Design Through Analysis

Möller, M.; Verhelst, H. M.

DOI
10.1007/978-3-031-47355-5_5

Publication date
2024

Document Version
Final published version

Published in
Fluids Under Control. Advances in Mathematical Fluid Mechanics.

Chapter 5
Design Through Analysis

Y. Ji, M. Möller, and H. M. Verhelst

5.1 Introduction

Numerical simulations of physical systems have become an indispensable third pillar in modern computational sciences and engineering (CSE) complementing theoretical and experimental analysis. Most numerical methods in use today like the finite element method (FEM), the boundary element method (BEM), the finite volume method (FVM), and the finite difference method (FDM) have their origin many decades ago when computers delivered only a marginal fraction of their today's performance and were moreover a scarcely available resource, and CSE was at its infancy. It is therefore no surprise that all aforementioned numerical methods were originally designed as validation tools to be utilized deliberately in one of the final stages of the entire design and analysis workflow and not as a repeatedly queried in-the-loop tool. Over many decades, upstream processes like the creation and iterative optimization of designs were typically disconnected from the downstream computer-aided simulation-based analysis, which has led to the separation of the computer-aided design (CAD) and computer-aided engineering (CAE) communities, an unsatisfactory situation that is lasting until today.

In fact, even modern textbooks on numerical methods for (initial-)boundary value problems ((I)BVPs) teach the traditional triad of pre-processing, analysis, and post-processing, thereby cementing the separation of CAD and CAE disciplines. If the focus lies on the theoretical study of the methods' mathematical properties, it is understandable why pre- and post-processing are considered secondary tasks. However, in engineering practice it is these upstream pre-processing steps that create the problem-specific inputs to the analysis, for instance, the geometry model on which the (I)BVP needs to be solved.

Y. Ji · M. Möller (✉) · H. M. Verhelst
Delft University of Technology, Delft, CD, The Netherlands
e-mail: M.Moller@tudelft.nl

T. Bodnár et al. (eds.), *Fluids Under Control*, Advances in Mathematical Fluid Mechanics, https://doi.org/10.1007/978-3-031-47355-5_5

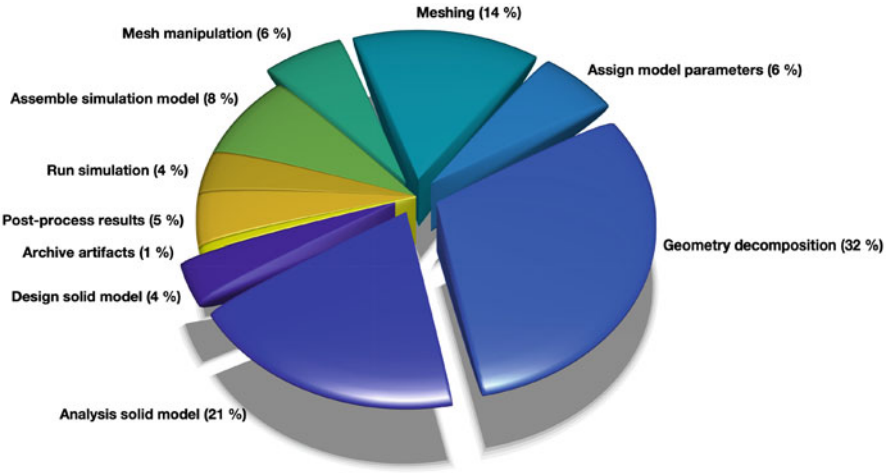


Fig. 5.1 Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories [8]. About 80% of the overall time is spent on mesh generation (20%) and the creation of the analysis-suitable geometry (60%) and only 20% go into the analysis per se

A study by Ted Blacker, Manager of Simulation Sciences, Sandia National Laboratories, has led to the often-cited 80/20 modeling/analysis ratio which states that about 80% of the overall time is spent on mesh generation (20%) and the creation of the analysis-suitable geometry (60%) and only 20% go into the analysis per se. The detailed breakdown in the percentage of time devoted to each task is given in Fig. 5.1, which is based on data from [8]. This 80/20 ratio is reported to be a common experience in industry and not specific to the Sandia report.

As it is often the case in disciplines that have a long but independent tradition, the disconnect between CAD and CAE is due to "historical reasons" and not caused by insuperable theoretical hurdles. Simply said, the CAD community has adopted non-uniform rational B-splines, in short NURBS [66], as industry standard for representing geometry models, whereas decades of research in numerical methods for (I)BVPs have brought a plethora of ways to express and compute solution fields and derived quantities of interest. As a consequence, the mathematical representation frameworks in CAD and CAE are often incompatible in the sense that a conversion is needed to cast, say, a continuously parametrized NURBS-based geometry model into a discrete grid with a finite number of cells.

The convergence step is not only time consuming as indicated in Blacker's study. It also introduces additional approximation errors and is not reversible in most cases. That is, the conversion of a parametric NURBS-based geometry model into a discrete mesh cannot be reversed easily if, say, the simulation involves some fluid-structure interaction which requires to update the geometry model from data stemming from the simulation-based analysis. The same problem arises in the PDE-constrained design optimization, where the shape of, say, an airfoil is adjusted

repeatedly within an outer optimization loop steered by quantities of interest such as drag and lift that are computed from the field variables of a flow analysis.

While the above scenario might leave the impression that the core of the problem is caused by CAE, we would like to stress that also the geometry models coming from CAD are far from being analysis suitable. Firstly, traditional CAD models typically provide a description of the geometries' surfaces, while numerical methods such as FEM and FVM often require volumetric meshes that need to be generated. Secondly, many CAD models of complex geometries consist of multiple NURBS patches that are often glued together in a non-watertight manner, thereby requiring tedious repair and reparameterization steps. Thirdly, as CAD models are primarily meant for visual inspection, they might contain fine-scale features below the resolution capability of the simulation that need to be removed by de-featuring processes. Even in the absence of such small-scale features, the parameterizations used in CAD models are typically not optimized for upstream analysis. For instance, NURBS surfaces with (locally) large Jacobians are good enough for the visual design but bring all kinds of problems in the analysis process, which is why even "good-looking" CAD models often require an overhaul in terms of reparameterization. Last but not least, the wide use of trimming, i.e., the selective deactivation of unwanted parts of NURBS surfaces, calls for special treatment in the simulation, e.g., in the framework if immersed methods [86, 95], through the adaptation of the discretization [56] or by decomposing trimmed models into untrimmed ones with many regular patches or cells [39].

Many of the abovementioned problems can be alleviated when basing CAD and CAE tools on a common mathematical representation framework that satisfies the needs of both worlds. For many applications in computational fluid dynamics (CFD) and computational solid mechanics (CSM), this common mathematical framework can be multi-variate splines such as NURBS [66], B-splines [15], hierarchical B-splines [25], truncated hierarchical B-splines [29, 30], T-splines [72] and their variants [49, 51, 50, 43, 52, 87], LR-splines [17], and T-splines and U-splines [78], to name some of the more prominent approaches.

From all the numerical methods that adopt the above philosophy of a unified representation framework for geometry modeling and PDE analysis, Isogeometric Analysis (IGA), which was introduced in 2005 by Hughes et al. [38], is by far the most prominent one nowadays. However, the quest for CAD/CAE integration is much older as described in the book "Precursors of Isogeometric Analysis" by Provatidis [68]. An approach that is conceptually close to IGA is the so-called NURBS-enhanced finite element method by Sevilla et al. [73, 74]. Despite common belief, the philosophy of a unified representation framework for geometry modeling and PDE analysis is not restricted to isogeometric variants of FEM and BEM but can also be applied in the context of FVM as demonstrated in [32, 57, 58].

Our main intention when writing this chapter was to sensitize the interested reader to the manifold opportunities that can arise from using a unified representation framework throughout the computer-aided design and analysis workflow be it in the context of IGA, NURBS-enhanced FEM/FVM, or any other numerical method that follows the underlying philosophy. At the very least, we want to stimulate

the reader to rethink whether the classical triad of pre-processing, analysis, and post-processing is still adequate at times when computer power is much less of a limiting factor than it was at the early days of CAD and CAE. Moreover, we want to illustrate the potential of genuine *design-through-analysis* workflows that, as the name suggests, enable practitioners to arrive at optimized designs in an iterative process through the repeated query to computer-aided analysis tools.

To the best of the authors' knowledge, the first occurrence of the term *design through analysis* in the literature dates back to the year 1977 when a team at the General Motors Research Laboratories wanted "to demonstrate and evaluate the design integrated analytic method when applied to the development of a complete automobile" [3]. Despite the very limited computer power available at the time the project was conducted, the visionary publication describes many aspects of modern design-through-analysis workflows, starting at "computer methods required for the evaluation of the entire vehicle system includ[ing] impact simulation analysis, static structural analysis and dynamic analysis" and ending at the "use of interactive graphics for modeling the vehicle." The publication concludes with the encouraging statement that "the potential value of design through analysis was demonstrated by a significant reduction in structural weight of the project vehicle." About five decades later, design-through-analysis workflows have become an integral part of daily engineering design practice. At the same time, it is a field of active research with novel machine learning techniques enabling yet another leap forward.

The rest of this chapter is organized as follows: Sect. 5.2 gives an introduction into splines followed by an overview of approaches to create analysis-suitable parameterizations in Sect. 5.3. Section 5.4 discusses several use-cases of isogeometric analysis, each of them illustrating a different facet of spline-based modeling and analysis. The chapter ends with conclusions and a brief outlook in Sect. 5.5.

5.2 A Spline Primer

The term "spline" has its roots in the aircraft and shipbuilding industries, where long and thin wooden, plastic, or metal strips (called "splines") were held in place at discrete points by lead weights (called "ducks" by Forrest because of their duck-like shape; Schoenberg refers to them as "dogs" or "rats"). The elasticity of the spline material combined with the constraint of the control points caused the strip to bend between the fixations into a shape of minimum strain energy, that is, the smoothest possible shape. This technique, illustrated in Fig. 5.2, had already been used for ship-hull design before the British aircraft industry adopted it during World War II to construct templates for airplanes. The first mathematical reference to splines goes back to the seminal work by Schoenberg [71] who established the characterization of splines as smooth, piecewise polynomial approximations.

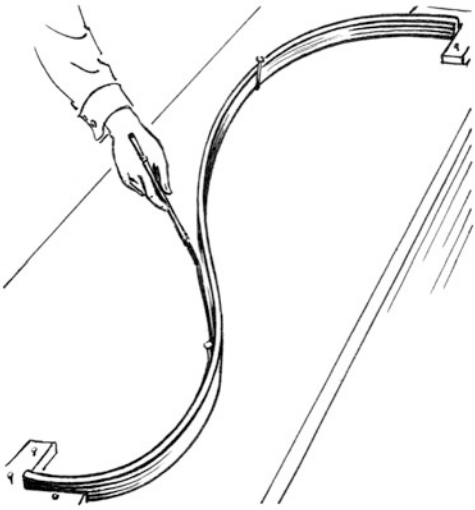


Fig. 5.2 Illustration of the working principle of "splines" as an early instrument for ship-hull and aircraft design. Thin wooden strips (called "splines") were held in place at discrete points by lead weights, which caused the spline to take the shape of minimum strain energy. The success design was then drawn on paper

5.2.1 B-Splines

This section is written in a tutorial style meaning that we present the relevant concepts "as is" without giving mathematical proofs, which can be found in many textbooks on splines. The description starts with an exhaustive discussion of univariate B-splines before it generalizes this concept to the multi-variate case.

Univariate B-Splines and Their Properties

The modern way to introduce splines is through the use of basis splines (B-splines) that are constructed by the Cox–de-Boor recursion formula from a so-called *knot vector*, which in its most general form is a sequence of non-decreasing real numbers

$$\Xi = [\xi_1, \xi_2, \ldots, \xi_{n+d+1}], \quad \xi_i \leqslant \xi_{i+1}, \ \forall i = 1, \ldots, n+d. \tag{5.1}$$

Here, $d \geqslant 0$ denotes the degree and n the number of B-splines to be created. Starting from the piecewise constant ($d = 0$) basis functions that are defined as

$$b^0_{i;\Xi}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leqslant \xi < \xi_{i+1}, \\ 0 \text{ otherwise}, \end{cases} \tag{5.2}$$

higher degree basis functions are constructed recursively [14]

$$b^d_{i;\Xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} b^{d-1}_{i;\Xi}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} b^{d-1}_{i+1;\Xi}(\xi). \tag{5.3}$$

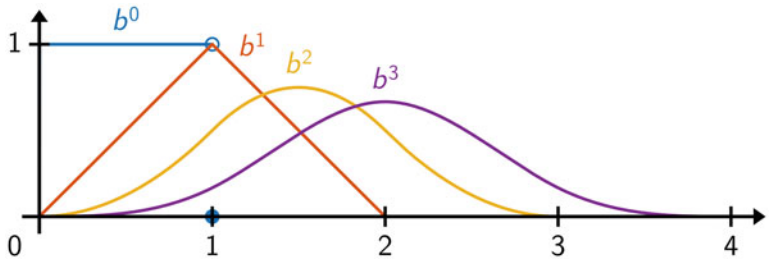


Fig. 5.3 Cardinal B-splines for the knot vector $\Xi = [0, 1, 2, 3, 4]$

Here, division by zero is precluded by assuming that "$0/0 := 0$." Figure 5.3 depicts a sequence of B-spline basis functions ranging from $d = 0$ up to degree $d = 3$.

The so-defined functions feature a set of amenable properties that make them particularly suited for both geometry modeling and PDE analysis. To start with, the functions are piecewise polynomials of degree d in the independent variable ξ. Moreover, like for standard finite element basis functions, their support is local and their value is, unlike in many finite element basis functions, non-negative:

$$b^d_{i;\Xi}(\xi) \begin{cases} > 0 \ \forall \xi \in \operatorname{supp}\left(b^d_{i;\Xi}\right) := [\xi_i, \xi_{i+d+1}), \\ = 0 \text{ otherwise.} \end{cases} \tag{5.4}$$

Looking at this property from a different perspective, in every *knot span* $[\xi_i, \xi_{i+1})$ at most $d + 1$ degree d basis functions are non-zero, namely, $b^d_{i-d;\Xi}, b^d_{i-d+1;\Xi}, \ldots, b^d_{i;\Xi}$. The set of basis functions moreover satisfies the partition of unity property

$$\sum_{i=1}^{n} b^d_{i;\Xi}(\xi) \equiv 1, \quad \forall \xi \in [\xi_{d+1}, \ldots \xi_n) \,. \tag{5.5}$$

It is moreover easy to show that the r-th derivative of a degree d B-spline with respect to the independent variable ξ is given by (cf. the proof of Lemma 3.20 in [60])

$$D^r b^d_{i;\Xi}(\xi) = d \left(\frac{D^{r-1} b^{d-1}_{i;\Xi}(\xi)}{\xi_{i+d} - \xi_i} - \frac{D^{r-1} b^{d-1}_{i+1;\Xi}(\xi)}{\xi_{i+d+1} - \xi_{i+1}} \right), \tag{5.6}$$

which can be worked out recursively to arrive at an explicit expression. However, computing derivatives in that way is tedious and not to be recommended. In the following section we will discuss a matrix representation of B-splines that simplifies the evaluation of function values and derivatives significantly.

Last but not least, B-spline basis functions have maximal continuity C^{d-1} provided that no two knots in the knot vector are the same. If the i-th knot ξ_i is

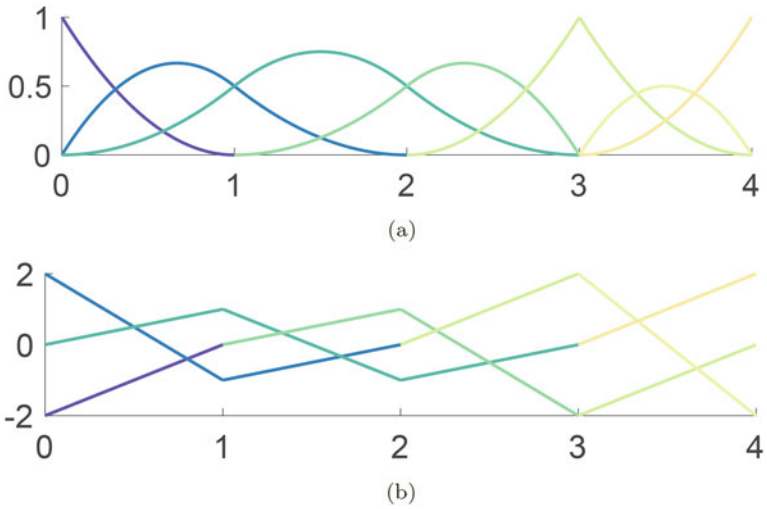


Fig. 5.4 Quadratic B-spline basis functions ($n = 7$, $d = 2$) and their first derivatives defined over the open knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$; the fifth basis function b_5^2 exhibits a local C^0 continuity around the repeated knot $\xi_6 = \xi_7 = 3$, which causes its derivative to jump at that point. (**a**) Univariate B-spline basis functions. (**b**) Derivatives of univariate B-spline basis functions

repeated $0 < m_i \leqslant d+1$ times, then the continuity reduces to C^{d-m_i} in the vicinity of that point. In practice, knot vectors are often chosen to be open, that is, the first and last knots are repeated $d+1$ times, which reduces the continuity locally to C^{-1} and causes the respective B-spline basis function to attain the function value one and all other basis functions to vanish at the two end points.

Figure 5.4 depicts the $n = 7$ degree $d = 2$ B-spline basis functions (top) and their first derivatives (bottom) generated from the open knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$ with a twice repeated knot $\xi_6 = \xi_7 = 3$, which leads to a local C^0 continuity of the fifth basis function b_5^2 around that point while elsewhere b_5^2 and all other basis functions are C^1. Consequently, Db_5^2 exhibits a jump at $\xi^* = 3$ and is C^0 continuous elsewhere as are all other basis functions.

Let us define the spline space $\mathbb{S}_\Xi^d$ as the space of all linear combinations of the B-spline basis functions as defined in (5.2)–(5.3), i.e.,

$$\mathbb{S}_\Xi^d = \text{span}\left\{b_{1;\Xi}^d, \ldots, b_{n;\Xi}^d\right\} \tag{5.7}$$

$$= \left\{\sum_{i=1}^{n} c_i b_{i;\Xi}^d \,:\, c_i \in \mathbb{R}, \text{ for } 1 \leqslant i \leqslant n\right\}. \tag{5.8}$$

An element $f(\xi) = \sum_{i=1}^n c_i b_{i;\Xi}^d(\xi)$ of this space is termed a *spline function* or just a *spline* and the set $(c_i)_{i=1}^n$ denotes the *B-spline coefficients* of f.

To improve readability, we drop subscript Ξ whenever there is no ambiguity about the knot vector used. Moreover, we introduce the vectors

$$\mathbf{b}^d = \begin{bmatrix} b_1^d & b_2^d & \cdots & b_n^d \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \tag{5.9}$$

so that the sum in (5.8) can be written as the dot product $\mathbf{b}^d \cdot \mathbf{c}$. Last but not least, let us define the selection operator "$[\cdot;\cdot]$" that selects a sub-vector, e.g.,

$$\mathbf{b}^d_{[j-d;j]} \cdot \mathbf{c}_{[j-d;j]} = \sum_{i=j-d}^{j} c_i b_i^d. \tag{5.10}$$

Among all possible points where a spline function $f(\xi)$ can be evaluated, the *Greville abscissae* play a special role. They are defined as the knot average [15]

$$\bar{\xi}_i = \frac{\xi_{i+1} + \cdots + \xi_{i+d}}{d}. \tag{5.11}$$

In general, $\bar{\xi}_i$ lies near the parameter value which corresponds to a maximum of the B-spline basis function b_i^d [66]. In combination with the B-spline coefficients $\mathbf{c}$, the pairs $(\bar{\xi}_i, c_i)$ form the so-called *control polygon* of the spline function f (cf. Theorem 2.8 in [60]). This is illustrated in Fig. 5.5 for the same knot vector as before and the vector of B-spline coefficients $\mathbf{c} = \begin{bmatrix} 0 & 2 & 1 & 1 & 3 & 1 & 2 \end{bmatrix}$. The repetition of knots causes the respective basis functions to attain the value one at the associated point and all other basis functions to vanish causing the spline function to become interpolatory, i.e., $f(\xi^* = 3) = c_5 = 3$ for the example above.

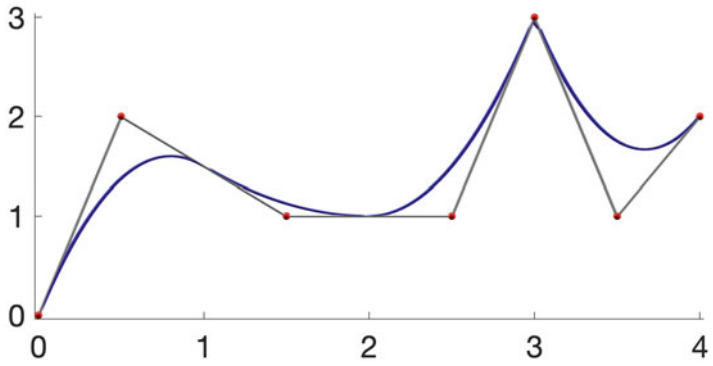


Fig. 5.5 Quadratic spline function and its control polygon constructed from $\Xi = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$ with B-spline coefficients $\mathbf{c} = [0 \quad 2 \quad 1 \quad 1 \quad 3 \quad 1 \quad 2]$

A Matrix Representation of B-Splines

Before we proceed to multi-variate B-splines, we would like to review the construction procedure (5.2)–(5.3) and discuss an alternative representation of B-splines that is hardly discussed in the literature but has several advantages from both the theoretical and the implementation point of view. The following description is based on chapters 2 and 3 from the lecture notes by Lyche and Mørken [60]. We encourage the reader to study these excellent notes for a detailed derivation of the matrix representation and for an in-depth discussion of splines and B-splines in general.

Exploiting the fact that at most $d+1$ degree d B-spline basis functions are non-zero per knot span, an element $f(\xi)$ of $\mathbb{S}^d$ can be written equivalently as

$$f(\xi) = \mathbf{b}^d_{[j-d;\,j]}(\xi) \cdot \mathbf{c}_{[j-d;\,j]}, \quad \forall \xi \in [\xi_j, \xi_{j+1}). \tag{5.12}$$

In Theorem 2.14 of [60], Lyche and Mørken provide an alternative representation of B-spline basis functions in terms of a chain of matrix products, namely

$$\mathbf{b}^d_{[j-d;\,j]}(\xi) = \mathbf{R}^d_1(\xi)\mathbf{R}^d_2(\xi)\cdots\mathbf{R}^d_d(\xi), \tag{5.13}$$

where for each positive integer $k \leqslant d$ the $k \times (k+1)$-dimensional *B-spline matrix* $\mathbf{R}^d_k$ is given by

$$\mathbf{R}^d_k(\xi) = \begin{bmatrix} \frac{\xi_{j+1}-\xi}{\xi_{j+1}-\xi_{j+1-k}} & \frac{\xi-\xi_{j+1-k}}{\xi_{j+1}-\xi_{j+1-k}} & 0 & \cdots & 0 \\ 0 & \frac{\xi_{j+2}-\xi}{\xi_{j+2}-\xi_{j+2-k}} & \frac{\xi-\xi_{j+2-k}}{\xi_{j+2}-\xi_{j+2-k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\xi_{j+k}-\xi}{\xi_{j+k}-\xi_j} & \frac{\xi-\xi_j}{\xi_{j+k}-\xi_j} \end{bmatrix}. \tag{5.14}$$

From the plentiful examples given in chapter 2 of the lecture notes [60], we would like to replicate Examples 2.12 and 2.13 to illustrate the above construction procedure. Let us consider linear B-splines ($d=1$) and let $\xi \in [\xi_j, \xi_{j+1})$. It follows from the local support property that the only two basis functions that are non-zero in that knot span are b^1_{j-1} and b^1_j. Their restriction to the interval can be given as

$$\begin{bmatrix} b^1_{j-1} & b^1_j \end{bmatrix} = \begin{bmatrix} \frac{\xi_{j+1}-\xi}{\xi_{j+1}-\xi_j} & \frac{\xi-\xi_j}{\xi_{j+1}-\xi_j} \end{bmatrix}. \tag{5.15}$$

Likewise, for quadratic B-splines ($d=2$) the row vector of B-splines that are non-zero in $[\xi_j, \xi_{j+1})$ can be written as the product of two matrices, namely

$$\begin{bmatrix} b^2_{j-2} & b^2_{j-1} & b^2_j \end{bmatrix} = \begin{bmatrix} \frac{\xi_{j+1}-\xi}{\xi_{j+1}-\xi_j} & \frac{\xi-\xi_j}{\xi_{j+1}-\xi_j} \end{bmatrix} \cdot \tag{5.16}$$

$$\begin{bmatrix} \frac{\xi_{j+1}-\xi}{\xi_{j+1}-\xi_{j-1}} & \frac{\xi-\xi_{j-1}}{\xi_{j+1}-\xi_{j-1}} & 0 \\ 0 & \frac{\xi_{j+2}-\xi}{\xi_{j+2}-\xi_j} & \frac{\xi-\xi_j}{\xi_{j+2}-\xi_j} \end{bmatrix}. \tag{5.17}$$

The matrix form moreover leads to "explicit" expressions for computing (higher order) derivatives of B-splines. Let us reconsider the general case (5.13). The vector of all r-th derivatives of degree d B-spline basis functions that are non-zero in $[\xi_j, \xi_{j+1})$ can be computed from (see Theorem 3.15 in [60])

$$D^r \mathbf{b}^d_{[j-d;j]}(\xi) = \frac{d!}{(d-r)!} \mathbf{R}^d_1(\xi) \cdots \mathbf{R}^d_{d-r}(\xi) D\mathbf{R}^d_{d-r+1} \cdots D\mathbf{R}^d_d, \tag{5.18}$$

where the (first ordinary) derivative of the k-th B-spline matrix is given by

$$D\mathbf{R}^d_k = \begin{bmatrix} \frac{-1}{\xi_{j+1}-\xi_{j+1-k}} & \frac{1}{\xi_{j+1}-\xi_{j+1-k}} & 0 & \cdots & 0 \\ 0 & \frac{-1}{\xi_{j+2}-\xi_{j+2-k}} & \frac{1}{\xi_{j+2}-\xi_{j+2-k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{-1}{\xi_{j+k}-\xi_j} & \frac{1}{\xi_{j+k}-\xi_j} \end{bmatrix}. \tag{5.19}$$

As remarked by Lyche and Mørken in [60], the derivative operator D can be applied to any of the d matrices $\mathbf{R}^d_k$ as long as we differentiate r of them.

From a computational point of view, it makes sense to choose the trailing r matrices as this reduces the number of arithmetic operations to be performed. Recall that the k-th B-spline matrix has dimension $k \times (k+1)$ and features $2k$ non-zero entries. Each entry of $\mathbf{R}^d_k$ involves two subtractions and one division, whereas this reduces to one subtraction and one division for $D\mathbf{R}^d_k$. Accumulated over all d matrices, this yields an overall count of arithmetic operations for assembling the non-zero matrix entries of $3d^2 - 2dr + 3d + r^2 - r$ when the derivatives are applied to the trailing r matrices and $3d^2 + 3d - r^2 - r$ when the leading r matrices are differentiated. A similar cost analysis can be performed for the order in which the chain of matrix–matrix multiplications is evaluated. Assuming that the matrices are stored in dense format, the product between an $n \times o$ and an $o \times m$ matrix gives rise to $nm(2o-1)$ arithmetic operations (o multiplications and $o-1$ additions per entry of the resulting $n \times m$ matrix). A quick calculation shows that for $d \geqslant 2$ a left-to-right evaluation requires $(4d^3 + 9d^2 - d - 12)/6$ arithmetic operations, whereas a right-to-left evaluation requires $(4d^4 + d^3 - 4d^2 - 1)/6$ operations. We therefore recommend to evaluate the matrix chains in (5.13) and (5.18) from left to right starting at $\mathbf{R}^d_1$ and applying the derivatives (if any) to the trailing matrices.

Putting it all together, a handy approach to evaluate an element f of the spline space $\mathbb{S}^d$ (or its r-th derivative $D^r f$) in a given point ξ^* is as follows:

Find the knot span $[\xi_j, \xi_{j+1})$ such that $\xi_j \leqslant \xi^* < \xi_{j+1}$ and compute

$$f(\xi^*) = \quad \mathbf{b}^d_{[j-d;\,j]}(\xi^*) \cdot \mathbf{c}_{[j-d;\,j]} \quad \text{or} \tag{5.20}$$

$$D^r f(\xi^*) = D^r \mathbf{b}^d_{[j-d;\,j]}(\xi^*) \cdot \mathbf{c}_{[j-d;\,j]} \tag{5.21}$$

using the matrix representations (5.13) and (5.18), respectively.

Efficient Evaluation of B-Splines

For the reader who is interested in implementing the above in their own code, we would like to remark that (5.20) and (5.21) is particularly suited for programming languages like Python or Matlab and linear algebra libraries that support tensor operations. In that case, **b**'s and **c**'s can be generalized to tensors, whereby the third dimension represents the different ξ^* values and respective sub-vectors of coefficients. Then the spline or its derivative can be evaluated in all given points simultaneously by simple tensor contraction along the first two dimensions.

Instead of assembling the matrices (5.17) and (5.19) (or their generalization to tensors) and performing the multiplications explicitly, which might be time consuming and require a lot of computer memory especially for many evaluation points at a time, we present a modified version of Algorithm 2.22 from chapter 2 of [60] in Algorithm 2. The main difference is the automated handling of repeated knots (i.e., the smart circumvention of the "$0/0 := 0$" check) and the uniform treatment of functions and their derivatives, respectively.

Algorithm 2 B-spline evaluation

Require: Find positive integer $j \leqslant n + d + 1$ such that $\xi^* \in [\xi_j, \xi_{j+1})$
1: $\mathbf{b} = 1$
2: **for** $k = 1, \ldots, d - r$ **do**
3: $\quad \mathbf{t}_1 = \begin{bmatrix} \xi_{i-k+1} & \ldots & \xi_i \end{bmatrix}$
4: $\quad \mathbf{t}_{21} = \begin{bmatrix} \xi_{i+1} & \ldots & \xi_{i+k} \end{bmatrix} - \mathbf{t}_1$
5: $\quad \mathbf{mask} = (\mathbf{t}_{21} < \text{tol})$ ▷ $<$ element-wise comparison
6: $\quad \mathbf{w} = (\xi^* - \mathbf{t}_1 - \mathbf{mask}) \div (\mathbf{t}_{21} - \mathbf{mask})$ ▷ $\div$ element-wise division
7: $\quad \mathbf{b} = \begin{bmatrix} (1 - \mathbf{w}) \odot \mathbf{b} & 0 \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{w} \odot \mathbf{b} \end{bmatrix}$ ▷ $\odot$ element-wise multiplication
8: **end for**
9: **for** $k = d - r + 1, \ldots, d$ **do**
10: $\quad \mathbf{t}_1 = \begin{bmatrix} \xi_{i-k+1} & \ldots & \xi_i \end{bmatrix}$
11: $\quad \mathbf{t}_{21} = \begin{bmatrix} \xi_{i+1} & \ldots & \xi_{i+k} \end{bmatrix} - \mathbf{t}_1$
12: $\quad \mathbf{mask} = (\mathbf{t}_{21} < \text{tol})$
13: $\quad \mathbf{w} = (1 - \mathbf{mask}) \div (\mathbf{t}_{21} - \mathbf{mask})$
14: $\quad \mathbf{b} = \begin{bmatrix} -\mathbf{w} \odot \mathbf{b} & 0 \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{w} \odot \mathbf{b} \end{bmatrix}$
15: **end for**

After the execution of the algorithm, vector **b** contains the values of the r-th derivatives of the $d + 1$ degree d B-splines that are non-zero at the point ξ^*. Like with (5.20) and (5.21), Algorithm 2 can be generalized to handle multiple points ξ^*

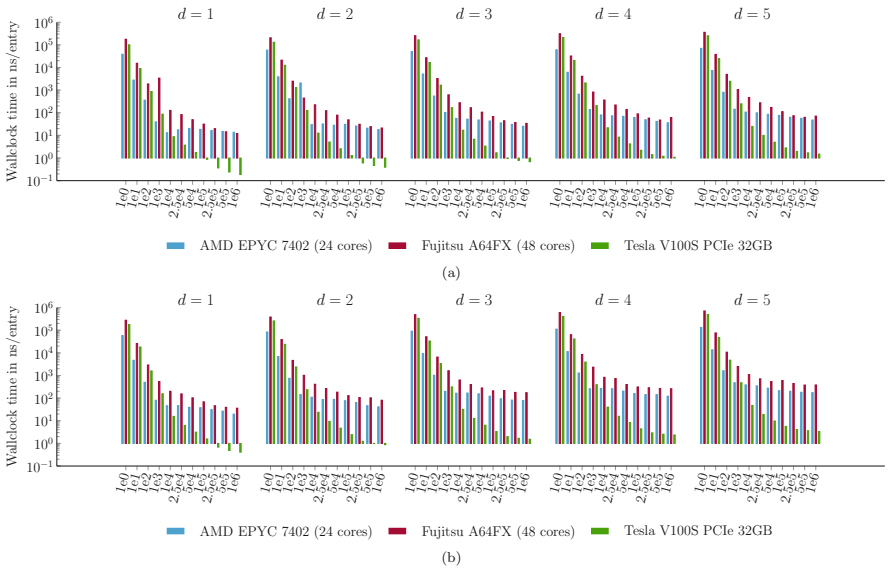


Fig. 5.6 Computational performance of vectorized B-spline evaluation as given in Algorithm 2 implemented in LibTorch 2.0 and executed on different CPU and GPU architectures: (**a**) univariate B-splines and (**b**) bi-variate B-splines

simultaneously by extending all vector quantities to matrices and adjusting lines 7 and 14 to not only append/prepend a scalar "0" to the vector but an entire column of zeros to the matrix. We have implemented this approach in an in-house research code based on the C++ API of LibTorch 2.0 and measured its computational performance for different processors (CPUs) and graphics cards (GPUs) for problem sizes between a single and a million evaluation points. A summary of the results is presented in Fig. 5.6 for B-splines of degree 1 to 5 both for the univariate case (described here) and for the case of bi-variate functions (to be addressed below). By exploiting vectorization and parallelization, the compute time for B-spline evaluation can be reduced by multiple orders of magnitude, especially when using GPUs.

It is easy to see that all vectors in lines 3–6 and 10–14 of Algorithm 2 are of length k and that the $\mathbf{b}$'s in lines 7 and 14 are after the appending/prepending with zeros of length $k+1$. Let us neglect this detail and estimate the cost of the loop body by $10k$ arithmetic operations (counting the element-wise comparison as arithmetic operation). Then the overall cost of Algorithm 2 is $5(d^2+d)$ operations which is significantly cheaper than assembling the matrices $\mathbf{R}_k^d$ and performing the matrix–matrix multiplications in (5.13) and (5.18), respectively.

Application of the vector of evaluated B-spline basis functions and their derivatives to the coefficient vector $\mathbf{c}$ in (5.20) and (5.21), respectively, gives rise to the costs of a standard inner product, namely, d multiplications and $d-1$ additions. Even though one might be tempted to evaluate (5.20) and (5.21) from right to left by

applying the trailing $\mathbf{R}_d^d$ matrix to the coefficient vector to obtain a smaller column vector and repeat the process until $\mathbf{R}_1^d$ is reached, we advise against this approach, since in many applications the B-spline basis functions can be pre-evaluated, e.g., in the Greville abscissae (5.11), and reused throughout the simulation, whereas the control points may change, e.g., in an analysis and optimization process.

Knot Insertion

The reader who is familiar with the finite element method might wonder if it is possible to "refine" the B-spline function space defined in (5.8). As $\mathbb{S}_\Xi^d$ is solely determined by the degree d and the knot vector $\Xi = (\xi_i)_{i=1}^{n+d+1}$, any "finer" knot vector $\Pi = (\pi_j)_{j=1}^{m+d+1}$ (with $m > n$) will generate a richer function space as $\mathbb{S}_\Pi^d$.

Let us consider the special case that $\Xi \subseteq \Pi$, that is, all knots from the original sequence are contained in the "refined" one. Moreover, let both knot vectors have common knots at the two ends, and let no knot occur with multiplicity higher than $d+1$ in which case we call the knot vectors $d+1$-regular. Then a spline f in $\mathbb{S}_\Xi^d$ with B-spline coefficients $\mathbf{c} = (c_i)_{i=1}^n$ can be represented exactly (!) in $\mathbb{S}_\Pi^d$ by calculating the B-Spline coefficients $\mathbf{d} = (d_j)_{j=1}^m$ relative to the knot vector Π by the second Oslo algorithm (see Algorithm 4.11 from chapter 4 in [60]):

Algorithm 3 Knot insertion (Oslo algorithm 2; cf. [60])

1: **for** $j = 1, \ldots, m$ **do**
2: Find positive integer i such that $\xi_i \leqslant \pi_j < \xi_{i+1}$
3: Compute the B-spline coefficient b_j as

$$b_j = \begin{cases} c_i & \text{if } d = 0, \\ \mathbf{R}_1^d(\pi_{j+1}) \cdots \mathbf{R}_d^d(\pi_{j+d}) \cdot \mathbf{c}_{[i-d;i]} & \text{if } d > 0. \end{cases}$$

4: **end for**

The chain of matrix–matrix products can again be computed efficiently by resorting to Algorithm 2. This approach of refining the spline space $\mathbb{S}_\Xi^d$ through augmenting its underlying knot vector Ξ is termed *knot insertion* and common practice. In particular the fact that spline functions f in $\mathbb{S}_\Xi^d$ have an equivalent and easy-to-compute representation in $\mathbb{S}_\Pi^d$ (with $\Xi \subseteq \Pi$) makes it an amenable tool for design-through-analysis workflows as it allows to first generate a spline space for representing the geometry model with sufficient level of detail and then refine this space for the analysis while preserving the capability to represent the original geometry model exactly within the refined spline space.

Next to the Oslo algorithm, there exist alternative approaches like blossoming to determine the B-spline coefficients after the insertion of knots into Ξ as discussed for instance in Section 4.4 of [60]. However, blossoming is mathematically more

advanced and requires a different implementation, whereas Algorithm 3 reuses the concept of B-spline matrices (5.17). It thereby suggests itself for programming languages like Python and Matlab and linear algebra libraries that support tensor operations since, like before, all **b** coefficients can be computed simultaneously.

Multi-variate B-Splines

A canonical approach to generalize B-splines into multiple parametric dimensions is by taking the tensor product of univariate ones. Let $\boldsymbol{\Xi} = \left(\Xi_1, \ldots, \Xi_p\right)$ denote the set of p independent knot vectors, $\mathbf{d} = \left(d_1, \ldots, d_p\right)$ the individual degrees for each direction, $\mathbf{n} = \left(n_1, \ldots, n_p\right)$ the number of univariate B-spline basis functions per direction, and $p \in \mathbb{N}_{\geqslant 1}$ the number of parametric dimensions.

The degree **d** tensor-product B-spline basis functions are then defined as

$$B^{\mathbf{d}}_{\mathbf{i};\boldsymbol{\Xi}}(\boldsymbol{\xi}) = \prod_{k=1}^{p} b^{d_k}_{i_k;\Xi_k}(\xi_k), \tag{5.22}$$

where $\mathbf{i} = \left(i_1, \ldots, i_p\right)$ is a multi-index from the admissible range $\mathbf{1} \leqslant \mathbf{i} \leqslant \mathbf{n}$ and $\boldsymbol{\xi} = \left(\xi_1, \ldots, \xi_p\right)$ denotes the multi-component independent variable.

In a practical implementation, the multi-index **i** is typically "flattened" to a global index i that varies between 1 and $n = n_1 n_2 \cdots n_p$ by imposing an ordering on the different dimensions (e.g., "smaller dimensions run faster") and computing

$$i := (i_3 - 1)n_1 n_2 + (i_2 - 1)n_1 + i_1. \tag{5.23}$$

Figure 5.7 illustrates the degree 2×3 B-splines basis functions defined over the knot vectors $\Xi_1 = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$ and $\Xi_2 = \{0, 0, 0, 0, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4\}$ in ξ_1- and ξ_2-direction, respectively. The interpolation property carries over to those basis functions for which both univariate components evaluate to one.

With definition (5.22) at hand, the corresponding spline space $\mathbb{S}^{\mathbf{d}}_{\boldsymbol{\Xi}}$ is defined analogously to its univariate counterparts (5.8) as follows:

$$\mathbb{S}^{\mathbf{d}}_{\boldsymbol{\Xi}} = \operatorname{span}\left\{B^{\mathbf{d}}_{\mathbf{1};\boldsymbol{\Xi}}, \ldots, B^{\mathbf{d}}_{\mathbf{n};\boldsymbol{\Xi}}\right\} \tag{5.24}$$

$$= \left\{\sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} c_{\mathbf{i}} B^{\mathbf{d}}_{\mathbf{i};\boldsymbol{\Xi}} \;:\; c_{\mathbf{i}} \in \mathbb{R}, \text{ for } \mathbf{1} \leqslant \mathbf{i} \leqslant \mathbf{n}\right\}. \tag{5.25}$$

We will drop subscript $\boldsymbol{\Xi}$ whenever there is no ambiguity about the knot vector.

For the efficient evaluation of multi-variate B-spline basis functions (or their derivatives), Algorithm 2 can be applied to each univariate component independently from which the final function value can be computed as follows:

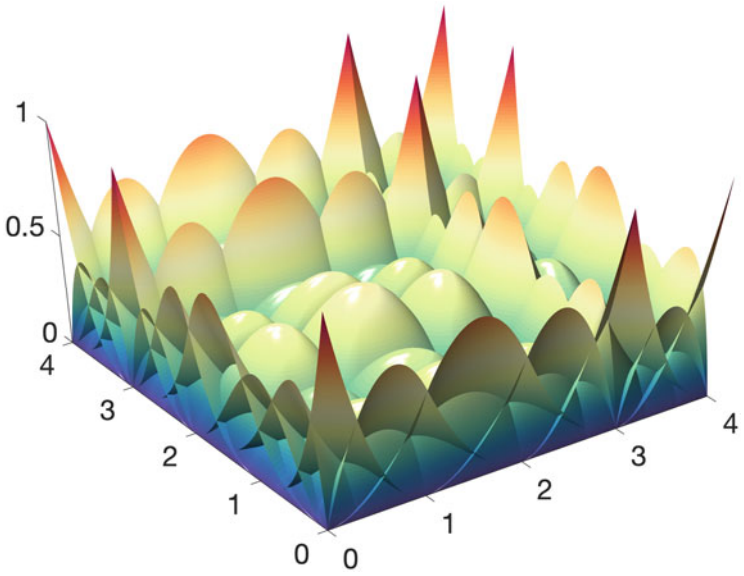


Fig. 5.7 Tensor-product degree 2×3 B-spline basis functions defined over the two knot vectors $\Xi_1 = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$ and $\Xi_2 = \{0, 0, 0, 0, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4\}$ in ξ_1- and ξ_2-direction, respectively

$$f(\boldsymbol{\xi}^*) = \left(\mathbf{b}^{d_1}_{1,[j_1-d_1;j_1]}(\xi_1^*) \otimes \mathbf{b}^{d_2}_{2,[j_2-d_2;j_2]}(\xi_2^*)\right) \cdot \mathbf{c}_{[\mathbf{j}-\mathbf{d};\mathbf{j}]}. \tag{5.26}$$

Here, $[\mathbf{j} - \mathbf{d}; \mathbf{j}]$ stands for the multi-dimensional generalization of the selection operator "$[\cdot;\cdot]$," that is, the operator that extracts a sub-matrix from the matrix of B-spline coefficients in the bi-variate case and a sub-tensor in the tri-variate case.

In a practical implementation, the values of $\mathbf{c}_{[\mathbf{j}-\mathbf{d};\mathbf{j}]}$ are typically not stored at contiguous memory positions unless the selection operator makes a deep copy. Often, the data is contiguous in the leading dimension, say, ξ_1 with offsets of n_1 along the second direction, offsets of $n_1 n_2$ along the third direction, etc. We therefore suggest to use Algorithm 993 [20] for the efficient computation of matrix–matrix products with matrices composed of Kronecker products. Let us drop sub- and superscripts for the moment. Then the tri-variate counterpart of (5.26) reads

$$(\mathbf{b}_1 \otimes \mathbf{b}_2 \otimes \mathbf{b}_3) \cdot \mathbf{c} = (\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{b}_3) \cdot (\mathbf{I} \otimes \mathbf{b}_2 \otimes \mathbf{I}) \cdot (\mathbf{b}_1 \otimes \mathbf{I} \otimes \mathbf{I}) \cdot \mathbf{c}. \tag{5.27}$$

This expression can be evaluated in three steps over the number of univariate directions as given in Algorithm 4. The reshape operation in line 2 assumes that matrices are stored in column-major format, that is, after the transposition in line 3 $\mathbf{c}$ is a matrix with n_d rows. After the successful execution of the algorithm, $\mathbf{c}$ contains the scalar value of f or its derivative in the point $\boldsymbol{\xi}^*$.

Algorithm 4 Efficient evaluation of tensor-product B-splines (Algorithm 993 [20])

Require: Univariate B-spline basis functions $\left[\mathbf{b}_1 \ \cdots \ \mathbf{b}_p\right]$ evaluated in $\boldsymbol{\xi}^*$ and selection of B-spline coefficients $\mathbf{c}$ both restricted to $[\mathbf{j}-\mathbf{d};\mathbf{j}]$

1: **for** $k = 1, \ldots, p$ **do**
2: $\quad \mathbf{c} = \text{reshape}(\mathbf{c}, [\cdot], n_k)$
3: $\quad \mathbf{c} = \mathbf{b}_k \cdot \mathbf{c}^\top$
4: **end for**

The arithmetic costs of this algorithm for the bi-variate case are $(2n_1+1)n_2-1$ arithmetic operations and $((2n_1+1)n_2)+1)n_3-1$ for the tri-variate case, respectively, assuming that reshaping and transposition are not performed explicitly but via clever memory accessing. As before, Algorithm 4 can be easily extended to a tensorized version that can evaluate f in multiple points simultaneously.

Geometry Modeling with B-Splines

In much the same way as we defined the spline function spaces (5.8) and (5.25), one can define higher dimensional spaces, i.e.,

$$\mathbb{S}^{\mathbf{d},s}_{\boldsymbol{\Xi}} = \left\{ \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbf{c}_{\mathbf{i}} B^{\mathbf{d}}_{\mathbf{i};\boldsymbol{\Xi}} \ : \ \mathbf{c}_{\mathbf{i}} \in \mathbb{R}^s, \ \text{for} \, \mathbf{1} \leqslant \mathbf{i} \leqslant \mathbf{n} \right\}, \tag{5.28}$$

with the main difference being that the B-spline coefficients are vector-valued data. Then, an element $\mathbf{f} \in \mathbb{S}^{\mathbf{d},s}_{\boldsymbol{\Xi}}$ realizes a mapping from the *parameter* space

$$\hat{\Omega}^p = \bigotimes_{k=1}^{p} \left[\xi_{k,d_k+1}, \xi_{k,n_k}\right] \subset \mathbb{R}^p, \tag{5.29}$$

a p-dimensional hypercube, to the s-dimensional *geometric* or *physical* space $\Omega^s \subset \mathbb{R}^s$. Canonically, $\mathbf{f} : \hat{\Omega}^1 \to \Omega^s$, $1 \leqslant s \leqslant 3$, defines a spline curve, $\mathbf{f} : \hat{\Omega}^2 \to \Omega^s$, $2 \leqslant s \leqslant 3$, a spline surface, and $\mathbf{f} : \hat{\Omega}^3 \to \Omega^3$ a spline volume. Higher order extensions for, e.g., space-time formulations or, in general, the parametric representation of s-dimensional data are also straightforward. In what follows, we will drop the superscripts p and s if the dimensions are clear from the context.

5.2.2 Truncated Hierarchical B-Splines

Since B-spline bases are typically constructed as a tensor product between the bases in each direction, tensor-product refinement of one element implies the refinement of multiple elements in one direction, see Fig. 5.9a. This implies a

quasi-local approach for adaptive refinement, which improves the speed of multi-scale simulation problems only partially. In order to provide local refinement of the spline basis, several spline basis constructions have been proposed in the literature such as hierarchical B-splines (HB-splines) [25], truncated hierarchical B-splines (THB-splines) [29, 30], T-splines [72] and their variations [49, 51, 50, 43, 52, 87], polynomial splines over hierarchical T-meshes (PHT-splines) [16], locally refined (LR) splines [17], and splines over unstructured meshes (U-splines) [78], the latter not to be confused with the unstructured splines to be presented in Sect. 5.2.4. Here, we review the (T)HB splines, and we refer the reader to the reference works for the other spline constructions.

The construction of the truncated (hierarchical) B-spline basis ($\mathcal{H}$) $\mathcal{T}$ is defined recursively as given in [29]:

1. Initialize $\mathcal{T}^0 = \mathcal{H}^0 = \{\varphi \in \mathcal{B}^0 : \text{supp}\, \varphi \neq \emptyset\}$, with the superscript denoting level 0, $\mathcal{B}^0$ a tensor B-spline basis on level 0, and φ a basis function with non-empty support.
2. Recursively define $\mathcal{T}^{\ell+1} = \mathcal{T}_A^{\ell+1} \cup \mathcal{T}_B^{\ell+1}$ or $\mathcal{H}^{\ell+1} = \mathcal{H}_A^{\ell+1} \cup \mathcal{H}_B^{\ell+1}$ for $\ell = 0, ..., N-2$ with N the maximum level. The truncated basis $\mathcal{T}_A^{\ell+1}$ is defined as

$$\mathcal{T}_A^{\ell+1} = \{\text{trunc}^{\ell+1}\, \tau : \tau \in \mathcal{T}^\ell \wedge \text{supp}\, \tau \not\subseteq \Omega^{\ell+1}\}$$

   and the hierarchical basis $\mathcal{H}_A^{\ell+1}$

$$\mathcal{H}_A^{\ell+1} = \{\varphi \in \mathcal{H}^\ell : \text{supp}\, \varphi \not\subseteq \Omega^{\ell+1}\}.$$

   Furthermore, the basis $\mathcal{T}_B^{\ell+1} = \mathcal{H}_B^{\ell+1}$ is given by

$$\mathcal{H}_B^{\ell+1} = \{\varphi \in \mathcal{B}^{\ell+1} : \text{supp}\, \varphi \subseteq \Omega^{\ell+1}\},$$

   with $\Omega^{\ell+1} \subseteq \Omega^\ell$ nested domains, $\mathcal{B}^\ell$ the B-spline basis on level ℓ, and $\text{trunc}^\ell\, \tau$ the truncation of τ with respect to $\mathcal{B}^{\ell+1}$ and $\Omega^{\ell+1}$.
3. Then the final THB-spline basis is defined as $\mathcal{T} = \mathcal{T}^{N-1}$ and the final HB-spline basis is defined as $\mathcal{H} = \mathcal{H}^{N-1}$.

Figure 5.8 illustrates the principle of local refinement with B-splines using (truncated) hierarchical B-splines (HB- and THB-splines, respectively). In the top row of this figure, an initial uniform degree 2 B-spline basis with uniform knot vector $\Xi = \{0, 1/8, 2/8, ..., 7/8, 1\}$ is presented. In the bottom row, a uniform refinement and (T)HB refinements of the indicated functions are presented.

The potential of refinement splines compared to knot insertions for local is illustrated in Fig. 5.9. When a knot insertion is performed in a tensor B-spline basis to refine a marked element, the refinement automatically introduces refinement of other elements in the knot line (see Fig. 5.9a). For hierarchical splines, the basis functions are inserted only locally, resulting in the addition of degrees of freedom only in the marked element, see Fig. 5.9b.

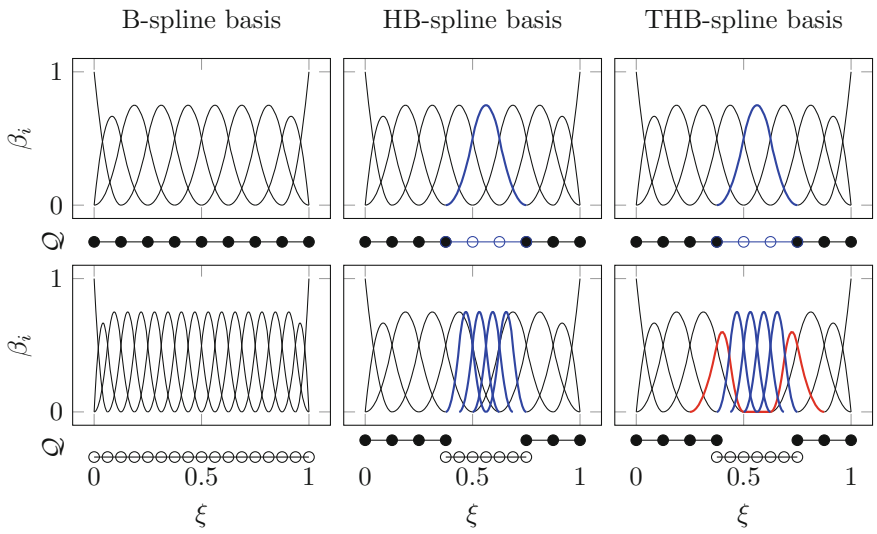
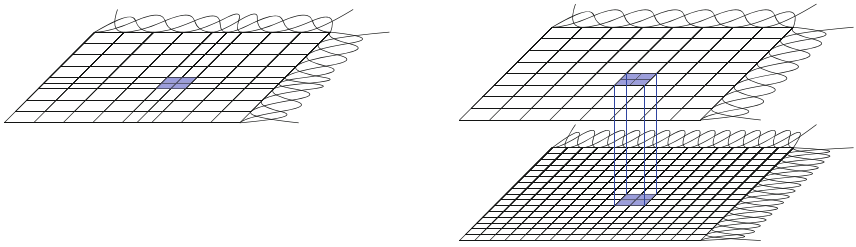


Fig. 5.8 Principles of refinement for different spline bases. The top plots represent the basis on level 0, optionally with refined basis functions given in blue color. The bottom plots illustrate the refined bases: uniform refinement (left), hence level 1; HB-refinement (middle); THB-refinement (right) with truncated basis functions in red color. The line $\mathcal{Q}$ represents the elements of the basis. The unrefined unique knot vector in all cases is $\Xi = \{0, 1/8, 2/8, \ldots, 7/8, 1\}$ and the degree of the basis is 2. All bases are generated with the open-source IGA library G+Smo [42]



(a) Refinement of a tensor-product B-spline basis. To refine the element, the knot 0.5625 is inserted in both knot vectors.

(b) Refinement of a (T)HB-spline basis. To refine the element, the basis functions with support on this element from the finer level (below) are inserted in the original basis (above) following the procedure from Fig. 5.8.

Fig. 5.9 Refinement of a two-dimensional tensor B-spline basis (**a**) and a (T)HB-spline basis (**b**) for a marked element with corners $(0.5, 0.5)$ and $(0.625, 0.625)$. The original B-spline basis has degree 2 and unique knot vector $\Xi = \{0, 1/8, 2/8, \ldots, 7/8, 1\}$ in both directions

5.2.3 Non-uniform Rational B-Splines

While B-spline basis functions offer great flexibility to model geometric freeform shapes like curves, surfaces, and volumes, they fall short in accurately representing

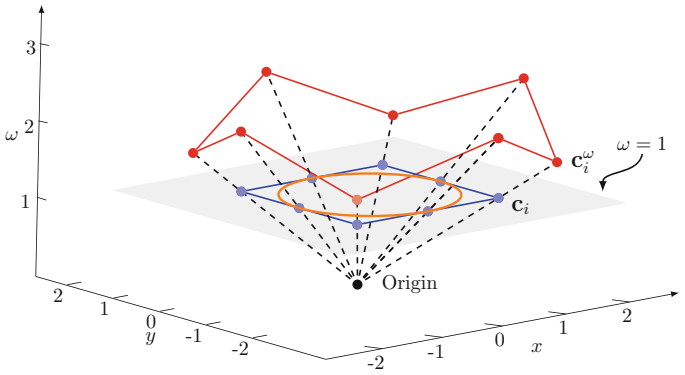


Fig. 5.10 Illustration of a univariate NURBS curve

conic curve surfaces such as ellipses and hyperbolas, which are widely encountered in industrial applications. To overcome this limitation, a natural progression is to generalize B-splines to Non-Uniform Rational B-Splines (NURBSs), which have become the industry standard in modern CAD systems. Essentially, NURBSs extend the concept of B-spline basis functions by incorporating rational functions, achieved through the introduction of weights $(\omega_i)_{i=1}^n$ as an additional mechanism to modulate the shape of the geometric object.

In essence, each control point $\mathbf{c}_i \in \mathbb{R}^s$ is assigned a weight factor $\omega_i \in \mathbb{R}^+$ with which we define the *projected control point* as follows:

$$\mathbf{c}_i^\omega = \begin{pmatrix} \omega_i \mathbf{c}_i \\ \omega_i \end{pmatrix} \in \mathbb{R}^{s+1}.$$

A NURBS geometry in $\mathbb{R}^s$ space is obtained by projecting a B-spline geometry in $\mathbb{R}^{s+1}$ onto the hyperplane of $\omega = 1$ through a central projection transformation, as illustrated in Fig. 5.10. Notably, this transformation enables the accurate representation of conic curves using piecewise quadratic polynomial curves, making quadratic NURBS curves an appropriate choice for precise representation.

In practice, NURBS curves (and other NURBS-based geometric objects) are not constructed via central projection transformation but by replacing the univariate B-spline basis functions (5.3) by their rational counterparts

$$N_{i;\Xi,\omega}^d(\xi) = \frac{\omega_i N_{i;\Xi,\omega}^d(\xi)}{\sum_{i=1}^n \omega_i N_{i;\Xi,\omega}^d(\xi)} \tag{5.30}$$

and defining the corresponding function space as follows:

$$\mathbb{S}^{d,s}_{\Xi,\omega} = \text{span}\left\{N^d_{1;\Xi,\omega}, \ldots, N^d_{n;\Xi,\omega}\right\} \tag{5.31}$$

$$= \left\{\sum_{i=1}^{n} \mathbf{c}_i N^d_{i;\Xi,\omega} \,:\, \mathbf{c}_i \in \mathbb{R}^s, \text{ for } 1 \leqslant i \leqslant n\right\}. \tag{5.32}$$

The generalization to multi-variate NURBS spaces follows the same tensor-product construction principle as was adopted for B-splines with a single weight per basis function. Let us conclude this section by remarking that NURBSs, although being the predominant industry standard, are often not necessary in practical design-through-analysis applications and that other features like local refinability of (T)HB-splines outweigh the shortcoming of B-spline basis functions to not being able to represent conic curves and their higher dimensional extensions exactly.

As before, we will drop sub- and superscripts whenever their information can be deduced from the context. Moreover, we will adopt N_i as generic notation for B-spline and NURBS basis functions alike unless stated otherwise.

5.2.4 Multi-patch Splines

Many geometries of practical interest like the one depicted in Fig. 5.11 cannot be represented by a single-patch B-spline or NURBS mapping $\mathbf{f} : \hat{\Omega} \to \Omega$ unless trimming is used excessively. It is therefore common practice to combine multiple such mappings to a multi-patch spline, i.e.,

$$\mathbf{f}_\ell : \hat{\Omega} \to \Omega_\ell, \quad \Omega = \bigcup_\ell \bar{\Omega}_\ell. \tag{5.33}$$

For the reader who is familiar with finite elements, each patch Ω_ℓ can be considered as a kind of macro element with $\mathbf{f}_\ell$ being the *push-forward* operator from the reference element $\hat{\Omega}$ into the physical space. Likewise, if the mapping $\mathbf{f}_\ell$ is bijective, its inverse $\mathbf{f}_\ell^{-1} : \Omega_\ell \to \hat{\Omega}$ exists and is termed the *pull-back* operator.

While most conformal finite element formulations do not go beyond C^0 continuity over element interfaces (i.e., continuity of the values of the basis functions but not of their derivatives), it would be beneficial to preserve at least part of the $C^{\mathbf{d}-1}$ continuity of higher order B-splines when coupling multiple patches to a multi-patch object. To achieve this, so-called *unstructured splines*, i.e., splines with higher order smoothness over patch interfaces, can be constructed.

In case of one-dimensional bases, the concept of patch smoothing is trivial, but illustrative for higher dimensions. The concept of interface smoothing is illustrated in Fig. 5.12 and can be interpreted as a construction where basis functions $\varphi \in \mathbb{S}^{d,1}_{\Xi}$ of a spline space $\mathbb{S}^{d,1}_{\Xi}$ with interface smoothness 1 are represented by a linear

Fig. 5.11 Illustration of a planar multi-patch geometry

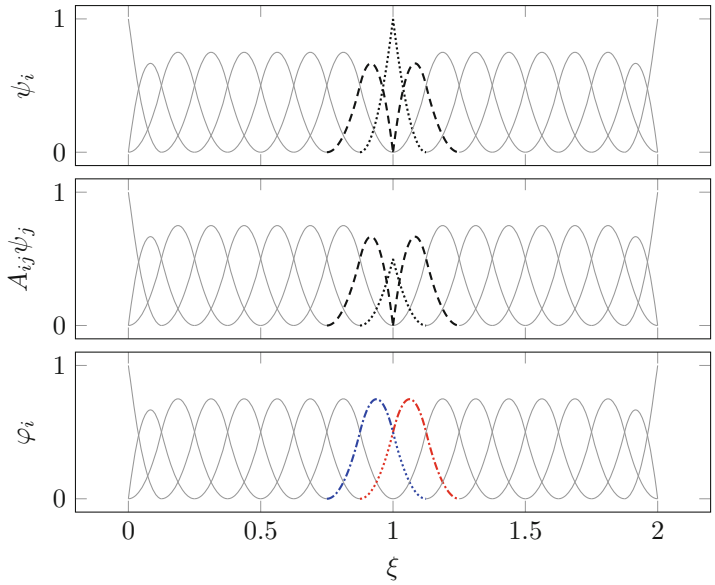


Fig. 5.12 The concept of interface smoothing between two bases of degree 2 with unique knot vector $\Xi = \{0, 1/8, 2/8, \ldots, 7/8, 1\}$ (left) and $\Xi = \{1, 9/8, 10/8, \ldots, 15/8, 2\}$ (right). The top figure provides the two bases with the dotted basis functions ψ_i, the functions that have non-zero derivatives and values on the interface, and the dashed basis functions represent the functions that have zero values but non-zero derivatives on the boundary. The middle row presents scaled basis functions $A_{ij}\psi_j$. Here, all functions are scaled by a factor of 1, except for the dotted functions, which are scaled by a factor of $1/2$. The bottom row presents the basis $\varphi_i = A_{ij}\psi_j$ where the sum is evaluated over the repeated index j. The blue and red functions are constructed by taking the sum of the dashed and dotted functions in their support on the one side (resulting in the dash-dotted line) and taking the dotted line on the other side

combination of functions $\psi \in \mathbb{S}^{d,0}$ from a space $\mathbb{S}^{d,0}$. For example, basis function φ_i is represented by all basis functions ψ_j weighted with coefficient A_{ij}:

$$\varphi_i = A_{ij}\psi_j, \tag{5.34}$$

By applying this procedure to all basis functions φ_i, we arrive at the relation

$$\boldsymbol{\varphi} = A\boldsymbol{\psi} \tag{5.35}$$

with A denoting the transformation matrix.

In higher dimensions, interface smoothing as illustrated in Fig. 5.12 can be performed to construct interface basis functions. However, the increased parametric dimension (see Fig. 5.13) introduces vertices where the smoothing of basis functions is non-trivial. Spline constructions that provide smoothing mappings like the matrix A are referred to as *unstructured splines*, providing bases with higher

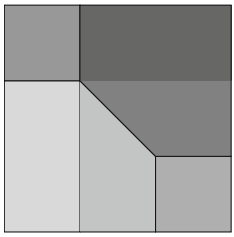


Fig. 5.13 Multi-patch decomposition of a simple domain into six patches. All the vertices on the boundary are regular since there are 1 or 2 patches joining. The vertices in the interior are irregular since their *valence* (i.e., the number of patches joining in the vertex) is not equal to 4. These vertices are also referred to as *extraordinary vertices*

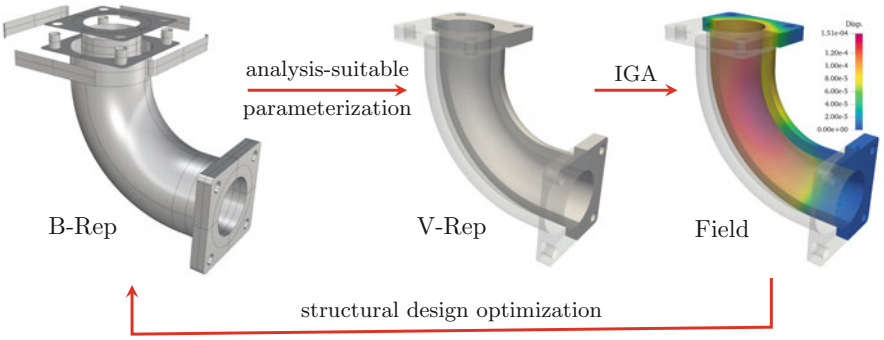


Fig. 5.14 Design-analysis-optimization pipeline

smoothness than C^0 over patch interfaces and vertices. Examples of unstructured spline constructions include the D-Patch [70, 80], the Almost-C^1 construction [77], the Approximate C^1 basis [89, 90], the Analysis-Suitable G^1 construction [12, 22], polar spline constructions [79], and constructions based on subdivision surfaces [54, 5].

5.3 Creation of Analysis-Suitable Parameterizations

In modern CAD systems, as shown in Fig. 5.14, Boundary Representations (B-Rep) are commonly used to represent CAD models. B-Rep describes the boundaries of an object through vertices, edges, and faces, organizing these elements using topological relationships. This representation provides high flexibility and efficiency in geometric modeling and design. However, before conducting simulation-based analysis, it is essential to construct a Volumetric Representation (V-Rep) for the interior of the CAD model. This process is typically known as domain parameterization or simply parameterization. Once an analysis-suitable parameterization is obtained,

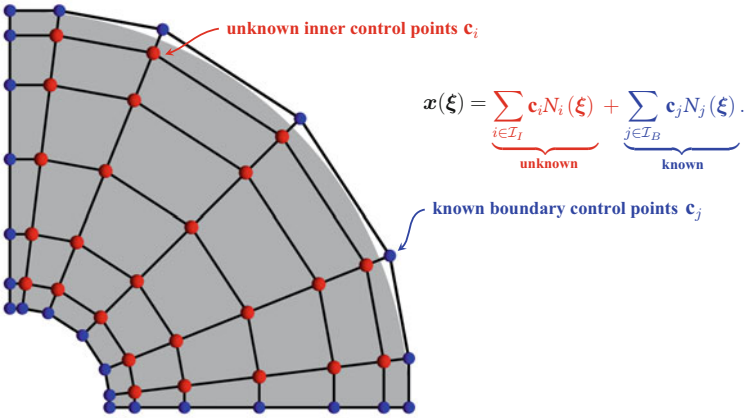


Fig. 5.15 Illustration of the problem statement for the creation of analysis-suitable parameterizations

simulation-based analysis can be performed on the volumetric representation model. For visual reference, please see Fig. 5.14.

5.3.1 Problem Statement

Unless noted otherwise, we will restrict ourselves to the single-patch case in what follows. For the sake of notation convenience, let $\mathcal{I}_I$ and $\mathcal{I}_B$ be the index sets for the unknown inner control points and the known boundary control points, respectively. Then, the parameterization $\boldsymbol{x}$ can be represented as follows:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \underbrace{\sum_{i\in\mathcal{I}_I} \mathbf{c}_i N_i(\boldsymbol{\xi})}_{\text{unknown}} + \underbrace{\sum_{j\in\mathcal{I}_B} \mathbf{c}_j N_j(\boldsymbol{\xi})}_{\text{known}}, \tag{5.36}$$

where $\mathbf{c}_i$, $i \in \mathcal{I}_I$ are unknown inner control points, $\mathbf{c}_j$, $j \in \mathcal{I}_B$ are the given boundary control points, $N_i(\boldsymbol{\xi})$ and $N_j(\boldsymbol{\xi})$ are the corresponding NURBS basis functions, and $\boldsymbol{\xi} \in \hat{\Omega}$. As depicted in Fig. 5.15, the known boundary control points $\mathbf{c}_j$ are represented by the blue points. Conversely, the unknown inner control points $\mathbf{c}_i$, are indicated by the red points. It should be noted that the black lines that connect the control points indicate the so-called *control net*, whereas the actual parameterized domain Ω is the gray quarter annulus underneath.

In fact, the quality of the parameterization significantly impacts the accuracy and efficiency of subsequent analysis tasks [11, 93, 67]. First and foremost, a high-quality analysis-suitable parameterization should be a bijection. Additionally, it should exhibit good orthogonality of "grid lines" and uniformity of "cell sizes"

or, in other words, minimal distortion in terms of angles and area/volume. Strictly speaking, the technical terms "grid lines" and "cells" are anachronisms from classical grid-based methods and should be replaced by "parametric curves with all but one parameter kept fixed" and "images of the push-forward operator for the tensor-product of knot spans in $\hat{\Omega}$," respectively. At the same time, use of this terminology might help the reader familiar with grid-based methods like finite elements to interpret grids as the lowest order C^0 parameterizations, where by virtue of the interpolation property the control net and the domain Ω fall together.

In light of the above requirements on the mapping $\boldsymbol{x} : \hat{\Omega} \rightarrow \Omega$, the parameterization problem can be formulated as follows: Given the set of boundary control points $\mathbf{c}_j$, the objective is to construct the unknown inner control points $\mathbf{c}_i$ in such a way that the resulting parameterization $\boldsymbol{x}$ guarantees bijectivity and minimizes angle, as well as area/volume distortion.

5.3.2 Classification of Parameterization Methods

For an analysis-suitable parameterization, the bijective property plays a crucial role. In this section, we categorize the existing methods based on the approaches they employ to handle the bijectivity constraints.

Algebraic Parameterization Methods

These methods rely on algebraic principles and involve, if at all, the solution of a linear system of equations, which makes them computationally efficient.

One notable example is the Discrete Coons method [24], which belongs to a special type of Transfinite Interpolation (TFI) methods. This explicit parameterization method does not require the solution of a linear system of equations and is generally considered highly efficient. Xu et al. extended this method to three-dimensional NURBS volumetric parameterization [93].

Several linear parameterization methods that solve a linear system, such as the spring model method and the mean value coordinates method, were discussed in [31]. Since these algebraic methods do not specifically consider the bijectivity constraint, they often yield self-intersecting parameterizations when dealing with complex domains. In these scenarios, algebraic methods are often used to compute an initial guess as a starting point for a more advanced parameterization approach. This two-step procedure, of course, requires that the downstream method is capable of turning a non-bijective parameterization into a bijective one.

Nonlinear Constrained Optimization Methods

These methods inherently treat the bijection constraint as constraint terms and employ energy functions that characterize the orthogonality and uniformity of the

parameterization as the objective function. An analysis-suitable parameterization is then generated by solving the constrained optimization problem.

Xu et al. [91] utilized the observation that the Jacobian determinant of the parameterization can be expressed using higher order NURBS functions. They employed the nonlinear coefficients of the Jacobian determinant as inequality constraints to ensure bijectivity in the parameterization results. They subsequently extended this method to volumetric parameterizations [93]. Wang et al. [85] proposed an accelerated constrained optimization framework by utilizing constraint aggregation, divide and conquer, and hierarchical optimization strategies. Xu et al. [94] introduced a computational reuse method for computation domains with consistent topological structures. Ugalde et al. [1] presented a series of sufficient and necessary conditions for achieving injectivity in quadratic B-spline parameterizations.

Despite the effective performance of these parameterization methods based on nonlinear constrained optimization in small-scale examples, the number of constraints significantly increases as the problem size grows, particularly in the context of volumetric parameterizations. To mitigate the computational burden, Pan et al. [65] introduced a constraint addition strategy that gradually incorporates collocation points in a coarse-to-fine resolution manner. Nevertheless, this method still incurs a substantial computational cost, necessitating the utilization of commercial optimization solvers.

Nonlinear Unconstrained Optimization Methods

In the aforementioned parameterization methods based on nonlinear constrained optimization, the significant quantity of nonlinear constraints represents a challenging problem to solve. Consequently, in recent years, unconstrained optimization-based parameterization methods have gained popularity.

Xu et al. [92] introduced a parameterization method that involves minimizing the variational harmonic mapping. Nguyen and Jüttler [61] initially computed a series of harmonic mappings from the computational domain to the parameter domain, subsequently employing spline approximation for the inverse mapping. Falini et al. [21] extended this method to planar THB-spline parameterization by first calculating the harmonic mapping from the computational domain to the parameter domain through the boundary element method. They then utilized spline least squares fitting for the inverse mapping. Nian et al. [62] presented a planar parameterization method that relies on Teichmüller mapping. Pan et al. [64] introduced a low-rank parameterization method that utilizes quasi-conformal mapping and employs an alternating direction multiplier method to minimize the objective functional. They subsequently extended this method to include volumetric parameterization [63].

The Radó–Kneser–Choquet theorem guarantees the injectivity of the solution to the Winslow functional, which resulted in its widespread usage in traditional mesh generation fields, where it is known as the Most Isometric ParameterizationS (MIPS) energy in computer graphics [37]. In the field of computer graphics, a three-

dimensional version of this energy is also recognized [26]. These energy functionals frequently necessitate an initial parameterization that possesses bijectivity. To tackle this issue, researchers have proposed various foldover elimination methods. Su et al. [76] projected the Jacobian matrix onto a space with bounded K-distortion, while Liu et al. [53] further enhanced this method by incorporating simultaneous optimization of boundary correspondence. Zheng et al. [96] recently employed this idea in THB-spline volumetric parameterization and introduced an efficient method for volumetric parameterization. Another approach involves the usage of penalty functions and Jacobian regularization techniques, which find their roots in the literature on grid distortion problems [27, 28]. Wang and Ma [84] implemented this idea in planar parameterization problems, successfully circumventing the need for extra foldover elimination steps.

Nonlinear Partial Differential Equation (PDE)-Based Methods

These methods either approximate the stationary points of the Dirichlet energy while satisfying known boundary conditions or solve the corresponding Euler–Lagrange equations.

Martin et al. [55] utilized discrete volumetric harmonic mappings to fit trivariate B-spline volumes. Shamanskiy et al. [75] developed analysis-oriented parameterizations through the solution of nonlinear elasticity equations using neo-Hookean hyperelastic material laws. Ali and Ma [2] utilized an isogeometric approach with equigeometric points to solve PDEs with boundary vector constraints in planar parameterization. Hinz et al. [35, 36, 34] proposed a series of parameterization construction methods based on nonlinear PDEs by discretizing the Laplace equation, drawing upon the principles of Elliptic Grid Generation (EGG). These elliptic parameterization methods based on EGG demonstrate favorable convergence properties and excel in slender domains with extreme aspect ratios.

5.3.3 Optimization-Based Parameterization Methods

This section introduces optimization-based parameterization methods. As mentioned in Sect. 5.3.2, constrained optimization methods are computationally inefficient due to the presence of numerous nonlinear constraints. Accordingly, we will present two unconstrained optimization methods: the barrier function-based method and the penalty function-based method. These methods are derived from the authors' two published papers: [40] and [41], respectively.

When considering energy functions that characterize angle distortion and area/volume distortion, the Jacobian matrix $\mathcal{J}$ plays a crucial role. The following fundamental quantities are frequently involved in this context.

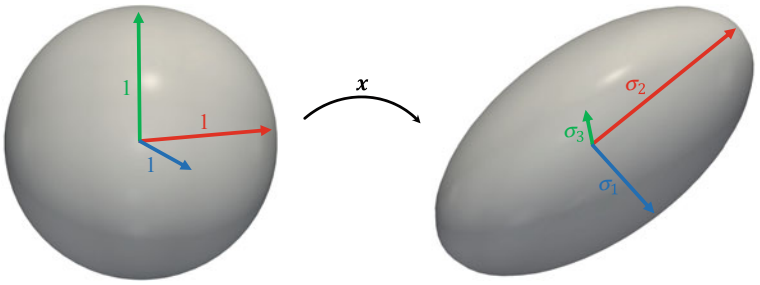


Fig. 5.16 The geometric interpretation of the singular values of the Jacobian matrix

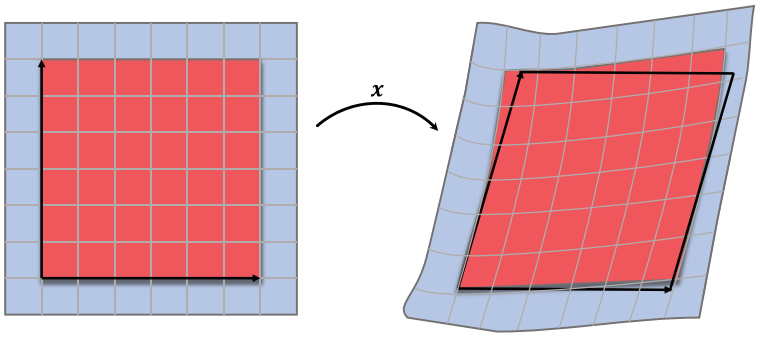


Fig. 5.17 The geometric interpretation of the Jacobian determinant

- The Jacobian matrix of the parameterization $\boldsymbol{x}$:

$$\text{For the 2D case:} \qquad \mathcal{J} = \begin{bmatrix} x_{1,\xi_1} & x_{1,\xi_2} \\ x_{2,\xi_1} & x_{2,\xi_2} \end{bmatrix}; \tag{5.37a}$$

$$\text{For the 3D case:} \qquad \mathcal{J} = \begin{bmatrix} x_{1,\xi_1} & x_{1,\xi_2} & x_{1,\xi_3} \\ x_{2,\xi_1} & x_{2,\xi_2} & x_{2,\xi_3} \\ x_{3,\xi_1} & x_{3,\xi_2} & x_{3,\xi_3} \end{bmatrix}. \tag{5.37b}$$

First, it is crucial to consider the singular value σ_i ($i = 1, 2, \ldots, s$, $1 \leq s \leq 3$) of the Jacobian matrix $\mathcal{J}$. Figure 5.16 illustrates how these singular values reflect the variations in the lengths of the principal axes when locally mapping the unit sphere to an ellipsoid. Ideally, we aim for uniform singular values across the Jacobian matrix to ensure minimal angle distortion.

Second, another significant quantity is the Jacobian determinant $|\mathcal{J}|$ of the parameterization $\boldsymbol{x}$. The Jacobian determinant at a specific point provides the optimal linear approximation of the distorted parallelogram in the vicinity of that point. As depicted in Fig. 5.17, the Jacobian determinant represents the ratio

between the area of the approximating parallelogram and that of the original square.

- The metric tensor $\mathcal{G} = \mathcal{J}^\top \mathcal{J}$:

$$\text{For the 2D case:} \qquad \mathcal{G} = \begin{bmatrix} \boldsymbol{x}_{,\xi_1} \cdot \boldsymbol{x}_{,\xi_1} & \boldsymbol{x}_{,\xi_1} \cdot \boldsymbol{x}_{,\xi_2} \\ \boldsymbol{x}_{,\xi_2} \cdot \boldsymbol{x}_{,\xi_1} & \boldsymbol{x}_{,\xi_2} \cdot \boldsymbol{x}_{,\xi_2} \end{bmatrix}; \tag{5.38a}$$

$$\text{For the 3D case:} \qquad \mathcal{G} = \begin{bmatrix} \boldsymbol{x}_{,\xi_1} \cdot \boldsymbol{x}_{,\xi_1} & \boldsymbol{x}_{,\xi_1} \cdot \boldsymbol{x}_{,\xi_2} & \boldsymbol{x}_{,\xi_1} \cdot \boldsymbol{x}_{,\xi_3} \\ \boldsymbol{x}_{,\xi_2} \cdot \boldsymbol{x}_{,\xi_1} & \boldsymbol{x}_{,\xi_2} \cdot \boldsymbol{x}_{,\xi_2} & \boldsymbol{x}_{,\xi_2} \cdot \boldsymbol{x}_{,\xi_3} \\ \boldsymbol{x}_{,\xi_3} \cdot \boldsymbol{x}_{,\xi_1} & \boldsymbol{x}_{,\xi_3} \cdot \boldsymbol{x}_{,\xi_2} & \boldsymbol{x}_{,\xi_3} \cdot \boldsymbol{x}_{,\xi_3} \end{bmatrix}. \tag{5.38b}$$

In particular, a parameterization $\boldsymbol{x}$ is locally conformal at a point $\boldsymbol{\xi}^*$ if and only if the metric tensor $\mathcal{G}$ satisfies $\mathcal{G} = n(\boldsymbol{\xi}^*) \cdot \mathbf{I}_{p\times p}$, where $\mathbf{I}_{p\times p}$ represents the identity matrix and $n(\boldsymbol{\xi}^*) > 0$. In this case, the Jacobian matrix $\mathcal{J}$ represents a combination of proportional scaling and rotation in each direction, leading to an orthogonal isoparametric structure for the parameterization $\boldsymbol{x}$. It is worth noting that under this condition, all singular values of the Jacobian matrix $\mathcal{J}$ are equal.

Based on the aforementioned fundamental quantities, the pointwise Most Isometric ParameterizationS (MIPS) energy [37, 26] can be employed to quantify angle distortion in the vicinity of a single point:

$$\mathscr{E}_p^{\text{angle}} = \begin{cases} \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}, & \text{2D case,} \\ \frac{1}{8}\left(\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}\right)\left(\frac{\sigma_2}{\sigma_3} + \frac{\sigma_3}{\sigma_2}\right)\left(\frac{\sigma_1}{\sigma_3} + \frac{\sigma_3}{\sigma_1}\right), & \text{3D case,} \end{cases} \tag{5.39}$$

where σ_i represent the singular values of $\mathcal{J}$. The minimum value of $\mathscr{E}^{\text{angle}}$ occurs when $\sigma_1 = \sigma_2 = \cdots = \sigma_p$, ensuring minimal angle distortion.

Furthermore, we utilize the following pointwise uniformity energy function $\mathscr{E}_p^{\text{unif.}}$ to assess the distortion in area/volume

$$\mathscr{E}_p^{\text{unif.}} = \frac{|\mathcal{J}|}{\text{vol}(\Omega)} + \frac{\text{vol}(\Omega)}{|\mathcal{J}|}, \tag{5.40}$$

where $\text{vol}(\Omega)$ denotes the area/volume of the computational domain Ω.

Recalling the angle distortion energy (5.39), in the 2D case, we have

$$\mathscr{E}_p^{\text{angle}} = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1\sigma_2} = \frac{\text{trace}(\mathcal{G})}{|\mathcal{J}|}. \tag{5.41}$$

This energy is also known as Winslow's functional, which may be more familiar to the mesh generation community. It is solely determined by the Jacobian determinant and the metric tensor $\mathcal{G}$ of the parameterization $\boldsymbol{x}$, making it an intrinsic geometric quantity. Given that the parameter domain $\hat{\Omega} = [0, 1]^2$ is convex, the

Radó–Kneser–Choquet theorem [69, 47] states that the unique minimum value of Winslow's functional establishes a differential homeomorphism between the interior of the parameter domain and the interior of the computational domain. Moreover, as the Jacobian determinant approaches zero, which is in the denominator, the energy $\mathscr{E}_p^{\text{angle}}$ that measures angle distortion tends to infinity. This property effectively prevents self-intersections from occurring.

Although there is no strict mathematical theory supporting the 3D case, a similar property holds, that is,

$$\begin{aligned}\mathscr{E}_p^{\text{angle}} &= \frac{1}{8}\left(\frac{\sigma_1}{\sigma_2}+\frac{\sigma_2}{\sigma_1}\right)\left(\frac{\sigma_2}{\sigma_3}+\frac{\sigma_3}{\sigma_2}\right)\left(\frac{\sigma_1}{\sigma_3}+\frac{\sigma_3}{\sigma_1}\right)\\ &= \frac{1}{8}\left(\frac{\left(\sigma_1^2+\sigma_2^2+\sigma_3^2\right)\left(\sigma_2^2\sigma_3^2+\sigma_1^2\sigma_3^2+\sigma_1^2\sigma_2^2\right)}{|\mathcal{J}|^2}-1\right).\end{aligned} \tag{5.42}$$

It can be observed that the Jacobian determinant appears in the denominator, which has the capability to prevent self-intersections.

Basically, this property appears to provide insight for constructing an analysis-suitable parameterization by minimizing the following energy function:

$$\mathscr{E} = \int_{\hat{\Omega}} \lambda^{\text{angle}} E_p^{\text{angle}} + \lambda^{\text{unif.}} E_p^{\text{unif.}} \mathrm{d}\hat{\Omega}, \tag{5.43}$$

where λ^{angle} and $\lambda^{\text{unif.}}$ are trade-off parameters used to balance the angle and area/volume distortion.

However, the situation is far more intricate than it initially appears. The presence of the Jacobian determinant in the denominator creates a barrier that can potentially prevent self-intersections. Conversely, starting from an infeasible initial parameterization, which is frequently encountered in complex domain parameterization problems, it introduces an arbitrary level of complexity. In essence, a bijective parameterization must be established prior to the minimization of the energy function (5.43). In the subsequent two subsections, we will investigate approaches to tackle this challenge.

Barrier Function-Based Method

In this section, we will introduce a three-step strategy known as the barrier function-based method, designed to generate high-quality parameterizations. We aim to overcome the challenges associated with numerous nonlinear constraints and achieve superior outcomes. The fundamental workflow of the barrier function-based method is illustrated in Fig. 5.18, providing a visual representation of the step-by-step process involved.

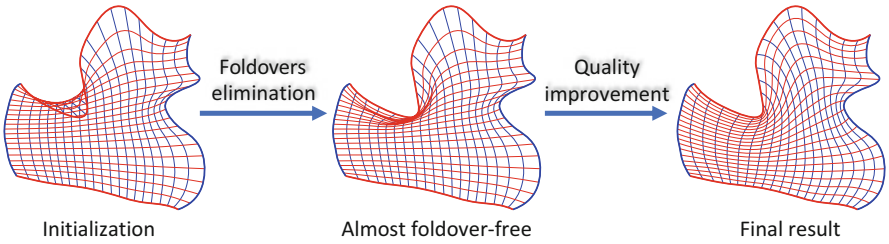


Fig. 5.18 The workflow of the barrier function-based method

Initialization

A common approach to solving nonlinear optimization problems is through iterative methods. Therefore, a reasonable initial guess is crucial to improve the convergence speed of subsequent solutions. As discussed in Sect. 5.3.2, algebraic parameterization methods such as the discrete Coons method [24], the spring model method [31], and the smoothness energy method [65] are commonly employed to generate an initial guess. In what follows, we adopt the smoothness energy method. Specifically, the unknown inner control points $\mathbf{c}_i,\ i \in \mathcal{I}_I$ are obtained by solving the following quadratic programming problem:

$$\underset{\mathbf{c}_i,\ i\in\mathcal{I}_I}{\arg\min} \int_{\hat{\Omega}} \|\Delta \boldsymbol{x}\|^2 \, d\hat{\Omega}. \tag{5.44}$$

It can be obtained by solving a sparse and symmetric linear system of equations. The preconditioned conjugate gradient method or the GMRES method with incomplete Cholesky decomposition is typically used for the solution. The initial parameterization constructed by this method is shown in the left part of Fig. 5.18. Note that this method does not guarantee a self-intersection-free parameterization, as it becomes evident from the presence of many self-intersections on the back of the duck.

Foldover Elimination

Generally, for complex computational domains, the initial parameterization constructed using algebraic parameterization methods does not guarantee bijectivity. Therefore, in our method, the second step is to eliminate foldovers.

To ensure a bijective parameterization, it is necessary for the Jacobian determinant to be greater than zero throughout the entire computational domain. To this end, we solve the following unconstrained optimization problem:

$$\underset{\mathbf{c}_i,\ i\in\mathcal{I}_I}{\arg\min}\ \mathscr{E}^{\text{fold}} = \int_{\hat{\Omega}} \max\{0,\ \delta - |\mathcal{J}|\}\, \mathrm{d}\hat{\Omega}, \tag{5.45}$$

where δ is a user-specified parameter value.

The objective function in problem (5.45) clearly attains a minimum value of zero. During practical computations, the objective function is commonly evaluated using Gaussian numerical integration. However, in this scenario, a zero value of the objective function only indicates that the integrand evaluates to zero at the Gaussian integration points, which does not guarantee a bijective parameterization. Therefore, we refer to parameterizations that satisfy the condition of the Jacobian determinant being greater than zero at all Gaussian integration points as almost foldover-free parameterizations. As depicted in the central portion of Fig. 5.18, solving the optimization problem (5.45) leads to a substantial reduction in self-intersections. Solving the problem (5.45) plays a crucial role in achieving success and improving the quality of the subsequent parameterization. However, it is important to note that addressing this problem alone is insufficient to completely eliminate self-intersections for the majority of complex computational domains. To this end, we further improve the parameterization quality in the next step.

In problem (5.45), the choice of parameter δ greatly influences the success of the problem-solving process. On the one hand, a larger value of δ is desired as it leads to a higher-quality parameterization, which in turn enhances the convergence efficiency in improving the subsequent parameterization. On the other hand, setting δ too large may result in the failure to solve the problem. To enhance the robustness of our method in this chapter, we employ an adaptive solving strategy that begins with a larger initial value of δ and gradually decreases it.

For a general parameterization $\boldsymbol{x}$, the determinant of the Jacobian at a specific parameter value represents the ratio of the area in the computational domain to that in the parameter domain near that point. Since we assume the parameter domain to be the unit square $[0, 1]^2$, the Jacobian determinant should be equal to the area of the computational domain. To achieve this, we initially set δ to 5% of the computational domain's area. If the problem cannot be solved with this value of δ, we progressively decrease it using a decay factor $decay\_factor$. The specific steps for solving the problem are outlined in Algorithm 5.

Algorithm 5 Foldover elimination

Require: $\boldsymbol{x}$: Planar NURBS parameterization;
Require: $decay\_factor$: Decay factor for parameter δ;
Require: $area(\Omega)$: Area of the computational domain;
Require: max_iter: Maximum number of iterations.
Ensure: $\boldsymbol{x}$: Parameterization after foldover elimination.
for $k = 0, 1, \ldots, max\_iter$ **do**
  Calculate $\delta = decay\_factor^k * 0.05 * area(\Omega)$;
  Solve the unconstrained optimization problem $\arg\min_{\mathbf{c}_i,\ i\in\mathcal{I}_I} \mathscr{E}^{\text{fold}}$; ▷ (5.45)
  Update the control points $\mathbf{c}_i,\ i \in \mathcal{I}_I$;
  if $\mathscr{E}^{\text{fold}} < 100 *$ MACHINE_PRECISION **then**
    return Parameterization $\boldsymbol{x}$ after foldover elimination;
  end if
end for
  return "Maximum number of iterations reached!";

Parameterization Quality Improvement

Considering that the Jacobian determinant is present in the energy function (5.43), it acts as a barrier to prevent self-intersections in the resulting parameterization. However, as the objective function exhibits discontinuous variation, the Jacobian determinant value can cross zero abruptly. In such cases, the objective function lacks a minimum value, and employing the original energy function (5.43) would lead to failure in solving. Therefore, we introduce the following modifications to the original energy function:

$$\mathscr{E}^c = \begin{cases} \mathscr{E}, & \text{if } \min|\mathcal{J}| > 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{5.46}$$

In other words, when the minimum value of the Jacobian determinant at the Gaussian integration points is negative, the original energy function $\mathscr{E}$ is modified to become infinite. This modification serves as a penalty for invalid parameterizations. The modified energy function, denoted as $\mathscr{E}^c$, acts as a barrier that distinguishes between bijective and non-bijective parameterizations. Hence, we refer to the approach presented in this subsection as the barrier function method. Modern nonlinear optimization solvers commonly employ line search techniques, such as the Armijo–Goldstein criterion, the Wolfe–Powell criterion, strong Wolfe criterion, and others, to ensure a significant decrease in the objective function value. By incorporating these techniques, the modified energy function $\mathscr{E}^c$ effectively prevents the occurrence of self-intersections, ensuring a smooth and valid parameterization. The resulting parameterization, depicted on the right side of Fig. 5.18, demonstrates the effectiveness of the approach.

Penalty Function-Based Method

The barrier function-based method presented in the previous section requires an initial parameterization that already exhibits bijectivity. However, obtaining such a parameterization efficiently can be challenging and may necessitate additional foldover elimination steps. It is crucial to highlight that these foldover elimination steps offer limited enhancements to the parameterization quality and may lead to redundant computations (see the middle of Fig. 5.18). To tackle this issue, we present a penalty function-based approach for parameterization. This method is straightforward to implement and effectively eliminates the need for extra foldover elimination steps. The fundamental workflow of this penalty function-based method is illustrated in Fig. 5.19. To the best of our knowledge, this concept was initially introduced by Garanzha to address mesh untangling problems [27, 28]. Notably, Wang and Ma recently applied this idea to planar parameterization problems [84].

The basic idea is quite simple. In order to accommodate invalid initial parameterizations and unfold folds during the optimization process, we introduce a novel

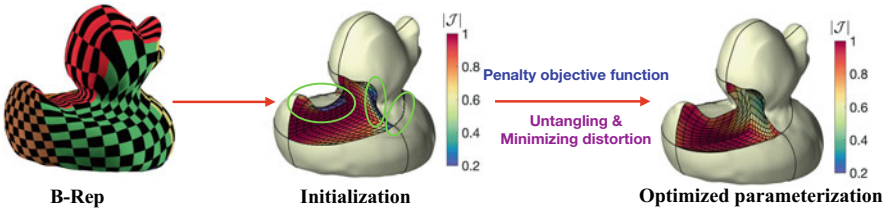


Fig. 5.19 The workflow of the penalty function-based method

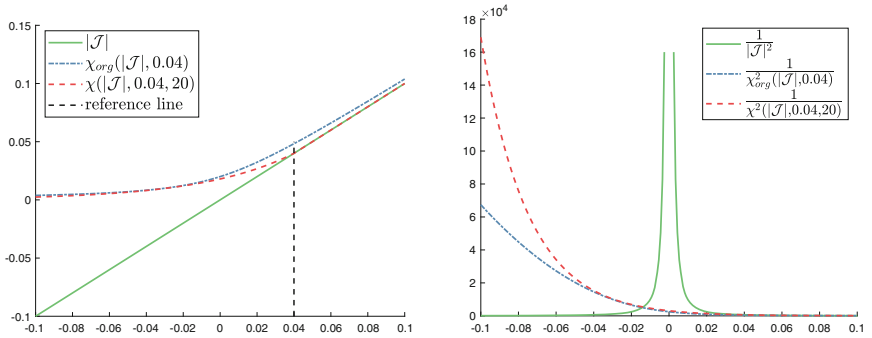


Fig. 5.20 Penalty function and Jacobian regularization techniques. (**a**) Jacobian determinant and different penalty functions. (**b**) Reciprocal square of Jacobian determinant and penalty functions

penalty function. The proposed penalty function, represented as $\chi(|\mathcal{J}|, \varepsilon, \beta)$, is defined as follows:

$$\chi(|\mathcal{J}|, \varepsilon, \beta) = \begin{cases} \varepsilon \cdot e^{\beta(|\mathcal{J}|-\varepsilon)} & \text{if } |\mathcal{J}| \leq \varepsilon, \\ |\mathcal{J}| & \text{if } |\mathcal{J}| > \varepsilon, \end{cases} \tag{5.47}$$

where ε is a small positive number, and β is a penalty factor used to control the slope of the penalty function.

As shown in Fig. 5.20a, if $|\mathcal{J}| < \varepsilon$, then $\chi(|\mathcal{J}|, \varepsilon, \beta)$ is equal to a small positive number. On the other hand, if $|\mathcal{J}| \geq \varepsilon$, it is exactly equal to the Jacobian determinant $|\mathcal{J}|$. Therefore, intuitively, $\frac{1}{\chi^2(|\mathcal{J}|,\varepsilon,\beta)}$ imposes a significant penalty for negative Jacobian determinants and has a smaller value to accept positive Jacobian determinants, as shown in Fig. 5.20b.

Remark 5.3.1 For mesh untangling problems, Garanzha [27] proposed a penalty function χ_{org}:

$$\chi_{org}(|\mathcal{J}|, \varepsilon) = \frac{|\mathcal{J}| + \sqrt{\varepsilon^2 + |\mathcal{J}|^2}}{2}, \tag{5.48}$$

where ε is a positive number. However, as shown in Fig. 5.20a, even when $|\mathcal{J}| > \varepsilon$, the penalty function χ_{org} is not strictly equal to the Jacobian determinant $|\mathcal{J}|$. This introduces additional errors, which may not be desirable in practical applications.

Based on the idea of Jacobian regularization, we replace the Jacobian determinant in the denominator of the pointwise angle distortion function (5.42) with the penalty function (5.47), and we obtain

$$\mathscr{E}_p^{\text{angle,c}} = \frac{1}{8}\left(\|\mathcal{J}\|_F^2 \left\|\mathcal{J}^{-1}\right\|_F^2 - 1\right). \tag{5.49}$$

Similarly, we also modify the uniformity energy function (5.40) as follows:

$$\mathscr{E}_p^{\text{unif.,c}} = \frac{\text{vol}(\Omega)}{\chi(|\mathcal{J}|, \varepsilon, \beta)} + \frac{\chi(|\mathcal{J}|, \varepsilon, \beta)}{\text{vol}(\Omega)}. \tag{5.50}$$

Eventually, the corrected weighted objective functional is expressed as follows:

$$\mathscr{E}^c = \int_{\hat{\Omega}} \left(\lambda^{\text{angle}} \mathscr{E}_p^{\text{mips,c}} + \lambda^{\text{unif.}} \mathscr{E}_p^{\text{unif.,c}}\right) \mathrm{d}\hat{\Omega}. \tag{5.51}$$

The resulting volumetric parameterization is depicted in Fig. 5.19 (right).

5.3.4 PDE-Based Methods

The theory of harmonic mapping has garnered considerable attention in the realm of planar parameterization owing to its exceptional mathematical properties and robust theoretical underpinnings. In actuality, the optimization-based approaches elucidated in the preceding sections fundamentally strive to approximate the inverse mapping of harmonic mapping in finite-dimensional spline spaces.

Figure 5.21 illustrates a cross-section of a twin-screw compressor, presenting notable geometric challenges, particularly with extreme aspect ratios between the rotor clearances. While numerous parameterization techniques perform admirably on established benchmark geometries, they often demonstrate subpar performance in practical applications. Figure 5.21a showcases the parameterization results achieved using the penalty function method outlined in Sect. 5.3.3 [41]. For such challenging geometric shapes, the precise selection of pertinent parameters becomes crucial. Moreover, during our research, we discovered an unreasonable influence of the parameter domain size on the parameterization results. In contrast, the PDE-based Elliptic Grid Generation (EGG) method [35] yields a satisfactory parameterization outcome, as depicted in Fig. 5.21b.

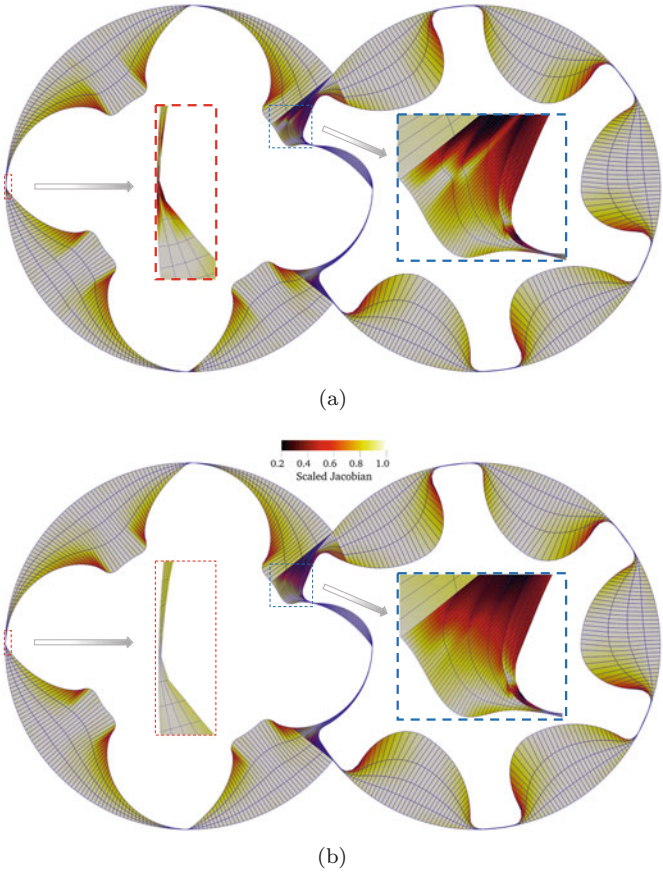


Fig. 5.21 Parameterizations generated by the penalty function-based method [41] and the Elliptic Grid Generation (EGG) method [34]. (**a**) Penalty function-based method. (**b**) Elliptic Grid Generation (EGG) method

The fundamental concept behind the Elliptic Grid Generation (EGG) method is to compute a harmonic mapping $\boldsymbol{x}$ from the parametric domain $\hat{\Omega}$ to the computational domain Ω by solving the following set of Laplace equations:

$$\begin{cases} \Delta\xi(x, y) = 0 \\ \Delta\eta(x, y) = 0 \end{cases} \text{s.t. } \boldsymbol{x}^{-1}|_{\partial\Omega} = \partial\hat{\Omega}. \tag{5.52}$$

The problem (5.52) belongs to a specific class of Dirichlet problems. The existence of a solution is ensured if the boundary $\partial\Omega$ satisfies the $C^{1,\alpha}$ Hölder continuity condition for some $\alpha \in (0, 1)$, and the uniqueness of the solution is guaranteed by the maximum principle. Given the assumption that the parametric domain $\hat{\Omega}$ is convex, typically represented as a unit square, the unique solution $\boldsymbol{x}^{-1}$

establishes a one-to-one correspondence, ensuring its non-vanishing Jacobian $\mathcal{J}$, between the interior regions of the parametric domain $\hat{\Omega}$ and the computational domain Ω [18].

Discretization in Sobolev Space H^2

In the context of generating parameterizations for IGA, the primary focus lies on the mapping $\boldsymbol{x}$ from the parametric domain $\hat{\Omega}$ to the computational domain Ω, which represents the inverse of the harmonic mapping $\boldsymbol{x}^{-1}$. Consequently, the set of Laplace equations (5.52) is transformed into an equivalent problem by Xu et al. [92]. The resulting problem is a nonlinear vector-valued second-order PDE:

$$\begin{cases} \widetilde{\mathcal{L}} x = 0 \\ \widetilde{\mathcal{L}} y = 0 \end{cases} \text{s.t. } \boldsymbol{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{5.53}$$

where

$$\widetilde{\mathcal{L}} = \frac{\mathcal{L}}{g_{11} + g_{22}}, \tag{5.54}$$

with the differential operator

$$\mathcal{L} = g_{22}\frac{\partial^2}{\partial\xi^2} - 2g_{12}\frac{\partial^2}{\partial\xi\,\partial\eta} + g_{11}\frac{\partial^2}{\partial\eta^2}, \tag{5.55}$$

and $g_{ij} = \boldsymbol{x}_{,\xi_i} \cdot \boldsymbol{x}_{,\xi_j}$ denotes the entries of the metric tensor $\boldsymbol{\mathcal{G}}$ in (5.38).

The scaled operator (5.54) was introduced by Hinz et al. [36] to enhance convergence. This operator provides a more consistent convergence criterion for geometries with varying length scales and demonstrates improved convergence properties in numerical experiments.

Let us denote by $\mathbb{S}$ the spline space spanned by NURBS basis functions. Let $\mathbb{S}_0 = \{N_i \in \mathbb{S} : N_i|_{\partial\hat{\Omega}} = 0\}$ be the collection of $N_i \in \mathbb{S}$ that vanish on $\partial\hat{\Omega}$. Following the IGA setting, we have the following variational counterpart of (5.53):

$$\forall N_i \in \mathbb{S}_0 : \begin{cases} \boldsymbol{\mathcal{F}}^x = \mathbf{0}, \\ \boldsymbol{\mathcal{F}}^y = \mathbf{0}, \end{cases} \text{s.t. } \boldsymbol{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{5.56}$$

where

$$\boldsymbol{\mathcal{F}}^x = \int_{\hat{\Omega}} \mathbf{N}\, \widetilde{\mathcal{L}} x \, \mathrm{d}\hat{\Omega}, \tag{5.57}$$

$$\boldsymbol{\mathcal{F}}^y = \int_{\hat{\Omega}} \mathbf{N}\, \widetilde{\mathcal{L}} y \, \mathrm{d}\hat{\Omega}, \tag{5.58}$$

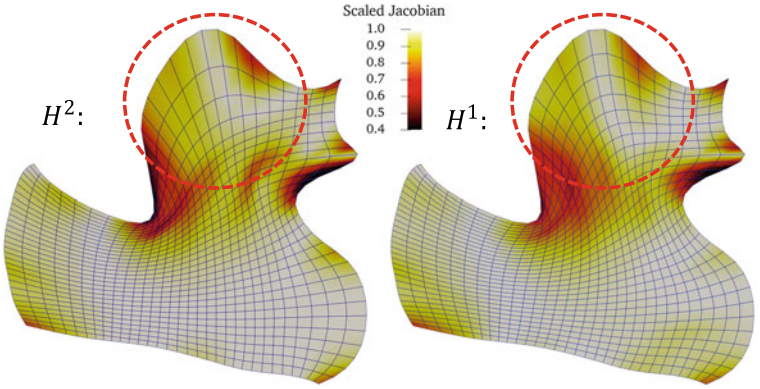


Fig. 5.22 Duck example: The left displays the parameterization resulting from the H^2 discretization (5.56), where some non-uniform elements can be observed inside the red circle. On the right side, the result achieved through the discretization (5.59) in the H^1 space is displayed. The color encodes the scaled Jacobian, with white representing optimal orthogonality

and $\mathbf{N}$ denotes the column collection of the NURBS basis functions $N_i \in \mathbb{S}_0$.

Then the unknown inner control points can be determined by solving the aforementioned nonlinear system, with the known boundary control points acting as Dirichlet boundary conditions.

The parameterization obtained from solving the nonlinear system (5.56) is presented on the left side of Fig. 5.22. It is evident that non-uniform elements appear near the head of the duck, highlighted by the red circle. As pointed out in [35], this issue can be partially alleviated by refining the current geometry to achieve a more accurate approximation of the harmonic mapping. However, such refinement operations introduce unnecessary control points and increase the complexity of CAD geometries, potentially leading to challenges in subsequent analyses and downstream processes. This phenomenon, widely observed in EGG [92], is an inherent characteristic. In the following section, our objective is to improve the quality of the parameterization while maintaining the same number of control points. To tackle this problem, we introduce a scale factor and propose a novel discretization for (5.52) in the Sobolev space H^1 instead of H^2.

Discretization in Sobolev Space H^1

In this section, to further enhance the quality of the parameterization, we introduce the following discretization in the Sobolev space H^1:

$$\forall N_i \in \mathbb{S}_0 : \begin{cases} \mathcal{F}^x_{H^1} = \mathbf{0}, \\ \mathcal{F}^y_{H^1} = \mathbf{0}, \end{cases} \text{ s.t. } \boldsymbol{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{5.59}$$

where

$$\begin{aligned} \mathcal{F}^{x}_{H^1} &= \int_{\hat{\Omega}} \nabla_{\boldsymbol{x}} \mathbf{N} \cdot \nabla_{\boldsymbol{x}} \xi \, \mathrm{d}\hat{\Omega}, \\ \mathcal{F}^{y}_{H^1} &= \int_{\hat{\Omega}} \nabla_{\boldsymbol{x}} \mathbf{N} \cdot \nabla_{\boldsymbol{x}} \eta \, \mathrm{d}\hat{\Omega}, \end{aligned} \tag{5.60}$$

and $\mathbf{N}$ denotes the column collection of the NURBS basis functions $N_i \in \mathbb{S}_0$.

Upon solving (5.59), the resulting parameterization is depicted on the right side of Fig. 5.22. It is evident that the parameterization quality has been substantially enhanced, as observed from the improved orthogonality and uniformity, and the absence of non-uniform elements. Importantly, the cardinality of control points remains unchanged.

5.3.5 *Experiments and Comparisons*

In this section, we embark on a comprehensive examination of the aforementioned parameterization techniques. Our main objective is to evaluate their effectiveness and applicability by utilizing the comprehensive test dataset [53] that comprises 977 planar models. Figure 5.23 show some representative examples of this dataset. By subjecting these techniques to rigorous scrutiny across a diverse range of models, we

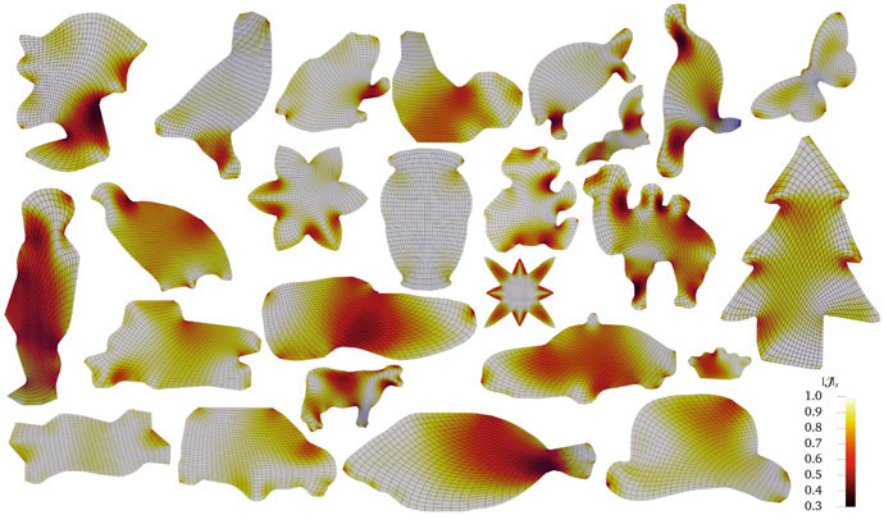


Fig. 5.23 Planar parameterization results gallery

aim to gain a deeper understanding of their performance and identify any potential limitations.

Quality Metrics for Parameterizations

In this section, we adopt the following quality metrics to measure the quality of parameterizations:

- Scaled Jacobian:

$$|\mathcal{J}|_s = \frac{|\mathcal{J}|}{\prod_{i=1}^{p} \boldsymbol{x}_{,\xi_i}}. \tag{5.61}$$

  The scaled Jacobian characterizes the orthogonality of the parameterization and falls within the range of $[-1, 1]$. A parameterization $\boldsymbol{x}$ is considered bijective only when $|\mathcal{J}|_s > 0$ for all $\boldsymbol{\xi} \in \hat{\Omega}$. If the scaled Jacobian $|\mathcal{J}|_s$ takes on negative values, it indicates that the parameterization is not bijective. A value close to 1.0 for the scaled Jacobian $|\mathcal{J}|_s$ across the entire parameter domain suggests a high degree of orthogonality.
- Uniformity metric:

$$unif. = \left(\frac{|\mathcal{J}|}{\text{area}(\Omega)} - 1 \right)^2, \tag{5.62}$$

  where area(Ω) denotes the area (or volume) of the computational domain Ω. Ideally, the Jacobian determinant $|\mathcal{J}|$ for the parameterization $\boldsymbol{x}$ should equal the ratio of the computational domain area (or volume) to the parameter domain area (or volume). Thus, the optimal value of the uniformity index $unif.$ is 0.0.

In our experiments, we evaluate both quality metrics using a dense sampling of 1001×1001 points, including the boundaries. We omit the maximum values of scaled Jacobian and the minimum values of uniformity metric in our statistics since they are attainable in most examples.

Effectiveness and Quality Assessment

Figure 5.24 illustrates the worst-case quality metrics for the resulting parameterizations, specifically $\min(|\mathcal{J}|_s)$ and $\max(unif.)$. It is worth noting that a negative value of $\min(|\mathcal{J}|_s)$ indicates the presence of self-intersections in the resulting parameterization. From the figure, it is evident that the optimization-based parameterization methods exhibit greater robustness in this planar dataset compared to the PDE-based methods. Table 5.1 presents the success rates of various parameterization approaches.

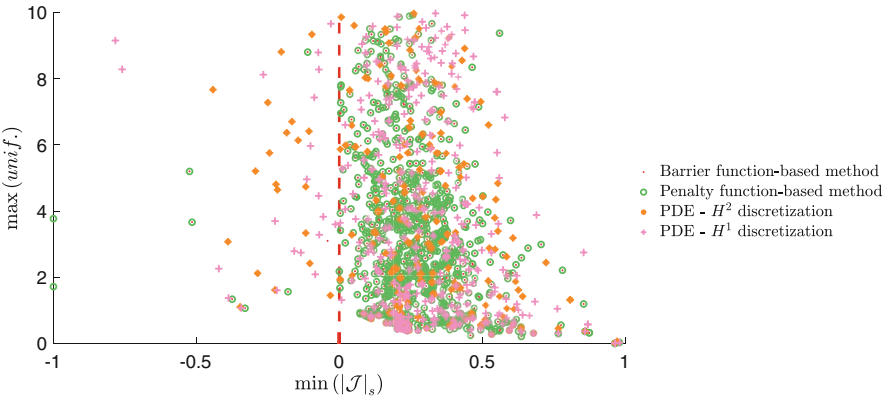


Fig. 5.24 Planar dataset: min $(|\mathcal{J}|_s)$ vs. max $(unif.)$

Table 5.1 Success rates of parameterization approaches

Method	Success rate
Barrier function-based method	$\frac{961}{977} \simeq 98.36\%$
Barrier function-based method	$\frac{956}{977} \simeq 97.85\%$
PDE-H2 discretization	$\frac{608}{977} \simeq 62.23\%$
PDE-H1 discretization	$\frac{721}{977} \simeq 73.80\%$

Figure 5.25 presents the mean values of the scaled Jacobian $|\mathcal{J}|_s$ and the uniformity metric. It is evident that the two optimization methods, namely the barrier function-based method and the penalty function-based method, exhibit convergence to similar results across most models. This outcome is expected, as the primary difference between these two methods lies in their approach to handling bijectivity constraints. In comparison to the PDE-H_2 discretization method, the H_1 discretization method demonstrates improved uniformity.

Computational Time

Figure 5.26 illustrates the performance of different parameterization approaches in terms of computational time, with the PDE-H_2 discretization exhibiting the best performance. In general, the PDE-based parameterization methods demonstrate faster computation times compared to the optimization-based methods.

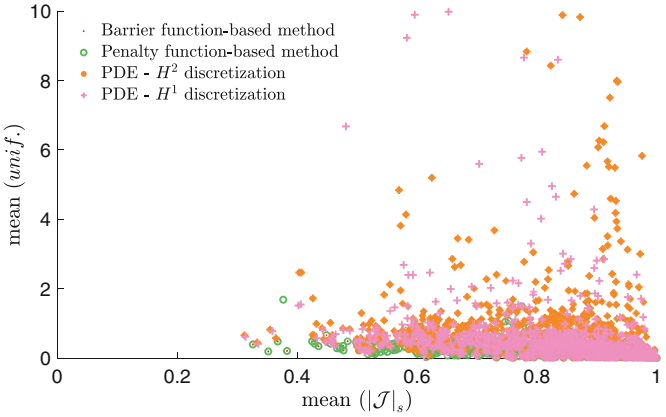


Fig. 5.25 Planar dataset: mean $(|\mathcal{J}|_s)$ vs. mean $(unif.)$

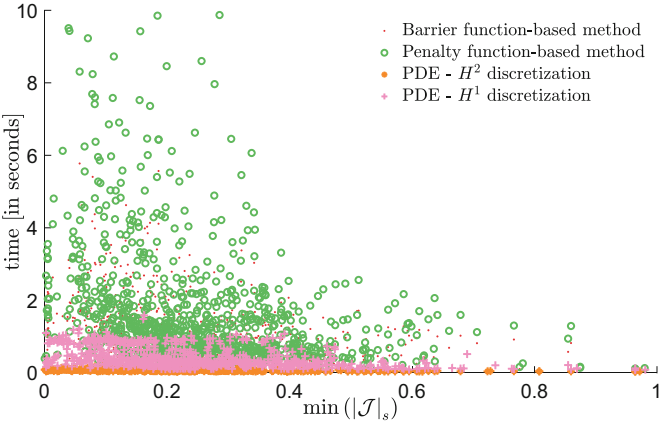


Fig. 5.26 Planar dataset: computational time

Volumetric Parameterizations

The aforementioned parameterization approaches can be seamlessly applied to 3D volumetric parameterization problems. Figure 5.27 showcases volumetric parameterization results obtained using the penalty function-based method. It is noteworthy that the minimum value of the scaled Jacobian indicates the resulting parameterizations are bijective.

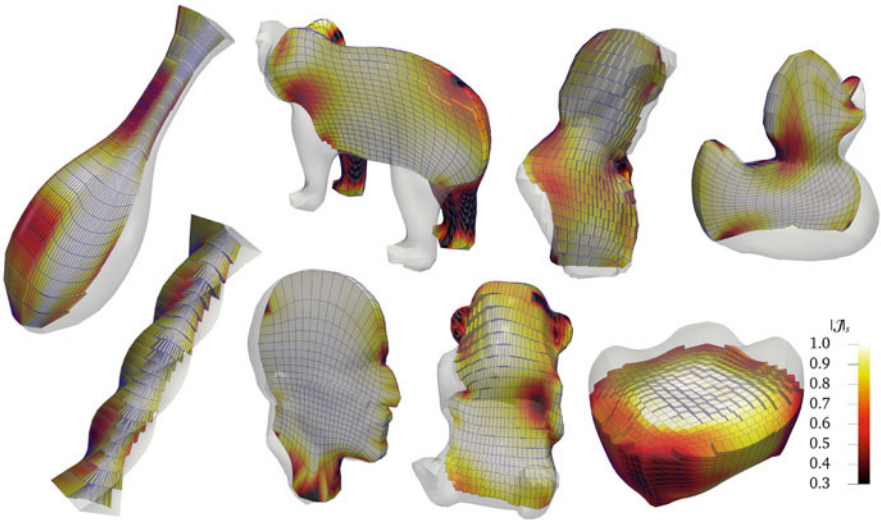


Fig. 5.27 Volumetric parameterization results gallery

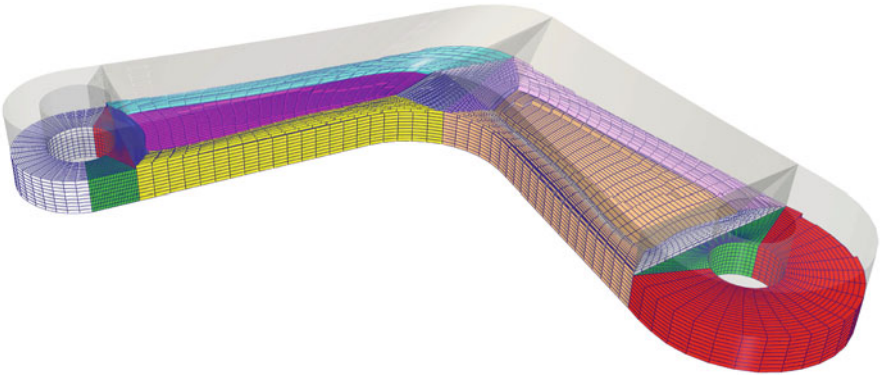


Fig. 5.28 Multi-patch volumetric parameterization

Extension to Multi-patch Parameterizations

Although our previous discussion primarily focused on the single-patch scenario, it is important to emphasize that our parameterization techniques can be seamlessly extended to handle multi-patch domain parameterizations. This versatility arises when the topological layout of the quadrilateral or hexahedral elements is determined. By intelligently integrating multiple patches, our techniques enable the effective representation of complex domains. Figure 5.28 serves as a visual demonstration, showcasing a volumetric parameterization that leverages a multi-patch configuration.

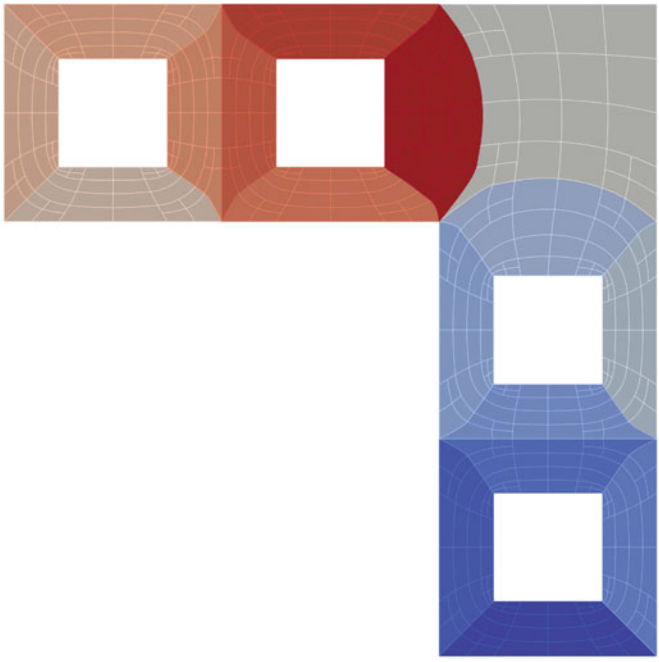


Fig. 5.29 Multi-patch THB-spline parameterization

Extension to THB-Spline Parameterizations

As illustrated in Fig. 5.29, it is important to highlight that our parameterization techniques exhibit impeccable compatibility with THB-spline parameterizations. This noteworthy characteristic indicates that our methods are adept at effectively managing the intricate complexities that arise in THB-spline-based parameterization. Consequently, our parameterization techniques offer a versatile and robust solution that can be applied across a broad spectrum of applications.

5.4 Isogeometric Kirchhoff–Love Shell Analysis

In this section, an example of isogeometric analysis for thin shell mechanics is provided. The aim of the section is to show how the isogeometric Kirchhoff–Love shell equations can be derived from geometric and mechanics principles, employing geometric and solution representations using splines and how they can be used in the analysis. The derivation of the shell element is provided in Sect. 5.4.1. Thereafter, three benchmark studies are presented in Sect. 5.4.2.

5.4.1 The Isogeometric Kirchhoff–Love Shell Element

The isogeometric Kirchhoff–Love shell was first presented by Kiendl in [45] and has been widely used within the isogeometric analysis community. The derivation of the Kirchhoff–Love shell model in this section follows the PhD Thesis of Kiendl [44]. For basic principles of continuum mechanics, the reader is referred to [6] or other continuum mechanics textbooks. The section concludes with a benchmark problem using snapping of a hyperelastic shell, based on [82].

Geometry

The Kirchhoff–Love shell theory describes the deformation of surfaces. Hence, let $\mathcal{S}(\xi_1, \xi_2) : \mathbb{R}^2 \to \mathbb{R}^3$ be a surface. For this surface, the *covariant basis vector* $\boldsymbol{a}_\alpha$ is defined by taking the derivatives of the surface with respect to the parametric coordinate ξ_α, i.e.,

$$\boldsymbol{a}_\alpha = \frac{\partial \mathcal{S}}{\partial \xi_\alpha}, \quad \alpha = 1, 2. \tag{5.63}$$

Using the covariant basis, the *covariant metric tensor* or *first fundamental form* is defined by

$$a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta. \tag{5.64}$$

Using the first fundamental form of the surface, the *contravariant metric tensor* is defined using the inverse of $[a_{\alpha\beta}]$, as $a^{\alpha\beta} = [a_{\alpha\beta}]^{-1}$. Furthermore, the *contravariant basis* $\boldsymbol{a}^\alpha$ is defined by

$$\boldsymbol{a}^\alpha = a^{\alpha\beta}\boldsymbol{a}_\beta. \tag{5.65}$$

Using the covariant basis vectors from (5.63), the *surface unit normal vector* is defined by

$$\hat{\boldsymbol{a}}_3 = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}. \tag{5.66}$$

In addition to the surface gradients, the curvature of the surface is a quantity of interest, typically related to bending. In the present derivation of the Kirchhoff–Love shell theory, the curvature is included via the *second fundamental form*, as

$$b_{\alpha\beta} = \hat{\boldsymbol{a}}_3 \cdot \boldsymbol{a}_{\alpha,\beta} = -\hat{\boldsymbol{a}}_{3,\beta} \cdot \boldsymbol{a}_\alpha. \tag{5.67}$$

Here, $\boldsymbol{a}_{\alpha,\beta}$ denotes the second derivative or Hessian of the surface and $\hat{\boldsymbol{a}}_{3,\alpha}$ denotes the derivative of the unit normal vector with respect to the parameter ξ_α. Via

Weingarten's formula [88] it holds that $\hat{\boldsymbol{a}}_{3,\alpha} = -b_\alpha^\beta \boldsymbol{a}_\beta$ with $b_\alpha^\beta = a^{\alpha\gamma} b_{\gamma\beta}$. Since the second fundamental form $b_{\alpha\beta}$ depends on the surface Hessian $\boldsymbol{a}_{\alpha,\beta}$, second derivatives of the surface description $\mathcal{S}(\xi_1, \xi_2)$ are required.

Assuming the Kirchhoff hypothesis [19], i.e., no shear of the shell cross-section, orthogonality of orthogonal vectors after deformation, and no thickness change, the Kirchhoff–Love shell formulation assumes that any point in the shell can be described by its position on the surface $\mathcal{S}(\xi_1, \xi_2)$ and its position along the surface normal $\boldsymbol{a}_3$ as

$$\boldsymbol{x}(\xi_1, \xi_2, \xi_3) = \mathcal{S}(\xi_1, \xi_2) + \xi_3 \hat{\boldsymbol{a}}_3. \tag{5.68}$$

The derivatives of the coordinate system $\boldsymbol{x}$ with respect to the parametric coordinates ξ_i $(i = 1, 2, 3)$ provide the full basis of the coordinate system used for the Kirchhoff–Love shell element. The covariant basis of $\boldsymbol{x}$ is given by

$$\begin{aligned} \boldsymbol{g}_\alpha &= \frac{\partial \boldsymbol{x}}{\partial \xi_\alpha} = \boldsymbol{a}_\alpha + \xi_3 \boldsymbol{a}_{3,\alpha}, \\ \boldsymbol{g}_3 &= \frac{\partial \boldsymbol{x}}{\partial \xi_\alpha} = \hat{\boldsymbol{a}}_3. \end{aligned} \tag{5.69}$$

Following from the covariant basis, the first fundamental form $g_{ij} = \boldsymbol{g}_i \cdot \boldsymbol{g}_j$ is defined using the first and second fundamental forms as

$$\begin{aligned} g_{\alpha\beta} &= (\boldsymbol{a}_\alpha + \xi_3 \boldsymbol{a}_{3,\alpha}) \cdot (\boldsymbol{a}_\beta + \xi_3 \boldsymbol{a}_{3,\beta}), \\ &= a_{\alpha\beta} - 2\xi_3 b_{\alpha\beta} + \xi_3^2 \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta, \\ g_{33} &= 1, \\ g_{i3} &= g_{3i} = 0. \end{aligned} \tag{5.70}$$

The last term, quadratic in ξ_3, can be neglected for thin or moderately thick shells [10]. The contravariant metric tensor g^{ij} and the contravariant basis $\boldsymbol{g}^i$ are derived like for the surface $\mathcal{S}$. Using the shell coordinate system (5.68) and the covariant basis (5.70), the kinematic relation for the Kirchhof–Love shell can be derived.

Kinematic Relation

The kinematic relation relates shell displacements to strains. Let $\mathring{\boldsymbol{x}}(\xi_1, \xi_2, \xi_3)$ denote the undeformed configuration of the shell and let $\boldsymbol{x}(\xi_1, \xi_2, \xi_3)$ denote the deformed configuration of the shell. Then, the deformation $\boldsymbol{u}(\xi_1, \xi_2, \xi_3)$ of a material point is defined as

$$\boldsymbol{u}(\xi_1, \xi_2, \xi_3) = \boldsymbol{x}(\xi_1, \xi_2, \xi_3) - \mathring{\boldsymbol{x}}(\xi_1, \xi_2, \xi_3). \tag{5.71}$$

Additionally, the *deformation gradient* $\mathbf{F}$ is a tensor that maps between the undeformed basis $\mathring{\boldsymbol{g}}_i$ and the deformed basis $\boldsymbol{g}_i$, meaning that an infinitesimal line element $\mathrm{d}\mathring{\boldsymbol{x}}$ in the undeformed configuration is defined as $\mathrm{d}\boldsymbol{x} = \mathbf{F} \cdot \mathrm{d}\mathring{\boldsymbol{x}}$ in the undeformed configuration [6]. Accordingly, the deformation gradient $\mathbf{F}$ is defined as

$$\mathbf{F} = \boldsymbol{g}_i \otimes \mathring{\boldsymbol{g}}^i. \tag{5.72}$$

Indeed, the deformation gradient maps $\mathring{\boldsymbol{g}}_i$ onto $\boldsymbol{g}_i$ via $\boldsymbol{g}_i = \mathbf{F}\mathring{\boldsymbol{g}}_i$ [6]. Using the deformation gradient, the Green–Lagrange strain tensor $\mathbf{E} = E_{ij}\, \mathring{\boldsymbol{g}}^i \otimes \mathring{\boldsymbol{g}}^j$ relates the nonlinear relation between deformations and strains

$$\mathbf{E} = \frac{1}{2}\Big(\mathbf{F}^\top \mathbf{F} - \mathbf{I}\Big) = \frac{1}{2}(\mathbf{C} - \mathbf{I}), \tag{5.73}$$

where $\mathbf{C}$ is the *deformation tensor*. Using the definition of the deformation gradient and the fact that the identity tensor $\mathbf{I}$ is equal to the metric tensor G^{ij} on $\mathring{\boldsymbol{g}}^i \otimes \mathring{\boldsymbol{g}}^j$ yields

$$E_{ij} = \frac{1}{2}\big(g_{ij} - G_{ij}\big). \tag{5.74}$$

Using the definition of the metric tensor from (5.70), the coefficients of the strain tensor can be expressed in terms of the surface metric and the curvature:

$$\begin{aligned} E_{\alpha\beta} &= \frac{1}{2}\big(a_{\alpha\beta} - 2\xi_3 b_{\alpha\beta}\big) - \mathring{a}_{\alpha\beta} + 2\xi_3 \mathring{b}_{\alpha\beta} x] \\ &= \frac{1}{2}\big(a_{\alpha\beta} - \mathring{a}_{\alpha\beta}\big) + \xi_3\Big(\mathring{b}_{\alpha\beta} - b_{\alpha\beta}\Big) = \varepsilon_{\alpha\beta} + \kappa_{\alpha\beta}. \end{aligned} \tag{5.75}$$

The shear strains E_{i3} and E_{3i} and the normal strain E_{33} vanish because of the orthogonality and unity of the basis vector $\boldsymbol{g}_3$ in deformed and undeformed configurations. This indeed shows that the shell formulation following from the assumed coordinate system in (5.68) yields a formulation free of cross-sectional shear and thickness change. Hence, the shell can be represented by its mid-surface only and the strain tensor is represented with respect to the first two components of the basis, i.e., $\mathbf{E} = E_{\alpha\beta}\, \mathring{\boldsymbol{g}}^\alpha \otimes \mathring{\boldsymbol{g}}^\beta$. The coefficients ε_{ij} and $\kappa_{\alpha\beta}$ relate to the *membrane strain tensor* $\boldsymbol{\varepsilon} = \varepsilon_{\alpha\beta}\, \mathring{\boldsymbol{g}}^\alpha \otimes \mathring{\boldsymbol{g}}^\beta$ and the *bending strain tensor* $\boldsymbol{\kappa} = \kappa_{\alpha\beta}\, \mathring{\boldsymbol{g}}^\alpha \otimes \mathring{\boldsymbol{g}}^\beta$.

Constitutive Relation

In general continuum mechanics, the *second Piola–Kirchhoff* stress tensor $\mathbf{S} = S^{ij}\, \mathring{\boldsymbol{g}}_i \otimes \mathring{\boldsymbol{g}}_j$ is energetically conjugate to the Green–Langrange strain tensor $\mathbf{E} =$

$E_{ij}\,\mathring{\boldsymbol{g}}^i \otimes \mathring{\boldsymbol{g}}^j$ [6]. For a 3D continuum, the coefficients of the second Piola–Kirchhoff stress tensor can be defined using a *strain energy density function* Ψ:

$$S^{ij} = 2\frac{\partial \Psi}{\partial C_{ij}}. \tag{5.76}$$

In addition, the *material tensor* or *elasticity tensor* $\mathcal{C} = \mathcal{C}^{ijkl}\,\mathring{\boldsymbol{g}}_i \otimes \mathring{\boldsymbol{g}}_j \otimes \mathring{\boldsymbol{g}}_k \otimes \mathring{\boldsymbol{g}}_l$ is a fourth-order tensor that relates the total differentials of the second Piola–Kirchhoff stress $\mathbf{S}$ and the Green–Lagrange strain $\mathbf{E}$. Its coefficients are defined by

$$\mathcal{C}^{ijkl} = \frac{\partial S_{ij}}{\partial E^{kl}} = 4\frac{\partial^2 \Psi}{\partial C_{ij}\partial C_{kl}}, \tag{5.77}$$

such that the coefficients of the total differential of the second Piola–Kirchhoff stress tensor, $\mathrm{d}S^{ij}$, relate to the total differential of the Green–Lagrange strain tensor, $\mathrm{d}E_{ij}$, via

$$\mathrm{d}S^{ij} = \mathcal{C}^{ijkl}\,\mathrm{d}E_{ij}\,. \tag{5.78}$$

For linear elastic materials, stress and strain are linearly dependent such that $\mathcal{C}$ has constant coefficients according to (5.77). Therefore, the following identity is valid for linear materials:

$$S^{ij} = \mathcal{C}^{ijkl} E_{kl}. \tag{5.79}$$

Furthermore, assuming small strains, through thickness deformation is neglected and $C_{33} = g_{33} = 1$, which allows to use 2D constitutive models. However, when strains are large, for example, in hyperelastic material models, the plane stress assumption that $S^{33} = 0$ is typically violated [46], hence $C_{33} \neq 1$. To use the in-plane components of the stress tensor, $S^{\alpha\beta}$ in the Kirchhoff–Love shell model, *static condensation* of the material tensor $\mathcal{C}$ needs to be performed to satisfy the plane stress condition. The formulations for the hyperelastic stress and material tensors for Kirchhoff–Love shells are provided in [46] and an extension for stretch-based material models was provided by [82].

Variational Formulation

The variational formulation for the Kirchhoff–Love shell is derived based on the *Principle of Virtual Work*. According to this principle, the total energy in the system, represented by $W(\boldsymbol{u}) = W^{\mathrm{int}}(\boldsymbol{u}) - W^{\mathrm{ext}}(\boldsymbol{u})$, is minimized for the deformation $\mathbf{u}$ if and only if its variation $\delta W(\boldsymbol{u}, \boldsymbol{v})$ with respect to $\boldsymbol{u}$ is equal to zero:

$$\delta W(\boldsymbol{u}, \boldsymbol{v}) = \delta W^{\mathrm{int}}(\boldsymbol{u}, \boldsymbol{v}) - \delta W^{\mathrm{ext}}(\boldsymbol{u}, \boldsymbol{v}). \tag{5.80}$$

Here, $\boldsymbol{v}$ denotes the virtual displacements. Since $\delta W(\boldsymbol{u}, \boldsymbol{v})$ can be nonlinear, the displacement $\boldsymbol{u}$ can be found using the Newton–Raphson method by solving

$$\delta_{\boldsymbol{v}} W + \delta^2_{\boldsymbol{vw}} W \Delta \boldsymbol{u} = 0, \tag{5.81}$$

where $\delta_{\boldsymbol{v}} W = \delta W(\boldsymbol{u}, \boldsymbol{v})$ and $\delta^2_{\boldsymbol{vw}} W = \delta^2 W(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ is the second variation of the energy in the system using virtual displacements $\boldsymbol{v}$ and $\boldsymbol{w}$ and $\Delta \boldsymbol{u}$ is the incremental update of the displacements.

The derivation of the external virtual work is rather straightforward. Assuming that the body force vector $\boldsymbol{f}$ and the boundary force vector $\boldsymbol{g}$ are independent of the deformation field $\boldsymbol{u}$, the first variation of the external work W^{ext} simply yields

$$\begin{aligned} \delta_{\boldsymbol{v}} W^{\text{ext}} = \int_{\Omega^\star} \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\Omega^\star + \int_{\partial\Omega^\star} \boldsymbol{g} \cdot \boldsymbol{v} \, \mathrm{d}\Gamma = \int_\tau \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\Omega \, \mathrm{d}\xi_3 \\ + \int_\tau \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{v}\boldsymbol{n} \, \mathrm{d}\Gamma \, \mathrm{d}\xi_3 \,, \end{aligned} \tag{5.82}$$

where $\Omega^\star = \tau \times \Omega$ with τ the thickness domain $\tau = [-t/2, t/2]$ of the shell and Ω the surface domain. For the internal virtual work, the first variation with respect to the displacements $\boldsymbol{u}$ is given by

$$\delta_{\boldsymbol{v}} W^{\text{int}} = \int_{\Omega^\star} \mathbf{S} : \delta_{\boldsymbol{v}} \mathbf{E} \, \mathrm{d}\Omega^\star = \int_\tau \int_\Omega \mathbf{S} : \delta_{\boldsymbol{v}} \mathbf{E} \, \mathrm{d}\Omega = \int_\Omega \mathbf{N} : \delta_{\boldsymbol{v}} \boldsymbol{\varepsilon} + \mathbf{M} : \delta_{\boldsymbol{v}} \boldsymbol{\kappa} \, \mathrm{d}\Omega \,. \tag{5.83}$$

Here, the definition of the strain tensors $\boldsymbol{\varepsilon}$ and $\boldsymbol{\kappa}$ from (5.75) is used and the *membrane force tensor* $\mathbf{N}$ and the bending moment tensor $\mathbf{M}$ are defined as moments of the stress tensor through thickness:

$$\begin{aligned} \mathbf{N} &= \int_\tau \mathbf{S} \, \mathrm{d}\xi_3 \,, \\ \mathbf{M} &= \int_\tau \xi_3 \mathbf{S} \, \mathrm{d}\xi_3 \,. \end{aligned} \tag{5.84}$$

The second variation of the energy of the system required for solving the nonlinear system of equations using the Newton–Raphson iterations (see (5.81)) solely depends on the second variation of the internal energy, assuming deformation-independent body forces. Taking the variation of the internal energy with respect to $\boldsymbol{u}$, the second variation becomes

$$\delta^2_{\boldsymbol{vw}} W(\boldsymbol{u}) = \int_\Omega \delta_{\boldsymbol{w}} \mathbf{N} : \delta_{\boldsymbol{v}} \boldsymbol{\varepsilon} + \mathbf{N} : \delta^2_{\boldsymbol{vw}} \boldsymbol{\varepsilon} + \delta_{\boldsymbol{w}} \mathbf{M} : \delta_{\boldsymbol{v}} \boldsymbol{\kappa} + \mathbf{M} : \delta^2_{\boldsymbol{vw}} \boldsymbol{\kappa} \, \mathrm{d}\Omega \,. \tag{5.85}$$

The variations of $\mathbf{N}$ and $\mathbf{M}$ can be obtained using the total differential of $\mathbf{S}$ and $\delta\mathbf{S}$ (see (5.78)). First, since

$$\delta\mathbf{E} = \delta(\boldsymbol{\varepsilon} + \xi_3\boldsymbol{\kappa}) = \delta\boldsymbol{\varepsilon} + \delta(\xi_3\boldsymbol{\kappa}) = \delta\boldsymbol{\varepsilon} + \boldsymbol{\kappa}\delta\xi_3 + \xi_3\delta\boldsymbol{\kappa} \tag{5.86}$$

and using the total differential of the strain, $\delta\mathbf{E}$, and integrating $\delta\mathbf{S}$ through the thickness, the total differentials of $\mathbf{N}$ and $\mathbf{M}$ are obtained:

$$\begin{aligned} \delta\mathbf{N} &= \int_\tau \delta\mathbf{S}\,\mathrm{d}\xi_3 \quad = \int_\tau (\mathcal{C} : \delta\boldsymbol{\varepsilon} + \xi_3\mathcal{C} : \delta\boldsymbol{\kappa} + \delta\xi_3\mathcal{C} : \boldsymbol{\kappa})\,\mathrm{d}\xi_3\,, \\ \delta\mathbf{M} &= \int_\tau \xi_3\delta\mathbf{S}\,\mathrm{d}\xi_3 = \int_\tau (\xi_3\mathcal{C} : \delta\boldsymbol{\varepsilon} + \xi_3\mathcal{C} : \delta\boldsymbol{\kappa} + \delta\xi_3\mathcal{C} : \boldsymbol{\kappa})\,\mathrm{d}\xi_3\,. \end{aligned} \tag{5.87}$$

From the first and second variations of the internal energy, respectively (5.83) and (5.85), it can be seen that the first and second variations of the membrane strain and bending strain tensors.

Discretization

The principle of virtual work derived in (5.80) is valid for any variation of the unknown displacement field $\boldsymbol{u}(\xi_1, \xi_2, \xi_3)$. In order to discretize the principle of virtual work, it is assumed that the undeformed and deformed configurations $\mathring{\boldsymbol{x}}$ and $\boldsymbol{x}$, respectively, are represented by a finite sum of basis functions $\varphi_k(\xi_1, \xi_2)$ weighted by coefficients $\mathring{\boldsymbol{x}}_k^h$ and $\boldsymbol{x}_k^h$, i.e.,

$$\begin{aligned} \mathring{\boldsymbol{x}}^h(\xi_1, \xi_2) &= \sum_k \varphi_k(\xi_1, \xi_2)\mathring{\boldsymbol{x}}_k^h, \\ \boldsymbol{x}^h(\xi_1, \xi_2) &= \sum_k \varphi_k(\xi_1, \xi_2)\boldsymbol{x}_k^h. \end{aligned} \tag{5.88}$$

Here, the superscript h indicates discrete approximations of $\mathring{\boldsymbol{x}}$ or $\boldsymbol{x}$ and the index k indicates the k-th component of this representation. Since the displacement field $\boldsymbol{u}$ is defined as the difference between $\mathring{\boldsymbol{x}}$ and $\boldsymbol{x}$, it can similarly be expressed as a discrete field $\boldsymbol{u}^h$ and the variations in the principle of virtual work are represented by virtual displacements $\boldsymbol{u}_k^h$. As a consequence, all variations in the virtual work equation are represented by derivatives with respect to components of the virtual nodal displacements $\boldsymbol{u}_k^h$. In the following, all quantities are referred to in the discrete setting, and hence the superscript h is omitted.

In the sequel, r denotes the global index of the degree of freedom u_r representing a component of one of the nodal displacement vectors. For the sake of brevity, the shorthand notation $(\cdot)_{,r} = \frac{\partial(\cdot)}{\partial u_r}$ is used to represent derivatives with respect to u_r. Using (5.71), the variation of the deformed configuration is

$$\boldsymbol{x}_{,r} = \sum_k \left(\mathring{\boldsymbol{x}}_{k,r} + \boldsymbol{u}_{k,r}\right) = \sum_k \varphi_k \boldsymbol{u}_{k,r}, \tag{5.89}$$

where the last equality follows from the fact that the undeformed configuration is trivially independent of the deformation field $\boldsymbol{u}$. Similarly, the derivatives of the covariant basis vectors $\boldsymbol{a}_\alpha$ of the discrete deformed configuration $\boldsymbol{x}_h$, see (5.63), are

$$\boldsymbol{a}_{\alpha,r} = \left(\frac{\partial \boldsymbol{x}_k}{\partial \xi_\alpha}\right)_{,r} = \sum_k \frac{\partial \varphi_k}{\partial \xi_\alpha} \boldsymbol{u}_{k,r}. \tag{5.90}$$

As a consequence, the variation of the surface metric tensor of the deformed configuration, $a_{\alpha\beta}$ (see (5.64)), becomes

$$a_{\alpha\beta,r} = \left(\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta\right)_r = \boldsymbol{a}_{\alpha,r} \cdot \boldsymbol{a}_\beta + \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_{\beta,r}. \tag{5.91}$$

Since the undeformed configuration is invariant to the deformation field $\boldsymbol{u}$, the first variation of the membrane strain tensor $\boldsymbol{\varepsilon}$ from (5.75) becomes

$$\varepsilon_{\alpha\beta,r} = \frac{1}{2} a_{\alpha\beta,r}. \tag{5.92}$$

Similarly, the second variations of the deformed configuration, the deformed surface metric tensor, and the membrane strain can be derived. Starting with the first variation of the deformed configuration from (5.89), the second variation becomes

$$\boldsymbol{x}_{,rs} = \sum_k \varphi_k \boldsymbol{u}_{k,rs} = \boldsymbol{0}. \tag{5.93}$$

The second variation of $\boldsymbol{u}_k$ is zero since the components of these nodal weights are linear in u_r. Similarly, $\boldsymbol{a}_{\alpha,rs} = \boldsymbol{0}$. As a consequence, the second variation of the surface metric tensor in the deformed configuration, $a_{\alpha\beta}$, becomes

$$a_{\alpha\beta,rs} = \boldsymbol{a}_{\alpha,rs} \cdot \boldsymbol{a}_\beta + \boldsymbol{a}_{\alpha,r} \cdot \boldsymbol{a}_{\beta,s} + \boldsymbol{a}_{\alpha,s} \cdot \boldsymbol{a}_{\beta,s} + \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_{\beta,rs}, \tag{5.94}$$

$$= \boldsymbol{a}_{\alpha,r} \cdot \boldsymbol{a}_{\beta,s} + \boldsymbol{a}_{\alpha,s} \cdot \boldsymbol{a}_{\beta,s}. \tag{5.95}$$

Again, since the undeformed configuration is invariant to the deformation field $\boldsymbol{u}$, the second variation of the membrane strain tensor becomes

$$\varepsilon_{\alpha\beta,rs} = \frac{1}{2} a_{\alpha\beta,rs}. \tag{5.96}$$

To derive the variations of the curvature tensor, the variations of the second fundamental form $b_{\alpha\beta}$ are needed, hence requiring variations of $\boldsymbol{a}_{\alpha,\beta}$ and $\hat{\boldsymbol{a}}_3$, see (5.67). Firstly, the variation of $\boldsymbol{a}_{\alpha,\beta}$ with respect to u_r is

$$\boldsymbol{a}_{(\alpha,\beta),r} = \left(\frac{\partial^2 \boldsymbol{x}}{\partial \xi_\alpha \partial \xi_\beta}\right)_{,r} = \sum_k \frac{\partial^2 \varphi_k}{\partial \xi_\alpha \partial \xi_\beta} \boldsymbol{u}_{k,r}. \tag{5.97}$$

Furthermore, using $\boldsymbol{a}_3 = \boldsymbol{a}_1 \times \boldsymbol{a}_2$, the variation of the unit normal vector $\hat{\boldsymbol{a}}_3$ is

$$\hat{\boldsymbol{a}}_{3,r} = \left(\frac{\boldsymbol{a}_3}{|\boldsymbol{a}_3|} \right)_{,r} = \frac{|\boldsymbol{a}_3| \boldsymbol{a}_{3,r} - \boldsymbol{a}_3 (|\boldsymbol{a}_3|)_{,r}}{|\boldsymbol{a}_3|^2}. \tag{5.98}$$

Here, the variation of the non-unit normal vector $\boldsymbol{a}_3$ is obtained by

$$\boldsymbol{a}_{3,r} = \boldsymbol{a}_{1,r} \times \boldsymbol{a}_2 + \boldsymbol{a}_1 \times \boldsymbol{a}_{2,r}, \tag{5.99}$$

and since $|\boldsymbol{a}_3| = \sqrt{\boldsymbol{a}_3 \cdot \boldsymbol{a}_3}$, the variation of the normalization $|\boldsymbol{a}_3|$ is

$$(|\boldsymbol{a}_3|)_{,r} = \frac{\boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,r}}{|\boldsymbol{a}_3|}, \tag{5.100}$$

such that the variation of the unit surface normal vector of the undeformed configuration, $\hat{\boldsymbol{a}}_3$, can be obtained. Together with the variation of the surface Hessian, $\boldsymbol{a}_{(\alpha,\beta),r}$ from (5.97), the variation of the second fundamental form becomes

$$b_{\alpha\beta,r} = \hat{\boldsymbol{a}}_{3,r} \cdot \boldsymbol{a}_{\alpha,\beta} + \hat{\boldsymbol{a}}_3 \cdot \boldsymbol{a}_{(\alpha,\beta),r}. \tag{5.101}$$

From the definition of the bending strain tensor $\boldsymbol{\kappa}$ in (5.75) and the fact that the undeformed configuration is invariant to the deformation field $\boldsymbol{u}$, the coefficients of the first variation of the bending strain tensor become

$$\kappa_{\alpha\beta,r} = -b_{\alpha\beta,r}. \tag{5.102}$$

To obtain the second variation of the bending strain tensor $\boldsymbol{\kappa}$, the second variations of $\boldsymbol{a}_{\alpha,\beta}$ and $\hat{\boldsymbol{a}_3}$ need to be obtained in order to compute the second variation of $b_{\alpha\beta}$. Firstly, from (5.97), it follows that $\boldsymbol{a}_{(\alpha,\beta),rs} = \boldsymbol{0}$ since the second variation of $\boldsymbol{u}_k$ is zero. Secondly, for the second variation of the unit normal vector $\hat{\boldsymbol{a}}_3$, the second variation of the non-unit normal vector $\boldsymbol{a}_3$ and its length $|\boldsymbol{a}_3|$ are needed. The second variation of the non-unit normal vector follows from the first variation in (5.98) and from $\boldsymbol{a}_{(\alpha,\beta),rs}$:

$$\boldsymbol{a}_{3,rs} = \boldsymbol{a}_{1,rs} \times \boldsymbol{a}_2 + \boldsymbol{a}_{1,r} \times \boldsymbol{a}_{2,s} + \boldsymbol{a}_{1,s} \times \boldsymbol{a}_{2,r} + \boldsymbol{a}_1 \times \boldsymbol{a}_{2,rs} \tag{5.103}$$

$$= \boldsymbol{a}_{1,r} \times \boldsymbol{a}_{2,s} + \boldsymbol{a}_{1,s} \times \boldsymbol{a}_{2,r}. \tag{5.104}$$

Furthermore, the second variation of $|\boldsymbol{a}_3|$ is

$$(|\boldsymbol{a}_3|)_{,rs} = \frac{|\boldsymbol{a}_3| (\boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,r})_{,s} - (\boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,r})(|\boldsymbol{a}_3|)_{,s}}{|\boldsymbol{a}_3|^2} \tag{5.105}$$

$$= \frac{\boldsymbol{a}_{3,s} \cdot \boldsymbol{a}_{3,r} + \boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,rs}}{|\boldsymbol{a}_3|} - \frac{(\boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,r})(\boldsymbol{a}_3 \cdot \boldsymbol{a}_{3,s})}{|\boldsymbol{a}_3|^3}. \tag{5.106}$$

Using the second variations of the non-unit normal $\boldsymbol{a}_3$ (see (5.103)) and its length $|\boldsymbol{a}_3|$ (see (5.105)), the second variation of the unit normal vector $\hat{\boldsymbol{a}}_3$ can be derived

$$\begin{aligned}\hat{\boldsymbol{a}}_{3,rs} &= \left(\frac{|\boldsymbol{a}_3|\boldsymbol{a}_{3,r} - \boldsymbol{a}_3(|\boldsymbol{a}_3|)_{,r}}{|\boldsymbol{a}_3|^2}\right)_{,s} \\ &= \frac{\left(|\boldsymbol{a}_3|\boldsymbol{a}_{3,r} - \boldsymbol{a}_3(|\boldsymbol{a}_3|)_{,r}\right)_{,s}}{|\boldsymbol{a}_3|^2} - \frac{\left(|\boldsymbol{a}_3|\boldsymbol{a}_{3,r} - \boldsymbol{a}_3(|\boldsymbol{a}_3|)_{,r}\right)2|\boldsymbol{a}_3|(|\boldsymbol{a}_3|)_{,s}}{|\boldsymbol{a}_3|^4} \\ &= \frac{\boldsymbol{a}_{3,rs}}{|\boldsymbol{a}_3|} - \frac{(|\boldsymbol{a}_3|)_{,s}\boldsymbol{a}_{3,r}}{|\boldsymbol{a}_3|^2} - \frac{\boldsymbol{a}_{3,s}(|\boldsymbol{a}_3|)_{,r}}{|\boldsymbol{a}_3|^2} - \frac{\boldsymbol{a}_3(|\boldsymbol{a}_3|)_{,rs}}{|\boldsymbol{a}_3|^2} + 2\frac{\boldsymbol{a}_3(|\boldsymbol{a}_3|)_{,r}(|\boldsymbol{a}_3|)_{,s}}{|\boldsymbol{a}_3|^3}.\end{aligned} \tag{5.107}$$

Additionally, taking the variation of $b_{\alpha\beta,r}$ and using the first and second variations of $\boldsymbol{a}_{\alpha,\beta}$ and $\hat{\boldsymbol{a}}_3$, the second variation of the second fundamental form $b_{\alpha\beta}$ can be obtained

$$\begin{aligned}b_{\alpha\beta,rs} &= \hat{\boldsymbol{a}}_{3,rs}\cdot\boldsymbol{a}_{\alpha,\beta} + \hat{\boldsymbol{a}}_{3,r}\cdot\boldsymbol{a}_{(\alpha,\beta),s} + \hat{\boldsymbol{a}}_{3,s}\cdot\boldsymbol{a}_{(\alpha,\beta),r} + \hat{\boldsymbol{a}}_3\cdot\boldsymbol{a}_{(\alpha,\beta),rs}, \\ &= \hat{\boldsymbol{a}}_{3,rs}\cdot\boldsymbol{a}_{\alpha,\beta} + \hat{\boldsymbol{a}}_{3,r}\cdot\boldsymbol{a}_{(\alpha,\beta),s} + \hat{\boldsymbol{a}}_{3,s}\cdot\boldsymbol{a}_{(\alpha,\beta),r}.\end{aligned} \tag{5.108}$$

From (5.108) and (5.75), it directly follows that the coefficients of the second variation of the bending strain tensor are

$$\kappa_{\alpha\beta,rs} = -b_{\alpha\beta,rs}. \tag{5.109}$$

Besides the first and second variations of the membrane strain tensor $\boldsymbol{\varepsilon}$ and the bending strain tensor $\boldsymbol{\kappa}$, the first variations of the membrane force tensor $\mathbf{N}$ and the bending moment tensor $\mathbf{M}$ also need to be obtained. Using the total differentials d$\mathbf{N}$ and d$\mathbf{M}$ (see (5.84)), the coefficients of the first variations of $\mathbf{N}$ and $\mathbf{M}$ with respect to u_r are

$$\begin{aligned}N^{\alpha\beta}_{,r} &= \left(\int_\tau \mathcal{C}^{\alpha\beta\gamma\delta}\,\mathrm{d}\xi_3\right)\varepsilon_{\gamma\delta,r} + \left(\int_\tau \xi_3\mathcal{C}^{\alpha\beta\gamma\delta}\,\mathrm{d}\xi_3\right)\kappa_{\gamma\delta,r}, \\ M^{\alpha\beta}_{,r} &= \left(\int_\tau \xi_3\mathcal{C}^{\alpha\beta\gamma\delta}\,\mathrm{d}\xi_3\right)\varepsilon_{\gamma\delta,r} + \left(\int_\tau \xi_3^2\mathcal{C}^{\alpha\beta\gamma\delta}\,\mathrm{d}\xi_3\right)\kappa_{\gamma\delta,r}.\end{aligned} \tag{5.110}$$

Note that the last term of (5.86) drops out because the variation of ξ_3 with respect to u_r is zero. Using the variations with respect to the nodal displacement components u_r, the first and second variations of the energy equation in the shell following from the virtual work statement in (5.80) can be defined for each component u_r. Firstly, the first variation of the energy statement provides the components of the residual vector $\boldsymbol{R}$ as

$$R_r(\boldsymbol{u}) = \int_\Omega \mathbf{N}(\boldsymbol{u}) : \boldsymbol{\varepsilon}_{,r}(\boldsymbol{u}) + \mathbf{M}(\boldsymbol{u}) : \boldsymbol{\kappa}_{,r}(\boldsymbol{u}) \, \mathrm{d}\Omega - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{u}_{,r} \, \mathrm{d}\Omega - \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{u}_{,r} \, \mathrm{d}\Gamma \,. \tag{5.111}$$

Secondly, the second variation of the energy statement from (5.81) provides the Jacobian matrix for the Newton–Raphson iterations, also known as the *(tangential) stiffness matrix* K, with coefficients

$$K_{rs} = \int_\Omega \mathbf{N}_{,s}(\boldsymbol{u}) : \boldsymbol{\varepsilon}_{,r}(\boldsymbol{u}) + \mathbf{N}(\boldsymbol{u}) : \boldsymbol{\varepsilon}_{,rs}(\boldsymbol{u}) + \mathbf{M}_{,s}(\boldsymbol{u}) : \boldsymbol{\kappa}_{,r}(\boldsymbol{u}) + \mathbf{M}(\boldsymbol{u}) : \boldsymbol{\kappa}_{,rs}(\boldsymbol{u}) \, \mathrm{d}\Omega \,. \tag{5.112}$$

In case of zero displacements, i.e., $\boldsymbol{u} = \mathbf{0}$, the deformation gradient $\mathbf{F}$ is an identity map and the deformation tensor $\mathbf{C}$ is the identity tensor. Therefore, the stress tensor becomes the null tensor, $\mathbf{S} = \mathbf{0}$, making the tensors $\mathbf{N}$ and $\mathbf{M}$ vanish as well. In this case, the first integral of the residual vector $\boldsymbol{R}$ is zero and the second and fourth terms drop out of the stiffness matrix K. This gives the *external force vector* $\boldsymbol{P}$ and the *linear stiffness matrix* K^L, with coefficients

$$\begin{aligned} P_r &= -R_r(\mathbf{0}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{u}_{,r} \, \mathrm{d}\Omega + \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{u}_{,r} \, \mathrm{d}\Gamma \,, \\ K^L_{rs} &= K_{rs}(\mathbf{0}) = \int_\Omega \mathbf{N}_{,s}(\mathbf{0}) : \boldsymbol{\varepsilon}_{,r}(\mathbf{0}) + \mathbf{M}_{,s}(\mathbf{0}) : \boldsymbol{\kappa}_{,r}(\mathbf{0}) \, \mathrm{d}\Omega \,. \end{aligned} \tag{5.113}$$

Up to this point, all quantities have been defined to be used in the variational formulation, except for the basis functions φ_k to define the undeformed and deformed configurations of the shell surface as well as the displacement field: $\mathring{\boldsymbol{x}}$, $\boldsymbol{x}$, and $\boldsymbol{u}$, respectively (see (5.88)). Since the Hessian of the metric tensor $a_{\alpha,\beta}$ is used in the definition of the second fundamental form and its variations, see (5.67), (5.101), and (5.108), the basis functions φ_k need to be differentiable up to the second derivative. Due to the higher order continuity that can be achieved using splines, they provide a suitable basis for the Kirchhoff–Love shell. In the paradigm of using the same splines for the representation of the geometry $\mathcal{S}(\xi_1, \xi_2)$ as well as for the discrete solution of the displacement field $\boldsymbol{u}^h(\xi_1, \xi_2, \xi_3)$, this choice of the basis introduces the *isogeometric Kirchhoff–Love shell*.

5.4.2 Benchmark Problems

In this section, we present some example problems using the isogeometric Kirchhoff–Love shell. In the first example, adopted from [82], the collapse of a truncated cone is simulated. This example uses a hyperelastic Ogden model such that the stress and material tensors are defined by (5.76)–(5.77). In the second example, adopted from [83], a mesh adaptivity example using THB-splines is

presented, using a basis as explained in Sect. 5.2.2. In the last example, a post-buckling problem on a multi-patch geometry is solved. The example is adopted from [23] and modelled using the Analysis-Suitable G^1 construction, which is a surface unstructured spline construction mentioned in Sect. 5.2.4.

Nonlinear Hyperelastic Shell Analysis

As a first example we model the collapse of a truncated cone (or *frustrum*). The geometry of the cone is given in Fig. 5.30 and is represented by NURBS from Sect. 5.2.3. The original benchmark problem is adopted from [7] and the presented results are published in [82].

The problem parameters from Fig. 5.30 are as follows. The shell has a height $H = 1$ [m], top radius $r = 1$ [m], bottom radius $R = 2$ [m], and thickness $t = 0.1$ [m]. Only a quarter of the shell is modelled since the original reference [6] uses axisymmetric elements. The shell is represented by quadratic NURBS with 32 elements over the height and one over the circumference. On the boundary Γ_1 the displacements are fixed in the x- and y-direction and free in z. Also, a uniform load p is applied on Γ_1, providing a uniform displacement Δ. On Γ_2 the displacements are fixed but rotations are free. On Γ_3 and Γ_4, symmetry conditions are applied, restricting in-plane deformations normal to the boundaries and restricting rotations on the boundary by applying clamped boundary conditions as described in [45].

The corresponding material model is of the Ogden type and has the following parameters:

$$\mu_1 = 6.300\ [\mathrm{N/m^2}], \qquad \alpha_1 = 1.3,$$

$$\mu_2 = 0.012\ [\mathrm{N/m^2}], \qquad \alpha_2 = 5.0,$$

$$\mu_3 = -0.100\ [\mathrm{N/m^2}], \qquad \alpha_3 = -2.0,$$

implying that $\mu = 4.225\ [\mathrm{N/m^2}]$. For more information about the stretch-based Ogden material model inside the isogeometric Kirchhoff–Love shell, the reader is referred to [82].

The load applied on the top boundary is either applied using displacement control (DC), by incrementally increasing the displacement of the boundary, or by arc-length control by employing Crisfield's spherical arc-length method [13] with extensions for resolving complex roots [48, 97]. If this method does not converge to an equilibrium point, the step size is bisected until a converged step is found. After this step, the step size is reset to its original value [81].

In Fig. 5.31, the results for the collapsing conical shell are presented and compared to the reference results from [6]. The displacement-controlled (DC) result

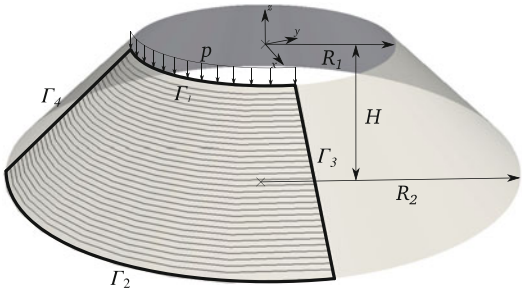


Fig. 5.30 Geometry of the collapsing conical shell with 32 quadratic elements over the height

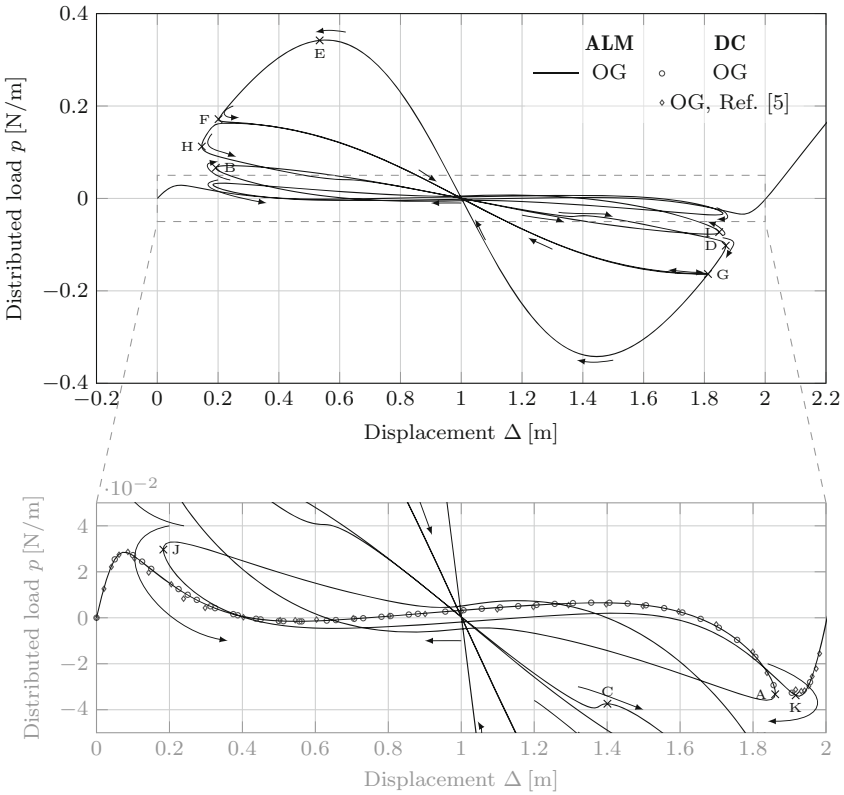


Fig. 5.31 Load-displacement diagram of the collapsing conical shell. The lines represent solutions obtained using the Arc-Length Method (ALM) and the markers represent solutions obtained by Displacement Control (DC) together with the reference solution by Başar and Itskov [7]. The capital letters in the diagram represent points for which the deformed geometry is plotted in Fig. 5.32 and the arrows describe the direction of the solution path

shows good agreement with the reference results, with differences that can be explained by shear locking as present in the reference results. When performing arc-length stepping on the problem, it can be seen that at the moment of snapping,

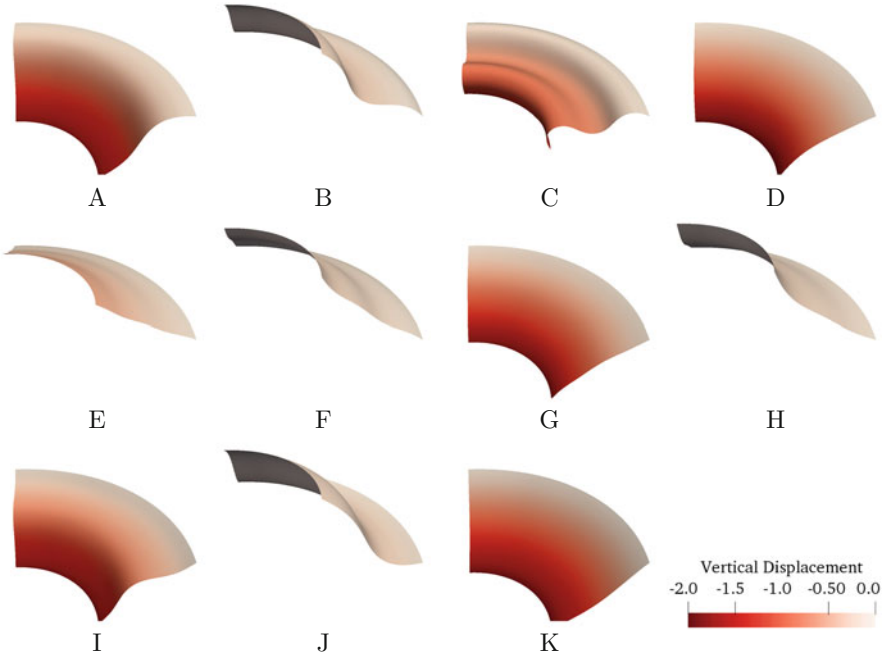
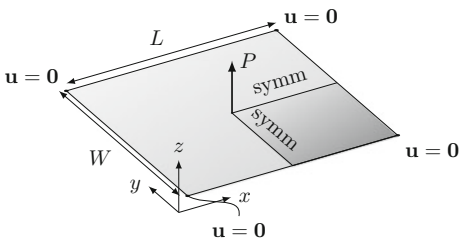


Fig. 5.32 Deformed geometries corresponding to the solution path presented in Fig. 5.31



E	=	1.0	[MPa]
ν	=	0.3	[-]
P	=	$4 \cdot 10^{-7}$	[N]
L	=	1	[mm]
W	=	1	[mm]
t	=	10^{-3}	[mm]

Fig. 5.33 Geometry and parameters for a square thin plate subject to a point load P in the middle. The plate is fully constrained in every corner. Because the problem is symmetric, only a quarter of the domain is modelled. Hence, symmetry conditions are applied. On the x-aligned symmetry axis, this implies that $u_y = \frac{\partial u_z}{\partial y} = \frac{\partial u_x}{\partial y} = 0$, and on the y-aligned symmetry axis this implies that $u_x = \frac{\partial u_z}{\partial x} = \frac{\partial u_y}{\partial x} = 0$

around $\Delta \sim 1.9$, a complex snapping phenomenon arises, coinciding with the moment of snapping in the DC path. Observing the deformations in Fig. 5.32, it can be seen that the collapsing mechanism consists of the formation of multiple waves in radial direction that invert after the loop with the highest force amplitude.

Fig. 5.34 Deformed surface from the benchmark presented in Fig. 5.33. The result is the last solution from the adaptive meshing routine with deformation norm goal functional of which the results are presented in Fig. 5.35

Nonlinear Adaptive Shell Analysis

As a next example, we consider adaptive refinement of isogeometric meshes. This example is adopted from [83]. The adaptivity iterations are performed using THB-splines and the marking of the element is driven by the Dual-Weighted Residual (DWR) method [4, 9] see [83] for the complete procedure.

We consider a square membrane subject to a point load in the middle and with corners fixed in all directions, see Fig. 5.33. As a material model, a linear Saint-Venant–Kirchhoff constitutive law with Young's modulus $E = 1.0\,[MPa]$ and a Poisson ratio $\nu = 0.3$ is used. The membrane is considered very thin compared to its in-plane dimensions, i.e., $t = 10^{-3}\,[\text{mm}]$, $L \times W = 1 \times 1\,[\text{mm}]$ $(L/t = 1000)$. In the middle of the membrane, $P = 4{\cdot}10^{-7}[\text{N}]$ is applied. As depicted in Fig. 5.33, the simulations are performed on a quarter of the domain, using symmetry conditions. As a goal functional for the DWR method, a displacement norm is used

$$\mathcal{L}(\mathbf{u}) = \int_{\Omega} \|\mathbf{u}\| \, \mathrm{d}\Omega \,. \tag{5.114}$$

The maximum number of levels for the THB refinement is 11, meaning that the finest level has $2^{11} \times 2^{11} = 2048 \times 2048$ elements.

The result of the deformed membrane for the last step of the adaptivity simulation is given in Fig. 5.34. In Fig. 5.35, the estimated error via the DWR method is given for the uniformly refined mesh as well as for the adaptively refined mesh. In addition, Fig. 5.36 provides the element errors on the meshes corresponding to the points with the marked border in Fig. 5.35, together with contour lines for the displaced geometry. From the results, it can be observed that the adaptively refined mesh is in general more efficient per degree of freedom than the uniformly refined mesh, due to localization of the error in the bottom-right corner. However, it can also be seen that the adaptive mesh is not monotonously decreasing, probably due to the nonlinearity of the problem where asymmetries in the mesh can result in large errors. From the bottom plot in Fig. 5.35, it can also be observed that the non-monotone decrease of the error is related to the percentage of the elements in the mesh that is eligible for refinement (i.e., that did not reach the maximum level yet).

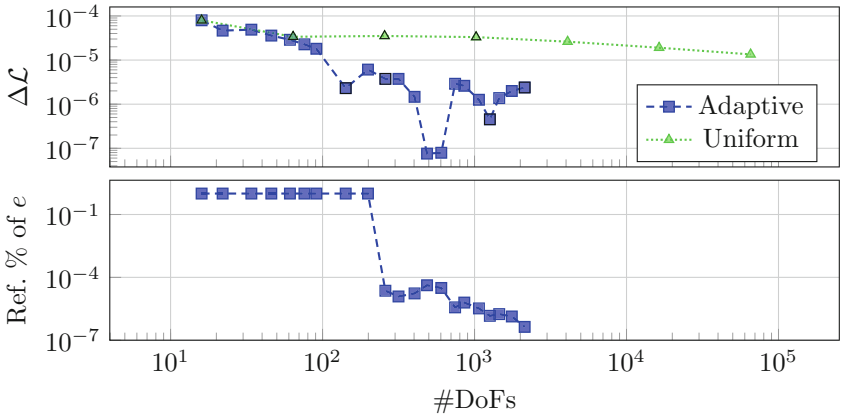


Fig. 5.35 Estimated error convergence (top) and the percentage of the total element error e that is available for refinement (bottom) against the number of degrees of freedom (DoFs) for adaptively and uniformly refined meshes with respect to the goal displacement-based goal functionals. The markers labeled with a black border are the markers for which the mesh is plotted in Fig. 5.36

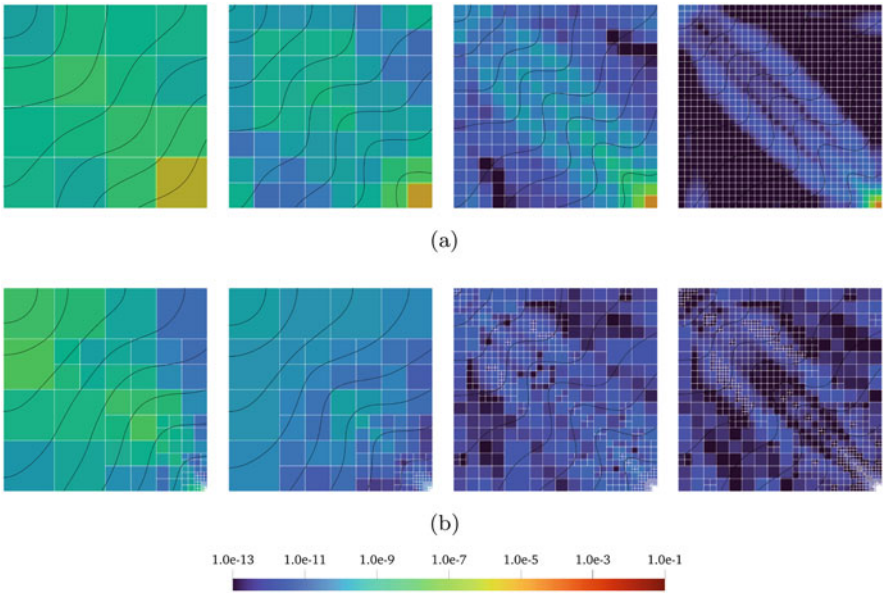


Fig. 5.36 Normalized element error values $e_k/\Delta\mathcal{L}$ for uniformly (top) and adaptively (bottom) refined meshes using goal function $\mathcal{L}(\mathbf{u}) = \int_\Omega \|\mathbf{u}\| \, \mathrm{d}\Omega$. The meshing steps increase from left to right. The contour lines represent the displacement of the membrane, with intervals of 0.1 [mm]. The bottom-right corner of the pictures indicates the fixed corner and the top-left corner is the corner where the load is applied. (**a**) Uniform refinement. (**b**) Adaptive refinement

When this percentage is small, refinement is performed using elements that have an insignificant contribution to the total error.

Nonlinear Multi-patch Shell Analysis

As a last example, a post-buckling problem on a multi-patch geometry is solved. This example is adopted from the paper [23] where the isogeometric Kirchhoff–Love shell is used on a multi-patch domain using the Analysis-Suitable G^1 basis from [22].

The geometry is given in Fig. 5.37 and is fixed on the left side and loaded with an in-plane load of λP and an out-of-plane load λP_s using $P/P_s = 100$ on the bottom-right side. The dimensions of the geometry are $L = 255$ [mm], $W = 30$ [mm], $L_h = 55$ [mm], and $W_h = 10$ [mm], and the thickness is $t = 0.6$ [mm]. Furthermore, the material parameters for a linear Saint-Venant–Kirchhoff material are $E = 71240$ [N/mm^2] and $\nu = 0.3$. Figure 5.38 shows the deformation of the L-shaped domain.

The load-displacement curves are given in Fig. 5.39 for a degree $p = 4$ basis with maximum regularity $r = p - 2$. The example has been computed by using the penalty method [33] with the penalty parameter $\alpha = 10^3$. The results show that the Analysis-Suitable G^1 basis provides a good alternative to penalty methods for multi-patch analysis of this post-buckling simulation.

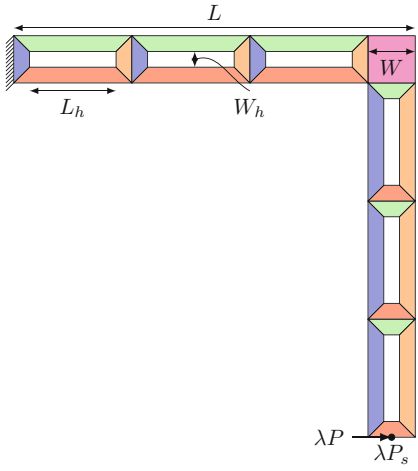


Fig. 5.37 Geometry of the L-shaped domain with length $L = 255$ [mm] and width $W = 30$ [mm] and with rectangular holes of size $L_h = 55$ [mm] and $W_h = 10$ [mm]. The geometry consists of 25 bilinear patches

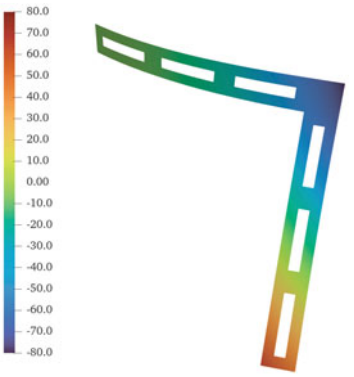


Fig. 5.38 Deformed geometry of the L-shaped domain with 25 patches corresponding to Fig. 5.37 on the last point of the load-displacement curve for the Analysis-Suitable G^1 method for $p = 4$ in Fig. 5.39. The color scale represents the out-of-plane displacement

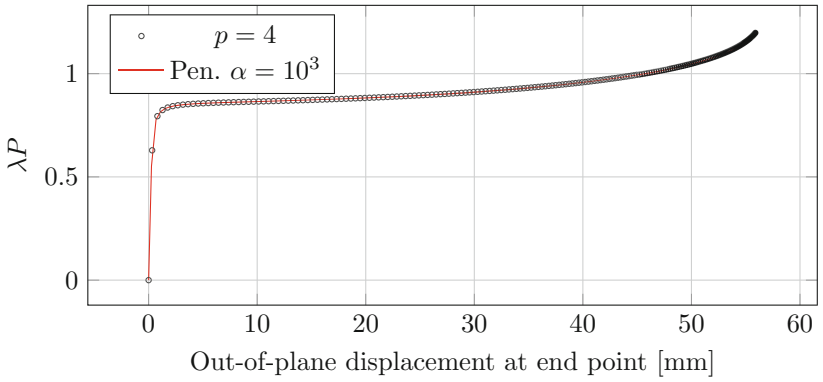


Fig. 5.39 Displacements at the point where the load is applied in Fig. 5.37. All results are plotted with the out-of-plane displacement component on the horizontal axis and the load λP on the vertical axis. The penalty parameter used for the penalty method is $\alpha = 10^3$

5.5 Conclusions and Outlook

Design through analysis—to the authors' best knowledge first mentioned in the literature in the 1970s in a report from the General Motors Research Laboratories has gained new impetus since the advent of Isogeometric Analysis in the early 2000s. It is obviously more than the bundling of CAD and CAE tools under a common user interface that hides (but does not cure) the tedious and error-prone back-and-forth conversion between genuinely incompatible representations of geometry and analysis models. As we tried to show in this chapter, breaking with the traditional triad of pre-processing, analysis, and post-processing and placing the entire workflow on common mathematical grounds—splines (cf. Sect. 5.2)—brings plentiful advantages. One of the most beneficial ones is the higher continuity that not only brings higher accuracy per degree of freedom but also simplifies the solution of higher order differential equations without auxiliary variables.

At the same time, the many advantages of IGA do not come for free: preserving higher continuity in the multi-patch case requires special techniques—unstructured spline constructions. Moreover, the task of creating analysis-suitable parameterizations from CAD models can be as complicated as generating admissible computational grids. In Sect. 5.3 we have presented several computational approaches to construct high-quality analysis-suitable parameterizations from boundary descriptions. For the reader who prefers working with classical computational grids, e.g., in the context of finite elements and finite volumes, we would like to remark that it is still possible to utilize the presented approaches to construct a bijective mapping $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$ and generate both structured and unstructured meshes by "pushing forward" a grid created in the parameter domain $\hat{\Omega}$ to the physical space Ω [59]. That way, the task of (i) ensuring the non-folding of elements and (ii) optimizing the shape and the size of elements gets decoupled as the former property is ensured by the bijectiveness of the push-forward operator.

Section 5.4 finally gave a brief insight into the analysis capabilities of IGA at the hand of Kirchhoff–Love shell models. Let us conclude this chapter with a word of caution that is likewise meant as motivation to continue research in IGA and design through analysis. We believe that all numerical methods have their strengths and weaknesses and none of them is superior to others at large. In our opinion, breaking with the traditional triad of pre-processing, analysis, and post-processing is what makes IGA stand out at this moment and, possibly, a precursor for other numerical approaches that will enable even more sophisticated design-through-analysis workflows in the future.

Acknowledgments The first author would like to thank Prof. Dr. Chun-Gang Zhu from Dalian University of Technology for his valuable contribution to domain parameterization in this chapter, as well as Dr. Kewang Chen and Prof. Dr. Cornelis Vuik from Delft University of Technology for their contributions to the development of the H^1 PDE-based methods. All three authors would like to express their gratitude to Dr. Angelos Mantzaflaris (Inria Sophia Antipolis Méditerranée), Dr. Andrea Farahat (Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Linz, Austria), Dr. Mario Kapl (ADMiRE Research Center, Carinthia University of Applied Sciences, Villach, Austria), Prof. Dr. Josef Kiendl (Institute of Engineering Mechanics Structural Analysis, Universität der Bundeswehr München, Munich, Germany), and Dr. Henk den Besten (Department of Maritime and Transport Technology, Delft University of Technology, The Netherlands) for fruitful discussions and collaboration on Kirchhoff–Love shell theory, multi-patch coupling and joint code development in the open-source IGA library G+Smo (https://github.com/gismo/gismo) [42].

References

1. I.A. Abelló Ugalde, V. Hernández Mederos, P. Barrera Sánchez, G. González Flores, Injectivity of B-spline biquadratic maps. Comput. Methods Appl. Mech. Eng. **341**, 586–608 (2018)
2. Z. Ali, W. Ma, Isogeometric collocation method with intuitive derivative constraints for PDE-based analysis-suitable parameterizations. Comput. Aided Geom. Des. **87**, 101994 (2021)
3. J.A. Augustitus, M.M. Kamal, L.J. Howell, Design through analysis of an experimental automobile structure. SAE Trans. **86**, 2186–2198 (1977)

4. W. Bangerth, R. Rannacher, *Adaptive Finite Element Methods for Differential Equations*, 1st edn. (Birkhäuser Basel, Basel, 2003). ISBN 978-3-7643-7009-1
5. P.J. Barendrecht, *Isogeometric Analysis for Subdivision Surfaces* (Eindhoven University of Technology, Eindhoven, 2013)
6. Y. Başar, R. Grytz, Incompressibility at large strains and finite-element implementation. Acta Mech. **168**(1), 75–101 (2004)
7. Y. Başar, M. Itskov, Finite element formulation of the Ogden material model with application to rubber-like shells. Int. J. Numer. Methods Eng. **42**(7), 1279–1305 (1998)
8. Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, T.W. Sederberg, Isogeometric analysis using T-splines. Comput. Methods Appl. Mech. Eng. **199**(5), 229–263 (2010). Computational Geometry and Analysis
9. R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods. Acta Numer. 10:1–102 (2001)
10. M. Bischoff, K.-U. Bletzinger, W.A. Wall, E. Ramm, Models and finite elements for thin-walled structures, in *Encyclopedia of Computational Mechanics*, chapter 3 (John Wiley & Sons, Ltd, Hoboken, 2004)
11. E. Cohen, T. Martin, R.M. Kirby, T. Lyche, R.F. Riesenfeld, Analysis-aware modeling: understanding quality considerations in modeling for isogeometric analysis. Comput. Methods Appl. Mech. Eng. **199**(5–8), 334–356 (2010)
12. A. Collin, G. Sangalli, T. Takacs, Analysis-suitable G1 multi-patch parametrizations for C1 isogeometric spaces. Comput. Aided Geom. Des. **47**, 93–113 (2016)
13. M.A. Crisfield, An arc-length method including line searches and accelerations. Int. J. Numer. Methods Eng. **19**(9), 1269–1289 (1983)
14. C. de Boor, Package for calculating with B-splines. SIAM J. Numer. Anal. **14**(3), 441–472 (32 pages) (1977). Published By: Society for Industrial and Applied Mathematics
15. C. De Boor, *A Practical Guide to Splines*. Applied Mathematical Sciences, 1 edn. (Springer, New York, 1978)
16. J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes. Graph. Models **70**(4), 76–86 (2008)
17. T. Dokken, T. Lyche, K.F. Pettersen, Polynomial splines over locally refined box-partitions. Comput. Aided Geom. Des. **30**(3), 331–356 (2013)
18. P. Duren, W. Hengartner, Harmonic mappings of multiply connected domains. Pac. J. Math. **180**(2), 201–220 (1997)
19. A. Edward, H. Love, G.H. Darwin, XVI. The small free vibrations and deformation of a thin elastic shell. Philos. Trans. R. Soc. London (A) **179**, 491–546 (1997)
20. P.L. Fackler, Algorithm 993: efficient computation with Kronecker products. ACM Trans. Math. Softw. **45**(2), 1–9 (2019)
21. A. Falini, J. Špeh, B. Jüttler, Planar domain parameterization with THB-splines. Comput. Aided Geom. Des. **35**, 95–108 (2015)
22. A. Farahat, B. Jüttler, M. Kapl, T. Takacs, Isogeometric analysis with C1-smooth functions over multi-patch surfaces. Comput. Methods Appl. Mech. Eng. **403**, 115706 (2023)
23. A. Farahat, H.M. Verhelst, J. Kiendl, M. Kapl, Isogeometric analysis for multi-patch structured Kirchhoff–Love shells. Comput. Methods Appl. Mech. Eng. **411**, 116060 (2023)
24. G. Farin, D. Hansford, Discrete Coons patches. Comput. Aided Geom. Des. **16**(7), 691–700 (1999)
25. D.R. Forsey, R.H. Bartels, Hierarchical B-spline refinement. ACM SIGGRAPH Comput. Graph. **22**(4), 205–212 (1988)
26. X.-M. Fu, Y. Liu, B.-N. Guo, Computing locally injective mappings by advanced MIPS. ACM Trans. Graph. **34**(4), 1–12 (2015)
27. V.A. Garanzha, I.E. Kaporin, Regularization of the barrier variational method. Comput. Math. Math. Phys. **39**(9), 1426–1440 (1999)
28. V. Garanzha, I. Kaporin, L. Kudryavtseva, F. Protais, N. Ray, D. Sokolov, Foldover-free maps in 50 lines of code. ACM Trans. Graph. **40**(4), 1–16 (2021)

29. C. Giannelli, B. Jüttler, H. Speleers, THB-splines: the truncated basis for hierarchical splines. Comput. Aided Geom. Des. **29**(7), 485–498 (2012)
30. C. Giannelli, B. Jüttler, S.K. Kleiss, A. Mantzaflaris, B. Simeon, J. Špeh, THB-splines: an effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. Comput. Methods Appl. Mech. Eng. **299**, 337–365 (2016)
31. J. Gravesen, A. Evgrafov, D.-M. Nguyen, P. Nørtoft, Planar parametrization in isogeometric analysis, in *Mathematical Methods for Curves and Surfaces: 8th International Conference, MMCS 2012, Oslo, June 28–July 3, 2012, Revised Selected Papers 8* (Springer, Berlin, 2014), pp. 189–212
32. Ch. Heinrich, B. Simeon, St. Boschert, A finite volume method on NURBS geometries and its application in isogeometric fluid–structure interaction. Math. Comput. Simul. **82**(9), 1645–1666 (2012)
33. A.J. Herrema, E.L. Johnson, D. Proserpio, M.C.H. Wu, J. Kiendl, M.-C. Hsu, Penalty coupling of non-matching isogeometric Kirchhoff–Love shell patches with application to composite wind turbine blades. Comput. Methods Appl. Mech. Eng. **346**, 810–840 (2019)
34. J.P. Hinz, PDE-Based Parameterization Techniques for Isogeometric Analysis Applications. PhD thesis, Delft University of Technology (2020)
35. J. Hinz, M. Möller, C. Vuik, Elliptic grid generation techniques in the framework of isogeometric analysis applications. Comput. Aided Geom. Des. **65**, 48–75 (2018)
36. J. Hinz, M. Möller, C. Vuik, Spline-based parameterization techniques for twin-screw machine geometries, in *IOP Conference Series: Materials Science and Engineering*, vol. 425 (IOP Publishing, Bristol, 2018), p. 012030
37. K. Hormann, G. Greiner, MIPS: an efficient global parametrization method. Technical report, Erlangen-Nürnberg University (Germany) Computer Graphics Group (2000)
38. T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. **194**(39), 4135–4195 (2005)
39. K.C. Hui, Y.-B. Wu, Feature-based decomposition of trimmed surface. Comput. Aided Des. **37**(8), 859–867 (2005). CAD '04 Special Issue: Modelling and Geometry Representations for CAD
40. Y. Ji, Y.-Y. Yu, M.-Y. Wang, C.-G. Zhu, Constructing high-quality planar NURBS parameterization for isogeometric analysis by adjustment control points and weights. J. Comput. Appl. Math. **396**, 113615 (2021)
41. Y. Ji, M.-Y. Wang, M.-D. Pan, Y. Zhang, C.-G. Zhu, Penalty function-based volumetric parameterization method for isogeometric analysis. Comput. Aided Geom. Des. **94**, 102081 (2022)
42. B. Jüttler, U. Langer, A. Mantzaflaris, S.E. Moore, W. Zulehner, Geometry + simulation modules: implementing isogeometric analysis. PAMM **14**(1), 961–962 (2014)
43. H. Kang, F. Chen, J. Deng, Modified T-splines. Comput. Aided Geom. Des. **30**(9), 827–843 (2013)
44. J. Kiendl, Isogeometric Analysis and Shape Optimal Design of Shell Structures. PhD thesis, Technische Universität München (2011)
45. J. Kiendl, K.-U. Bletzinger, J. Linhard, R. Wüchner, Isogeometric shell analysis with Kirchhoff–Love elements. Comput. Methods Appl. Mech. Eng. **198**(49–52), 3902–3914 (2009)
46. J. Kiendl, M.-C. Hsu, M.C.H. Wu, A. Reali, Isogeometric Kirchhoff–Love shell formulations for general hyperelastic materials. Comput. Methods Appl. Mech. Eng. **291**, 280–303 (2015)
47. H. Kneser, Lösung der Aufgabe 41. Jber. Deutsch. Math.-Verein. **35**, 123–124 (1926)
48. W.F. Lam, C.T. Morley, Arc-length method for passing limit points in structural calculation. J. Struct. Eng. **118**(1), 169–185 (1992)
49. X. Li, M.A. Scott, Analysis-suitable T-splines: characterization, refineability, and approximation. Math. Models Methods Appl. Sci. **24**(06), 1141–1164 (2014)
50. X. Li, T.W. Sederberg, S-splines: a simple surface solution for IGA and CAD. Comput. Methods Appl. Mech. Eng. **350**, 664–678 (2019)

51. X. Li, J. Zhang, AS++ T-splines: linear independence and approximation. Comput. Methods Appl. Mech. Eng. **333**, 462–474 (2018)
52. L. Liu, Y.J. Zhang, X. Wei, Weighted T-splines with application in reparameterizing trimmed NURBS surfaces. Comput. Methods Appl. Mech. Eng. **295**, 108–126 (2015)
53. H. Liu, Y. Yang, Y. Liu, X.-M. Fu, Simultaneous interior and boundary optimization of volumetric domain parameterizations for IGA. Comput. Aided Geom. Des. **79**, 101853 (2020)
54. M. Marsala, A. Mantzaflaris, B. Mourrain, G1 – Smooth biquintic approximation of Catmull-Clark subdivision surfaces. Comput. Aided Geom. Des. **99**, 102158 (2022)
55. T. Martin, E. Cohen, R.M. Kirby, Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Comput. Aided Geom. Des. **26**(6), 648–664 (2009)
56. B. Marussig, J. Zechner, G. Beer, T.-P. Fries, Stable isogeometric analysis of trimmed geometries. Comput. Methods Appl. Mech. Eng. **316**, 497–521 (2017). Special Issue on Isogeometric Analysis: Progress and Challenges
57. X. Meng, G. Hu, A NURBS-enhanced finite volume solver for steady Euler equations. J. Comput. Phys. **359**, 77–92 (2018)
58. X. Meng, Y. Gu, G. Hu, A fourth-order unstructured NURBS-enhanced finite volume WENO scheme for steady Euler equations in curved geometries. Commun. Appl. Math. Comput. **5**(1), 315–342 (2021)
59. M. Möller, J. Hinz, Isogeometric analysis framework for the numerical simulation of rotary screw machines. I. general concept and early applications. IOP Conf. Ser. Mat. Sci. Eng. **425**(1), 012032 (2018)
60. T. Lyche, K. Mørken, Spline Methods. Lecture notes from the Department of Mathematics, University of Oslo (2018). https://www.uio.no/studier/emner/matnat/math/MAT4170/v18/pensumliste/splinebook-2018.pdf
61. T. Nguyen, B. Jüttler, Parameterization of contractible domains using sequences of harmonic maps, in *International Conference on Curves and Surfaces* (Springer, Berlin, 2010), pp. 501–514
62. X. Nian, F.-L. Chen, Planar domain parameterization for isogeometric analysis based on Teichmüller mapping. Comput. Methods Appl. Mech. Eng. **311**, 41–55 (2016)
63. M.-D. Pan, F.-L. Chen, Low-rank parameterization of volumetric domains for isogeometric analysis. Comput. Aided Des. **114**, 82–90 (2019)
64. M.-D. Pan, F.-L. Chen, W.-H. Tong, Low-rank parameterization of planar domains for isogeometric analysis. Comput. Aided Geom. Des. **63**, 1–16 (2018)
65. M.-D. Pan, F.-L. Chen, W.-H. Tong, Volumetric spline parameterization for isogeometric analysis. Comput. Methods Appl. Mech. Eng. **359**, 112769 (2020)
66. L. Piegl, W. Tiller, *The NURBS Book* (Springer, Berlin, 1995)
67. E. Pilgerstorfer, B. Jüttler, Bounding the influence of domain parameterization and knot spacing on numerical stability in isogeometric analysis. Comput. Methods Appl. Mech. Eng. **268**, 589–613 (2014)
68. C.G. Provatidis, *Precursors of Isogeometric Analysis* (Springer International Publishing, Berlin, 2019)
69. T. Rado, Aufgabe 41. Jber. Deutsch. Math.-Verein. **35**, 49 (1926)
70. U. Reif, A refineable space of smooth spline surfaces of arbitrary topological genus. J. Approximation Theory **90**(2), 174–199 (1997)
71. I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. Q. Appl. Math. **4**, 45–99 and 112–141 (1946)
72. T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs. ACM Trans. Graph. **22**(3), 477–484 (2003)
73. R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced finite element method (NEFEM). Int. J. Numer. Methods Eng. **76**(1), 56–83 (2008)
74. R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced finite element method (NEFEM). Arch. Comput. Methods Eng. **18**(4), 441–484 (2011)
75. A. Shamanskiy, M.H. Gfrerer, J. Hinz, B. Simeon, Isogeometric parametrization inspired by large elastic deformation. Comput. Methods Appl. Mech. Eng. **363**, 112920 (2020)

76. J.-P. Su, X.-M. Fu, L.-G. Liu, Practical foldover-free volumetric mapping construction, in *Computer Graphics Forum*, vol. 38 (Wiley Online Library, Hoboken, 2019), pp. 287–297
77. T. Takacs, D. Toshniwal, Almost-C1 splines: Biquadratic splines on unstructured quadrilateral meshes and their application to fourth order problems. Comput. Methods Appl. Mech. Eng. **403**, 115640 (2023)
78. D.C. Thomas, L. Engvall, S.K. Schmidt, K. Tew, M.A. Scott, U-splines: splines over unstructured meshes. Comput. Methods Appl. Mech. Eng. **401**, 115515 (2022)
79. D. Toshniwal, H. Speleers, R.R. Hiemstra, T.J.R. Hughes, Multi-degree smooth polar splines: a framework for geometric modeling and isogeometric analysis. Comput. Methods Appl. Mech. Eng. **316**, 1005–1061 (2017)
80. D. Toshniwal, H. Speleers, T.J.R. Hughes, Smooth cubic spline spaces on unstructured quadrilateral meshes with particular emphasis on extraordinary points: geometric design and isogeometric analysis considerations. Comput. Methods Appl. Mech. Eng. **327**, 411–458 (2017)
81. H.M. Verhelst, M. Möller, J.H. Den Besten, F.J. Vermolen, M.L. Kaminski, Equilibrium path analysis including bifurcations with an arc-length method avoiding a priori perturbations, in *Proceedings of ENUMATH2019 Conference* (2020)
82. H.M. Verhelst, M. Möller, J.H. Den Besten, A. Mantzaflaris, M.L. Kaminski, Stretch-based hyperelastic material formulations for Isogeometric Kirchhoff–Love Shells with application to wrinkling. Comput. Aided Des. **139**, 103075 (2021)
83. H.M. Verhelst, A. Mantzaflaris, M. Möller, J.H. Den Besten, Goal-adaptive meshing of isogeometric Kirchhoff-Love shells. arXiv:2307.08356 (2023)
84. X. Wang, W. Ma, Smooth analysis-suitable parameterization based on a weighted and modified Liao functional. Comput. Aided Des. **140**, 103079 (2021)
85. X. Wang, X. Qian, An optimization approach for constructing trivariate B-spline solids. Comput. Aided Des. **46**, 179–191 (2014)
86. B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, E. Rank, Integrating CAD and numerical analysis: 'dirty geometry' handling using the finite cell method. Comput. Methods Appl. Mech. Eng. **351**, 808–835 (2019)
87. X. Wei, Y. Zhang, L. Liu, T.J.R. Hughes, Truncated T-splines: fundamentals and methods. Comput. Methods Appl. Mech. Eng. **316**, 349–372 (2017)
88. J. Weingarten, Über eine Klasse auf einander abwickelbarer Flächen. J. Reinen Angew. Math. **1861**(59), 382–393 (1861)
89. P. Weinmüller, T. Takacs, Construction of approximate C1 bases for isogeometric analysis on two-patch domains. Comput. Methods Appl. Mech. Eng. **385**, 114017 (2021)
90. P. Weinmüller, T. Takacs, An approximate C1 multi-patch space for isogeometric analysis with a comparison to Nitsche's method. Comput. Methods Appl. Mech. Eng. **401**(Part B), 115592 (2022). ISSN 0045–7825. https://doi.org/10.1016/j.cma.2022.115592
91. G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Parameterization of computational domain in isogeometric analysis: methods and comparison. Comput. Methods Appl. Mech. Eng. **200**(23–24), 2021–2031 (2011)
92. G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Constructing analysis-suitable parameterization of computational domain from CAD boundary by variational harmonic method. J. Comput. Phys. **252**, 275–289 (2013)
93. G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Optimal analysis-aware parameterization of computational domain in 3D isogeometric analysis. Comput. Aided Des. **45**(4), 812–821 (2013)
94. G. Xu, T.-H. Kwok, C.C.L. Wang, Isogeometric computation reuse method for complex objects with topology-consistent volumetric parameterization. Comput. Aided Des. **91**, 1–13 (2017)
95. L. Zhang, A. Gerstenberger, X. Wang, W.K. Liu, Immersed finite element method. Comput. Methods Appl. Mech. Eng. **193**(21), 2051–2067 (2004). Flow Simulation and Modeling
96. Y. Zheng, F.-L. Chen, Volumetric parameterization with truncated hierarchical B-splines for isogeometric analysis. Comput. Methods Appl. Mech. Eng. **401**, 115662 (2022)
97. Z. Zhou, D.W. Murray, An incremental solution technique for unstable equilibrium paths of shell structures. Comput. Struct. **55**(5), 749–759 (1995)