho_1}, \dots, \lambda^{\rho_{d+1}}\right), \end{array}$$

the function that permutes the components of a vector according to permutation $\rho$. Now, in order to determine which nodes define the symmetric nodal distribution, we define the equivalence class

$$[\boldsymbol{\lambda}_i] = \{P[\rho](\boldsymbol{\lambda}_i) \mid \rho \in \mathcal{S}_{d+1}\}.$$

Equivalently, nodes with the same barycentric coordinates up to permutation belong to the same equivalence class. We choose the greatest node under the lexicographical order $\succeq$ to be the *representative* of a class of related nodes. More precisely, assume node $i$ is related to $k$ nodes, $[\boldsymbol{\lambda}_i] = \{\boldsymbol{\lambda}_{i_1}, \dots, \boldsymbol{\lambda}_{i_k}\}$. Then, the representative is the node $i_j$ such that $\boldsymbol{\lambda}_{i_j} \succeq \boldsymbol{\lambda}_{i_s}$, $\forall s = 1, \dots, k$. The number of elements in $[\boldsymbol{\lambda}_i]$ is called the multiplicity of the node and it is denoted as $\mu_i$.

To illustrate these definitions, in Fig. C.1, we show the representatives of the equispaced nodal distribution of degree $p = 12$ for the triangle and for the tetrahedron. In the two-dimensional simplex, Fig. C.1(a), the representatives are a vertex, six edge nodes, three nodes on the symmetry hyper-plane $\{\lambda^2 = \lambda^3\}$, one node on the symmetry hyper-plane $\{\lambda^1 = \lambda^2\}$, one node at the intersection of these two symmetry hyper-planes, and seven free interior nodes. For the tetrahedron, Fig. C.1(b), note that the representatives on the face $\{\lambda^4 = 0\}$ are the representative of the triangle described before. Furthermore, there are thirteen interior nodes that belong to at least one of the symmetry hyper-planes $\{\lambda^1 = \lambda^2\}$, $\{\lambda^2 = \lambda^3\}$ or $\{\lambda^3 = \lambda^4\}$, and two free interior nodes.




(a) (b)

Figure C.1: Representatives of the (a) triangle, and the (b) tetrahedron. Colors indicate the number of degrees of freedom. Blue nodes have zero degrees of freedom, cyan nodes identify nodes with one degree of freedom, yellow nodes have two degrees of freedom, and red nodes describe nodes with three degrees of freedom.

## C.2 Parameterization of a nodal representative

A nodal representative is constrained to move along the intersection of the symmetry hyper-planes it belongs to. Consider first the representative of an interior node with barycentric coordinates $\boldsymbol{\lambda} = \left(\lambda^1, \ldots, \lambda^{d+1}\right)$, $\sum_{i=1}^{d+1} \lambda^i = 1$. Let us assume that the first $k$ components are equal and the rest $d+1-k$ are pairwise different, that is,

$$\boldsymbol{\lambda} = (\underbrace{\lambda^1, \ldots, \lambda^1}_{k}, \lambda^{k+1}, \ldots, \lambda^{d+1}),$$

with $\lambda^i \neq \lambda^1$ for $i = k+1, \ldots, d+1$, and $\lambda^i \neq \lambda^j$ for $i, j = k+1, \ldots, d+1$, $i \neq j$. Since the sum of the components equals one, this node has $d-(k-1)$ degrees of freedom. To determine the domain of a node, we compute the intersection of the simplex with the symmetry hyper-planes the node belongs to. In particular, a node with $k$ equal components belongs to $\binom{k}{2}$ symmetry hyper-planes, and the intersection of these hyper-planes with the whole simplex $K^d$ determines a simplex of dimension $d-(k-1)$. In general, a node with $s$ subsets of equal components of lengths $k_1, \ldots, k_s$ has $d-(k_1-1)-\cdots-(k_s-1)$ degrees of freedom, and the intersection of the symmetry hyper-planes with the simplex $K^d$ determines a simplex of dimension $d-(k_1-1)-\cdots-(k_s-1)$.

For the representative of a boundary node, we determine its domain by studying

the face $\{\lambda^{d+1} = 0\}$ as a $(d-1)$-dimensional simplex. On the one hand, the nodes in the interior of the boundary face are interior nodes of the $(d-1)$-dimensional simplex and the reasoning above applies. On the other hand, the nodes on the boundary of the boundary face belong to a $(d-2)$-dimensional simplex. Therefore, we recursively compute the domain of any nodal representative.

Finally, the position of the representative is a linear combination of the position of the vertices of its sub-simplicial domain. More precisely, consider the representative node $i$ and suppose that it has $n_i$ degrees of freedom. Then, it is constrained to move on its associated $n_i$-dimensional simplicial domain. Expressing the node in barycentric coordinates with respect to the vertices of its sub-simplicial domain leads to a point in $\mathbb{R}^{n_i+1}$ denoted as $\boldsymbol{\mu}_i = \left(\mu_i^1, \ldots, \mu_i^{n_i+1}\right)$, with $\sum_{j=1}^{n_i+1} \mu_i^j = 1$. The first $n_i$ components are enough to determine the position of the representative and serve as the degrees of freedom describing such node, $\boldsymbol{y}_i = (y_i^1, \ldots, y_i^{n_i})$, with $y_i^j = \mu_i^j$.

We remark that the degrees of freedom should satisfy

$$\begin{aligned} &0 < y_i^j < 1, \text{ for } j = 1, \ldots, n_i, \\ &\sum_{j=1}^{n_i} y_i^j < 1, \end{aligned} \tag{C.1}$$

to describe a point inside its associated $n_i$-simplex. Moreover, even though a node can freely move therein, there are some regions where it would not define a unisolvent nodal distribution. For instance, in the two-dimensional case, although a free interior node can move throughout all the triangle, this node cannot be in the median. If it was on the median, there would be one of its related nodes on the same position. Therefore, these two nodes would overlap, and they would not determine a unisolvent nodal distribution.

In Fig. C.1(a), the representatives of the equispaced nodal distribution of degree $p = 12$ for the triangle are colored accordingly to their number of degrees of freedom. Nodes colored in blue have zero degrees of freedom, color cyan represents nodes with one degree of freedom, while yellow nodes have two degrees of freedom. We remark that the barycenter belongs to the three symmetry hyper-planes and, therefore, it is fixed. The node at the midpoint of the edge is a boundary node that belongs to a symmetry hyper-plane of the one-dimensional simplex, i.e. the edge, hence it has zero degrees of freedom. Analogously, in Fig. C.1(b), we color the representatives of the equispaced nodal distribution of degree $p = 12$ for the tetrahedron. Besides the colors already described, red nodes are nodes with three degrees of freedom. Note

that the representatives on the boundary are the same as for the triangle case and with the same degrees of freedom.

## C.3 Parameterization of a symmetric nodal distribution

We encode the degrees of freedom of all the representatives in a single vector $\boldsymbol{y} \in \mathbb{R}^n$, with $n$ being the sum of degrees of freedom of all the representatives. Vector $\boldsymbol{y}$ is simply a concatenation of the parameters describing the representatives. In the first positions, we find the degrees of freedom (if any) of the first representative. Following, we encounter the degrees of freedom (if any) describing the second representative. This process is repeated for all the representatives, and the result is a vector of dimension $n$ with all the degrees of freedom that define the nodal distribution. We note that the domain for $\boldsymbol{y}$ is not the whole space $\mathbb{R}^n$, but the subspace defined by the product of each of the simplicial domain in Eq. (C.1).

In the example of Fig. C.1(a), the vector encoding the degrees of freedom of the nodal distribution has length $3 \cdot 0 + 9 \cdot 1 + 7 \cdot 2 = 23$. The first component corresponds to the unique degree of freedom describing the position of the second edge node, $y_2^1$. Similarly, the degrees of freedom describing the position of the edge nodes $3, \ldots, 6$ follow. Next, we find the degrees of freedom describing node 15. Since it belongs to the symmetry line, it has one degree of freedom, $y_{15}^1$. Two degrees of freedom describing the position node 16, $(y_{16}^1, y_{16}^2)$, appear immediately after in the vector $\boldsymbol{y}$. Proceeding similarly for the rest of the representatives, the vector describing the degrees of freedom is

$$\begin{aligned}\boldsymbol{y} = \big(&y_2^1, y_3^1, y_4^1, y_5^1, y_6^1, y_{15}^1, y_{16}^1, y_{16}^2, y_{17}^1, y_{17}^2, y_{18}^1, y_{18}^2,\\ &y_{19}^1, y_{19}^2, y_{28}^1, y_{29}^1, y_{29}^2, y_{30}^1, y_{30}^2, y_{31}^1, y_{40}^1, y_{41}^1, y_{41}^2\big)\,.\end{aligned}$$

For optimization purposes, it is convenient to consider the inverse mapping of the procedure described above, that is, the mapping from the degrees of freedom of each representative, $\boldsymbol{y}$, to the cartesian coordinates of all the nodes, $\boldsymbol{Z}$. This mapping consists of three steps. First, we recover the barycentric coordinates of the representative from the degrees of freedom. Then, we permute these coordinates to retrieve all the symmetric nodes. Finally, all the nodes expressed in barycentric coordinates are expressed in cartesian coordinates.

More precisely, consider the components of the vector $\boldsymbol{y}$ that correspond to the $n_i$ degrees of freedom describing the representative node $i$, $\boldsymbol{y}_i = (y_i^1, \dots, y_i^{n_i})$. First, we recover the barycentric coordinates, $\boldsymbol{\lambda}_i$, from its degrees of freedom via a mapping $\sigma_1$,

$$\begin{array}{rccc} \sigma_1: & \mathbb{R}^{n_i} & \longrightarrow & \mathbb{R}^{d+1} \\ & \boldsymbol{y}_i = (y_i^1, \dots, y_i^{n_i}) & \longmapsto & \sigma_1(\boldsymbol{y}_i) = \boldsymbol{\lambda}_i = \left(\lambda_i^1, \dots, \lambda_i^{d+1}\right). \end{array}$$

Next, by means of a mapping $\sigma_2$ the barycentric coordinates of all the elements in the equivalence class $[\boldsymbol{\lambda}_i]$ are obtained by permuting the components of $\boldsymbol{\lambda}_i$,

$$\begin{array}{rccc} \sigma_2: & \mathbb{R}^{d+1} & \longrightarrow & \mathbb{R}^{d+1} \times \overset{\mu_i}{\cdots} \times \mathbb{R}^{d+1} \\ & \boldsymbol{\lambda}_i & \longmapsto & \sigma_2(\boldsymbol{\lambda}_i) = \{\boldsymbol{\lambda}_{i_1}, \dots, \boldsymbol{\lambda}_{i_{\mu_i}}\} = \{P[\rho](\boldsymbol{\lambda}_i) \,|\forall \rho \in \mathcal{S}_{d+1}\}. \end{array}$$

Finally, $\sigma_3$ maps points in barycentric coordinates onto points in cartesian coordinates,

$$\begin{array}{rccc} \sigma_3: & \mathbb{R}^{d+1} \times \overset{\mu_i}{\cdots} \times \mathbb{R}^{d+1} & \longrightarrow & \mathbb{R}^{d} \times \overset{\mu_i}{\cdots} \times \mathbb{R}^{d} \\ & \{\boldsymbol{\lambda}_{i_1}, \dots, \boldsymbol{\lambda}_{i_{\mu_i}}\} & \longmapsto & \sigma_3\left(\{\boldsymbol{\lambda}_{i_1}, \dots, \boldsymbol{\lambda}_{i_{\mu_i}}\}\right) = \{\boldsymbol{z}_{i_1}, \dots, \boldsymbol{z}_{i_{\mu_i}}\} \end{array}$$

The composition of these linear mappings, $\sigma = \sigma_3 \circ \sigma_2 \circ \sigma_1$, maps a sub-vector of degrees of freedom $\boldsymbol{y}_i$ to a subset of nodes of the nodal distribution expressed in cartesian coordinates, $\{\boldsymbol{z}_{i_1}, \dots, \boldsymbol{z}_{i_{\mu_i}}\}$. Applying the mapping $\sigma$ to the degrees of freedom of all the representatives consecutively, we recover the whole nodal distribution. Specifically, if we denote by $m$ the number of representative nodes, and by $\boldsymbol{\sigma}$ the mapping applied to the whole vector of degrees of freedom, mapping $\boldsymbol{\sigma}$ consists in applying the mapping $\sigma$ to each sub-vector describing the representatives

$$\begin{array}{rccccccc} \boldsymbol{\sigma}: & \mathbb{R}^n & \to & \mathbb{R}^{n_1} \times \overset{m}{\cdots} \times \mathbb{R}^{n_m} & \to & \{\mathbb{R}^d\}^{\mu_1} \times \overset{m}{\cdots} \times \{\mathbb{R}^d\}^{\mu_m} & \to & \{\mathbb{R}^d\}^{N_{p,d}} \\ & \boldsymbol{y} & \mapsto & (\boldsymbol{y}_1, \dots, \boldsymbol{y}_m) & \mapsto & (\sigma(\boldsymbol{y}_1), \dots, \sigma(\boldsymbol{y}_m)) & \mapsto & \boldsymbol{Z} \end{array}, \quad \text{(C.2)}$$

where the first and last mappings are simply a split and a concatenation, respectively. We highlight that this mapping is linear and, therefore, the second derivatives with respect to the degrees of freedom are null. The use of this mapping allows us to properly define the functional to optimize in terms of the degrees of freedom of the representatives, i.e. the actual design variables.

# Appendix D

# Chen-Babuska derivatives and optimization

In this appendix, we describe how we optimize a symmetric nodal distribution with respect to the Chen-Babuska functional. Sect. D.1 is devoted to a careful derivation of the first and second derivatives of the Chen-Babuska functional. Remarkably, it is possible to get rid of the integral term using an orthogonal basis. We refer the reader to Eq. (D.7) and Eq. (D.8) for the final expressions. In Sect. D.2, we detail the second-order method used to optimize the Chen-Babuska value of a given initial nodal distribution.

## D.1 Derivatives of the functional

Following, we derive the expressions for the evaluation of the Chen-Babuska functional, as well as the first and second derivatives. We follow the content in Roth (2005) — the expressions are stated in this thesis just for completeness.

### D.1.1 Preliminaries

Let us denote by $\boldsymbol{z}_k$ the $k$-th node with components $\boldsymbol{z}_k = (z_k^{x_1}, \cdots, z_k^{x_d})$, and by $\frac{\partial}{\partial z_k^{x_r}}$ the derivative with respect to the $r$-th coordinate of the $k$-th node. We introduce the

mapping $\pi_i(\boldsymbol{Z}) := \boldsymbol{z}_i$ which retrieves the $i$-th node from a nodal set $\boldsymbol{Z}$. Note that

$$\frac{\partial}{\partial z_k^{x_r}} \pi_i(\boldsymbol{Z}) = \boldsymbol{e}_r \delta_{ki} = \begin{cases} \boldsymbol{0}, & \text{if } k \neq i \\ \boldsymbol{e}_r, & \text{if } k = i \end{cases},$$

where $\boldsymbol{e}_r$ is the canonical vector of $\mathbb{R}^d$ with all the components equal to zero, except for the $r$-th that equals one, and $\delta_{ki}$ is the Kronecker delta.

Let $\boldsymbol{b} = \{b_j(\boldsymbol{x})\}_{j=1,\dots,N_{p,d}}$ be an orthonormal basis of the space of polynomials in $d$ variables of degree equal or less than $p$, and denote by $V(\boldsymbol{Z}, \boldsymbol{b})$ the Vandermonde matrix with entries $V_{ij}(\boldsymbol{Z}, \boldsymbol{b}) = b_j(\pi_i(\boldsymbol{Z}))$. The nodal derivatives of the polynomial $b_j$ evaluated at the node $\boldsymbol{z}_i$ are

$$\begin{aligned} \frac{\partial}{\partial z_k^{x_r}} \left(b_j(\pi_i(\boldsymbol{Z}))\right) = \nabla b_j(\pi_i(\boldsymbol{Z})) \frac{\partial}{\partial z_k^{x_r}} \pi_i(\boldsymbol{Z}) = \nabla b_j(\pi_i(\boldsymbol{Z})) \boldsymbol{e}_r \delta_{ki} = \\ = \frac{\partial}{\partial x_r} b_j(\pi_i(\boldsymbol{Z})) \delta_{ki}. \end{aligned} \tag{D.1}$$

Hence, the nodal derivative of the Vandermonde matrix, denoted by $\frac{\partial}{\partial z_k^{x_r}} V(\boldsymbol{Z}, \boldsymbol{b})$, is a matrix with all the entries equal to zero, except for those of the $k$-th row that correspond to the terms $\frac{\partial}{\partial x_r} b_j(\pi_k(\boldsymbol{Z}))$, for $j = 1, \dots, N_{p,d}$. We highlight that the choice of the polynomial basis affects the entries of the Vandermonde matrix and, in particular, using the Lagrangian basis $\boldsymbol{\phi} = \{\phi_i(\boldsymbol{x}; \boldsymbol{Z})\}_{i=1,\dots,N_{p,d}}$ yields to $V_{ij}(\boldsymbol{Z}, \boldsymbol{\phi}) = \delta_{ij}$. We remark that since $\boldsymbol{\phi}$ is an interpolative basis, we have

$$V^T(\boldsymbol{Z}, \boldsymbol{b}) \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{Z}) = \boldsymbol{b}(\boldsymbol{x}). \tag{D.2}$$

Since the Vandermonde matrix does not depend on $\boldsymbol{x}$, the spatial derivatives satisfy a similar relation. In particular, for the first derivative it reads as

$$V^T(\boldsymbol{Z}, \boldsymbol{b}) \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{Z}) = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{b}(\boldsymbol{x}). \tag{D.3}$$

### D.1.2 Evaluation of the functional

In Sect. 5.4.1, we introduced the Chen-Babuska functional as a proxy of the Lebesgue constant,

$$\begin{array}{rccc} \tilde{\beta}: & \mathbb{R}^d \times \overset{N_{p,d}}{\cdots} \times \mathbb{R}^d & \longrightarrow & \mathbb{R} \\ & \boldsymbol{Z} & \longmapsto & \tilde{\beta}(\boldsymbol{Z}) = \int_{K^d} \sum_{i=1}^{N_{p,d}} \phi_i^2(\boldsymbol{x}; \boldsymbol{Z}) \, \mathrm{d}\boldsymbol{x}. \end{array}$$

We are interested in the computation of the first and second derivatives of $\tilde{\beta}(\boldsymbol{Z})$ with respect to a concrete spatial component of a particular node, namely $\frac{\partial}{\partial z_k^{x_r}}\tilde{\beta}(\boldsymbol{Z})$ and $\frac{\partial}{\partial z_j^{x_s}}\frac{\partial}{\partial z_k^{x_r}}\tilde{\beta}(\boldsymbol{Z})$, for $k, j = 1, \ldots, d$, and $r, s = 1, \ldots, N_{p,d}$. Since the integral is a linear operator, we will focus on the term

$$W_{ij} = W_{ij}(\boldsymbol{Z}) := \int_{K^d} \phi_i(\boldsymbol{x}; \boldsymbol{Z})\, \phi_j(\boldsymbol{x}; \boldsymbol{Z})\, \mathrm{d}\boldsymbol{x}.$$

We define the matrix $W$ as

$$W = \int_{K^d} \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{Z})\, \boldsymbol{\phi}^T(\boldsymbol{x}; \boldsymbol{Z})\, \mathrm{d}\boldsymbol{x},$$

where the integral is applied element-wise. Now, using Eq. (D.2), it expands to

$$W = \int_{K^d} \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{Z})\, \boldsymbol{\phi}^T(\boldsymbol{x}; \boldsymbol{Z})\, \mathrm{d}\boldsymbol{x} = \int_{K^d} \left(V^{-T}(\boldsymbol{Z}, \boldsymbol{b})\, \boldsymbol{b}(\boldsymbol{x})\right) \left(V^{-T}(\boldsymbol{Z}, \boldsymbol{b})\, \boldsymbol{b}(\boldsymbol{x})\right)^T \mathrm{d}\boldsymbol{x},$$

and since $V(\boldsymbol{Z}, \boldsymbol{b})$ does not depend on $\boldsymbol{x}$,

$$W = V^{-T}(\boldsymbol{Z}, \boldsymbol{b}) \left( \int_{K^d} \boldsymbol{b}(\boldsymbol{x})\, \boldsymbol{b}^T(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \right) V^{-1}(\boldsymbol{Z}, \boldsymbol{b}).$$

Now, using that the basis $\boldsymbol{b}$ is orthonormal,

$$W = V^{-T}(\boldsymbol{Z}, \boldsymbol{b})\, V^{-1}(\boldsymbol{Z}, \boldsymbol{b}) = \left(V(\boldsymbol{Z}, \boldsymbol{b})\, V^T(\boldsymbol{Z}, \boldsymbol{b})\right)^{-1}.$$

Finally, the functional $\tilde{\beta}(\boldsymbol{Z})$ reduces to the trace of the matrix $W$

$$\tilde{\beta}(\boldsymbol{Z}) = \sum_{i=1}^{N_{p,d}} W_{ii} = \mathrm{tr}(W)$$

We highlight that we have got rid of the integrals, and only a matrix product and inverse are needed.

### D.1.3 First-order derivatives

Taking derivatives with respect to $z_k^{x_r}$ leads to

$$\frac{\partial}{\partial z_k^{x_r}}\tilde{\beta}(\boldsymbol{Z}) = \sum_{i=1}^{N_{p,d}} \int_{K^d} 2\phi_i(\boldsymbol{x}; \boldsymbol{Z})\, \frac{\partial}{\partial z_k^{x_r}}\phi_i(\boldsymbol{x}; \boldsymbol{Z})\, \mathrm{d}\boldsymbol{x}. \tag{D.4}$$

In order to provide a more explicit expression for the term $\frac{\partial}{\partial z_k^{x_r}}\phi_i(\boldsymbol{x}; \boldsymbol{Z})$, we differentiate Eq. (D.2) with respect to $z_k^{x_r}$,

$$\left(\frac{\partial}{\partial z_k^{x_r}} V^T(\boldsymbol{Z}, \boldsymbol{b})\right) \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{Z}) + V^T(\boldsymbol{Z}, \boldsymbol{b}) \left(\frac{\partial}{\partial z_k^{x_r}} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{Z})\right) = \frac{\partial}{\partial z_k^{x_r}} \boldsymbol{b}(\boldsymbol{x}). \tag{D.5}$$

Since $\boldsymbol{b}(\boldsymbol{x})$ does not depend on any of the nodes $\boldsymbol{Z}$, the right hand side is the zero vector. The $i$-th row of the first term can be written as

$$\sum_{j=1}^{N_{p,d}} \left( \frac{\partial}{\partial z_k^{x_r}} V^T(\boldsymbol{Z}, \boldsymbol{b}) \right)_{ij} \phi_j(\boldsymbol{x}, \boldsymbol{Z}). \tag{D.6}$$

Recall that by Eq. (D.1),

$$\left( \frac{\partial}{\partial z_k^{x_r}} V^T(\boldsymbol{Z}, \boldsymbol{b}) \right)_{ij} = \frac{\partial}{\partial z_k^{x_r}} b_i(\pi_j(\boldsymbol{Z})) = \frac{\partial}{\partial x_r} b_i(\pi_j(\boldsymbol{Z})) \delta_{kj},$$

and, therefore, substituting in Eq. (D.6),

$$\sum_{j=1}^{N_{p,d}} \left( \frac{\partial}{\partial z_k^{x_r}} V^T(\boldsymbol{Z}, \boldsymbol{b}) \right)_{ij} \phi_j(\boldsymbol{x}, \boldsymbol{Z}) = \frac{\partial}{\partial x_r} b_i(\pi_k(\boldsymbol{Z})) \phi_k(\boldsymbol{x}, \boldsymbol{Z}),$$

and using the relation in Eq. (D.3), we get

$$\frac{\partial}{\partial x_r} b_i(\pi_k(\boldsymbol{Z})) \phi_k(\boldsymbol{x}, \boldsymbol{Z}) = \left( \sum_{j=1}^{N_{p,d}} V_{ji}(\boldsymbol{Z}, \boldsymbol{b})(\boldsymbol{Z}) \frac{\partial \phi_j}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}) \right) \phi_k(\boldsymbol{x}, \boldsymbol{Z}).$$

Replacing this expression back in Eq. (D.5), we obtain

$$\phi_k(\boldsymbol{x}, \boldsymbol{Z}) V^T(\boldsymbol{Z}, \boldsymbol{b})(\boldsymbol{Z}) \frac{\partial \boldsymbol{\phi}}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}) + V^T(\boldsymbol{Z}, \boldsymbol{b}) \left( \frac{\partial}{\partial z_k^{x_r}} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{Z}) \right) = \boldsymbol{0},$$

and isolating,

$$\frac{\partial}{\partial z_k^{x_r}} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{Z}) = -\phi_k(\boldsymbol{x}, \boldsymbol{Z}) \frac{\partial \boldsymbol{\phi}}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}).$$

Now, we can use this explicit expression in Eq. (D.4),

$$\begin{aligned}
\frac{\partial}{\partial z_k^{x_r}} \tilde{\beta}(\boldsymbol{Z}) &= \sum_{i=1}^{N_{p,d}} \int_{K^d} 2\phi_i(\boldsymbol{x}; \boldsymbol{Z}) \frac{\partial}{\partial z_k^{x_r}} \phi_i(\boldsymbol{x}; \boldsymbol{Z}) \, \mathrm{d}\boldsymbol{x} = \\
&= \sum_{i=1}^{N_{p,d}} \int_{K^d} 2\phi_i(\boldsymbol{x}; \boldsymbol{Z}) \left( -\phi_k(\boldsymbol{x}, \boldsymbol{Z}) \frac{\partial \phi_i}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}) \right) \mathrm{d}\boldsymbol{x} = \\
&= -2 \sum_{i=1}^{N_{p,d}} \frac{\partial \phi_i}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}) \int_{K^d} \phi_i(\boldsymbol{x}; \boldsymbol{Z}) \phi_k(\boldsymbol{x}, \boldsymbol{Z}) \, \mathrm{d}\boldsymbol{x}
\end{aligned}$$

Finally, the first derivatives of the Chen-Babuska functional are

$$\frac{\partial}{\partial z_k^{x_r}} \tilde{\beta}(\boldsymbol{Z}) = -2 \sum_{i=1}^{N_{p,d}} \frac{\partial \phi_i}{\partial x_r}(\pi_k(\boldsymbol{Z}); \boldsymbol{Z}) W_{ik} \tag{D.7}$$

### D.1.4 Second-order derivatives

For the second derivatives, we differentiate Eq. (D.7) with respect to $z_j^{x_s}$,

$$\frac{\partial}{\partial z_j^{x_s}}\frac{\partial}{\partial z_k^{x_r}}\tilde{\beta}\left(\boldsymbol{Z}\right)=-2\sum_{i=1}^{N_{p,d}}\left(\frac{\partial}{\partial z_j^{x_s}}\frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\right)W_{ik}+\frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\left(\frac{\partial}{\partial z_j^{x_s}}W_{ik}\right).$$

Let us start with the term $\frac{\partial}{\partial z_j^{x_s}}W_{ik}$. Expanding terms we get

$$\frac{\partial}{\partial z_j^{x_s}}W_{ik}=\int_{K^d}\left(\frac{\partial}{\partial z_j^{x_s}}\phi_i\left(\boldsymbol{x};\boldsymbol{Z}\right)\right)\phi_k\left(\boldsymbol{x},\boldsymbol{Z}\right)\mathrm{d}\boldsymbol{x}+\int_{K^d}\phi_i\left(\boldsymbol{x},\boldsymbol{Z}\right)\left(\frac{\partial}{\partial z_j^{x_s}}\phi_k\left(\boldsymbol{x};\boldsymbol{Z}\right)\right)\mathrm{d}\boldsymbol{x},$$

and using the same reasoning as for the first derivatives we obtain

$$\frac{\partial}{\partial z_j^{x_s}}W_{ik}=-\frac{\partial\phi_i}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)W_{jk}-\frac{\partial\phi_k}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)W_{ij}.$$

In order to provide an explicit expression of the term $\frac{\partial}{\partial z_j^{x_s}}\frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)$, we differentiate

$$V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\frac{\partial\boldsymbol{\phi}}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)=\frac{\partial}{\partial x_r}\boldsymbol{b}\left(\pi_k\left(\boldsymbol{Z}\right)\right)$$

with respect to $z_j^{x_s}$ to obtain

$$\begin{aligned}&\left(\frac{\partial}{\partial z_j^{x_s}}V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\right)\frac{\partial\boldsymbol{\phi}}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)+V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\left(\frac{\partial}{\partial z_j^{x_s}}\frac{\partial\boldsymbol{\phi}}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\right)=\\&=\frac{\partial}{\partial z_j^{x_s}}\frac{\partial}{\partial x_r}\boldsymbol{b}\left(\pi_k\left(\boldsymbol{Z}\right)\right)\end{aligned}$$

Note that now the right hand side is non-zero because it depends on the nodes. For the left hand side, we proceed analogously as for the first derivatives to get

$$\left(\frac{\partial}{\partial z_j^{x_s}}V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\right)\frac{\partial\boldsymbol{\phi}}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)=V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\left(\boldsymbol{Z}\right)\frac{\partial\boldsymbol{\phi}}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\frac{\partial\phi_j}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right).$$

The right hand side becomes

$$\frac{\partial}{\partial z_j^{x_s}}\frac{\partial}{\partial x_r}\boldsymbol{b}\left(\pi_k\left(\boldsymbol{Z}\right)\right)=\frac{\partial^2}{\partial x_s\partial x_r}\boldsymbol{b}\left(\pi_k\left(\boldsymbol{Z}\right)\right)\delta_{jk}=V^T\left(\boldsymbol{Z},\boldsymbol{b}\right)\left(\boldsymbol{Z}\right)\frac{\partial^2}{\partial x_s\partial x_r}\boldsymbol{\phi}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\delta_{jk}.$$

Coupling all together,

$$\frac{\partial}{\partial z_j^{x_s}}\frac{\partial\boldsymbol{\phi}}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)=-\frac{\partial\boldsymbol{\phi}}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\frac{\partial\phi_j}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)+\frac{\partial^2}{\partial x_s\partial x_r}\boldsymbol{\phi}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\delta_{jk}$$

In conclusion, the second derivatives of the Chen-Babuska functional are

$$\begin{aligned}\frac{\partial}{\partial z_j^{x_s}}\frac{\partial}{\partial z_k^{x_r}}\tilde{\beta}\left(\boldsymbol{Z}\right) &= -2\sum_{i=1}^{N_{p,d}}\left(\frac{\partial}{\partial z_j^{x_s}}\frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\right)W_{ik} + \frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\left(\frac{\partial}{\partial z_j^{x_s}}W_{ik}\right)\\ &= 2\sum_{i=1}^{N_{p,d}}\frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\frac{\partial\phi_i}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)W_{jk} + \frac{\partial\phi_i}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\frac{\partial\phi_k}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)W_{ij} +\\ &+ \frac{\partial\phi_i}{\partial x_s}\left(\pi_j\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\frac{\partial\phi_j}{\partial x_r}\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)W_{ik} - \frac{\partial^2}{\partial x_s\partial x_r}\phi_i\left(\pi_k\left(\boldsymbol{Z}\right);\boldsymbol{Z}\right)\delta_{jk}W_{ik}.\end{aligned} \tag{D.8}$$

### D.1.5 Symmetric nodal distributions

In this thesis we consider symmetric nodal distributions on the simplex. As detailed in Appendix C, a vector $\boldsymbol{y}$ of length $n$ encodes the position of all the points. Thus, the actual functional we consider is

$$\beta\left(\boldsymbol{y}\right) := \frac{1}{\mathrm{vol}\left(K^d\right)}\int_{K^d}\sum_{i=1}^{N_{p,d}}\phi_i^2\left(\boldsymbol{x};\boldsymbol{\sigma}\left(\boldsymbol{y}\right)\right)\mathrm{d}\boldsymbol{x}.$$

where $\boldsymbol{\sigma}$ generates the cartesian coordinates of the nodal distributions from the vector of degrees of freedom. To compute the derivatives of $\beta$, we apply the chain rule. Fortunately, the mapping $\boldsymbol{\sigma}$ is linear and computations are simplified.

## D.2 Optimization

In this section, we detail the second-order method used to optimize the Chen-Babuska functional. Specifically, we use Newton's method with a trust-region globalization. The radius of confidence accounts for the point resolution and ensures smooth dynamics of the points. Moreover, feasibility of the nodal distribution is preserved throughout all the optimization process.

The complete algorithm is stated in Algorithm D.1. Given a symmetric nodal configuration $\boldsymbol{Z}_0$, in Line 2 we compute the corresponding degrees of freedom $\boldsymbol{y}^{(0)}$. Next, in Lines 3-5, we set the algorithm parameters. We denote by $\delta$ the maximum allowed displacement, that is, the maximum spherical distance a node can move between two consecutive iterations. We set $\delta$ to a fraction of the spherical edge length in terms of the polynomial degree. Next, we set the largest allowable trust

**Algorithm D.1** Optimizing the functional $\beta(\boldsymbol{y})$.

**Input:** Nodal distribution $\boldsymbol{Z}_0$, Function to optimize $\beta$
**Output:** Nodal distribution $\boldsymbol{Z}$

1: **function** `OptimizePoints`($\boldsymbol{Z}_0$, $\beta$)
2: $\quad\boldsymbol{y}^{(0)} \leftarrow$ `GetVectorOfDOFs`($\boldsymbol{Z}_0$)
3: $\quad\delta = \frac{\pi}{2}\frac{1}{16p}$
4: $\quad r_{\max} = \delta\sqrt{n}$
5: $\quad r^{(0)} \leftarrow r_{\max}$
6: $\quad k \leftarrow 0$
7: $\quad$**while** $\left\|\nabla\beta\left(\boldsymbol{y}^{(k)}\right)\right\|_{\text{RMS}} > \varepsilon$ and $k < k_{\max}$ **do**
8: $\quad\quad\boldsymbol{y}^{(k+1)}, r^{(k+1)} \leftarrow$ `OneIterationTrustRegion`($\beta$,$\boldsymbol{y}^{(k)}$,$r^{(k)}$,$r_{\max}$)
9: $\quad\quad$**while** not `isFeasible`($\boldsymbol{y}^{(k+1)}$) or `maxNodalDistance`($\boldsymbol{y}^{(k)}$,$\boldsymbol{y}^{(k+1)}$) $> \delta$ **do**
10: $\quad\quad\quad r^{(k)} \leftarrow r^{(k)}/2$
11: $\quad\quad\quad\boldsymbol{y}^{(k+1)}, r^{(k+1)} \leftarrow$ `OneIterationTrustRegion`($\beta$,$\boldsymbol{y}^{(k)}$,$r^{(k)}$,$r_{\max}$)
12: $\quad\quad\quad r^{(k+1)} \leftarrow r^{(k)}$
13: $\quad\quad$**end while**
14: $\quad\quad k \leftarrow k + 1$
15: $\quad$**end while**
16: $\quad\boldsymbol{Z} \leftarrow \boldsymbol{\sigma}\left(\boldsymbol{y}^{(k)}\right)$
17: $\quad$**return** $\boldsymbol{Z}$
18: **end function**

region radius $r_{\max}$ and the starting trust region radius $r^{(0)}$. The maximum radius $r_{\max}$ is defined in terms of the maximum allowed displacement $\delta$ and the number of degrees of freedom $n$, and the radius for the first iteration $r^{(0)}$ is set to the maximum radius.

Once all the parameters have been specified, we compute the new nodal position. We do so by performing one iteration of trust region method, see Line 8. The call to this function returns the new vector of degrees of freedom $\boldsymbol{y}^{(k+1)}$ and the suggested radius for the next iteration $r^{(k+1)}$. Then, we check if $\boldsymbol{y}^{(k+1)}$ satisfies the domain and optimization constraints. More precisely, in Line 9, we check if the degrees of freedom are feasible, i.e., if they belong to their simplicial domain, and if the nodes that correspond to the vector $\boldsymbol{y}^{(k+1)}$ have moved more than the maximum allowed displacement $\delta$. If the nodes are valid, we proceed to the next iteration. On the contrary, while any of these constraints is violated, we recompute the new position by calling the trust region method with half the starting radius, Lines 10 and 11.

At some point, the degrees of freedom become feasible and the nodes do not move more than $\delta$. Since the estimation for the next radius does not take into account

these constraints, the method usually suggests incrementing it. In our experience, using this larger radius in the next iteration we find unfeasible nodal distributions. Thus, we discard the estimation for the next radius obtained from the trust region method and use the initial radius that worked in the current iteration, see Line 12.

This process is repeated until gradient convergence is achieved, $\left\|\nabla\beta\left(\boldsymbol{y}^{(k)}\right)\right\|_{\mathrm{RMS}} > \varepsilon$, see Line 7. We remark that we use the root mean square of the gradient in the stopping criterion. This is so because the length of the gradient vector depends on the number of degrees of freedom, which vary in terms of the dimension and the polynomial degree and, therefore, we need a normalized measure to solve all the problems with the same accuracy. Hence, we set $\varepsilon = 10^{-6}$ independently of the dimension and the polynomial degree. If gradient convergence is not achieved, the algorithm stops once a limit of iterations $k_{\max}$ is reached, $k_{\max} = 1000$. In such case, we solve the optimization problem using a specific steepest descent algorithm. We expect this more robust method to exit the conflicting zone and improve the current solution.

# Bibliography


Angelos, J. R., Kaufman Jr, E. H., Henry, M. S., and Lenker, T. D. Optimal nodes for polynomial interpolation. *Approximation theory VI*, 1:17–20, 1989.

Argyris, J. H., Fried, I., and Scharpf, D. W. The TUBA family of plate elements for the matrix displacement method. *The Aeronautical Journal*, 72(692):701–709, 1968.

Badia, S. and Olm, M. Space-time balancing domain decomposition. *SIAM journal on scientific computing*, 39(2):C194–C213, 2017.

Baran, M. Complex equilibrium measure and Bernstein type theorems for compact sets in $\mathbb{R}^n$. *Proceedings of the American Mathematical Society*, 123(2):485–494, 1995.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

Belda-Ferrín, G., Gargallo-Peiró, A., and Roca, X. Local bisection for conformal refinement of unstructured 4D simplicial meshes. *27th International Meshing Roundtable 27*, pages 229–247, 2019.

Belda-Ferrín, G., Ruiz-Gironés, E., Gargallo-Peiró, A., and Roca, X. Conformal Marked Bisection for Local Refinement of n-Dimensional Unstructured Simplicial Meshes. *Computer-Aided Design*, 154:103419, 2023.

Bell, K. A refined triangular plate bending finite element. *International journal for numerical methods in engineering*, 1(1):101–122, 1969.

Bernstein, S. Sur la limitation des valeurs d'un polynôme $P_n(x)$ de degré $n$ sur tout un segment par ses valeurs en $(n+1)$ points du segment. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, (8):1025–1050, 1931.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL `https://epubs.siam.org/doi/10.1137/141000671`.

Bloom, T. The Lebesgue constant for Lagrange interpolation in the simplex. *Journal of approximation theory*, 54(3):338–353, 1988.

Bloom, T., Bos, L., Christensen, C., and Levenberg, N. Polynomial interpolation of holomorphic functions in $\mathbb{C}$ and $\mathbb{C}^n$. *The Rocky Mountain Journal of Mathematics*, 22(2):441–470, 1992.

Blyth, M. and Pozrikidis, C. A Lobatto interpolation grid over the triangle. *IMA journal of applied mathematics*, 71(1):153–169, 2005.

Bos, L. Bounding the Lebesgue function for Lagrange interpolation in a simplex. *Journal of Approximation Theory*, 38(1):43–59, 1983.

Briani, M., Sommariva, A., and Vianello, M. Computing Fekete and Lebesgue points: simplex, square, disk. *Journal of Computational and Applied Mathematics*, 236(9): 2477–2486, 2012.

Buss, S. R. and Fillmore, J. P. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics (TOG)*, 20(2):95–126, 2001.

Byrd, R. H., Hribar, M. E., and Nocedal, J. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

Calvi, J.-P. and Levenberg, N. Uniform approximation by discrete least squares polynomials. *Journal of Approximation Theory*, 152(1):82–100, 2008.

Caplan, P. C., Haimes, R., Darmofal, D. L., and Galbraith, M. C. Four-dimensional anisotropic mesh adaptation. *Computer-Aided Design*, 129:102915, 2020.

Chaurasia, H., Roca, X., Persson, P., and Peraire, J. A coarse-to-fine approach for efficient deformation of curved high-order meshes. *Research Notes, 21st Int. Meshing Roundtable, Springer International Publishing*, pages 1–5, 2012.

Chaurasia, H. K. *Active pitch control of an oscillating foil with biologically-inspired boundary layer feedback*. PhD thesis, Massachusetts Institute of Technology, 2010.

Chen, Q. and Babuška, I. Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Computer Methods in Applied Mechanics and Engineering*, 128(3-4):405–417, 1995.

Chen, Q. and Babuška, I. The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. *Computer methods in applied mechanics and engineering*, 137(1):89–94, 1996.

Chuluunbaatar, G., Chuluunbaatar, O., Gusev, A., and Vinitsky, S. PI-type fully symmetric quadrature rules on the 3-,..., 6-simplexes. *Computers & Mathematics with Applications*, 124:89–97, 2022.

Dan, W. and Wang, R.-h. A fourth degree integration formula for the n-dimensional simplex. *Applied numerical mathematics*, 59(12):2990–2993, 2009.

Davis, P. J. *Interpolation and approximation*. Courier Corporation, 1975.

De Boor, C. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

De Boor, C. and Pinkus, A. Proof of the conjectures of Bernstein and Erdös concerning the optimal nodes for polynomial interpolation. 1978.

Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., and Liu, J. W. H. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

Dubiner, M. Spectral methods on triangles and other domains. *Journal of Scientific Computing*, 6(4):345–390, 1991.

Dunning, I., Huchette, J., and Lubin, M. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

Dyn, N., Levine, D., and Gregory, J. A. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)*, 9(2): 160–169, 1990.

Fekete, M. Über die Verteilung der Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten. *Mathematische Zeitschrift*, 17(1):228–249, 1923.

Foucault, G., Cuillière, J.-C., François, V., Léon, J.-C., and Maranzana, R. Adaptation of CAD model topology for finite element analysis. *Computer-Aided Design*, 40(2):176–196, 2008.

Foucault, G., Cuillière, J.-C., François, V., Léon, J.-C., and Maranzana, R. Generalizing the advancing front method to composite surfaces in the context of meshing constraints topology. *Computer-Aided Design*, 45(11):1408–1425, 2013.

Frontin, C. V., Walters, G. S., Witherden, F. D., Lee, C. W., Williams, D. M., and Darmofal, D. L. Foundations of space-time finite element methods: Polytopes, interpolation, and integration. *Applied Numerical Mathematics*, 166:92–113, 2021.

Gargallo-Peiró, A., Avila, M., Owen, H., Prieto, L., and Folch, A. Mesh generation for atmospheric boundary layer simulation in wind farm design and management. *Procedia Engineering*, 124:239–251, 2015a.

Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J. Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 103(5):342–363, 2015b. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.4888`.

Gargallo-Peiró, A., Houzeaux, G., and Roca, X. Subdividing triangular and quadrilateral meshes in parallel to approximate curved geometries. *Procedia Engineering*, 203:310 – 322, 2017. ISSN 1877-7058. URL `http://www.sciencedirect.com/science/article/pii/S1877705817343771`. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

Gargallo-Peiró, A. *Validation and generation of curved meshes for high-order unstructured methods*. PhD thesis, Universitat Politècnica de Catalunya. Departament de Matemàtica Aplicada III, July 2014.

Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J. Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Engineering with Computers*, 31(3):423–437, Jul 2015c. ISSN 1435-5663.

Gargallo-Peiró, A., Folch, A., and Roca, X. Representing urban geometries for unstructured mesh generation. *Procedia engineering*, 163:175–185, 2016a.

Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J. A distortion measure to validate and generate curved high-order meshes on CAD surfaces with independence of parameterization. *International Journal for Numerical Methods in Engineering*, 106(13):1100–1130, 2016b.

Gargallo-Peiró, A., Avila, M., Owen, H., Prieto-Godino, L., and Folch, A. Mesh generation, sizing and convergence for onshore and offshore wind farm Atmospheric Boundary Layer flow simulation with actuator discs. *Journal of Computational Physics*, 375:209–227, 2018.

George, P., Hecht, F., and Saltel, E. Automatic 3D mesh generation with prescribed meshed boundaries (alternator). *IEEE Transactions on magnetics*, 26(2):771–774, 1990.

Giraldo, F. X. and Taylor, M. A. A diagonal mass matrix triangular spectral element method based on cubature points. Technical report, Naval postgraduate school, Monterey, CA, Dept of Operations Research, 2006.

GLPK. GNU Linear Programming Kit. URL `http://www.gnu.org/software/glpk/glpk.html`.

Grundmann, A. and Möller, H.-M. Invariant integration formulas for the n-simplex by combinatorial methods. *SIAM Journal on Numerical Analysis*, 15(2):282–290, 1978.

Hammer, P., Marlowe, O., and Stroud, A. Numerical integration over simplexes and cones. *Mathematical Tables and Other Aids to Computation*, 10(55):130–137, 1956.

Hammer, P. C. and Stroud, A. H. Numerical integration over simplexes. *Mathematical tables and other aids to computation*, 10(55):137–139, 1956.

Heinrichs, W. Improved Lebesgue constants on the triangle. *Journal of Computational Physics*, 207(2):625–638, 2005.

Hesthaven, J. S. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM Journal on Numerical Analysis*, 35(2):655–676, 1998.

Hesthaven, J. S. and Teng, C.-H. Stable spectral methods on tetrahedral elements. *SIAM Journal on Scientific Computing*, 21(6):2352–2380, 2000.

Hesthaven, J. S. and Warburton, T. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

Ims, J., Duan, Z., and Wang, Z. J. meshCurve: An automated low-order to high-order mesh generator. In *22nd AIAA computational fluid dynamics conference*, page 2293, 2015.

Isaac, T. Recursive, parameter-free, explicitly defined interpolation nodes for simplices. *SIAM Journal on Scientific Computing*, 42(6):A4046–A4062, 2020.

Jameson, A., Vincent, P. E., and Castonguay, P. On the non-linear stability of flux reconstruction schemes. *Journal of Scientific Computing*, 50:434–445, 2012.

Jayasinghe, S., Darmofal, D. L., Burgess, N. K., Galbraith, M. C., and Allmaras, S. R. A space-time adaptive method for reservoir flows: formulation and one-dimensional application. *Computational Geosciences*, 22:107–123, 2018.

Jiao, X. and Wang, D. Reconstructing high-order surfaces for meshing. *Engineering with Computers*, 28(4):361–373, Oct 2012. ISSN 1435-5663.

Jiménez Ramos, A. Incorporating curvature to the boundary of linear and high-order meshes when a target geometry is unavailable. Master's thesis, Universitat Politècnica de Catalunya, 2018.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Subdivided Linear and Curved Meshes Preserving Features of a Linear Mesh Model. In *Proceedings of the 28th International Meshing Roundtable (IMR)*. Zenodo, February 2020. doi: 10.5281/zenodo.3653357.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Interpolation of subdivision features for curved geometry modeling. *Computer-Aided Design*, 145:103185, 2022.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Exploring locally optimal nodal distributions of a Lebesgue constant proxy in the high-dimensional simplex. *In preparation*, 2023a.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Computing interpolation-aware numerical quadratures in the high-dimensional simplex. *In preparation*, 2023b.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Computing nodal distributions with quasi-optimal Lebesgue constant in the high-dimensional simplex. *In preparation*, 2023c.

Jiménez-Ramos, A., Gargallo-Peiró, A., and Roca, X. Refining simplex points for scalable estimation of the Lebesgue constant. In *SIAM International Meshing Roundtable*, March 2023d.

Johnen, A., Roca, X., Toulorge, T., and Remacle, J. A new framework for curving structured boundary-layer meshes, 2018.

Johnen, A., Remacle, J.-F., and Geuzaine, C. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372, 2013.

Jones, D. R., Perttunen, C. D., and Stuckman, B. E. Lipschitzian optimization without the Lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.

Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

Karniadakis, G. E. and Sherwin, S. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press on Demand, 2005.

Keast, P. Moderate-degree tetrahedral quadrature formulas. *Computer methods in applied mechanics and engineering*, 55(3):339–348, 1986.

Kilgore, T. A. Optimization of the norm of the Lagrange interpolation operator. *Bulletin of the American Mathematical Society*, 83(5):1069–1071, 1977.

Kilgore, T. A. A characterization of the Lagrange interpolating projection with minimal Tchebycheff norm. 1978.

Kirby, R. C. Singularity-free evaluation of collapsed-coordinate orthogonal polynomials. *ACM Transactions on Mathematical Software (TOMS)*, 37(1):1–16, 2010.

Koornwinder, T. Two-variable analogues of the classical orthogonal polynomials. In *Theory and application of special functions*, pages 435–495. Elsevier, 1975.

Kopriva, D. A. and Gassner, G. On the quadrature and weak form choices in collocation type discontinuous Galerkin spectral element methods. *Journal of Scientific Computing*, 44:136–155, 2010.

Lalee, M., Nocedal, J., and Plantenga, T. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, 1998.

Lane, J. M. and Riesenfeld, R. F. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):35–46, 1980.

Li, X., Demmel, J., Gilbert, J., iL. Grigori, Shao, M., and Yamazaki, I. SuperLU Users' Guide. Technical Report LBNL-44289, Lawrence Berkeley National Laboratory, September 1999. `https://portal.nersc.gov/project/sparse/superlu/ug.pdf` Last update: June 2018.

Löhner, R. and Parikh, P. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10):1135–1149, 1988.

Loop, C. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, Department of Mathematics, The University of Utah, Masters Thesis, January 1987.

Loseille, A. and Rochery, L. P3 Bézier CAD Surrogates for anisotropic mesh adaptation. *Computer-Aided Design*, 160:103515, 2023.

Luo, H. and Pozrikidis, C. A Lobatto interpolation grid in the tetrahedron. *IMA journal of applied mathematics*, 71(2):298–313, 2006.

Luttmann, F. W. and Rivlin, T. J. Some numerical experiments in the theory of polynomial interpolation. *IBM Journal of Research and Development*, 9(3):187–191, 1965.

Lyness, J. and Jespersen, D. Moderate degree symmetric quadrature rules for the triangle. *IMA Journal of Applied Mathematics*, 15(1):19–32, 1975.

Maeztu, J. and Sáinz de La Maza, E. Consistent structures of invariant quadrature rules for the $n$-simplex. *Mathematics of computation*, 64(211):1171–1192, 1995.

Marchildon, A. L. and Zingg, D. W. Unisolvency for polynomial interpolation in simplices with symmetrical nodal distributions. *Journal of Scientific Computing*, 92(2):1–24, 2022.

Mogensen, P. K. and Riseth, A. N. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24):615, 2018. doi: 10.21105/joss.00615.

Moxey, D., Ekelschot, D., Keskin, Ü., Sherwin, S., and Peiró, J. High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72:130 – 139, 2016. ISSN 0010-4485. URL `http://www.sciencedirect.com/science/article/pii/S0010448515001530`. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

Murman, S. M., Diosady, L., Garai, A., and Ceze, M. A space-time discontinuous-Galerkin approach for separated flows. In *54th AIAA Aerospace Sciences Meeting*, page 1059, 2016.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Nolan, D. C., Tierney, C. M., Armstrong, C. G., and Robinson, T. T. Defining simulation intent. *Computer-aided design*, 59:50–63, 2015.

Pasquetti, R. and Rapetti, F. Spectral element methods on unstructured meshes: comparisons and recent advances. *Journal of Scientific Computing*, 27:377–387, 2006.

Pasquetti, R. and Rapetti, F. Spectral element methods on simplicial meshes. In *Spectral and High Order Methods for Partial Differential Equations-ICOSAHOM 2012: Selected papers from the ICOSAHOM conference, June 25-29, 2012, Gammarth, Tunisia*, pages 37–55. Springer, 2013.

Patera, A. T. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *Journal of computational Physics*, 54(3):468–488, 1984.

Paulavičius, R. and Žilinskas, J. Simplicial Lipschitz optimization without Lipschitz constant. In *Simplicial Global Optimization*, pages 61–86. Springer, 2014.

Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C. Adaptive remeshing for compressible flow computations. *Journal of computational physics*, 72(2):449–466, 1987.

Perronnet, A. Interpolation transfinie sur le triangle, le tétraèdre et le pentaèdre. Application à la création de maillages et à la condition de Dirichlet. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 326(1):117–122, 1998.

Persson, P.-O. and Peraire, J. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 12 2008.

Persson, P.-O., Aftosmis, M. J., and Haimes, R. On the Use of Loop Subdivision Surfaces for Surrogate Geometry. In Pébay, P. P., editor, *Proceedings of the 15th International Meshing Roundtable*, pages 375–392, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-34958-7.

Quadros, W. R. and Owen, S. J. Defeaturing CAD models using a geometry-based size field and facet-based reduction operators. *Engineering with Computers*, 28(3): 211–224, 2012.

Rack, H.-J. An example of optimal nodes for interpolation. *International Journal of Mathematical Education in Science and Technology*, 15(3):355–357, 1984.

Rapetti, F., Sommariva, A., and Vianello, M. On the generation of symmetric Lebesgue-like points in the triangle. *Journal of Computational and Applied Mathematics*, 236(18):4925–4932, 2012.

Roth, M. J. *Nodal configurations and Voronoi tessellations for triangular spectral elements*. PhD thesis, 2005.

Ruiz-Gironés, E., Roca, X., and Sarrate, J. High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation. *Computer-Aided Design*, 72:52–64, 2016a.

Ruiz-Gironés, E., Sarrate, J., and Roca, X. Generation of curved high-order meshes with optimal quality and geometric accuracy. *Procedia engineering*, 163:315–327, 2016b.

Ruiz-Gironés, E., Gargallo-Peiró, A., Sarrate, J., and Roca, X. Automatically imposing incremental boundary displacements for valid mesh morphing and curving. *Computer-Aided Design*, 2019.

Rumsey, C. L., Slotnick, J. P., and Sclafani, A. J. Overview and Summary of the Third AIAA High Lift Prediction Workshop. *Journal of Aircraft*, 56(2):621–644, 2019.

Runge, C. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243):20, 1901.

Shapiro, V., Tsukanov, I., and Grishin, A. Geometric issues in computer aided design/computer aided engineering integration. *Journal of Computing and Information Science in Engineering*, 11(2), 2011.

Sheffer, A. Model simplification for meshing using face clustering. *Computer-Aided Design*, 33(13):925–934, 2001.

Sheffer, A., Bercovier, M., Blacker, T., and Clements, J. Virtual topology operators for meshing. *International Journal of Computational Geometry & Applications*, 10 (03):309–331, 2000.

Shewchuk, J. R. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering: FCRC'96 Workshop, WACG'96 Philadelphia, PA, May 27–28, 1996 Selected Papers*, pages 203–222. Springer, 2005.

Shunn, L. and Ham, F. Symmetric quadrature rules for tetrahedra based on a cubic close-packed lattice arrangement. *Journal of Computational and Applied Mathematics*, 236(17):4348–4364, 2012.

Si, H. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11, 2015.

Silvester, P. Symmetric quadrature formulae for simplexes. *Mathematics of Computation*, 24(109):95–100, 1970.

Stam, J. Evaluation of Loop Subdivision Surfaces. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, 01 1998.

Sturmfels, B. *Algorithms in invariant theory*. Springer Science & Business Media, 2008.

Taylor, M. and Wingate, B. The Fekete collocation points for triangular spectral elements. *SIAM Journal of Numerical Analysis*, 1999.

Taylor, M. A. and Wingate, B. A generalized diagonal mass matrix spectral element method for non-quadrilateral elements. *Applied Numerical Mathematics*, 33(1-4): 259–265, 2000.

Taylor, M. A., Wingate, B. A., and Vincent, R. E. An algorithm for computing Fekete points in the triangle. *SIAM Journal on Numerical Analysis*, 38(5):1707–1720, 2000.

Taylor, M. A., Wingate, B. A., and Bos, L. P. A cardinal function algorithm for computing multivariate quadrature points. *SIAM Journal on Numerical Analysis*, 45(1):193–205, 2007.

Thakur, A., Banerjee, A. G., and Gupta, S. K. A survey of CAD model simplification techniques for physics-based simulation applications. *Computer-Aided Design*, 41 (2):65–80, 2009.

The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition, 2023. URL `https://doc.cgal.org/5.5.2/Manual/packages.html`.

Toulorge, T., Geuzaine, C., Remacle, J.-F., and Lambrechts, J. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013.

Van Barel, M., Humet, M., and Sorber, L. Approximating optimal point configurations for multivariate polynomial interpolation. *Electronic Transactions on Numerical Analysis*, 42:41–63, 2014.

Van Rossum, G. and Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Wandzura, S. and Xiao, H. Symmetric quadrature rules on a triangle. *Computers & Mathematics with Applications*, 45(12):1829–1840, 2003.

Wang, L. and Persson, P.-O. A high-order discontinuous Galerkin method with unstructured space–time meshes for two-dimensional compressible flows on domains with large deformations. *Computers & Fluids*, 118:53–68, 2015.

Wanner, G. and Hairer, E. Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems. *Springer Berlin, Heidelberg*, 1996.

Warburton, T. An explicit construction of interpolation nodes on the simplex. *Journal of engineering mathematics*, 56(3):247–262, 2006.

White, D. R., Saigal, S., and Owen, S. J. Meshing complexity: predicting meshing difficulty for single part CAD models. *Engineering with Computers*, 21(1):76–90, 2005.

Williams, D. M. *Energy stable high-order methods for simulating unsteady, viscous, compressible flows on unstructured grids*. Stanford University, 2013.

Williams, D. M. and Jameson, A. Nodal points and the nonlinear stability of high-order methods for unsteady flow problems on tetrahedral meshes. In *21st AIAA Computational Fluid Dynamics Conference*, page 2830, 2013.

Williams, D. M., Frontin, C. V., Miller, E. A., and Darmofal, D. L. A family of symmetric, optimized quadrature rules for pentatopes. *Computers & Mathematics with Applications*, 80(5):1405–1420, 2020.

Williams, D., Shunn, L., and Jameson, A. Symmetric quadrature rules for simplexes based on sphere close packed lattice arrangements. *Journal of Computational and Applied Mathematics*, 266:18–38, 2014.

Witherden, F. D. and Vincent, P. E. An analysis of solution point coordinates for flux reconstruction schemes on triangular elements. *Journal of Scientific Computing*, 61:398–423, 2014.

Witherden, F. D. and Vincent, P. E. On the identification of symmetric quadrature rules for finite element methods. *Computers & Mathematics with Applications*, 69 (10):1232–1241, 2015.

Xiao, H. and Gimbutas, Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & mathematics with applications*, 59(2):663–676, 2010.

Yang, H. Q., Zhou, X., Harris, R. E., and Yang, S. An Open Source, Geometry Kernel Based High-Order Element Mesh Generation Tool. In *AIAA Scitech 2019 Forum*, page 1719, 2019.

Zhang, L., Cui, T., and Liu, H. A set of symmetric quadrature rules on triangles and tetrahedra. *Journal of Computational Mathematics*, pages 89–96, 2009.

Zorin, D. A method for analysis of $C^1$-continuity of subdivision surfaces. *SIAM Journal on Numerical Analysis*, 37(5):1677–1708, 2000.