Unstructured Transfinite Elements Including T-Splines

Christopher G. Provatidis *

Posted Date: 20 October 2025

doi: 10.20944/preprints202510.1500.v1

Keywords: transfinite elements; B-spline functions; T-splines; macro-elements; unstructured grids; partition of unity; isogeometric analysis; finite element method

 doi:10.20944/preprints202510.1500.v1

Article

Unstructured Transfinite Elements Including T-Splines

Christopher G. Provatidis

National Technical University of Athens, 9 Iroon Polytechniou Str., 15780 Athens, Greece; cprovat@mail.ntua.gr; Tel.: +30-210-772-1520

Abstract

This paper presents a unified framework for constructing partially unstructured B-spline transfinite finite elements with arbitrary nodal distributions. Four novel distinct classes of elements are investigated. The first consists of classical transfinite elements reformulated using B-spline basis functions. The second includes elements defined by arbitrary control point networks arranged in parallel layers along one direction. The third features arbitrarily placed boundary nodes combined with a tensor-product structure in the interior. The fourth class comprises classical T-spline elements, characterized by selectively omitted internal nodes that produce sparsely populated, incomplete grids. For all four classes, novel macro-element formulations are introduced, enabling flexible and customizable nodal configurations while preserving the partition of unity property. The key innovation lies in reinterpreting the generalized coefficients as discrete samples of an underlying continuous univariate function, which is independently approximated at each station in the T-index space. This perspective generalizes the classical transfinite interpolation by allowing both the blending functions and the univariate trial functions to be defined using non-cardinal bases such as Bernstein polynomials or B-splines, offering enhanced adaptability for complex geometries and nonuniform node layouts.

Keywords: transfinite elements; B-spline functions; T-splines; macro-elements; unstructured grids; partition of unity; isogeometric analysis; finite element method

MSC: 41A10; 41A15; 65N30

1. Introduction

Computational methods have undergone several stages of development and continue to evolve. The progression began with global approximation techniques, such as those introduced by Rayleigh [1], Ritz [2], and the Bubnov-Galerkin method (1913-1915), eventually giving way to more localized approaches like the finite element method [3,4]. Among these developments, transfinite interpolation occupies a particularly important role [5,6]. Initially developed to interpolate the geometry of solid objects and surfaces [7], it soon became apparent that the method could also be used to interpolate scalar and even vector fields within three-dimensional domains. Notably, transfinite interpolation allows the construction of finite elements with differing numbers of nodes along opposing edges or surfaces–without requiring transitional elements. This flexibility extends to the merging of dissimilar domains in two or three dimensions.

An effort to integrate ideas from computer-aided geometric design (CAGD) into analysis methods—such as the finite element, boundary element, and collocation methods—was initiated at the National Technical University of Athens (NTUA) in the summer of 1984, within the research group to which the present author belonged; the first publication appeared later in 1989 [8]. A comprehensive monograph, compiled from approximately 60 journal and conference papers produced by this group, along with an extensive literature review until 2018 (here omitted, for the sake of brevity), is presented in reference [9]. In that monograph, the term *macroelement* is used to describe an entire patch, whether a surface or a volumetric block. In this framework, although B-splines required domain decomposition at individual breakpoints [8], they were initially regarded more as integration cells than as B-spline

 doi:10.20944/preprints202510.1500.v1

elements in the modern sense [10]. Concurrently, similar ideas were being explored by other research groups, as documented in [11–15].

Classical literature on transfinite interpolation (see Refs. [5,6] and references therein) primarily employs linear blending functions, typically Lagrange polynomials. The univariate trial functions defined along the mesh-lines (also referred to as 'stations') are commonly chosen as *piecewise linear*, *piecewise Hermite*, or *cubic splines*. However, in the case of cubic splines, specific implementation details are generally omitted or not fully addressed.

While transfinite interpolation works perfectly well in conjunction with Lagrange polynomials, there is not much experience when it is combined with Bernstein polynomials and B-splines. This is because the former are cardinal polynomials of $[0,1]$-type, in the sense they are interpolatory (take the value of unity) along the stations, whereas the latter are not (take a value less than unity). Nevertheless, both sets hold the partition of unity property. This fact has recently sustained the *conjecture* that transfinite interpolation can be extended in such a way that blending functions can be Bernstein polynomials or B-splines and at the same time they can also be trial functions which interpolate the primary variable U along the stations of the curvilinear patch [16].

Figure 1 presents a collection of multiple classes of transfinite finite elements, starting from the older Coons element (studied in detail in [8,9]) and ending to a multi-layer element. The well-known case of the T-mesh element [17–19] —which is a variation of cases (d,e)– is still missing in this collection, but will be extensively discussed later in Sect. 6.

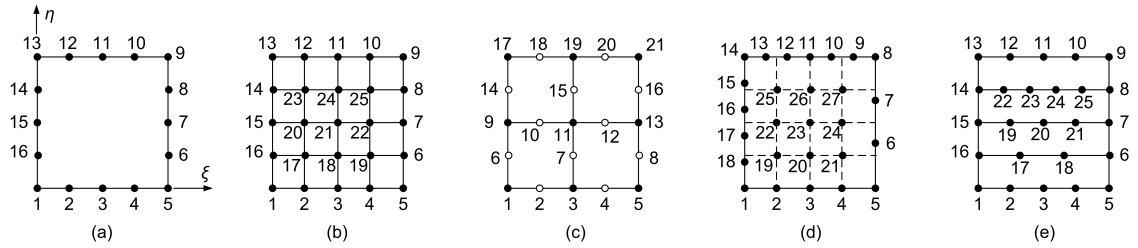


Figure 1. (**a**) Coons element. (**b**) Tensor-product element. (**c**) Classical transfinite element. (**d**) Arbitrary-noded boundary with inner tensor-product element. (**e**) One-directional element.

Recent studies have shown that a direct substitution of Lagrange polynomials with Bernstein polynomials in the final expressions of the shape functions for all types of transfinite elements is feasible, and sometimes yields equivalent results for both polynomial sets. To date, this direct replacement–accompanied by a pointwise agreement between the two polynomial families–has been verified for the following four classes of transfinite elements [20,21]:

- Coons elements (Figure 1a).
- Tensor product elements (Figure 1b).
- Classical transfinite elements of structured pattern (Figure 1c).
- Deformed tensor-product elements (not shown; similar to Figure 1b, but with boundary nodes arbitrarily positioned relative to the orthogonal projection of the internal nodes).

While the primary variable U is a straightforward physical quantity to interpret, the associated discrete generalized coefficients often cause confusion—particularly among engineers rather than applied mathematicians. In general, however, the coefficients a_i form a set of dual quantities corresponding to the nodal values U_i, and the two sets are interconnected through a linear relationship.

First of all, we must report that the expression of tensor-product Bernstein polynomials is not an axiom but a direct sequence of the pre-existed tensor-product of Lagrange polynomials. Clearly, the existing linear relationship between univariate Lagrange and Bernstein polynomials –which share the same monomials– is adequate to offer a mathematical proof for the equivalence between tensor-product Lagrange and tensor-product (non-rational) Bernstein polynomials [9].

Moreover, having established the above explanation for the tensor product of Bernstein polynomials, the interpretation of the tensor-product B-spline follows as a straightforward extension. However,

 doi:10.20944/preprints202510.1500.v1

while the rationale behind the tensor-product B-spline is relatively easy to grasp, the same cannot be said for transfinite interpolation, which—even to this day—remains a powerful and versatile tool for constructing novel, large-scale finite elements, as will be demonstrated in the following sections.

The difficulty and delay in advancing the transfinite interpolation method stem from the following issue. Although replacing Lagrange polynomials with Bernstein polynomials is generally straightforward, the challenge lies in substituting the univariate functions $U(\bar{\xi}, \eta)$ along each section (station) orthogonal to the ξ-axis at point $\bar{\xi}$. This is due to the fact that *all* the generalized coefficients a_i associated with the two opposite sides (i.e., AB and DC) of the quadrilateral patch $ABCD$ (Figure 2b), which are perpendicular to the station under consideration (thus containing its ends), *contribute* to the determination of function $U(\bar{\xi}, \eta)$. Therefore, the classical cardinal blending functions of Lagrange type cannot be directly combined with standard B-splines along the stations, simply because the latter are of global character, that is, it is *not* possible to interpolate the function $U(\bar{\xi}, \eta)$ through a *self-contained* expression (trial functions) along a $\bar{\xi}$-station. Addressing this incompatibility is the central objective of the present study.

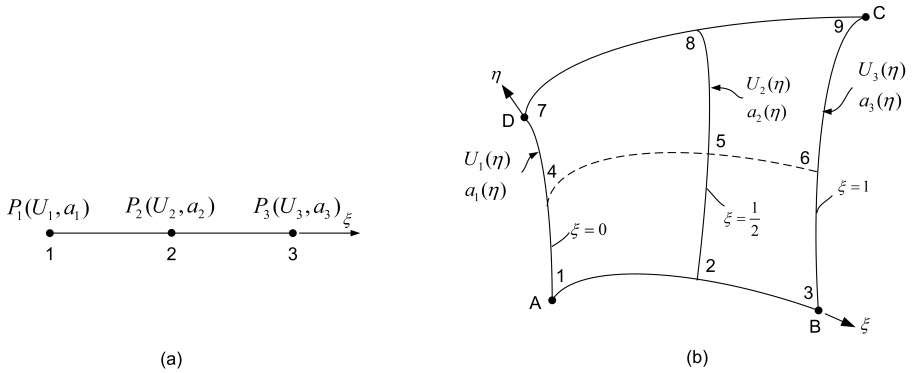


Figure 2. (**a**) One-dimensional curve. (**b**) Two-dimensional patch.

Aiming to address the above issue, a recent preliminary study on transfinite elements has indicated that the classical blending functions –which usually are *cardinal* functions of [1,0]-type– can be substituted by non-cardinal functions such as Bernstein polynomials and B-splines [16]. The beginning of this inspiration is the 2D analogue of the 1D interpolation using Lagrange polynomials from one side and any other IGA-based functions such as Bernstein polynomials, B-splines, and NURBS. The lowest level of this concept is to think about the quadratic interpolation, which can be implemented either using Lagrange polynomials: $U(\xi) = L_{1,2}(\xi)U_1 + L_{2,2}(\xi)U_2 + L_{3,2}(\xi)U_2$ or using Bernstein polynomials: $U(\xi) = B_{1,2}(\xi)a_1 + B_{2,2}(\xi)a_2 + B_{3,2}(\xi)a_2$. The issue is as follows: It is quite reasonable that the aforementioned three Lagrange polynomials may serve the role of blending (interpolation) functions in the ξ-direction since they fullfil the delta-Kronecker property ($L_i(\xi_j) = \delta_{ij}$) and hold the partition the unity property. Therefore, they can accurately represent a constant, a linear, and a quadratic univariate function. Similarly, if the generalized coefficients (a_1, a_2, a_3) are properly chosen each time, the Bernstein polynomials may again accurately represent the same functions (constant, linear, and quadratic).

Clearly, when the blending functions $E_i(\xi)$ are Bernstein polynomials (or B-splines), the variation in the η-direction should be performed in terms of a (supposed existing) univariate function $a(\eta)$, which for $\eta = 0$ (i.e., on edge AB) coincides with the usual generalized coefficients (a_1, a_2, a_3) at points (ξ_1, ξ_2, ξ_3), respectively. The same holds for a triplet (ξ_1', ξ_2', ξ_3') on the opposite edge DC at $\eta = 1$.

The critical point is how to describe the abovementioned function $a(\bar{\xi}_i, \eta)$, which is perpendicularly oriented at point $\bar{\xi}_i$. Within the IGA framework, the most easy way is to employ a set of B-spline functions extended in the interval $\eta \in [0, 1]$. To this purpose, a safe way is to use clamped splines (for degree $p = 3$: $\Xi = [0, 0, 0, 0, \ldots, 1, 1, 1, 1]$), whose the end values will be the boundary variable a (and not U). In this way, the previously mentioned difficulty in handling $a(\eta)$ is removed.

 doi:10.20944/preprints202510.1500.v1

The paper is divided into two main parts. The first part focuses on the construction and performance of simplified elements, both classical and newly introduced, including those composed of layers oriented in a single direction. This serves as a pilot study to validate the approach of treating the discrete coefficients as a bivariate function. The second part addresses the well-known T-spline elements.

The present paper is structured as follows: Section 2 introduces the general formulation of transfinite interpolation, employing either Lagrange or Bernstein polynomials, as well as B-splines, for the representation of blending and trial functions. Section 3 describes the construction of classical transfinite elements. Section 4 presents the development of transfinite elements with nodes arranged in parallel layers. Section 5 focuses on elements that combine tensor-product interior nodes with nonuniform boundary node placement. Section 6 addresses T-spline elements of increasing complexity. Section 7 discusses software-related aspects. Section 8 provides numerical results for all models, Section 9 briefly refers to possible implementation of the collocation method as an alternative, and Section 10 offers a detailed discussion. The paper is supplemented by seven appendices.

2. Transfinite Interpolation Using B-Splines

2.1. Terminology

In classical computer-aided geometric design (CAGD) [22], the term *blending function* typically refers to univariate interpolation functions used in a specific parametric direction. For instance, consider a Coons patch defined over the quadrilateral $ABCD$, with the origin at vertex A and the ξ-axis oriented along edge AB (Figure 2b) [7]. The blending functions $E_1(\xi)$ and $E_2(\xi)$ interpolate between the boundary functions $U_{AD}(\eta)$ and $U_{BC}(\eta)$ in the ξ-direction.

This usage of the term differs from its meaning in T-spline literature, where blending functions are often synonymous with basis functions [18]. In this paper, we adopt the classical interpretation: the term blending function will refer specifically to the interpolation functions used in Coons or Gordon patch construction, as adopted in Refs. [5–7,22]. Throughout the present paper, the term *basis functions* will be used uniformly for both B-splines/NURBS and T-splines.

A second key concept in classical transfinite interpolation theory–as well as in finite element methodology, as emphasized in Zienkiewicz's seminal work–is that of *trial functions* [3, p. 21]. In the present study, this term refers to univariate functions defined along boundary edges, or more generally, along horizontal and vertical mesh-lines (stations) within the domain.

A third issue concerns the use of the terms *transfinite element* and *macroelement*. In this paper, these terms are used synonymously to refer to the *parametric patch* defined over the domain $0 \leq \xi, \eta \leq 1$. Regardless of the choice of blending or trial functions, the resulting basis functions share the *same* closed-form expression. The main difference lies in the interpolation approach: Lagrange and Bernstein polynomials perform global interpolation over the entire patch, whereas B-splines use piecewise polynomial interpolation between breakpoints, which define the *integration cells*. Naturally, in the B-spline formulation, not all degrees of freedom (DOFs) are active within every integration cell. However, to maintain a uniform treatment across all three cases (Lagrange, Bernstein, and B-splines), we do not emphasize this distinction.

2.2. State of the Art

The general expression of transfinite interpolation for 2D patches is given in [5]:

$$\begin{aligned} U(\xi,\eta) &= P_\xi \oplus P_\eta \\ &= P_\xi\{U\} + P_\eta\{U\} - P_{\xi\eta}\{U\}\,, \end{aligned} \tag{1}$$

where $P_\xi\{U\}$ and $P_\eta\{U\}$ are projectors of the quantity U along the parametric axes ξ and η, respectively, and $P_{\xi\eta}\{U\}$ is the corrective projector, defined as the tensor product of nodal values at the intersections of the coordinate stations.

 doi:10.20944/preprints202510.1500.v1

It is well known that, for vertical stations passing through $\boldsymbol{\xi} = \xi_1 = 0, \ldots, \xi_{m+1} = 1$, and for horizontal stations located at $\boldsymbol{\eta} = \eta_1 = 0, \ldots, \eta_{n+1} = 1$, we construct univariate blending functions $E_i(\xi)$ and $E_j(\eta)$ of degrees m and n, respectively (assuming Lagrange polynomials are used). Based on the univariate functions $U(\xi_i, \eta)$ and $U(\xi, \eta_j)$ defined along the vertical and horizontal stations, respectively, we obtain:

$$P_\xi\{U\} = \sum_{i=1}^{m+1} E_i(\xi) U(\xi_i, \eta), \tag{2}$$

$$P_\eta\{U\} = \sum_{j=1}^{n+1} E_j(\eta) U(\xi, \eta_j), \tag{3}$$

$$P_{\xi\eta}\{U\} = \sum_{j=1}^{n+1} \sum_{i=1}^{m+1} E_i(\xi) E_j(\eta) U(\xi_i, \eta_j)\,. \tag{4}$$

On the other hand, along each station, the variable $U(\xi, \eta)$ is interpolated using trial functions $\tilde{B}_i(s)$, where the variable s denotes either of the parametric directions ξ or η, as follows:

$$\text{Horizontal station:} \quad U(\xi, \bar{\eta}) = \sum_{i=1}^{\tilde{m}+1} \tilde{B}_i(\xi) U(\xi_i, \bar{\eta}), \tag{5}$$

$$\text{Vertical station:} \quad U(\bar{\xi}, \eta) = \sum_{j=1}^{\tilde{n}+1} \tilde{B}_j(\eta) U(\bar{\xi}, \eta_j)\,. \tag{6}$$

Here, the overbar denotes a prescribed value of a given parameter (ξ or η) along the corresponding station—vertical for $\bar{\xi}$ or horizontal for $\bar{\eta}$. In general, the number of nodes ($\tilde{m}+1$ or $\tilde{n}+1$) along a station may differ—either smaller or greater—from the number of stations in the same direction. The case of equality (e.g., in a tensor-product configuration) is also included.

Henceforth, for brevity, the notation $\{U\}$ following each projector is suppressed; for example, $P_\xi\{U\}$ is written simply as P_ξ, and similarly for the others.

2.3. Extension of the Projectors Using Bernstein Polynomials

The classical interpretation of the well-known projectors (P_ξ, P_η, and $P_{\xi\eta}$) is founded on their application to the primary variable $U(\xi, \eta)$, in accordance with the use of Lagrange polynomials as blending functions $E_i(\xi)$ and $E_j(\eta)$, as presented in Ref. [5]. It is well established that Coons interpolation represents a direct extension of one-dimensional linear interpolation to two-dimensional domains [9]. Similarly, classical transfinite (Gordon) interpolation generalizes one-dimensional polynomial interpolation to bivariate patches. Since one-dimensional polynomial interpolation can be formulated using either Lagrange or Bernstein polynomials [23], it follows that the classical projectors may likewise be expressed in terms of Bernstein polynomials.

The above claim will be demonstrated—in full detail for the first time—for one projector, namely P_ξ; the same holds for the remaining ones. Let us consider three points $P_1(\xi_1)$, $P_2(\xi_2)$, and $P_3(\xi_3)$ on a parametric curve $C(\xi)$ and the associated values U_1, U_2, U_3 of a univariate function $U(\xi)$ (Figure 2a). These can be interpolated in terms of the nodal values (U_1, U_2, U_3), using the quadratic Lagrange polynomials $E_1(\xi) = L_{1,2}(\xi)$, $E_2(\xi) = L_{2,2}(\xi)$, $E_3(\xi) = L_{3,2}(\xi)$:

$$U(\xi) = E_1(\xi) U_1 + E_2(\xi) U_2 + E_3(\xi) U_3\,. \tag{7}$$

Alternatively, the same points can be interpolated in terms of the generalized coefficients (a_1, a_2, a_3) using the quadratic Bernstein polynomials $\tilde{E}_1(\xi) = B_{1,2}(\xi)$, $\tilde{E}_2(\xi) = B_{2,2}(\xi)$, and $\tilde{E}_3(\xi) = B_{3,2}(\xi)$:

$$U(\xi) = \tilde{E}_1(\xi) a_1 + \tilde{E}_2(\xi) a_2 + \tilde{E}_3(\xi) a_3\,. \tag{8}$$

 doi:10.20944/preprints202510.1500.v1

The above equations can be extended from one to two dimensions, beginning with the projector P_ξ, in which the nodal values (U_1, U_2, U_3) are replaced by univariate functions $(U_1(\eta), U_2(\eta), U_3(\eta))$, as illustrated in Figure 2b:

$$P_\xi\{U\} = E_1(\xi)U_1(\eta) + E_2(\xi)U_2(\eta) + E_3(\xi)U_3(\eta)\,. \tag{9}$$

Similarly, in terms of Bernstein polynomials, the same projector is expressed in the form of a separation of variables, as follows:

$$P_\xi\{a\} = \tilde{E}_1(\xi)a_1(\eta) + \tilde{E}_2(\xi)a_2(\eta) + \tilde{E}_3(\xi)a_3(\eta)\,. \tag{10}$$

Of course, a significant distinction exists between the two formulations presented above. The Lagrange interpolation (Eq. (9)) is a local scheme applied to univariate functions that directly represent the primary variable $U(\eta)$ at discrete stations located at $\xi = 0, \frac{1}{2}, 1$. In contrast, the Bernstein interpolation (Equation (10)) is a global approach, involving univariate functions $a(\eta)$ constructed from generalized coefficients.

This implies that while the interpolation of the primary variable U is straightforward and intuitive, the corresponding interpolation of the coefficient a is less direct. Nevertheless, the coefficient a embodies a definitive quantity that, in essence, is not fundamentally different from the variable U. For instance, in the case of quadratic interpolation, we observe:

$$a_1 = U_1, \quad a_2 = -\frac{1}{2}U_1 + 2U_2 + \frac{1}{2}U_3, \quad a_3 = U_3\,. \tag{11}$$

Therefore, if—for instance—the extreme values are equal (i.e., $U_1 = U_3$), the coefficient a_2 corresponds to $2U_2$. This implies that a duplication of the associated basis function can be employed to ensure that a_2 effectively represents the actual variable U_2.

Numerical investigations have confirmed that this concept—promoting the use of Bernstein polynomials, and more generally B-splines, as blending functions—yields robust performance, provided that the remaining two projectors (P_ξ, P_η) are incorporated in a consistent manner [9,16].

2.4. B-Splines Projectors

A comprehensive investigation into the use of B-splines as blending functions was recently presented in Ref. [16], although the treatment therein was primarily speculative. In contrast, the methodology introduced in this paper adopts a more rigorous framework and may be interpreted in analogy with a function of separated variables, namely $U(\xi, \eta) = X(\xi)Y(\eta)$, where the role of $Y(\eta)$ is assumed by the function $a(\eta)$, which is intrinsically linked to the generalized coordinates.

In general, the number of mesh lines (stations) along each parametric direction determines the length of the corresponding knot vector used for the blending functions. Likewise, the number of control points defined at each station governs the length of the associated knot vector for the trial functions. Further clarification and illustrative examples will be provided in the subsequent sections.

3. Construction of Classical B-Spline Transfinite Elements

A representative transfinite element $ABCD$, comprising 113 control points, is depicted in Figure 3d. It can be observed that the element is structured with four horizontal and five vertical stations (including the boundary edges), upon which the control points are distributed. The horizontal set of stations is oriented perpendicular to the η-axis and positioned at $\eta = 0, \frac{1}{3}, \frac{2}{3}, 1$, while the vertical set is oriented perpendicular to the ξ-axis and located at $\xi = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1$.

 doi:10.20944/preprints202510.1500.v1

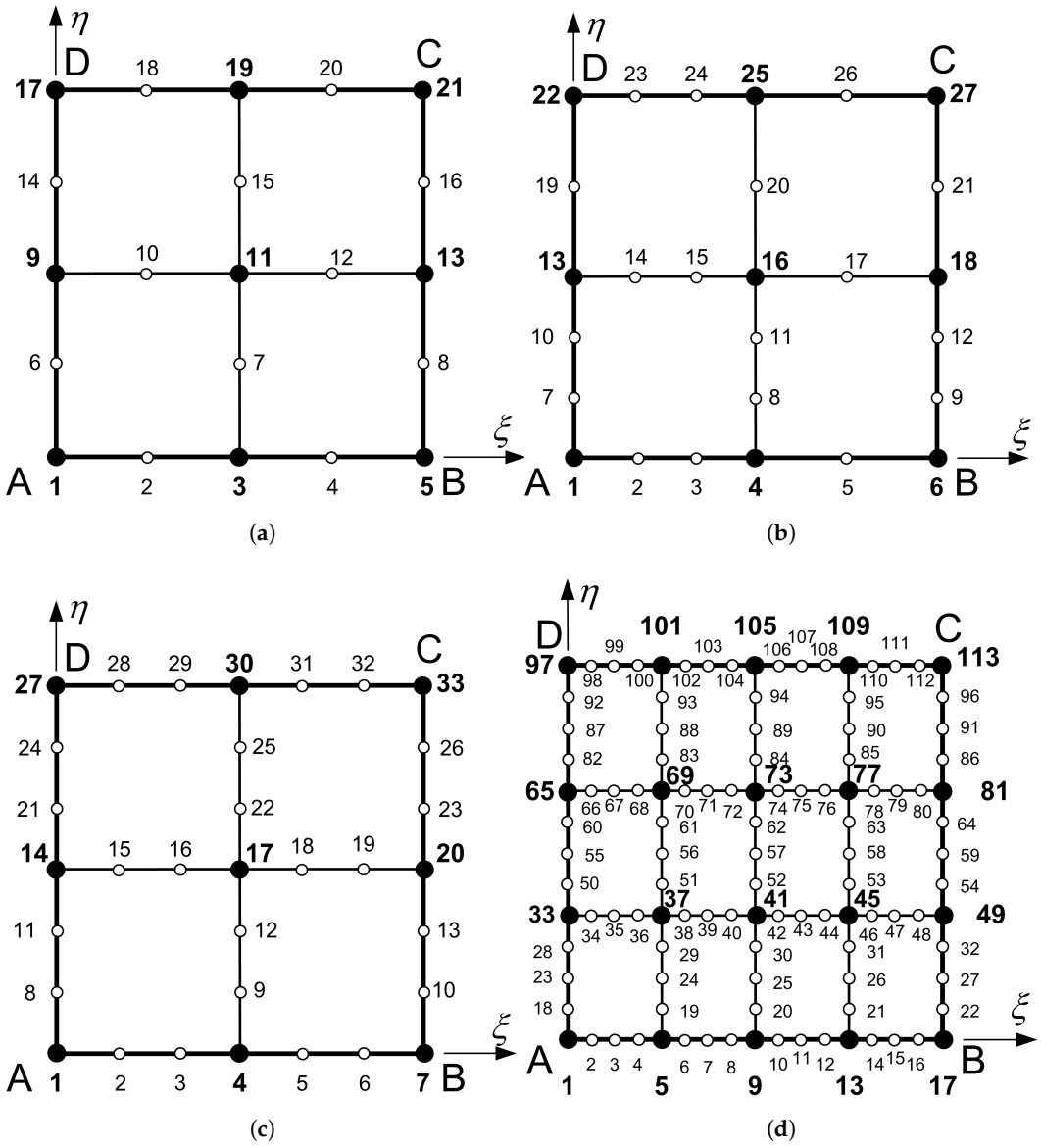


Figure 3. A collection of classical transfinite elements: (**a**) 21-node element. (**b**) 27-node element. (**c**) 33-node element. (**d**) 113-node element.

Based on the aforementioned station configuration, the blending functions in the ξ-direction may be chosen as Lagrange or Bernstein polynomials of degree 4, corresponding to five station values. Similarly, in the η-direction, the blending functions may be Lagrange or Bernstein polynomials of degree 3, associated with four station values.

Alternatively, when employing cubic B-splines, the ξ-direction can be represented using the knot vector

$$\Xi_{\text{blend}} = [0, 0, 0, 0, \tfrac{1}{2}, 1, 1, 1, 1],$$

which corresponds to five distinct station values and ensures C^2 continuity. In the η-direction, the blending functions may be defined by the knot vector

$$H_{\text{blend}} = [0, 0, 0, 0, 1, 1, 1, 1],$$

which corresponds to four station values and yields Bernstein polynomials of degree 3.

In all cases, the construction of the knot vectors for the blending functions must guarantee that the number of control points aligns with the number of stations, thereby ensuring proper interpolation and continuity across the transfinite element.

 doi:10.20944/preprints202510.1500.v1

Under these conditions, the three projectors defined by Eqs. (2) through (4) are now reformulated in terms of B-spline-based blending and trial functions, where the trial functions are associated with the generalized coefficients a_i.

Upon further elaboration, it is found that the Boolean sum in Equation (1) yields a total of 113 basis functions. These functions can be categorized into two distinct groups, as follows:

1. **Nodes at section intersections** (including corner nodes A, B, C, and D), for example:

$$\psi_{69}(\xi,\eta) = E_2(\xi)N_{9,3}(\eta) + E_3(\eta)N_{5,3}(\xi) - E_2(\xi)E_3(\eta)\,. \tag{12}$$

   In this expression, $E_2(\xi)$ denotes the second blending function in the ξ-direction, corresponding to the second vertical station where node 69 is located. Likewise, $N_{9,3}(\eta)$ refers to the 9th B-spline basis function in the η-direction, as node 69 is the 9th node from the bottom. The same logic applies to the remaining terms.
2. **Intermediate nodes on single stations**, for example:

$$\psi_{71}(\xi,\eta) = E_3(\eta)N_{7,3}(\xi)\,. \tag{13}$$

   Here, $E_3(\eta)$ is the third blending function in the η-direction, indicating that node 71 lies on the third horizontal station. Similarly, $N_{7,3}(\xi)$ represents the 7th B-spline basis function in the ξ-direction, as node 71 is the 7th node from the left boundary.

By classifying the 113 degrees of freedom into the two aforementioned categories, the complete set of basis functions can be systematically constructed. Specifically, 20 of these functions—corresponding to the black-filled circles in Figure 3d—are composed of three terms, as exemplified by Equation (12), and reflect contributions from all three projectors. The remaining 93 functions—associated with the white-filled circles—are expressed as simple tensor products, as illustrated in Equation (13).

With respect to the blending and trial functions, the formulations presented above are of general applicability. In the following sections, several alternative modeling approaches will be examined and compared.

3.1. Model-1: Lagrange Polynomials

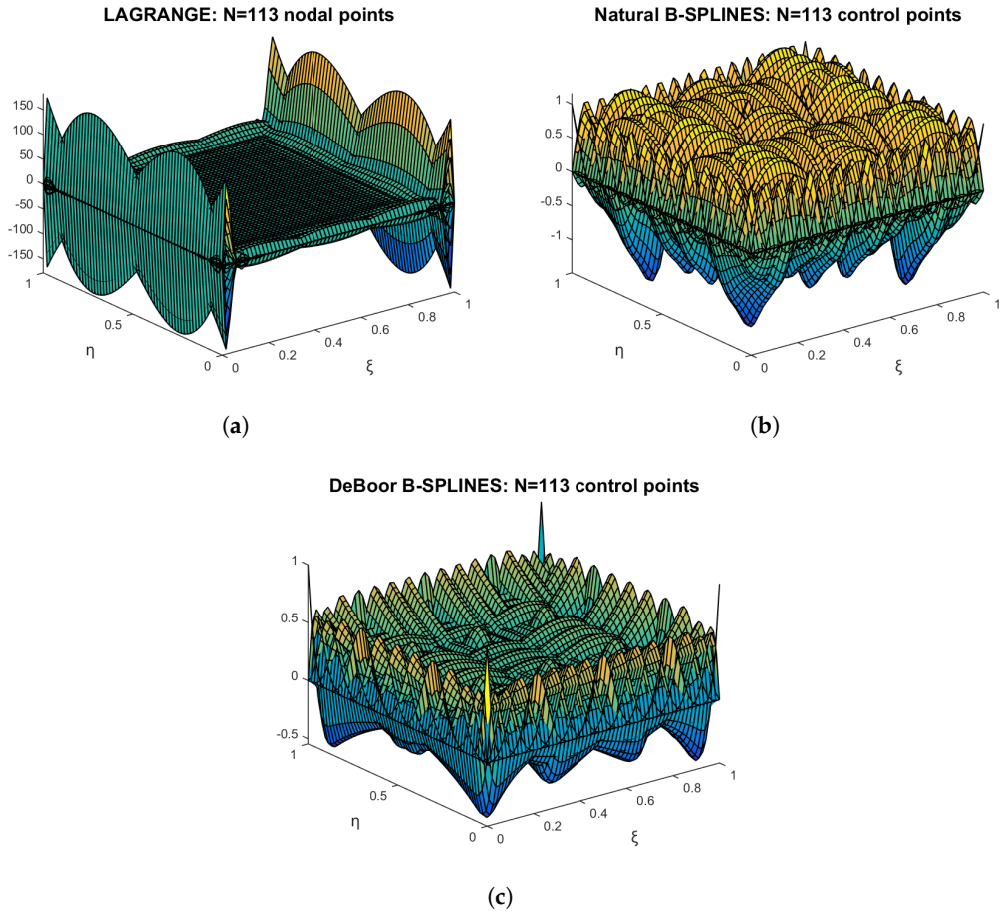
As a representative example, we consider the following configuration (Figure 3d):

1. Lagrange polynomials of degree $p_\xi = 4$ and $p_\eta = 3$ are employed as blending functions in the ξ- and η-directions, respectively.
2. Lagrange polynomials of degree $p'_\xi = 16$ and $p'_\eta = 12$ are used as trial functions in the ξ- and η-directions, respectively.

For this Model-1 setup, the resulting bivariate global shape functions $\phi(\xi,\eta)$ are depicted in Figure 4a. Notably, elevated function values are observed along the vertical edges, indicating localized influence. Furthermore, the constructed basis satisfies the partition of unity property:

$$\sum_{i=1}^{113} \phi_i(\xi,\eta) = 1\,. \tag{14}$$

 doi:10.20944/preprints202510.1500.v1



(**a**) (**b**)

(**c**)

Figure 4. Bivariate basis functions for the 113-node transfinite element, using Lagrange polynomials as blending functions, while trial functions are: (**a**) Lagrange polynomials; (**b**) natural cubic cardinal B-splines; (**c**) de Boor cubic B-splines.

3.2. Model-2: Mixed Scheme

In this model, the blending functions are retained from Model-1, namely Lagrange polynomials of degree $p_\xi = 4$ and $p_\eta = 3$ in the ξ- and η-directions, respectively. For the trial functions, however, we adopt the classical *natural cubic cardinal* B-splines, which are directly associated with the nodal values U. This approach mirrors the behavior of Lagrange polynomials in Model-1, thereby eliminating the need for generalized coefficients a_i.

This formulation is particularly well-suited to uniform nodal distributions with single knots and facilitates direct comparison with Model-1. It is worth noting that this type of spline interpolation—natural cubic B-splines—was widely used during the mid-1980s (see [24], among others).

In brief, the knot vector for the trial functions in the ξ-direction consists of 23 entries:

$$\Xi = [0, 0, 0, 0, \tfrac{1}{16}, \ldots, \tfrac{15}{16}, 1, 1, 1, 1],$$

which generates 19 control points. From these, 17 univariate cardinal trial functions are constructed, corresponding to the 17 nodal points along the horizontal stations.

Based on the formulations in Eqs. (12) and (13), the graphical representation of the resulting 113 basis functions is shown in Figure 4b. These basis functions have been verified to satisfy the partition of unity property:

$$\sum_{i=1}^{113} \psi_i(\xi, \eta) = 1\,, \tag{15}$$

and exhibit unity as their upper bound.

 doi:10.20944/preprints202510.1500.v1

Further details regarding the construction of the cardinal univariate set of natural cubic B-splines can be found in Ref. [9, pp. 103–109], and relevant implementation code is provided in A.

3.3. Model-3: Cubic B-Splines

This model closely aligns with the current state-of-the-art practices in Isogeometric Analysis (IGA), where classical B-splines (de Boor formulation) or NURBS are commonly employed. In this framework, B-splines are utilized for constructing both the blending and trial function sets. Classical clamped splines are adopted, reflecting the fact that for a polynomial degree p, the minimum number of control points is $p+1$ in the case of Bernstein polynomials, whereas for pure B-splines, the condition $n_{\text{ctrl}} > p+1$ must be satisfied.

Within this context, the 113-node transfinite element illustrated in Figure 3d should be interpreted as an index space rather than a physical nodal layout. In Models 1 and 2 (see Sects. 3.1 and 3.2), each of the 113 nodal points was directly associated with a unique bivariate shape function linked to a nodal value U_i, leaving no ambiguity in the trial function definition.

Assuming that the 17 points along each horizontal station are treated as single breakpoints, it is well known that the corresponding number of control points becomes 19. Conversely, if only 17 control points per horizontal station are desired, one must define 15 breakpoints, resulting in 14 knot spans in the ξ-direction. The associated knot vector is then given by:

$$\Xi = [0,0,0,0,\tfrac{1}{14},\tfrac{2}{14},\ldots,\tfrac{13}{14},1,1,1,1]\,. \tag{16}$$

Similarly, to obtain 13 control points in the η-direction (based on 11 breakpoints and 10 knot spans), the corresponding knot vector is:

$$H = [0,0,0,0,\tfrac{1}{10},\tfrac{2}{10},\ldots,\tfrac{9}{10},1,1,1,1]\,. \tag{17}$$

The graphical representation of the resulting basis functions is shown in Figure 4c.

4. Construction of Multi-Layer Finite Elements with Nodes Arranged in Parallel Layers

We now examine multi-layer finite elements containing internal nodes, where the number of nodes per station (i.e., per parallel layer) may vary. Such configurations arise when nodes—both internal and boundary—are aligned along unidirectional stations, for instance, distributed horizontally, as depicted in Figure 1e.

4.1. Unidirectional Multi-Layer Transfinite Lagrange and Bernstein Elements

The unidirectional analog of Equation (10), corresponding to the configuration shown in Figure 1e where all five stations are parallel to the horizontal ξ-axis, is given by the following projector:

$$\begin{aligned} P_\eta\{a\} = {} & \tilde{E}_1(\eta)a_1(\xi) + \tilde{E}_2(\eta)a_2(\xi) + \tilde{E}_3(\eta)a_3(\xi) \\ & + \tilde{E}_4(\eta)a_4(\xi) + \tilde{E}_5(\eta)a_5(\xi)\,. \end{aligned} \tag{18}$$

In this formulation, the approximation function is expressed as $U(\xi,\eta) = P_\eta\{a\}$. Each component function $a_i(\xi)$, for $i = 1,\ldots,5$, can be represented using clamped B-splines. The corresponding knot vectors are constructed such that the number of control points matches the number of nodes shown in Figure 1e.

It is important to note that the interpolatory nature of these B-splines—exhibiting unit values at the endpoints of each horizontal station (i.e., at $\xi = 0$ and $\xi = 1$)—does not introduce any complications. This is because each function $a_j(\xi)$ is multiplied by a non-cardinal blending function $\tilde{E}_j(\eta)$, analogous to the behavior observed in one-dimensional problems.

 doi:10.20944/preprints202510.1500.v1

For instance, if $a(\xi)$ is a constant function, the clamped B-spline basis will represent it exactly. This observation underscores that the value of U is influenced by all coefficients a within the patch, a property particularly relevant for Bernstein polynomials. However, in the present formulation, the focus is on the coefficient functions $a_i(\xi)$ rather than the potentially problematic direct interpolation of U along each horizontal station.

A previous report, Ref. [16], proposed the conjecture that transfinite interpolation can be effectively employed using a wide variety of blending functions, provided that the partition of unity condition is satisfied and the function set is complete. Within this framework, not only cardinal Lagrange polynomials but also non-cardinal blending functions—such as Bernstein polynomials and B-splines—have been successfully explored.

Furthermore, in the context of trial functions, any well-established univariate functional basis may be considered a viable candidate for use within numerical analysis modules, including Galerkin and collocation methods. This flexibility allows for the integration of diverse approximation schemes while maintaining consistency with the underlying transfinite interpolation principles.

Let us consider a class of transfinite elements with unidirectional nodes, as shown in Figure 5. Let us focus on the 12-node element (Figure 5a). Replacing Equation (9) by the vertical projection $P_\eta = E_1(\eta)U_1(\xi) + E_2(\eta)U_2(\xi) + E_3(\eta)U_3(\xi)$, and then expressing the involved univariate functions $U_1(\xi), U_2(\xi), U_3(\xi)$ in terms of Lagrange polynomials $L_{i,p}(\xi)$ of variable degree p associated with each horizontal station (layer), we obtain the following set of bivariate global shape functions:

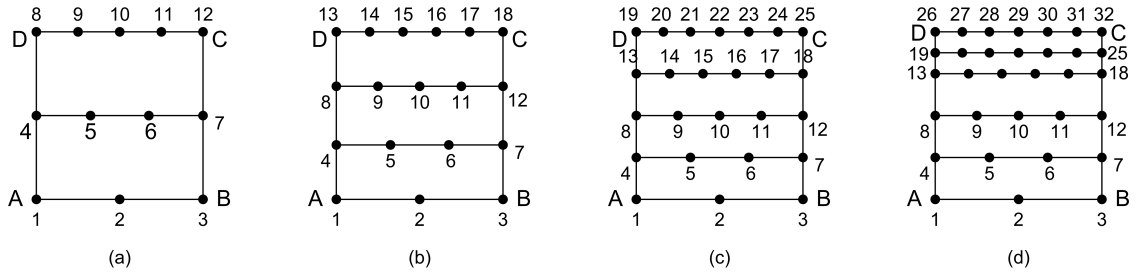


Figure 5. Transitional elements: (**a**) 12-node; (**b**) 18-node; (**c**) 25-node; (**d**) 32-node element.

$$
\begin{aligned}
\phi_1(\xi,\eta) &= L_{1,2}(\xi)L_{1,2}(\eta),\\
\phi_2(\xi,\eta) &= L_{2,2}(\xi)L_{1,2}(\eta),\\
\phi_3(\xi,\eta) &= L_{3,2}(\xi)L_{1,2}(\eta),\\
\hline
\phi_4(\xi,\eta) &= L_{1,3}(\xi)L_{2,2}(\eta),\\
\phi_5(\xi,\eta) &= L_{2,3}(\xi)L_{2,2}(\eta),\\
\phi_6(\xi,\eta) &= L_{3,3}(\xi)L_{2,2}(\eta),\\
\phi_7(\xi,\eta) &= L_{3,3}(\xi)L_{2,2}(\eta),\\
\hline
\phi_8(\xi,\eta) &= L_{1,4}(\xi)L_{3,2}(\eta),\\
\phi_9(\xi,\eta) &= L_{2,4}(\xi)L_{3,2}(\eta),\\
\phi_{10}(\xi,\eta) &= L_{3,4}(\xi)L_{3,2}(\eta),\\
\phi_{11}(\xi,\eta) &= L_{4,4}(\xi)L_{3,2}(\eta),\\
\phi_{12}(\xi,\eta) &= L_{5,4}(\xi)L_{3,2}(\eta)
\end{aligned}
\tag{19}
$$

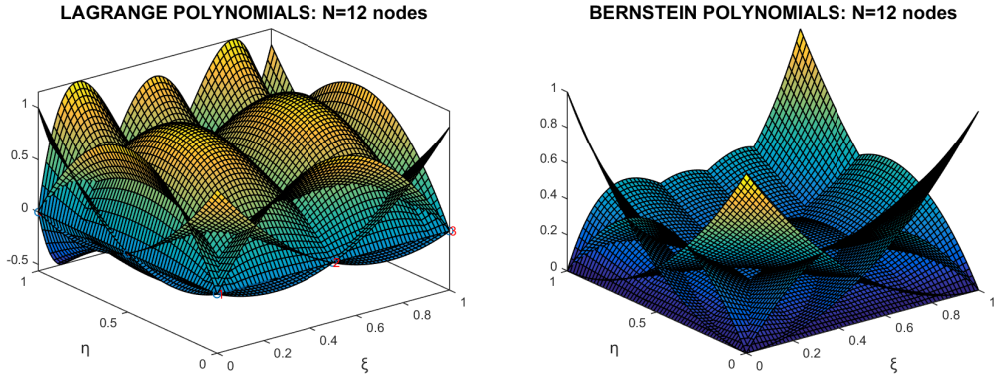
Next, replacing Lagrange polynomials with their Bernstein counterparts, or alternatively applying the projector defined by the analog of Equation (18) (including three terms associated with the three

 doi:10.20944/preprints202510.1500.v1

layers in Figure 5a) by expressing the involved univariate functions $a_1(\xi), a_2(\xi), a_3(\xi)$ in terms of Bernstein polynomials, we obtain the following set of non-negative basis functions:

$$\begin{aligned}
\psi_1(\xi,\eta) &= B_{1,2}(\xi)B_{1,2}(\eta),\\
\psi_2(\xi,\eta) &= B_{2,2}(\xi)B_{1,2}(\eta),\\
\psi_3(\xi,\eta) &= B_{3,2}(\xi)B_{1,2}(\eta),\\
\hline
\psi_4(\xi,\eta) &= B_{1,3}(\xi)B_{2,2}(\eta),\\
\psi_5(\xi,\eta) &= B_{2,3}(\xi)B_{2,2}(\eta),\\
\psi_6(\xi,\eta) &= B_{3,3}(\xi)B_{2,2}(\eta),\\
\psi_7(\xi,\eta) &= B_{3,3}(\xi)B_{2,2}(\eta),\\
\hline
\psi_8(\xi,\eta) &= B_{1,4}(\xi)B_{3,2}(\eta),\\
\psi_9(\xi,\eta) &= B_{2,4}(\xi)B_{3,2}(\eta),\\
\psi_{10}(\xi,\eta) &= B_{3,4}(\xi)B_{3,2}(\eta),\\
\psi_{11}(\xi,\eta) &= B_{4,4}(\xi)B_{3,2}(\eta),\\
\psi_{12}(\xi,\eta) &= B_{5,4}(\xi)B_{3,2}(\eta).
\end{aligned} \tag{20}$$

The shape functions $\phi_i(\xi,\eta)$ based on Lagrange polynomials (Equation (19)) are illustrated in Figure 6a, while the basis functions $\psi_i(\xi,\eta)$ corresponding to Bernstein polynomials (Equation (20)) are shown in Figure 6b. This element is not provided for B-spline interpolation, as it contains too few nodes to support such a representation.



(**a**) (**b**)

Figure 6. 12-node transition element: (**a**) Lagrange polynomials; (**b**) Bernstein polynomials.

4.2. Unidirectional Transfinite B-Spline Elements

Below we focus on the 18-node element shown in Figure 5b. The characteristic of this element is that—in addition to the four edges—it includes two internal stations which are both oriented in the ξ-direction (horizontal stations), and thus (as also happened with the 12-node element) the only non-zero projector among the three is P_η, i.e., $P_\xi = P_{\xi\eta} = 0$. Since the total number of horizontal stations is 4, there is no cubic B-spline to cover this case (unless we select quadratic B-spline), and therefore we resort to cubic Bernstein polynomials (hybrid scheme), as follows:

1. The blending functions in vertical (η)-direction were taken as cubic Bernstein polynomials, $B_{i,3}(\eta), i = 1,2,3,4$, where $B_{1,3} = (1-\eta)^3, B_{2,3} = 3(1-\eta)^2\eta, B_{3,3} = 3(1-\eta)\eta^2, B_{4,3} = \eta^3$.
2. The base 1-2-3 was modelled by quadratic Bernstein polynomials: $B_{1,2} = (1-\eta)^2, B_{2,2} = 2(1-\eta)\eta, B_{3,2} = \eta^2$.
3. The second layer, with nodes 4-5-6-7, was modelled by cubic Bernstein polynomials, $B_{i,3}(\xi), i = 1,2,3,4$.

 doi:10.20944/preprints202510.1500.v1

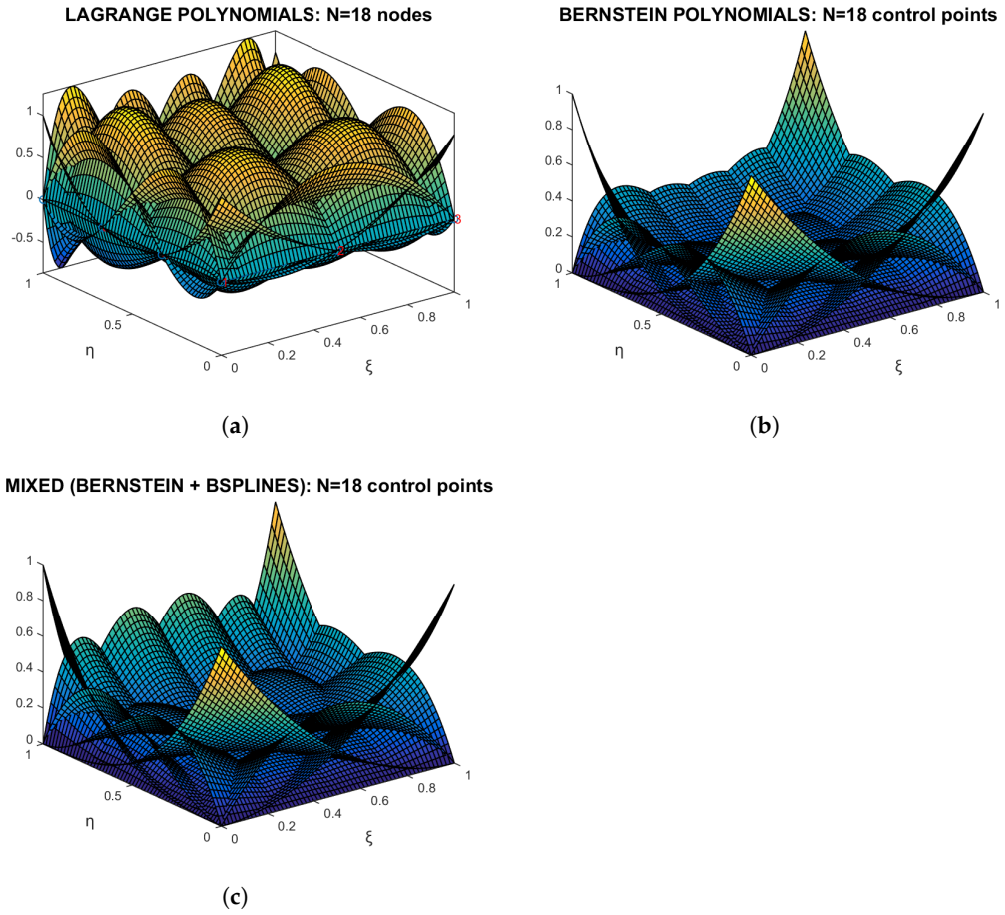
4. The third layer, with nodes 8-9-10-11-12, was modelled as a set of five cubic B-splines, $N_{i,3}^{\eta=2/3}(\xi), i = 1, \ldots, 5$, with knot vector $[0, 0, 0, 0, \frac{1}{2}, 1, 1, 1, 1]$.
5. The fourth (top) layer, with nodes 13-14-15-16-17-18, was modelled as a set of six cubic B-splines, $N_{i,3}^{\eta=1}(\xi), i = 1, \ldots, 6$, with knot vector $[0, 0, 0, 0, \frac{2}{5}, \frac{3}{5}, 1, 1, 1, 1]$.

Writing a projector similar to that of Equation (18), in conjunction with four parallel layers (stations) at $\eta = 0, \frac{1}{3}, \frac{2}{3}, 1$, we obtain the following basis functions:

$$\begin{aligned}
\psi_1(\xi,\eta) &= B_{1,2}(\xi)B_{1,3}(\eta),\\
\psi_2(\xi,\eta) &= B_{2,2}(\xi)B_{1,3}(\eta),\\
\psi_3(\xi,\eta) &= B_{3,2}(\xi)B_{1,3}(\eta),\\
\hline
\psi_4(\xi,\eta) &= B_{1,3}(\xi)B_{2,3}(\eta),\\
\psi_5(\xi,\eta) &= B_{2,3}(\xi)B_{2,3}(\eta),\\
\psi_6(\xi,\eta) &= B_{3,3}(\xi)B_{2,3}(\eta),\\
\psi_7(\xi,\eta) &= B_{3,3}(\xi)B_{2,3}(\eta),\\
\hline
\psi_8(\xi,\eta) &= N_{1,3}^{\eta=2/3}(\xi)B_{3,3}(\eta),\\
\psi_9(\xi,\eta) &= N_{2,3}^{\eta=2/3}(\xi)B_{3,3}(\eta),\\
\psi_{10}(\xi,\eta) &= N_{3,3}^{\eta=2/3}(\xi)B_{3,3}(\eta),\\
\psi_{11}(\xi,\eta) &= N_{4,3}^{\eta=2/3}(\xi)B_{3,3}(\eta),\\
\psi_{12}(\xi,\eta) &= N_{5,3}^{\eta=2/3}(\xi)B_{3,3}(\eta),\\
\hline
\psi_{13}(\xi,\eta) &= N_{1,3}^{\eta=1}(\xi)B_{4,3}(\eta),\\
\psi_{14}(\xi,\eta) &= N_{2,3}^{\eta=1}(\xi)B_{4,3}(\eta),\\
\psi_{15}(\xi,\eta) &= N_{3,3}^{\eta=1}(\xi)B_{4,3}(\eta),\\
\psi_{16}(\xi,\eta) &= N_{4,3}^{\eta=1}(\xi)B_{4,3}(\eta),\\
\psi_{17}(\xi,\eta) &= N_{5,3}^{\eta=1}(\xi)B_{4,3}(\eta),\\
\psi_{18}(\xi,\eta) &= N_{6,3}^{\eta=1}(\xi)B_{4,3}(\eta).
\end{aligned} \tag{21}$$

Obviously, the same form as Equation (21) holds when the blending and trial functions are all of Lagrange type or all of Bernstein type. For all these three cases, the graphs of the basis functions are shown in Figure 7.

 doi:10.20944/preprints202510.1500.v1



(**a**) (**b**)

(**c**)

Figure 7. Basis functions for the 18-node transfinite element: (**a**) Lagrange polynomials; (**b**) Bernstein polynomials; (**c**) Mixed, Bernstein polynomials and B-splines.

Regarding the estimation of the stiffness matrix of the 18-node transfinite element in Figure 5b, the fully-Lagrange and the fully-Bernstein models require an integration scheme of 6×4 Gauss points in the ξ- and η-directions, without local support. This is because the maximum polynomial degrees per single basis function are $p = 5$ and $q = 3$, respectively, and thus the multiplication in the integrand will contain terms up to $x^{10}y^6$. In contrast, for the same patch, the piecewise cubic B-spline interpolation requires 10×3 integration cells (i.e., 30 B-spline elements, as they have been called in standard IGA [10]) with 4×4 Gauss points per element (i.e., a total of 480 integration points, however with local support).

Note: When the number of parallel sections increases to 5 (and thus the number of nodes becomes 25, as shown in Figure 5c), the 25-node transfinite element may be handled using blending functions of B-spline type, based on the knot vector $\Xi_{\text{blend}} = [0, 0, 0, 0, \frac{1}{2}, 1, 1, 1, 1]$. Moreover, regarding the 32-node element made of 6 horizontal sections (partially non-uniform, as shown in Figure 5d), results and discussion are given in Sect. 8.1.2.

5. Finite Elements Featuring Tensor-Product Interior Nodes and Nonuniform Boundary Node Placement

This is a class of elements (the third in the present paper) in which the boundary nodes are placed arbitrarily, and the interior is populated using a tensor product grid of $m \times n$ internal nodes. A common rule of thumb is to choose the number of internal points in each direction (m or n) based on the average number of intermediate nodes on opposite edges. This class of elements has previously been formulated using Lagrange polynomials as both blending and trial functions [16,20]. In the

 doi:10.20944/preprints202510.1500.v1

present paper, the methodology is extended by introducing B-splines as an alternative choice for both blending and trial functions.

5.1. 27-Node Transfinite Element

As an example, we consider a 27-node transfinite element, in which the four edges are uniformly divided into 4, 3, 6, and 5 segments, respectively, as shown in Figure 8. For this configuration, we can construct at least three different formulations (blending and trial functions) of the transfinite element as follows:

- Lagrange polynomials;
- Bernstein polynomials.
- B-splines

Regarding Lagrange and Bernstein polynomials, the associated polynomial degrees per edge are as follows: $p_{AB} = 4$, $p_{BC} = 3$, $p_{DC} = 6$, and $p_{AD} = 5$. The internal nodes form a uniform tensor product pattern of scheme 3×3 (nodes 19 to 27 in Figure 8), which means that the degree of the blending functions is $n_{I,\xi} = n_{I,\eta} = 4$ (between edges and internal nodes, four node spans per direction are created). Therefore, this element consists of 18 boundary and 9 internal nodes.

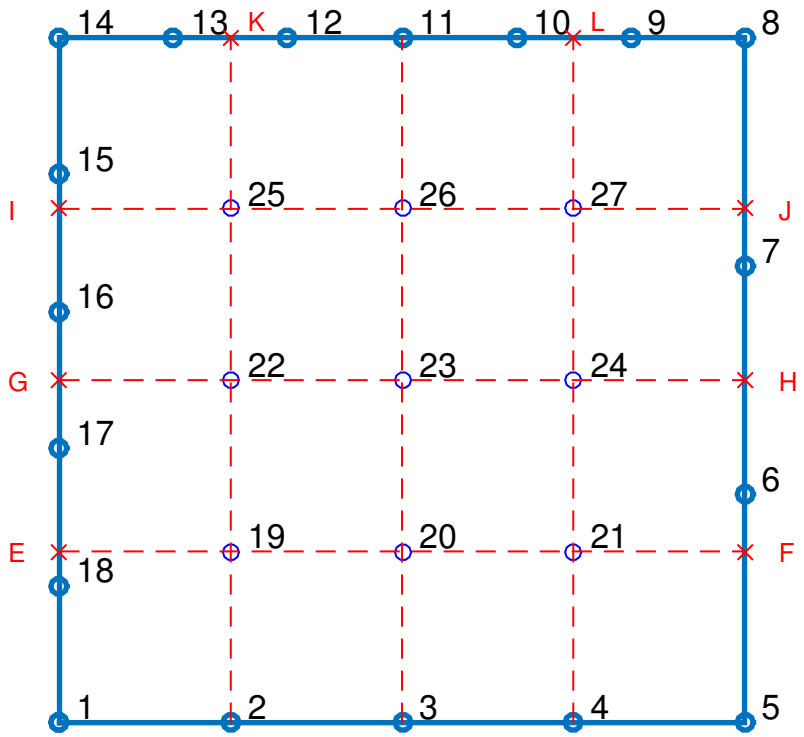


Figure 8. 27-node transfinite element.

The above 27-node B-spline element is defined by a regular tensor-product arrangement of internal nodes in a 3×3 grid. However, each of the four edges features a different discretization: edge AB contains 5 control points, edge BC has 4, edge DC includes 7, and edge AD consists of 6 control points. For clarity, node numbering starts from the boundary and proceeds to the interior. The internal grid lines intersect edge AD at points (E, G, I), edge BC at (F, H, J), and the top edge DC at $(K, 11, L)$. Notably, in this example, the bottom edge AB is defined such that its intermediate points coincide exactly with the orthogonal projections of the internal nodes (19 to 27) onto it, for the sake of alignment and illustrative symmetry.

 doi:10.20944/preprints202510.1500.v1

For this element, the transfinite interpolation (Equation (1)) consists of the following projectors (the auxiliary points, which are necessary to define the projectors, are denoted in red $\times$):

$$\begin{aligned}
P_\xi = {} & E_1(\xi)\Big(N_{1,3}^{AD}(\eta)a_1 + N_{2,3}^{AD}(\eta)a_{18} + N_{3,3}^{AD}(\eta)a_{17} + N_{4,3}^{AD}(\eta)a_{16} + N_{5,3}^{AD}(\eta)a_{15} + N_{6,3}^{AD}(\eta)a_{14}\Big) \\
& + E_2(\xi)\Big(N_{1,3}^{L\#2}(\eta)a_2 + N_{2,3}^{L\#2}(\eta)a_{19} + N_{3,3}^{L\#2}(\eta)a_{22} + N_{4,3}^{L\#2}(\eta)a_{25} + N_{5,3}^{L\#2}(\eta)a_K\Big) \\
& + E_3(\xi)\Big(N_{1,3}^{L\#3}(\eta)a_3 + N_{2,3}^{L\#3}(\eta)a_{20} + N_{3,3}^{L\#3}(\eta)a_{23} + N_{4,3}^{L\#3}(\eta)a_{26} + N_{5,3}^{L\#3}(\eta)a_{11}\Big) \\
& + E_4(\xi)\Big(N_{1,3}^{L\#4}(\eta)a_4 + N_{2,3}^{L\#4}(\eta)a_{21} + N_{3,3}^{L\#4}(\eta)a_{24} + N_{4,3}^{L\#4}(\eta)a_{27} + N_{5,3}^{L\#4}(\eta)a_L\Big) \\
& + E_5(\xi)\Big(N_{1,3}^{DC}(\eta)a_5 + N_{2,3}^{DC}(\eta)a_6 + N_{3,3}^{DC}(\eta)a_7 + N_{4,3}^{DC}(\eta)a_8\,,
\end{aligned} \tag{22}$$

also

$$\begin{aligned}
P_\eta = {} & E_1(\eta)\Big(N_{1,3}^{AB}(\xi)a_1 + N_{2,3}^{AB}(\xi)a_2 + N_{3,3}^{AB}(\xi)a_3 + N_{4,3}^{AB}(\xi)a_4 + N_{5,3}^{AB}(\xi)a_5\Big) \\
& + E_2(\eta)\Big(N_{1,3}^{L\#2}(\xi)a_E + N_{2,3}^{L\#2}(\xi)a_{19} + N_{3,3}^{L\#2}(\xi)a_{20} + N_{4,3}^{L\#2}(\xi)a_{21} + N_{5,3}^{L\#2}(\xi)a_F\Big) \\
& + E_3(\eta)\Big(N_{1,3}^{L\#3}(\xi)a_G + N_{2,3}^{L\#3}(\xi)a_{22} + N_{3,3}^{L\#3}(\xi)a_{23} + N_{4,3}^{L\#3}(\xi)a_{24} + N_{5,3}^{L\#3}(\xi)a_H\Big) \\
& + E_4(\eta)\Big(N_{1,3}^{L\#4}(\xi)a_I + N_{2,3}^{L\#4}(\xi)a_{25} + N_{3,3}^{L\#4}(\xi)a_{26} + N_{4,3}^{L\#4}(\xi)a_{27} + N_{5,3}^{L\#4}(\xi)a_J\Big) \\
& + E_5(\eta)\Big(N_{1,3}^{DC}(\xi)a_{14} + N_{2,3}^{DC}(\xi)a_{13} + N_{3,3}^{DC}(\xi)a_{12} + N_{4,3}^{DC}(\xi)a_{11} + N_{5,3}^{DC}(\xi)a_{10} \\
& + \quad N_{6,3}^{DC}(\xi)a_9 + N_{7,3}^{DC}(\xi)a_8\Big)\,,
\end{aligned} \tag{23}$$

and

$$\begin{aligned}
P_{\xi\eta} = {} & E_1(\xi)E_1(\eta)a_1 + E_2(\xi)E_1(\eta)a_2 + E_3(\xi)E_1(\eta)a_3 + E_4(\xi)E_1(\eta)a_4 + E_5(\xi)E_1(\eta)a_4 \\
& + E_1(\xi)E_2(\eta)a_E + E_2(\xi)E_2(\eta)a_{19} + E_3(\xi)E_2(\eta)a_{20} + E_4(\xi)E_2(\eta)a_{21} + E_5(\xi)E_2(\eta)a_F \\
& + E_1(\xi)E_3(\eta)a_G + E_2(\xi)E_3(\eta)a_{22} + E_3(\xi)E_3(\eta)a_{23} + E_4(\xi)E_3(\eta)a_{24} + E_5(\xi)E_3(\eta)a_H \\
& + E_1(\xi)E_4(\eta)a_I + E_2(\xi)E_4(\eta)a_{25} + E_3(\xi)E_4(\eta)a_{26} + E_4(\xi)E_4(\eta)a_{27} + E_5(\xi)E_4(\eta)a_J \\
& + E_1(\xi)E_5(\eta)a_{14} + E_2(\xi)E_5(\eta)a_K + E_3(\xi)E_5(\eta)a_{11} + E_4(\xi)E_5(\eta)a_L + E_5(\xi)E_5(\eta)a_8\,.
\end{aligned} \tag{24}$$

Furthermore, if we consider that for either $s = \xi, \eta$ we have:

$$N_{i,3}^{L\#j}(s) = E_i(s), \quad i = 1, \ldots, 5 \quad \text{for all internal layers L\#j}\,, \tag{25}$$

 doi:10.20944/preprints202510.1500.v1

the substitution of Equation (22) to Equation (25) into the Boolean sum given by Equation (1) cancels all the auxiliary red-colored terms, and thus the bivariate function $U(\xi,\eta)$ is approximated by:

$$\begin{aligned}
U(\xi,\eta) = &\Big(N_{1,3}^{AD}(\eta)E_1(\xi) + \cancel{N_{1,3}^{AB}(\xi)E_1(\eta)} - \cancel{E_1(\xi)E_1(\eta)}\Big)a_1 \\
&+ N_{2,3}^{AB}(\xi)E_1(\eta)a_2 + N_{3,3}^{AB}(\xi)E_1(\eta)a_3 + N_{4,3}^{AB}(\xi)E_1(\eta)a_4 \\
&+ \Big(N_{1,3}^{BC}(\eta)E_5(\xi) + \cancel{N_{5,3}^{AB}(\xi)E_1(\eta)} - \cancel{E_5(\xi)E_1(\eta)}\Big)a_5 \\
&+ N_{2,3}(\eta)E_5(\xi)a_6 + N_{3,3}(\eta)E_5(\xi)a_7 \\
&+ \Big(N_{4,3}^{BC}(\eta)E_5(\xi) + N_{7,3}^{DC}(\xi)E_5(\eta) - E_5(\xi)E_5(\eta)\Big)a_8 \\
&+ N_{6,3}(\xi)E_5(\eta)a_9 + N_{5,3}(\xi)E_5(\eta)a_{10} + N_{4,3}(\xi)E_5(\eta)a_{11} \\
&+ N_{3,3}(\xi)E_5(\eta)a_{12} + N_{2,3}(\xi)E_5(\eta)a_{13} \\
&+ (N_{6,3}(\eta)E_1(\xi) + N_{1,3}(\xi)E_5(\eta) - E_1(\xi)E_5(\eta))a_{14} \\
&+ N_{5,3}(\eta)E_1(\xi)a_{15} + N_{4,3}(\eta)E_1(\xi)a_{16} + N_{3,3}(\eta)E_1(\xi)a_{17} + N_{2,3}(\eta)E_1(\xi)a_{18} \\
&+ E_2(\xi)E_2(\eta)a_{19} + E_3(\xi)E_2(\eta)a_{20} + E_4(\xi)E_2(\eta)a_{21} \\
&+ E_2(\xi)E_3(\eta)a_{22} + E_3(\xi)E_3(\eta)a_{23} + E_4(\xi)E_3(\eta)a_{24} \\
&+ E_2(\xi)E_4(\eta)a_{25} + E_3(\xi)E_4(\eta)a_{26} + E_4(\xi)E_4(\eta)a_{27}\,.
\end{aligned} \tag{26}$$

The above model offers flexibility in the choice of the univariate blending and trial functions. Therefore, we can proceed as follows:

1. The five blending functions per direction, horizontal or vertical, can be ensured by the knot vector $\Xi = [0,0,0,0,\frac{1}{2},1,1,1,1]$ which guarantees five control points.
2. The trial functions along the edge AB are chosen to match the blending functions in the ξ-direction, since all associated control points are orthogonal projections of the internal points onto this edge.
3. The four control points along the edge BC are ensured by the knot vector $H = [0,0,0,0,1,1,1,1]$, which effectively corresponds to a set of four Bernstein polynomials of degree 3.
4. The seven control points along the edge DC are ensured by the knot vector $\Xi = [0,0,0,0,\frac{1}{4},\frac{2}{4},\frac{3}{4},1,1,1,1]$.
5. The six control points along the edge AD are determined by the knot vector $\Xi = [0,0,0,0,\frac{1}{3},\frac{2}{3},1,1,1,1]$.

Under these assumptions, the 27 basis functions $\psi_j(\xi,\eta)$ are defined as follows:

Regardless of the element type, the formulas for the shape or basis functions remain identical. According to the Boolean sum formulation, the corner nodes (A,B) on the bottom edge—discretized identically to the interior—are not influenced by the blending functions, whereas the corner nodes (C,D) on the top edge are affected.

For instance, using cubic B-splines for the blending functions:

$$E_1 = N_{1,3}, \quad E_2 = N_{2,3}, \quad E_3 = N_{3,3}, \quad E_4 = N_{4,3}, \quad E_5 = N_{5,3},$$

and also employing cubic B-splines for the trial functions along the edges, the resulting set of bivariate basis functions is:

 doi:10.20944/preprints202510.1500.v1

$$
\begin{aligned}
\psi_1(\xi,\eta) &= N_{1,3}(\xi)E_{1,3}(\eta),\\
\psi_2(\xi,\eta) &= N_{2,3}(\xi)E_{1,3}(\eta),\\
\psi_3(\xi,\eta) &= N_{3,3}(\xi)E_{1,3}(\eta),\\
\psi_4(\xi,\eta) &= N_{4,3}(\xi)E_{1,3}(\eta),\\
\psi_5(\xi,\eta) &= N_{5,3}(\xi)E_{1,3}(\eta),\\
\psi_6(\xi,\eta) &= E_{5,3}(\xi)N_{2,3}(\eta),\\
\psi_7(\xi,\eta) &= E_{5,3}(\xi)N_{3,3}(\eta),\\
\psi_8(\xi,\eta) &= B_{4,3}(\eta)E_{5,3}(\xi) - E_{5,3}(\xi)E_{5,3}(\eta) + E_{5,3}(\eta)B_{7,3}(\xi),\\
\psi_9(\xi,\eta) &= N_{6,3}(\xi)E_{5,3}(\eta),\\
\psi_{10}(\xi,\eta) &= N_{5,3}(\xi)E_{5,3}(\eta),\\
\psi_{11}(\xi,\eta) &= N_{4,3}(\xi)E_{5,3}(\eta),\\
\psi_{12}(\xi,\eta) &= N_{3,3}(\xi)E_{5,3}(\eta),\\
\psi_{13}(\xi,\eta) &= N_{2,3}(\xi)E_{5,3}(\eta),\\
\psi_{14}(\xi,\eta) &= N_{1,3}(\xi)E_{5,3}(\eta) - E_{1,3}(\xi)E_{5,3}(\eta) + E_{1,3}(\xi)N_{6,3}(\eta),\\
\psi_{15}(\xi,\eta) &= E_{1,3}(\xi)N_{5,3}(\eta),\\
\psi_{16}(\xi,\eta) &= E_{1,3}(\xi)N_{4,3}(\eta),\\
\psi_{17}(\xi,\eta) &= E_{1,3}(\xi)N_{3,3}(\eta),\\
\psi_{18}(\xi,\eta) &= E_{1,3}(\xi)N_{2,3}(\eta),\\
\psi_{19}(\xi,\eta) &= E_{2,3}(\xi)E_{2,3}(\eta),\\
\psi_{20}(\xi,\eta) &= E_{3,3}(\xi)E_{2,3}(\eta),\\
\psi_{21}(\xi,\eta) &= E_{4,3}(\xi)E_{2,3}(\eta),\\
\psi_{22}(\xi,\eta) &= E_{2,3}(\xi)E_{3,3}(\eta),\\
\psi_{23}(\xi,\eta) &= E_{3,3}(\xi)E_{3,3}(\eta),\\
\psi_{24}(\xi,\eta) &= E_{4,3}(\xi)E_{3,3}(\eta),\\
\psi_{25}(\xi,\eta) &= E_{2,3}(\xi)E_{4,3}(\eta),\\
\psi_{26}(\xi,\eta) &= E_{3,3}(\xi)E_{4,3}(\eta),\\
\psi_{27}(\xi,\eta) &= E_{4,3}(\xi)E_{4,3}(\eta)\,.
\end{aligned}
\tag{27}
$$

The graph of the 27 basis functions $\psi_j(\xi,\eta)$, which are involved in Equation (26) and described by Equation (27), is shown in Figure 9. It has also been verified that the functional set $\{\psi_j(\xi,\eta)\}$ satisfies the partition of unity property, i.e.,

$$\sum_{i=1}^{27} \psi_i(\xi,\eta) = 1\,. \tag{28}$$

 doi:10.20944/preprints202510.1500.v1

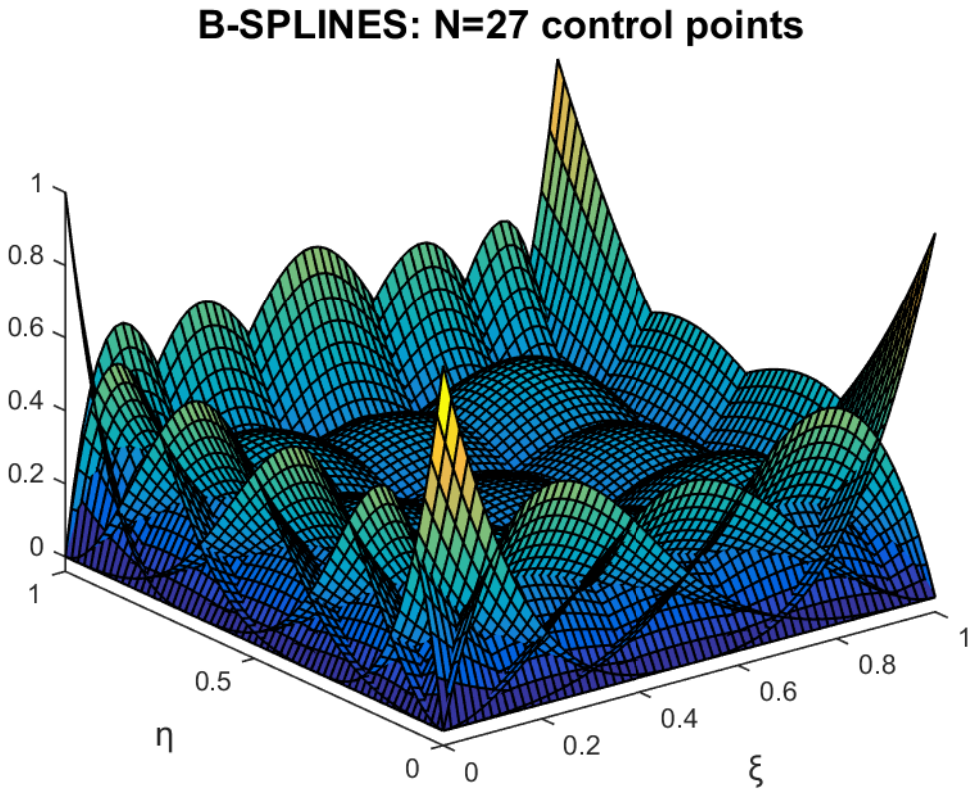


Figure 9. 27-DOF B-spline transfinite element.

It is noted that the basis functions can be categorized into three groups as follows:

1. **Internal nodes:** Defined by a simple tensor product of blending functions.
2. **Intermediate boundary nodes:** Constructed as local tensor products of trial functions along the edge and blending functions in the perpendicular direction.
3. **Corner nodes:** Formulated using a Boolean sum of three terms. Two terms correspond to projectors perpendicular to the edges connected to the corner node, while the third term is a tensor product of the associated blending functions, serving as a correction.

Although the symbol E in Equation (27) is identical to N, it was used for the sake of readability, allowing one to distinguish between blending functions (E) and trial functions (N). A graph of the set of basis functions is illustrated in Figure 9.

6. Handling T-Spline Elements

Although the previous sections referred to T-spline-like elements, this section systematically addresses the classical T-spline element. This element is defined by a T-index, which includes m knots $\{\xi_1, \ldots, \xi_m\}$ in the ξ-direction and n knots $\{\eta_1, \ldots, \eta_n\}$ in the η-direction of the parametric domain $ABCD$, where $0 \leq \xi, \eta \leq 1$.

In the common case of a cubic T-spline, the anchors of the element form a tensor-product grid of $m \times n$ potential anchor positions. However, some of these anchors may be missing, depending on the local refinement and topology of the T-mesh.

6.1. The Two Approaches

In general, two equivalent approaches can be developed, as follows [16]:

1. Transfinite interpolation
2. Background tensor-product

 doi:10.20944/preprints202510.1500.v1

Similar techniques have previously been applied in conjunction with Lagrange polynomials [21] and Bernstein polynomials [20]. In those cases, it was possible to globally eliminate the degrees of freedom (DOFs) associated with the missing nodes in the $m \times n$ tensor-product grid.

Since the central approach of this paper is to first control the stations (sections or isolines) and then blend them, it is immaterial whether the focus is on a boundary edge or an internal station (inter-boundary). Naturally, when operating along a boundary edge (e.g., edge AB), the computations are strictly one-dimensional, as each edge is independent of the remaining data associated with the patch $ABCD$.

In contrast, knot insertion in the interior of a cubic T-spline typically affects a surrounding region spanning two steps in each direction. However, in our numerical scheme—regarded as a first approximation—this influence is intentionally neglected.

6.2. Transfinite Interpolation Versus Tensor-Product

Transfinite interpolation applies to cases where each internal node lies on a single line—either horizontal or vertical—so that the associated degree of freedom (DOF) is involved exclusively in either the P_η or the P_ξ projector, respectively. However, if the internal point lies at the intersection of a horizontal and a vertical line, an alternative approach must be adopted—for example, by taking the average of the two lines.

As an example, let us consider a T-mesh with a maximum of 9 control points per section (Figure 10). Suppose that two knots are missing on the bottom edge AB, and thus the function $U(\xi, 0)$ is approximated as a B-spline associated with 7 univariate functions:

$$U(\xi, 0) = \sum_{i=1}^{7} N_{i,3}(\xi) a_i,$$

as illustrated by the black-colored nodes 1 to 7 in Figure 10b (red-colored nodes 77 and 78 at $\xi = \frac{3}{6}, \frac{5}{6}$ are missing). Since this set of B-splines is involved in the projector P_η, it follows that the associated bivariate basis functions are given by:

$$\psi_i(\xi, \eta) = N_{i,3}(\xi) E_1(\eta), \quad i = 1, \ldots, 7.$$

The seven B-spline functions along the edge AB are also plotted as solid lines in Figure 11.

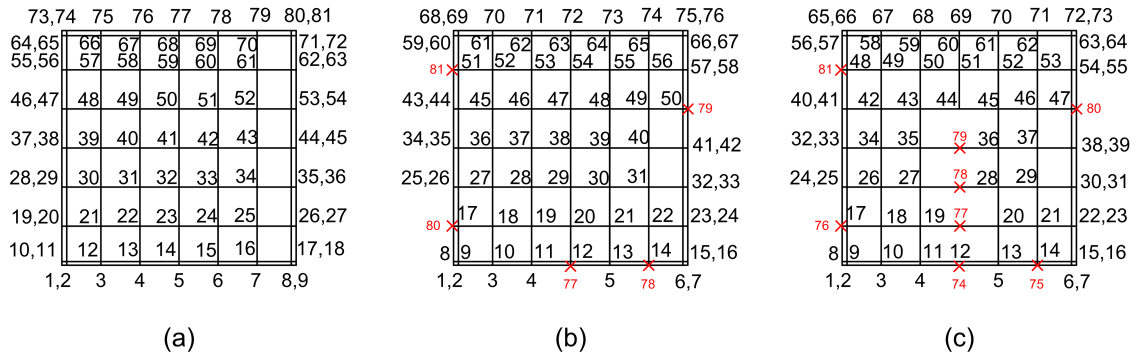


Figure 10. T-spline transfinite elements: (**a**) Tensor-product 81-node; (**b**) 76-node; (**c**) 73-node element. Missing nodes in tensor product are denoted by a skew red cross (×).

 doi:10.20944/preprints202510.1500.v1

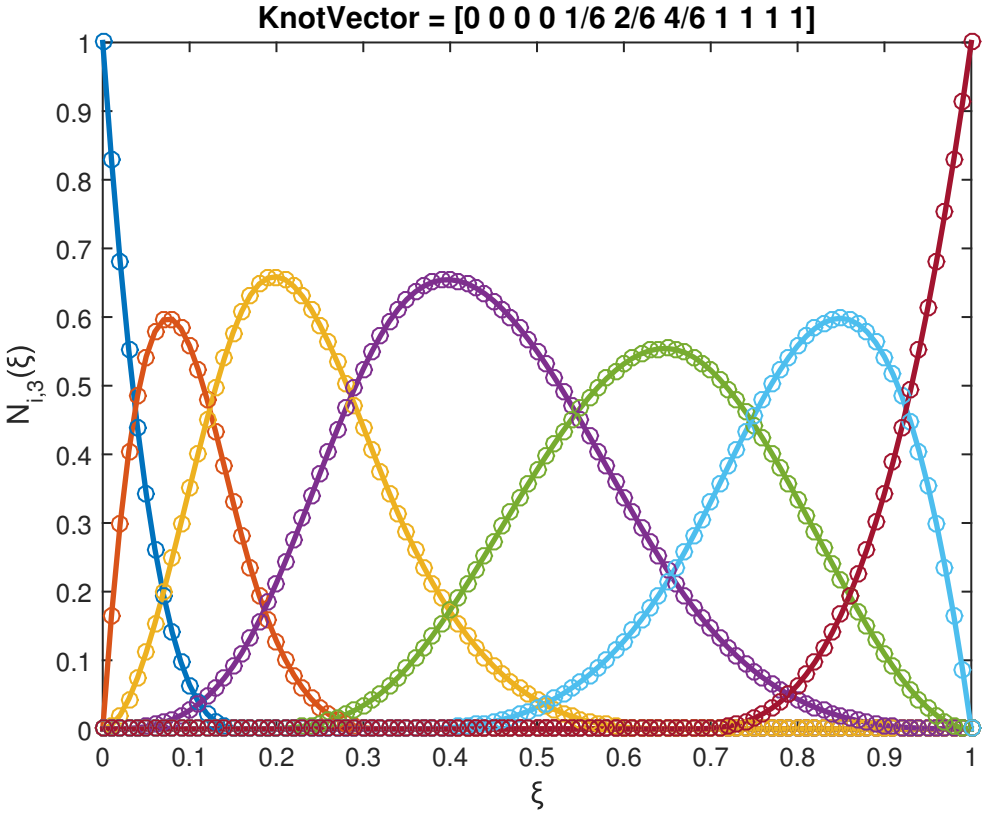


Figure 11. B-splines along the bottom edge AB.

The same result along the edge AB can also be obtained via an alternative approach. We virtually assume that all nine control points—fully occupying the nine uniformly spaced positions—contribute to the interpolation of the primary variable U along the edge AB. Furthermore, we consider this complete configuration as resulting from a previously incomplete one, following knot insertion at the previously missing locations: $\xi = \frac{3}{6}, \frac{5}{6}$.

The relationship between the assumed initial (incomplete) and the final complete configuration is detailed in Appendices B and C. More specifically, the complete set ($\mathbf{a}_Q$ of size 9×1) is related to the incomplete set ($\mathbf{a}_P$ of size 7×1) through the transformation matrix $\mathbf{T_{AB}}$, as follows:

$$\underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4^{tp} \\ a_{77} \\ a_5^{tp} \\ a_{78} \\ a_6^{tp} \\ a_7 \end{bmatrix}}_{\substack{(\mathbf{a}_Q) \\ \text{(Tensor-product)}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 3/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3/5 & 2/5 & 0 & 0 \\ 0 & 0 & 0 & 3/20 & 53/80 & 3/16 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{(\mathbf{T}_{AB}^t)} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}}_{\substack{(\mathbf{a}_P) \\ \text{(Primary)}}} . \tag{29}$$

It is noted that the matrix $\mathbf{T}_{AB}^t$ was obtained by calling the subroutine cited in C twice. The first call corresponds to the insertion of the knot at $\xi = \frac{3}{6}$, resulting in the transformation matrix $\mathbf{T}_1$, while the second call corresponds to the insertion of the knot at $\xi = \frac{5}{6}$, yielding the transformation matrix $\mathbf{T}_2$.

 doi:10.20944/preprints202510.1500.v1

The cumulative effect of inserting both knots leads to the overall transformation matrix $\mathbf{T}_{AB} = \mathbf{T}_1\mathbf{T}_2$, and consequently, its transpose is given by $\mathbf{T}^t_{AB} = \mathbf{T}^t_2\mathbf{T}^t_1$.

Substituting Equation (29) into the full B-spline sum expansion, we obtain:

$$
\begin{aligned}
U(\xi) &= \begin{bmatrix} N_{1,3}(\xi) & N_{2,3}(\xi) & \cdots & N_{9,3}(\xi) \end{bmatrix} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_{77} \\ a_6 \\ a_{78} \\ a_8 \\ a_9 \end{bmatrix}}_{\text{Tensor-product}} \\
&= \begin{bmatrix} N_{1,3}(\xi) & N_{2,3}(\xi) & \cdots & N_{9,3}(\xi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 3/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3/5 & 2/5 & 0 & 0 \\ 0 & 0 & 0 & 3/20 & 53/80 & 3/16 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}}_{\text{Primary}}
\end{aligned}
\tag{30}
$$

Obviously, Equation (30) dictates that the set of B-splines based on the seven control points (incomplete system: $\bar{N}_{i,3}(\xi)$, $i = 1, \ldots, 7$) is a linear combination of the basis functions of the complete system ($N_{i,3}(\xi)$, $i = 1, \ldots, 9$), which is defined by the product of a row vector and a transformation matrix:

$$
\begin{aligned}
\bar{N}_{1,3}(\xi) &= N_{1,3}(\xi), \\
\bar{N}_{2,3}(\xi) &= N_{2,3}(\xi), \\
\bar{N}_{3,3}(\xi) &= N_{3,3}(\xi) + \tfrac{1}{4}N_{4,3}(\xi), \\
\bar{N}_{4,3}(\xi) &= \tfrac{3}{4}N_{4,3}(\xi) + \tfrac{3}{5}N_{5,3}(\xi) + \tfrac{3}{20}N_{6,3}(\xi), \\
\bar{N}_{5,3}(\xi) &= \tfrac{2}{5}N_{5,3}(\xi) + \tfrac{53}{80}N_{6,3}(\xi) + \tfrac{1}{4}N_{7,3}(\xi), \\
\bar{N}_{6,3}(\xi) &= \tfrac{3}{16}N_{6,3}(\xi) + \tfrac{3}{4}N_{7,3}(\xi) + \tfrac{1}{2}N_{8,3}(\xi), \\
\bar{N}_{7,3}(\xi) &= \tfrac{1}{2}N_{8,3}(\xi) + N_{9,3}(\xi)\,.
\end{aligned}
\tag{31}
$$

Based on the full set and using Equation (31), the B-spline functions are also plotted as small circles in Figure 11. One may observe the coincidence between the B-splines directly generated by the knot vector $\Xi = [0, 0, 0, 0,\ 1/6,\ 2/6,\ 4/6,\ 1, 1, 1, 1]$ and those obtained via Equation (31).

6.3. Application of Knot Insertion Functions on the Entire Boundary of T-Spline

The above procedure is straightforward and can be extended from edge AB to the remaining boundary edges: BC, CD, and DA. From a numerical standpoint, substituting the incomplete basis with the complete one does not require matrix inversion, but only a simple matrix multiplication: $\mathbf{T}\mathbf{N}_Q = \mathbf{N}_P$, where $\mathbf{N}_P$ denotes the incomplete vector of basis functions, $\mathbf{T}$ is the transformation matrix, and $\mathbf{N}_Q$ is the complete vector associated with the tensor-product basis. Since the background tensor-product basis satisfies the partition of unity property, the same property holds after the substitution described by Equation (A19) in B.

 doi:10.20944/preprints202510.1500.v1

Let us now complete the discussion of Figure 10b, which illustrates a T-spline patch composed of 76 control points. This configuration is derived from a full tensor-product grid of 81 points, with five control points missing from the boundary. In this extreme case, the patch interior is fully populated, while the boundary is incomplete—lacking five control points (indicated by red crosses in Figure 10b and highlighted in red below) required to complete the full tensor structure. By considering the 76 actual (or primary) control points—shown in black—and the five missing (secondary) ones, we conceptually reconstruct a complete tensor-product grid in the background. The expansion of the primary variable over this virtual tensor-product grid (in which the updated variables—after elimination using Equation (29) and subsequent steps—are denoted by the superscript 'tp') is then expressed as:

$$
\begin{aligned}
U(\xi,\eta) = {} & (E_{1x}E_{1y})a_1 + (E_{2x}E_{1y})a_2 + (E_{3x}E_{1y})a_3 + (E_{4x}E_{1y})a_4^{tp} \\
& + (E_{5x}E_{1y})a_{77} + (E_{6x}E_{1y})a_5^{tp} + (E_{7x}E_{1y})a_{78} + (E_{8x}E_{1y})a_6^{tp} + (E_{9x}E_{1y})a_7 \\
& + (E_{1x}E_{2y})a_8^{tp} + (E_{2x}E_{2y})a_9 + (E_{3x}E_{2y})a_{10} + (E_{4x}E_{2y})a_{11} \\
& + (E_{5x}E_{2y})a_{12} + (E_{6x}E_{2y})a_{13} + (E_{7x}E_{2y})a_{14} + (E_{8x}E_{2y})a_{15} + (E_{9x}E_{2y})a_{16} \\
& + (E_{1x}E_{3y})a_{80}^{tp} + (E_{2x}E_{3y})a_{17} + (E_{3x}E_{3y})a_{18} + (E_{4x}E_{3y})a_{19} \\
& + (E_{5x}E_{3y})a_{20} + (E_{6x}E_{3y})a_{21} + (E_{7x}E_{3y})a_{22} + (E_{8x}E_{3y})a_{23} + (E_{9x}E_{3y})a_{24} \\
& + (E_{1x}E_{4y})a_{25}^{tp} + (E_{2x}E_{4y})a_{26} + (E_{3x}E_{4y})a_{27} + (E_{4x}E_{4y})a_{28} \\
& + (E_{5x}E_{4y})a_{29} + (E_{6x}E_{4y})a_{30} + (E_{7x}E_{4y})a_{31} + (E_{8x}E_{4y})a_{32} + (E_{9x}E_{4y})a_{33} \\
& + (E_{1x}E_{5y})a_{34} + (E_{2x}E_{5y})a_{35} + (E_{3x}E_{5y})a_{36} + (E_{4x}E_{5y})a_{37} \\
& + (E_{5x}E_{5y})a_{38} + (E_{6x}E_{5y})a_{39} + (E_{7x}E_{5y})a_{40} + (E_{8x}E_{5y})a_{41} + (E_{9x}E_{5y})a_{42}^{tp} \\
& + (E_{1x}E_{6y})a_{43}^{tp} + (E_{2x}E_{6y})a_{44} + (E_{3x}E_{6y})a_{45} + (E_{4x}E_{6y})a_{46} \\
& + (E_{5x}E_{6y})a_{47} + (E_{6x}E_{6y})a_{48} + (E_{7x}E_{6y})a_{49} + (E_{8x}E_{6y})a_{50} + (E_{9x}E_{6y})a_{79} \\
& + (E_{1x}E_{7y})a_{81}^{tp} + (E_{2x}E_{7y})a_{51} + (E_{3x}E_{7y})a_{52} + (E_{4x}E_{7y})a_{53} \\
& + (E_{5x}E_{7y})a_{54} + (E_{6x}E_{7y})a_{55} + (E_{7x}E_{7y})a_{56} + (E_{8x}E_{7y})a_{57} + (E_{9x}E_{7y})a_{58}^{tp} \\
& + (E_{1x}E_{8y})a_{59}^{tp} + (E_{2x}E_{8y})a_{60} + (E_{3x}E_{8y})a_{61} + (E_{4x}E_{8y})a_{62} \\
& + (E_{5x}E_{8y})a_{63} + (E_{6x}E_{8y})a_{64} + (E_{7x}E_{8y})a_{65} + (E_{8x}E_{8y})a_{66} + (E_{9x}E_{8y})a_{67} \\
& + (E_{1x}E_{9y})a_{68} + (E_{2x}E_{9y})a_{69} + (E_{3x}E_{9y})a_{70} + (E_{4x}E_{9y})a_{71} \\
& + (E_{5x}E_{9y})a_{72} + (E_{6x}E_{9y})a_{73} + (E_{7x}E_{9y})a_{74} + (E_{8x}E_{9y})a_{75} + (E_{9x}E_{9y})a_{76}
\end{aligned}
\tag{32}
$$

The main concept is that the basis functions associated with the 76 primary DOFs of this T-spline element will be produced by eliminating the red-colored secondary DOFs. This idea has been successfully applied in conjunction with Lagrange polynomials, where the values of the incomplete polynomials were calculated at the missing points and then substituted into virtual polynomials of higher degree in the tensor product [21], in a global manner for the entire station under consideration.

However, in the context of B-splines, if we wish to preserve the same polynomial degree, a different technique is required to substitute the red-colored variables.

Since the tensor-product configuration is fully controlled and the associated basis functions are standard B-splines, it can be considered as a state produced by virtual knot insertion into a coarser mesh—namely, the given T-mesh (defined by its index space). In this example, we must eliminate two DOFs along edge AB, one DOF along edge BC, and two DOFs along edge AD.

 doi:10.20944/preprints202510.1500.v1

In total, the model includes 76 primary nodes and 5 auxiliary (secondary) nodes—located on the boundary—which merely complete (or close) the tensor-product structure in the background.

More specifically, the treatment of edge AB has already been discussed in Equation (29).

Similarly, for the edge BC we have:

$$\underbrace{\begin{bmatrix} a_7 \\ a_{16} \\ a_{24} \\ a_{33} \\ a_{42} \\ a_{79} \\ a_{58} \\ a_{67} \\ a_{76} \end{bmatrix}}_{\text{Tensor-product}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 3/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2/3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{(\mathbf{T}_{BC}^T)} \underbrace{\begin{bmatrix} a_7 \\ a_{16} \\ a_{24} \\ a_{33} \\ a_{42} \\ a_{58} \\ a_{67} \\ a_{76} \end{bmatrix}}_{\text{Primary}} . \tag{33}$$

Since the top edge DC is complete, we keep it intact and proceed to the final edge AD, for which we have:

$$\underbrace{\begin{bmatrix} a_1 \\ a_8 \\ a_{80} \\ a_{25} \\ a_{34} \\ a_{43} \\ a_{81} \\ a_{59} \\ a_{68} \end{bmatrix}}_{\text{Tensor-product}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 3/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{(\mathbf{T}_{AD}^T)} \underbrace{\begin{bmatrix} a_1 \\ a_8 \\ a_{25} \\ a_{34} \\ a_{43} \\ a_{59} \\ a_{68} \end{bmatrix}}_{\text{Primary}} . \tag{34}$$

Substituting the three sets of equations—Equation (29), Equation (33), and Equation (34)—each corresponding to an active edge, into the tensor product defined by Equation (32), and performing factorization, we obtain:

$$U(\xi, \eta) = \sum_{i=1}^{76} \psi_i(\xi, \eta) a_i \,, \tag{35}$$

where the basis functions are given by:

 doi:10.20944/preprints202510.1500.v1

$$
\begin{aligned}
\psi_1 &= E_{1x}E_{1y} + \tfrac{1}{2}E_{1x}E_{2y} & \psi_2 &= E_{2x}E_{1y} \\
\psi_3 &= E_{3x}E_{1y} + \tfrac{1}{4}E_{4x}E_{1y} & \psi_4 &= \tfrac{3}{4}E_{4x}E_{1y} + \tfrac{3}{5}E_{5x}E_{1y} + \tfrac{3}{20}E_{6x}E_{1y} \\
\psi_5 &= \tfrac{2}{5}E_{5x}E_{1y} + \tfrac{53}{80}E_{6x}E_{1y} + \tfrac{1}{4}E_{7x}E_{1y} & \psi_6 &= \tfrac{3}{16}E_{6x}E_{1y} + \tfrac{3}{4}E_{7x}E_{1y} + \tfrac{1}{2}E_{8x}E_{1y} \\
\psi_7 &= \tfrac{1}{2}E_{8x}E_{1y} + E_{9x}E_{1y} & \psi_8 &= \tfrac{1}{2}E_{1x}E_{2y} + \tfrac{2}{3}E_{1x}E_{3y} \\
\psi_9 &= E_{2x}E_{2y} & \psi_{10} &= E_{3x}E_{2y} \\
\psi_{11} &= E_{4x}E_{2y} & \psi_{12} &= E_{5x}E_{2y} \\
\psi_{13} &= E_{6x}E_{2y} & \psi_{14} &= E_{7x}E_{2y} \\
\psi_{15} &= E_{8x}E_{2y} & \psi_{16} &= E_{9x}E_{2y} \\
\psi_{17} &= E_{2x}E_{3y} & \psi_{18} &= E_{3x}E_{3y} \\
\psi_{19} &= E_{4x}E_{3y} & \psi_{20} &= E_{5x}E_{3y} \\
\psi_{21} &= E_{6x}E_{3y} & \psi_{22} &= E_{7x}E_{3y} \\
\psi_{23} &= E_{8x}E_{3y} & \psi_{24} &= E_{9x}E_{3y} \\
\psi_{25} &= \tfrac{1}{3}E_{1x}E_{3y} + \tfrac{3}{4}E_{1x}E_{4y} & \psi_{26} &= E_{2x}E_{4y} \\
\psi_{27} &= E_{3x}E_{4y} & \psi_{28} &= E_{4x}E_{4y} \\
\psi_{29} &= E_{5x}E_{4y} & \psi_{30} &= E_{6x}E_{4y} \\
\psi_{31} &= E_{7x}E_{4y} & \psi_{32} &= E_{8x}E_{4y} \\
\psi_{33} &= E_{9x}E_{4y} + \tfrac{1}{4}E_{9x}E_{5y} & \psi_{34} &= \tfrac{1}{4}E_{1x}E_{4y} + E_{1x}E_{5y} + \tfrac{1}{4}E_{1x}E_{6y} \\
\psi_{35} &= E_{2x}E_{5y} & \psi_{36} &= E_{3x}E_{5y} \\
\psi_{37} &= E_{4x}E_{5y} & \psi_{38} &= E_{5x}E_{5y} \\
\psi_{39} &= E_{6x}E_{5y} & \psi_{40} &= E_{7x}E_{5y} \\
\psi_{41} &= E_{8x}E_{5y} & \psi_{42} &= \tfrac{3}{4}E_{9x}E_{5y} + \tfrac{1}{2}E_{9x}E_{6y} \\
\psi_{43} &= \tfrac{3}{4}E_{1x}E_{6y} + \tfrac{1}{3}E_{1x}E_{7y} & \psi_{44} &= E_{2x}E_{6y} \\
\psi_{45} &= E_{3x}E_{6y} & \psi_{46} &= E_{4x}E_{6y} \\
\psi_{47} &= E_{5x}E_{6y} & \psi_{48} &= E_{6x}E_{6y} \\
\psi_{49} &= E_{7x}E_{6y} & \psi_{50} &= E_{8x}E_{6y} \\
\psi_{51} &= E_{2x}E_{7y} & \psi_{52} &= E_{3x}E_{7y} \\
\psi_{53} &= E_{4x}E_{7y} & \psi_{54} &= E_{5x}E_{7y} \\
\psi_{55} &= E_{6x}E_{7y} & \psi_{56} &= E_{7x}E_{7y} \\
\psi_{57} &= E_{8x}E_{7y} & \psi_{58} &= \tfrac{1}{2}E_{9x}E_{6y} + \tfrac{2}{3}E_{9x}E_{7y} \\
\psi_{59} &= \tfrac{2}{3}E_{1x}E_{7y} + \tfrac{1}{2}E_{1x}E_{8y} & \psi_{60} &= E_{2x}E_{8y} \\
\psi_{61} &= E_{3x}E_{8y} & \psi_{62} &= E_{4x}E_{8y} \\
\psi_{63} &= E_{5x}E_{8y} & \psi_{64} &= E_{6x}E_{8y} \\
\psi_{65} &= E_{7x}E_{8y} & \psi_{66} &= E_{8x}E_{8y} \\
\psi_{67} &= \tfrac{1}{3}E_{9x}E_{7y} + E_{9x}E_{8y} & \psi_{68} &= \tfrac{1}{2}E_{1x}E_{8y} + E_{1x}E_{9y} \\
\psi_{69} &= E_{2x}E_{9y} & \psi_{70} &= E_{3x}E_{9y} \\
\psi_{71} &= E_{4x}E_{9y} & \psi_{72} &= E_{5x}E_{9y} \\
\psi_{73} &= E_{6x}E_{9y} & \psi_{74} &= E_{7x}E_{9y} \\
\psi_{75} &= E_{8x}E_{9y} & \psi_{76} &= E_{9x}E_{9y}
\end{aligned}
\tag{36}
$$

In Equation (36), we have assumed that E_{1x} through E_{9x} represent the B-splines $N_{i,3}(\xi)$ for $i = 1, \ldots, 9$, while E_{1y} through E_{9y} correspond to the B-splines $N_{i,3}(\eta)$ for $i = 1, \ldots, 9$. The symbol E is also consistent with the interpretation of blending functions. However, the tensor formulation—whether based on Lagrange, Bernstein, or B-spline functions—is fundamentally governed by one of the three projectors: P_ξ, P_η, or $P_{\xi\eta}$, as thoroughly discussed in Ref. [9].

The graph of these basis functions is shown in Figure 12. One may observe that:

 doi:10.20944/preprints202510.1500.v1

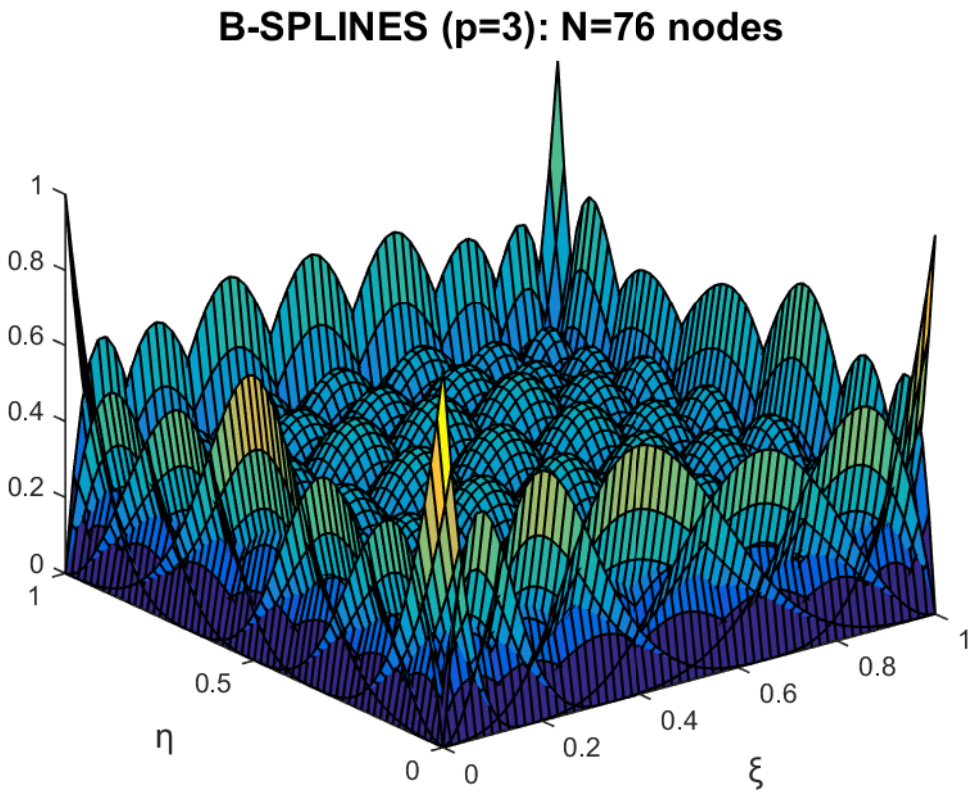


Figure 12. 76-node T-spline transfinite element with affected boundary.

1. None of the degrees of freedom (DOFs) associated with the top edge are affected, as this edge is fully occupied.
2. Among the 76 DOFs, the only affected ones are those associated with nodes $(1, 3, 4, 5, 6, 7, 8, 25, 33, 34, 42, 43, 58, 59, 67,$ All of these nodes lie on the affected boundary, comprising the three edges: AB, BC, and AD.
3. Although node 2 lies on edge AB, it is not affected by the nearest inserted knot at point '77', since it is located three units away.
4. None of the interior tensor-product DOFs are influenced by the coarse mesh along the boundary.
5. The shape and its parametric representation remain unchanged.
6. The partition of unity is satisfied *a priori* for all $(\xi, \eta) \in \Omega$:

$$\sum_{i=1}^{76} \phi_i(\xi, \eta) = 1\,, \tag{37}$$

   and therefore, no normalization is required.
7. The L_2 error for the tensor-product element (81-node) is 5.6581×10^{-4}%.
8. The L_2 error for the 76-node element was found to be 0.0011%, which is approximately double that of the tensor-product element.

6.4. Treatment of Internal Nodes

Regarding the internal nodes, any missing knot can be treated in an analogous manner to the boundary case. The key difference is that when a gap is filled by inserting a knot, the influence may extend in two parametric directions. The simplest scenario occurs when the missing knot affects only a single station—e.g., a horizontal line. This concept is directly applicable when using Lagrange or Bernstein polynomials.

Let us consider a simple case inspired by Ref. [25]. Specifically, three internal vertices are missing along three horizontal lines at $\xi = \frac{1}{2}$, corresponding to the parametric positions $\eta = \frac{1}{6}, \frac{2}{6}, \frac{3}{6}$, as

 doi:10.20944/preprints202510.1500.v1

illustrated in Figure 10c. For each of these missing vertices, we define the following transformation matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3/4 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 3/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{38}$$

so that the tensor-product degrees of freedom (DOFs), denoted by $\mathbf{a}_Q$, are related to the actual ones, $\mathbf{a}_P$, through the transformation

$$\mathbf{a}_Q = \mathbf{T}^t \mathbf{a}_P.$$

In more detail, we impose the following sets of constraints for each missing internal vertex:

$$\text{Isoline} \quad \eta = \tfrac{1}{6}: \quad a_{19}^{\text{tp}} = \tfrac{1}{4}a_{18} + \tfrac{3}{4}a_{19}, \quad a_{77}^{\text{tp}} = \tfrac{1}{2}a_{19} + \tfrac{1}{2}a_{20}, \quad a_{20}^{\text{tp}} = \tfrac{3}{4}a_{20} + \tfrac{1}{4}a_{21}\,. \tag{39}$$

$$\text{Isoline} \quad \eta = \tfrac{2}{6}: \quad a_{27}^{\text{tp}} = \tfrac{1}{4}a_{26} + \tfrac{3}{4}a_{27}, \quad a_{78}^{\text{tp}} = \tfrac{1}{2}a_{27} + \tfrac{1}{2}a_{28}, \quad a_{28}^{\text{tp}} = \tfrac{3}{4}a_{28} + \tfrac{1}{4}a_{29}\,. \tag{40}$$

$$\text{Isoline} \quad \eta = \tfrac{3}{6}: \quad a_{35}^{\text{tp}} = \tfrac{1}{4}a_{34} + \tfrac{3}{4}a_{35}, \quad a_{79}^{\text{tp}} = \tfrac{1}{2}a_{35} + \tfrac{1}{2}a_{36}, \quad a_{36}^{\text{tp}} = \tfrac{3}{4}a_{36} + \tfrac{1}{4}a_{37}\,. \tag{41}$$

In Eqs. (39)–(41), the superscript "tp" (standing for tensor product) in a_k^{tp} indicates that the k-th degree of freedom (DOF), shown in Figure 10c and subject to the constraint, represents the final value after the elimination process. This final value is subsequently employed in the expression of the initial tensor-product form, whereas the variables on the right-hand side correspond to the values prior to elimination.

After accounting for the five missing boundary nodes (Eqs. (29), (33), and (34)), along with the three internal nodes (Eqs. (39) to (41)), the resulting graph of the basis function set is illustrated in Figure 13.

 doi:10.20944/preprints202510.1500.v1

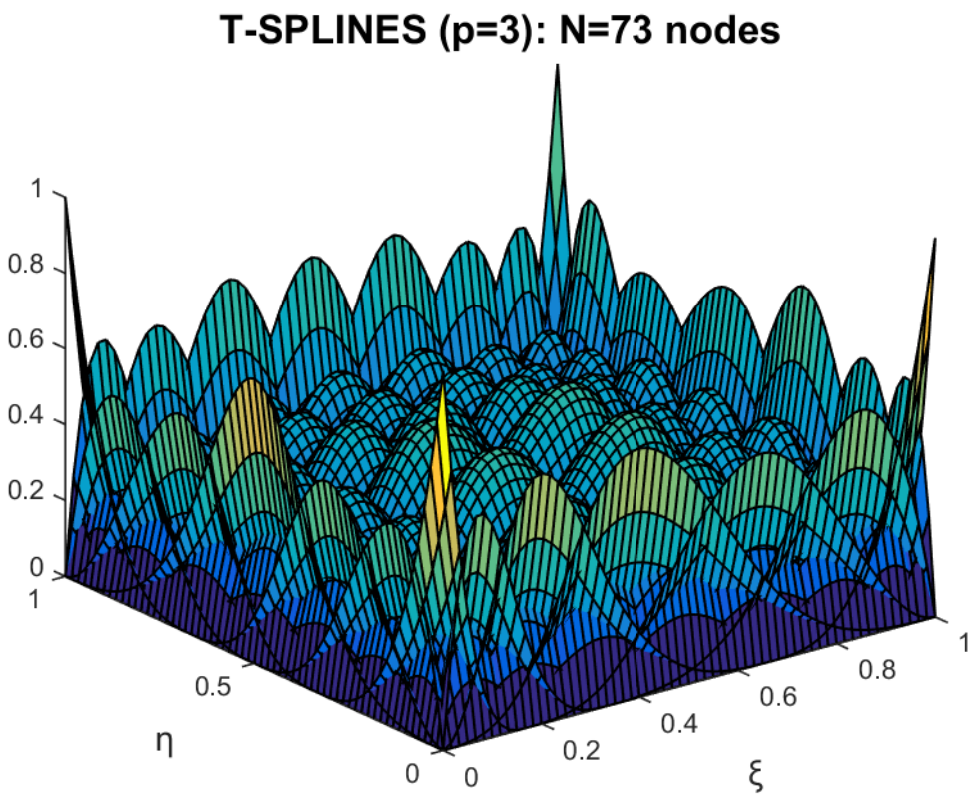


Figure 13. 73-node T-spline transfinite element with affected boundary plus three internal nodes.

6.5. Intersected Isolines

While handling missing vertices along the isolines of a T-mesh is relatively straightforward—as demonstrated in Sect. 6.4—complications arise when a missing vertex is located at the intersection of two isolines. In such cases, the numerical results may vary depending on the choice of isoline along which the elimination is performed.

Extensive numerical experimentation suggests a more robust strategy: perform the elimination independently along both isolines and then take the arithmetic mean of the two results as the final expression for elimination.

A simple test case is illustrated in Figure 14, where the central node (located at $\xi = \eta = \frac{1}{2}$) has been omitted from the tensor-product grid of 81 nodes.

 doi:10.20944/preprints202510.1500.v1

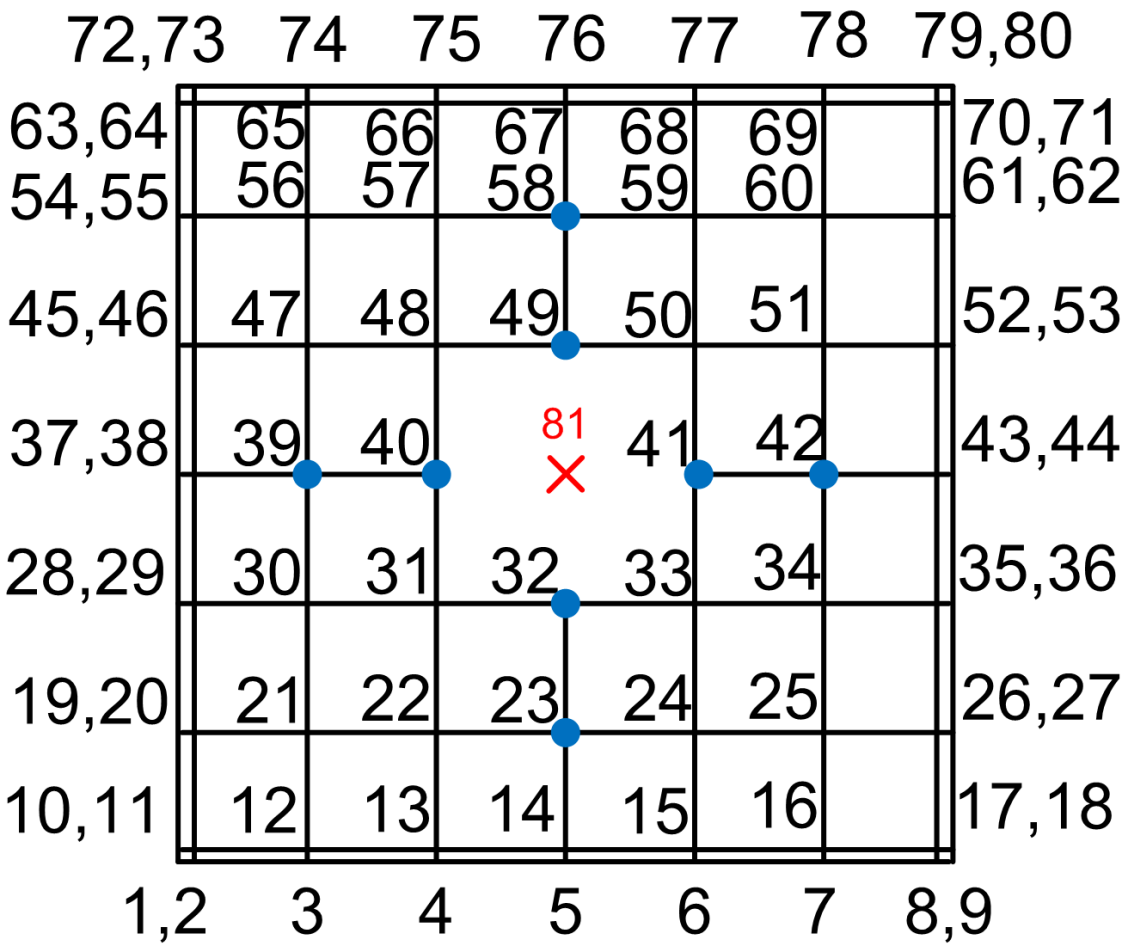


Figure 14. Missing central node in an 80-node transfinite element.

The imposed constraints are:
Regarding the horizontal ξ-direction:

$$\text{Isoline} \quad \eta = \tfrac{1}{2}: \quad a^{\text{tp}}_{40} = \tfrac{1}{4}a_{39} + \tfrac{3}{4}a_{40}, \quad a^{\text{tp}}_{81,x} = \tfrac{1}{2}a_{40} + \tfrac{1}{2}a_{41}, \quad a^{\text{tp}}_{41} = \tfrac{3}{4}a_{41} + \tfrac{1}{4}a_{42}\,. \tag{42}$$

Regarding the vertical η-direction:

$$\text{Isoline} \quad \xi = \tfrac{1}{2}: \quad a^{\text{tp}}_{32} = \tfrac{1}{4}a_{23} + \tfrac{3}{4}a_{32}, \quad a^{\text{tp}}_{81,y} = \tfrac{1}{2}a_{32} + \tfrac{1}{2}a_{49}, \quad a^{\text{tp}}_{49} = \tfrac{3}{4}a_{49} + \tfrac{1}{4}a_{58}\,. \tag{43}$$

 doi:10.20944/preprints202510.1500.v1

After substituting Equation (42) and Equation (43) into the tensor-product expression, we find that only the eight blue-colored vertices in Figure 14 are affected. Using the MATLAB code provided in Appendix D, the reader may verify that the corresponding (modified) basis functions are given by:

$$
\begin{aligned}
\psi_1 &= E_{1x}E_{1y} & \psi_2 &= E_{2x}E_{1y} & \psi_3 &= E_{3x}E_{1y} \\
\psi_4 &= E_{4x}E_{1y} & \psi_5 &= E_{5x}E_{1y} & \psi_6 &= E_{6x}E_{1y} \\
\psi_7 &= E_{7x}E_{1y} & \psi_8 &= E_{8x}E_{1y} & \psi_9 &= E_{9x}E_{1y} \\
\psi_{10} &= E_{1x}E_{2y} & \psi_{11} &= E_{2x}E_{2y} & \psi_{12} &= E_{3x}E_{2y} \\
\psi_{13} &= E_{4x}E_{2y} & \psi_{14} &= E_{5x}E_{2y} & \psi_{15} &= E_{6x}E_{2y} \\
\psi_{16} &= E_{7x}E_{2y} & \psi_{17} &= E_{8x}E_{2y} & \psi_{18} &= E_{9x}E_{2y} \\
\psi_{19} &= E_{1x}E_{3y} & \psi_{20} &= E_{2x}E_{3y} & \psi_{21} &= E_{3x}E_{3y} \\
\psi_{22} &= E_{4x}E_{3y} & \psi_{23} &= E_{5x}E_{3y} + \tfrac{1}{4}E_{5x}E_{4y} & \psi_{24} &= E_{6x}E_{3y} \\
\psi_{25} &= E_{7x}E_{3y} & \psi_{26} &= E_{8x}E_{3y} & \psi_{27} &= E_{9x}E_{3y} \\
\psi_{28} &= E_{1x}E_{4y} & \psi_{29} &= E_{2x}E_{4y} & \psi_{30} &= E_{3x}E_{4y} \\
\psi_{31} &= E_{4x}E_{4y} & \psi_{32} &= \tfrac{3}{4}E_{5x}E_{4y} + \tfrac{1}{4}E_{5x}E_{5y} & \psi_{33} &= E_{6x}E_{4y} \\
\psi_{34} &= E_{7x}E_{4y} & \psi_{35} &= E_{8x}E_{4y} & \psi_{36} &= E_{9x}E_{4y} \\
\psi_{37} &= E_{1x}E_{5y} & \psi_{38} &= E_{2x}E_{5y} & \psi_{39} &= E_{3x}E_{5y} + \tfrac{1}{4}E_{4x}E_{5y} \\
\psi_{40} &= \tfrac{3}{4}E_{4x}E_{5y} + \tfrac{1}{4}E_{5x}E_{5y} & \psi_{41} &= \tfrac{1}{4}E_{5x}E_{5y} + \tfrac{3}{4}E_{6x}E_{5y} & \psi_{42} &= \tfrac{1}{4}E_{6x}E_{5y} + E_{7x}E_{5y} \\
\psi_{43} &= E_{8x}E_{5y} & \psi_{44} &= E_{9x}E_{5y} & \psi_{45} &= E_{1x}E_{6y} \\
\psi_{46} &= E_{2x}E_{6y} & \psi_{47} &= E_{3x}E_{6y} & \psi_{48} &= E_{4x}E_{6y} \\
\psi_{49} &= \tfrac{1}{4}E_{5x}E_{5y} + \tfrac{3}{4}E_{5x}E_{6y} & \psi_{50} &= E_{6x}E_{6y} & \psi_{51} &= E_{7x}E_{6y} \\
\psi_{52} &= E_{8x}E_{6y} & \psi_{53} &= E_{9x}E_{6y} & \psi_{54} &= E_{1x}E_{7y} \\
\psi_{55} &= E_{2x}E_{7y} & \psi_{56} &= E_{3x}E_{7y} & \psi_{57} &= E_{4x}E_{7y} \\
\psi_{58} &= \tfrac{1}{4}E_{5x}E_{6y} + E_{5x}E_{7y} & \psi_{59} &= E_{6x}E_{7y} & \psi_{60} &= E_{7x}E_{7y} \\
\psi_{61} &= E_{8x}E_{7y} & \psi_{62} &= E_{9x}E_{7y} & \psi_{63} &= E_{1x}E_{8y} \\
\psi_{64} &= E_{2x}E_{8y} & \psi_{65} &= E_{3x}E_{8y} & \psi_{66} &= E_{4x}E_{8y} \\
\psi_{67} &= E_{5x}E_{8y} & \psi_{68} &= E_{6x}E_{8y} & \psi_{69} &= E_{7x}E_{8y} \\
\psi_{70} &= E_{8x}E_{8y} & \psi_{71} &= E_{9x}E_{8y} & \psi_{72} &= E_{1x}E_{9y} \\
\psi_{73} &= E_{2x}E_{9y} & \psi_{74} &= E_{3x}E_{9y} & \psi_{75} &= E_{4x}E_{9y} \\
\psi_{76} &= E_{5x}E_{9y} & \psi_{77} &= E_{6x}E_{9y} & \psi_{78} &= E_{7x}E_{9y} \\
\psi_{79} &= E_{8x}E_{9y} & \psi_{80} &= E_{9x}E_{9y} & . &
\end{aligned}
\tag{44}
$$

One may observe that only the blue-colored vertices are influenced—a well-known characteristic in standard T-spline practice. The complete set of basis functions is illustrated in Figure 15.

 doi:10.20944/preprints202510.1500.v1

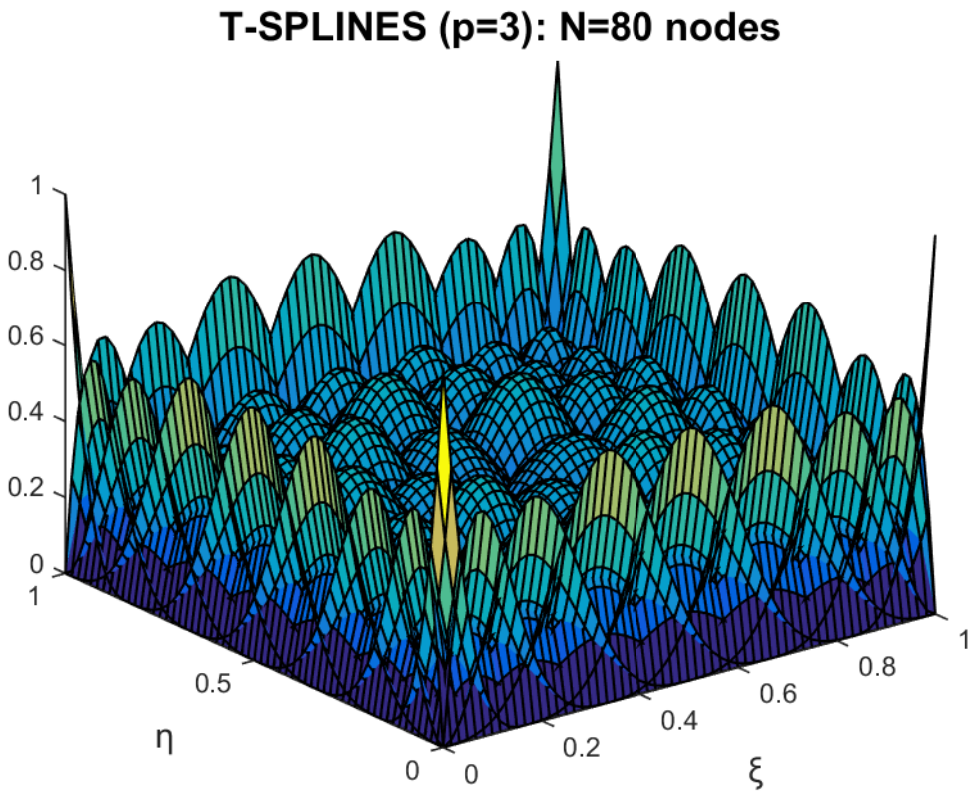


Figure 15. 80-node T-spline transfinite element with missing central vertex.

6.6. Continuity Issues

All bivariate basis functions are ultimately constructed as tensor products of blending functions and trial functions—both of which are *a priori* continuous. Consequently, the numerical solution, being an algebraic sum of such tensor products, inherits this continuity.

For B-splines, where the blending functions are piecewise polynomials of degree p_{blend} in the ξ-direction and the trial functions are of degree p_{trial} in the η-direction, the continuity at internal breakpoints is $C^{p_{\text{blend}}-1}$ and $C^{p_{\text{trial}}-1}$, respectively.

In contrast, tensor-product Lagrange (or Bernstein) polynomials of degrees (p, q) yield bivariate shape (or basis) functions that are infinitely differentiable within the open patch—effectively $C^p \times C^q$ continuous throughout the interior, which is treated as a single macroelement. However, globally, the assembled surface exhibits only $C^0 \times C^0$ continuity across element (patch) interfaces.

7. Software Issues

All computations were performed within the standard MATLAB® environment (version R2024b), which includes the Spline Toolbox. When applicable, the evaluation of B-splines, $N_{i,3}(\xi, \eta)$, and their derivatives was carried out using the built-in function `spcol`, originally developed in `Fortran77` by Carl de Boor and later ported to MATLAB [26].

In addition to built-in tools, a suite of self-contained in-house codes was developed to support the workflow, comprising: (i) pre-processing modules for mesh generation and data preparation, (ii) analysis routines for solving the governing equations, and (iii) post-processing utilities for visualization and error assessment.

Given that the primary objective was proof-of-concept—specifically, to evaluate the accuracy of transfinite elements under various unstructured configurations—the computational strategy was tailored accordingly.

 doi:10.20944/preprints202510.1500.v1

7.1. Lagrange and Bernstein Polynomials

A MATLAB® function named `lagrange`, which computes univariate Lagrange polynomials (both uniform and non-uniform) along with their first derivatives, is available in [23, p. 26]. Additionally, a tensor-product implementation of uniform Lagrange polynomials, named `shape_full_lagrange`, is provided in [23, p. 27]. The output of this function includes the bivariate shape functions and their partial derivatives (stored in the array `shp`), as well as the determinant of the Jacobian matrix (variable `xsj`). With minor modifications, this function can be extended to handle non-uniform polynomials.

An auxiliary in-house function, `Bernstein(tau, p)`, for computing Bernstein–Bézier polynomials is listed in E. Gaussian integration was performed using the subroutine `lgwt` (see, [27]).

For each pair of parameters (ξ, η), a subroutine `shapeX` was invoked. This subroutine utilizes either closed-form expressions from the main text or formulations from D, depending on context. Its general form is:

```
[shp, xsj] = shapeX(xi, eta, XL, NEL, ...)
```

where:
- `shp` contains the basis functions and their partial derivatives:
`shp(1,i)` $= \partial N/\partial x$,
`shp(2,i)` $= \partial N/\partial y$,
`shp(3,i)` $= N$, for $i = 1, \ldots,$ `NEL`.
- `xsj` is the determinant of the Jacobian matrix.
- `xi, eta` are the parametric coordinates (ξ, η).
- `XL` contains the Cartesian coordinates of the element nodes.
- `NEL` is the number of nodes in the element.

For the simplest case—a 12-node transfinite element described by Equation (19)—the complete subroutine is provided in F.

This subroutine structure aligns with standard finite element implementation practices, as discussed in Ref. [3, p. 754]. The full computer code is included in G.

7.2. B-Spline Interpolation

The MATLAB® function `spcol`, part of the Spline Toolbox, generates a collocation matrix for B-splines by evaluating spline basis functions and their derivatives at specified sites.

In the context of B-spline interpolation, the function `shapeX` is enhanced to accept additional input arguments: the polynomial degrees `px`, `py`, and the knot vectors `knotx`, `knoty`. This extended formulation enables direct comparison of different blending and trial functions within a unified in-house finite element framework, without requiring special treatment for their local support characteristics.

Numerical integration was performed over cells defined by adjacent breakpoints—i.e., cubic B-spline elements in the sense of isogeometric analysis (IGA)—using a 4×4 or 6×6 Gaussian quadrature scheme for rectangular or curvilinear elements, respectively. Notably, both Lagrange and Bézier elements do not require domain subdivision and can be integrated using a global quadrature over the entire patch.

The determination of integration cells (elements) can be efficiently performed by applying the MATLAB function `unique` to the knot vectors. Although optimizing the code for computational efficiency using a connectivity vector `IEN` is straightforward, this enhancement has been deferred to future work.

7.3. T-Spline

In this context, a dedicated routine was developed to compute the transformation matrix $\mathbf{T}$ associated with knot insertion (see, for example, Equation (38), and for further details, refer to B and C). These formulas are ultimately incorporated manually into the implementation provided in D.

 doi:10.20944/preprints202510.1500.v1

8. Numerical Solution of Boundary-Value Problems

To illustrate the proposed method, four representative examples are presented below. The first involves a rectangular domain, while the second considers a semicircular ring (i.e., a half-annulus), both addressing the solution of the Laplace equation.

For these two cases, the L_2 error norm (expressed as a percentage) is computed over the entire domain Ω using the following formula:

$$L_2 = \frac{\int_\Omega (U_{\text{calculated}} - U_{\text{exact}})^2 \, d\Omega}{\int_\Omega (U_{\text{exact}})^2 \, d\Omega} \times 100\,(\%)\,. \tag{45}$$

The third example concerns an eigenvalue problem, for which details on the error are provided in the dedicated subsection 8.3. Finally, the fourth example is a patch test in plane elasticity (Sect. 8.4).

8.1. Example 1

A square plate ABCD is subjected to steady-state heat conduction, with boundary conditions illustrated in Figure 16. The exact solution is given by:

$$U(x,y) = U_m \frac{\sinh\left(\frac{\pi y}{2a}\right)}{\sinh\left(\frac{\pi b}{2a}\right)} \cos\left(\frac{\pi x}{2a}\right). \tag{46}$$

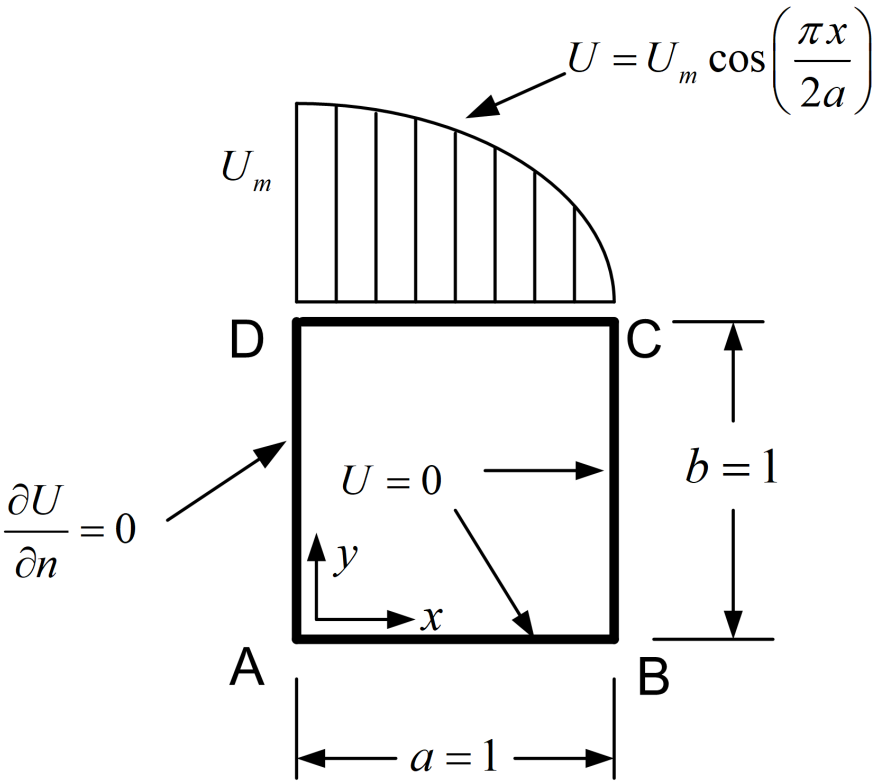


Figure 16. Geometry and boundary conditions.

This problem was solved using the following nine transfinite element types:

- Classical B-spline transfinite element (21-27-33-113 nodes; see Figure 3).
- 12-node element (see Figure 5a).
- 18-node element (see Figure 5b).
- 25-node element (see Figure 5c).
- 27-node element (see Figure 8).
- 81-node element (tensor product; see Figure 10a).

 doi:10.20944/preprints202510.1500.v1

- 80-node element (central node No. 41 is missing; see Figure 14).
- 76-node element (five boundary nodes are missing; see Figure 10b).
- 73-node element (five boundary nodes plus three internal nodes are missing; see Figure 10c).

For the sake of comparison, in addition to B-spline interpolation—which serves as the reference in the present paper—Lagrange and Bernstein polynomials have also been employed in selected cases.

The numerical results are summarized below:

8.1.1. Classical Transfinite Elements (21-27-33-113 Nodes)

These elements belong to the *first class* studied in the present paper. The overall results of the three models mentioned in Sect. 3 (each with 21, 27, 33, and 113 DOFs, shown in Figure 3) are presented in Table 1, which clearly illustrates the convergence trend.

Table 1. Errors (L_2 in %) for the Classical Transfinite Elements shown in Figure 3.

Element type	Model-1 (Lagrange)	Model-2 (Natural B-spline)	Model-3 (De Boor B-spline)
21-node	0.0505	0.5690	0.0579
27-node	0.0464	0.4525	0.0506
33-node	0.0460	0.2177	0.0464
113-node	1.9140×10^{-5}	0.0324	6.2379×10^{-5}

For the sake of brevity, below we discuss only the case of the 113-DOF element (Figure 3d):

- The implementation of Model-1 (Sect. 3.1), based entirely on Lagrange polynomials for both blending and trial functions (of degree $p_\xi = 16, p_\eta = 12$), yields an excellent result: $L_2 = 1.9140 \times 10^{-5}$%, using a 17×13 Gaussian quadrature scheme. The same result was obtained using Bernstein polynomials.
- For Model-2 (Sect. 3.2), using 16×12 integration cells (also referred to as natural B-spline elements), and applying a 4×4 Gaussian quadrature per cell, the error was found to be $L_2 = 0.0324$%.
- Finally, Model-3 (Sect. 3.3) requires 14×10 integration cells (also referred to as Cox–de Boor spline elements), with 4×4 Gauss points per cell, and yields an error ($L_2 = 6.2379 \times 10^{-5}$%) of the same order of magnitude when using either Lagrange or Bernstein polynomials.

8.1.2. Unidirectional 12-18-25-32-Node Elements

These elements belong to the *second class* studied in the present paper (Figure 5). The results for all three element types are presented in Table 2. As the number of nodes increases (from 12 to 25), a monotonic convergence is observed across all types of polynomial and B-spline approximations.

Table 2. Errors (L_2 in %) for the Unidirectional Elements shown in Figure 5.

Element type	Lagrange	Bernstein	De Boor B-spline
12-node	2.6671	2.6654	2.6704
18-node	0.1638	0.1556	0.1567
25-node	0.0385	0.0192	0.0273
32-node[a]	0.0327	0.0039	0.0173
32-node[b]	0.0327	0.0039	0.0182

[a] Nodes 19–25 uniformly distributed; B-spline knot vector: $\Xi = [0,0,0,0,\frac{1}{4},\frac{2}{4},\frac{3}{4},1,1,1,1]$. [b] Nodes 19–25 non-uniformly distributed; B-spline knot vector: $\Xi = [0,0,0,0,\frac{1}{10},\frac{3}{10},\frac{6}{10},1,1,1,1]$.

The 32-node element (shown in Figure 5d) is derived from the 25-node element by subdividing the upper strip into two equal parts, thereby producing *non-uniform* blending functions. When Lagrange or Bernstein polynomials are employed, the six horizontal station positions are uniquely determined at

 doi:10.20944/preprints202510.1500.v1

$\eta = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \frac{7}{8}, 1$. In contrast, the use of B-splines as blending functions is not unique, since six control points can be generated from multiple combinations of breakpoints.

Nevertheless, despite the non-uniform placement of horizontal stations near the top (nodes 19 to 25), adopting the uniform knot vector $H = [0,0,0,0,\frac{1}{3},\frac{2}{3},1,1,1,1]$ yields an error of $L_2 = 0.0173\%$ (last column in Table 2), indicating convergence for the B-spline formulation as well.

Moreover, when the station at $\eta = \frac{7}{8}$ is defined with nodes 19 to 25 placed non-uniformly—e.g., following an *arithmetic progression* measured from the left at $\xi = \{0, \frac{1}{12}, \frac{1}{5}, \frac{7}{20}, \frac{8}{15}, \frac{3}{4}, 1\}$—the Lagrange and Bernstein formulations were affected only in the seventh and sixth decimal places, respectively. In contrast, using the knot vector $\Xi = [0,0,0,0,\frac{1}{10},\frac{3}{10},\frac{6}{10},1,1,1,1]$, the B-spline formulation showed a change in the third decimal place, yielding $L_2 = 0.0182\%$ instead of the previous $L_2 = 0.0173\%$.

8.1.3. Arbitrary-Noded 27-Node Transfinite Element

This element belongs to the *third class* examined in the present study (Sect. 5.1, Figure 8). The L_2 error norms of the numerical solution obtained from the three models applied to the 27-node element are presented in Table 3.

Table 3. Errors (L_2 in %) for the 27-node element shown in Figure **??**.

Lagrange	Bernstein	De Boor B-spline
0.0191	0.0126	0.0245

To demonstrate the effect of mesh non-uniformity, boundary nodes 15–18 and 9–13 were clustered toward the concealed vertex D (node 14) according to an arithmetic progression. The resulting difference in the error norm L_2 (%) was observed only beyond the fifth decimal place.

Similarly, regardless of the location of the boundary nodes, the tensor product of the internal nodes may be either uniform or non-uniform, for example, arranged as shifted Legendre roots or as Gauß–Lobatto–Legendre (GLL) points, as discussed later in Sect. 8.2.1.

8.1.4. T-Spline Elements

I. Proposed formulation

This element belongs to the *fourth class* examined in this paper (Sect. 6). We begin with a tensor-product T-spline element comprising 81 degrees of freedom (DOFs) arranged in a 9×9 configuration. To investigate the impact of vertex reduction on solution accuracy, the number of control points is gradually decreased. The models considered include: one with a missing central node (80 DOFs, Figure 14), one with five missing boundary nodes (76 DOFs, Figure 10b), and another with five boundary and three internal nodes removed (73 DOFs, Figure 10c). In all cases, numerical integration was performed within the 36 cells (in 6×6 setup) which are determined by the seven unique knots (breakpoints) per direction (at $\xi, \eta = [0, 1/6, 2/6, 3/6, 4/6, 5/6, 1]$). In each case, the control points were selected such that the parameterization coincides with the physical coordinates, i.e., $x(\xi,\eta) = \xi$ and $y(\xi,\eta) = \eta$, and thus the determinant of the Jacobian was a constant ($det\mathbf{J} = 1$). The corresponding results of the proposed model are presented in Table 4, which demonstrate a progressive increase in error as the number of DOFs decreases. In contrast, the condition number oscillates in the interval $[38.1, 61.2]$.

Table 4. Errors and condition numbers of proposed T-spline transfinite elements using de Boor B-splines.

	81-node	80-node	76-node	73-node
$\mathbf{L_2}$ (in %)	5.6581×10^{-4}	0.0010	0.0011	0.1120
Condition number	54.2	38.1	61.2	43.2

II. Conventional T-spline

 doi:10.20944/preprints202510.1500.v1

For completeness, the same T-index was employed for the conventional T-spline [17,18]. For the case of 73 nodes (Figure 10c), two matrices, `Uvec` and `Vvec`, each of size 73×5, were constructed to represent the vertex topology. Using this local vertex information, the de Boor tensor-product B-spline basis function

$$T_k(\xi, \eta) = N_{i,3}(\xi) N_{j,3}(\eta) \tag{47}$$

was evaluated over a uniform grid of points. The spatial distribution of the sum $\sum_{k=1}^{73} T_k(\xi, \eta)$ is illustrated in Figure 17, revealing that the partition of unity (PU) property is not satisfied everywhere in the conventional T-spline formulation. In contrast, the proposed method inherently preserves this property.

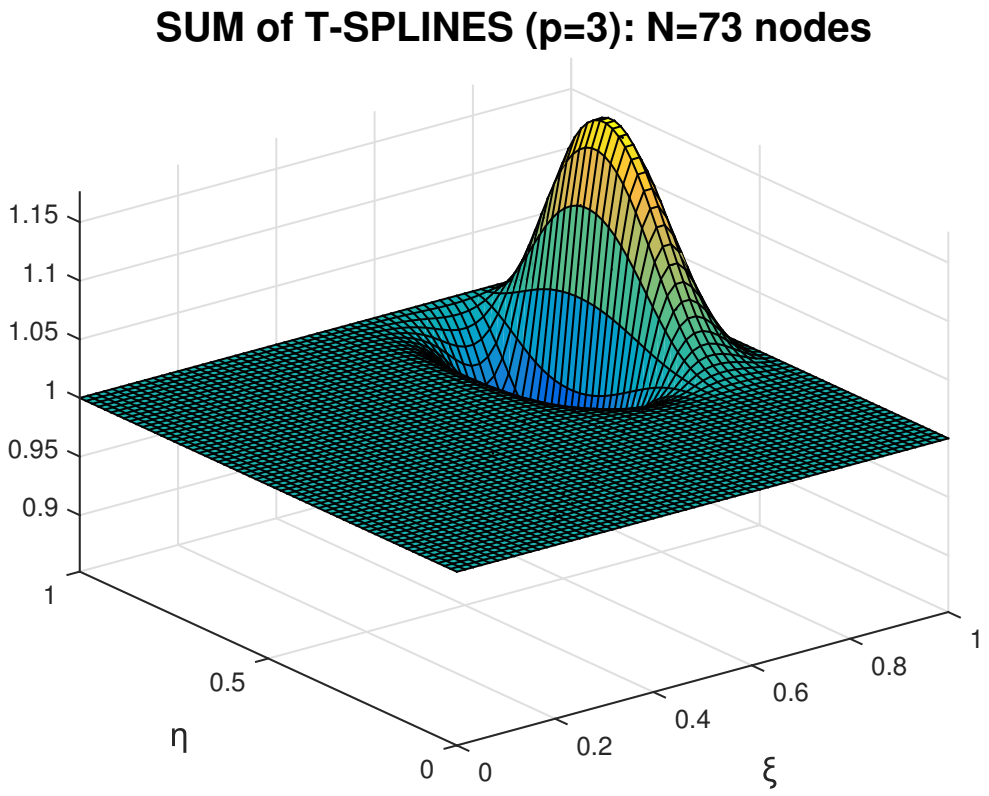


Figure 17. Sum of basis functions in the conventional T-spline, before normalization.

To impose the PU property in the conventional T-spline, at each pair of parameters (ξ, η), Equation (47) is normalized as follows:

$$\tilde{T}_i(\xi, \eta) = \frac{w_i T_i(\xi, \eta)}{\sum_{k=1}^{73} w_k T_k(\xi, \eta)}, \quad i = 1, \ldots, 73\,. \tag{48}$$

Obviously, although the denominator in Equation (48) theoretically includes 73 terms, in reality only a few of them are different than zero (local support property).

Remark: The comparison between the basis functions of this conventional formulation and the abovementioned proposed T-spline is as follows:

1. **Before normalization:** 59 out of the 73 basis functions (Equation (47)) are identical between the two formulations. The remaining 14 functions—specifically those numbered 18, 19, 20, 21, 26, 27, 28, 29, 34, 35, 36, 37, 46, and 53—exhibit differences within the interval $[-0.12, 0.15]$.
2. **After normalization:** Considering the weights in Equation (48) equal to 1 (i.e., $w_i = 1, i = 1, \ldots, 73$), 32 of the 73 basis functions are identical in both formulations. These include the basis

 doi:10.20944/preprints202510.1500.v1

functions associated with vertices numbered 1–17, 24, 25, 32, 33, 40, 41, 42, 48, 49, 56, 57, 58, 65, 66, and 67.

For a fair comparison between conventional T-splines (Equation (48)) and the proposed method, we employed the same control points, namely those of the latter (as explained in the first paragraph of this subsection). The convergence behavior of the conventional T-spline is summarized in Table 5. Similar to the proposed T-spline formulation, the results exhibit monotonic convergence up to a certain level, followed by a progressive increase in error as the number of DOFs decreases. In contrast, the condition number varies in the interval $[39.8, 61.2]$.

Table 5. Errors and Condition numbers for conventional normalized T-spline.

	81-node	**80-node**	**76-node**	**73-node**
$\mathbf{L_2}$ (in %)	5.6581×10^{-4}	0.0454	0.0011	0.1030
Condition number	54.2	39.8	61.2	51.8

Comparing Table 4 with Table 5, one may observe that—for the same control points—the proposed method is competitive to the conventional T-spline in terms of the error norm L_2, while also exhibiting a smaller condition number.

8.1.5. Coons-Patch Elements

This represents a type of element in addition to the four classes mentioned above. Although Coons-patch macroelements have been extensively discussed in numerous papers reviewed in the monograph [9], this model is included here for direct comparison with the models presented in Example 1. For the boundary-node configurations shown in Figure 5a–d (with 10, 13, 16, and 18 nodes, respectively), all illustrated in Figure 18, the computed results are presented in the left portion of Table 6.

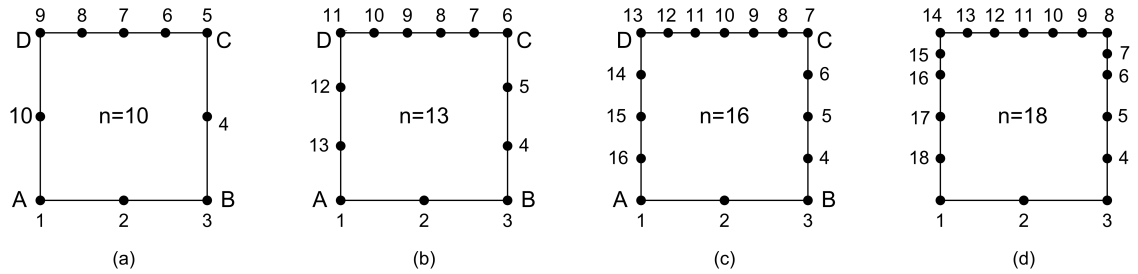


Figure 18. Coons elements.

Table 6. Errors (L_2 in %) of Coons-patch elements using uniform Lagrange polynomials as trial functions, and two alternative sets of blending functions.

First set: $E_1(\xi) = 1 - \xi,\ E_2(\xi) = \xi$				**Second set:** $E_1(\xi) = \cos\left(\frac{\pi\xi}{2}\right),\ E_2(\xi) = 1 - \cos\left(\frac{\pi\xi}{2}\right)$			
10-node	**13-node**	**16-node**	**18-node**	**10-node**	**13-node**	**16-node**	**18-node**
3.4886	2.2919	2.3012	2.3013	2.6577	0.1547	0.0178	7.5040×10^{-4}

For completeness, we examined both the standard *linear* blending functions, $E_1(\xi) = 1 - \xi$, $E_2(\xi) = \xi$ (first set), and the *cosine-like* blending functions, $E_1(\xi) = \cos\left(\frac{\pi\xi}{2}\right)$, $E_2(\xi) = 1 - \cos\left(\frac{\pi\xi}{2}\right)$ (second set). The results indicate that, with linear blending functions, the numerical solution converges to an erroneous value of approximately $L_2 = 2.3\%$, which is substantially larger than the competing values reported in Table 2. In contrast, the second set of blending functions—although heuristic in

 doi:10.20944/preprints202510.1500.v1

nature and inspired by the boundary conditions imposed on the top edge—leads rapidly to the exact solution.

Overall, the results of Example 1 suggest the following:

1. In a fully two-dimensional problem without any symmetry, the conventional Coons interpolation—implemented with linear blending functions—is insufficient for accurately representing the exact solution. Therefore, the inclusion of internal nodes is generally necessary to enhance accuracy.
2. These internal nodes may be arranged along horizontal and/or vertical stations and can follow a transfinite interpolation formula that is applied *globally* across the entire patch.
3. One practical approach is to place internal nodes along horizontal layers (stations), i.e., parallel to the ξ-axis. The station locations may correspond to either uniform or non-uniform η-values, and the associated *blending* functions will be constructed accordingly—based on uniform or non-uniform nodal points (breakpoints).
4. Regardless of whether the blending functions are uniform or non-uniform, the *trial* functions along each station may also be chosen to be uniform or non-uniform.
5. Once the closed-form expressions for the *global* bivariate shape functions have been derived using Lagrange polynomials, they can be readily extended to Bernstein polynomials and B-splines. While the local support property of B-splines affects the resulting bivariate basis functions, splines should be viewed as a specific choice of trial functions for univariate interpolation along each station. The same flexibility applies to the blending functions, which need not be restricted to cardinal types; in addition to Lagrange polynomials, Bernstein polynomials and B-splines are also valid options.

8.2. Example 2

Steady-State Conduction in a Cylindrical Wall (Half-Annulus): This example analyzes a long, hollow cylinder subjected to steady-state heat conduction, with a uniform inner surface temperature of $T_i = 1000°C$ and a uniform outer surface temperature of $T_o = 0°C$. The cylinder, defined by inner and outer radii $R_i = 1$ and $R_o = 32$, is assumed to be insulated to prevent axial heat flow (see Figure 19).

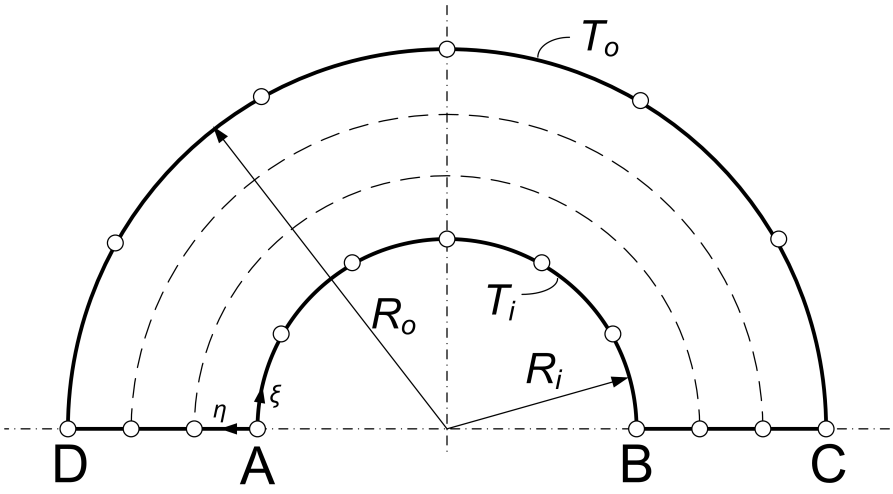


Figure 19. Half-Annulus ($n_\xi = 6, n_\eta = 3$).

According to [79], the steady-state temperature distribution in the radial direction is given by:

$$T(r) = T_i + \frac{(T_o - T_i)}{\log\left(\frac{R_o}{R_i}\right)} \log\left(\frac{r}{R_i}\right) \tag{49}$$

The radii are chosen to produce a sufficiently steep temperature gradient, making this example suitable for a convergence test. This test case was previously presented using single Coons macroele-

 doi:10.20944/preprints202510.1500.v1

ments based on cardinal natural cubic B-splines [8], formulated via truncated power series (i.e., not the procedure shown in A, but mathematically equivalent).

In the present study, we introduce several additional models and clarify all relevant implementation details. Unlike Example 1, where the Coons-patch element failed to converge accurately, here the Coons formulation does converge to the exact solution. Therefore, we adopt a different strategy: we begin with the Coons model in several variations (Sect. 8.2.1) and progressively introduce internal nodes (Sect. 8.2.2).

8.2.1. Coons-Patch Element

In this subsection, we investigate the convergence behavior of several models constructed using a single Coons-patch element, assuming the standard linear blending functions: $E_1(s) = 1 - s$, $E_2(s) = s$, where s denotes either of the parametric coordinates ξ or η.

As reviewed in Ref. [9, Chapter 3], previous studies have employed various cardinal trial functions, including:
(i) piecewise-linear functions,
(ii) cardinal natural cubic B-splines,
(iii) Lagrange polynomials.
In the present work, we extend this framework by also implementing non-uniform Lagrange polynomials and Cox–de Boor B-splines [29–31].

Although each edge of the Coons patch may consist of an *arbitrary* number of nodal or control points (each associated with degrees of freedom), for brevity of the presentation we assume equal number of points on opposite edges, as follows:
– The number of spans along the parallel edges (AB, CD) is denoted n_ξ.
– The number of spans along the edges (BC, AD) is denoted n_η.

For global Lagrange and Bernstein polynomials, the polynomial degrees in each parametric direction are $p = n_\xi$ and $q = n_\eta$, respectively. Interpolation using piecewise-linear or Lagrange polynomials requires $n_\xi + 1$ and $n_\eta + 1$ nodal points per side in the ξ and η directions, respectively. These nodal points correspond to the breakpoints used in the Cox–de Boor formulation. For cubic Cox–de Boor B-splines, the number of control points per edge exceeds the number of breakpoints by two (e.g., $n_{c,\xi} = n_\xi + 3$) [26].

The parameterization of the patch is illustrated in Figure 19:
- Point A is the origin of the axis
- Arc AB defines the ξ-axis
- Segment AD defines the η-axis
- Edges AB and CD are subjected to Dirichlet boundary conditions
- Edges BC and AD are subjected to Neumann boundary conditions ($\partial T/\partial n = 0$), due to symmetry.

In the numerical setup, each circular arc is uniformly subdivided into six breakpoint spans ($n_\xi = 6$). The number of uniform breakpoint spans along the radial direction (e.g., edge AD) is varied incrementally: $n_\eta = 1, 2, 3, \ldots$

Model C1: Piecewise-Linear. The easiest case for implementing a Coons-patch macroelement is the adoption of *piecewise-linear* trial functions. This implies minimal cost in the estimation of these functions and also ensures local support. The vector of unknowns includes $2(n_\xi + n_\eta)$ nodal values, T_i, all of them located on the boundary of the patch. Although this model is generally *not* equivalent to a set of $n_\xi \times n_\eta$ bilinear finite elements, the quality of the numerical solution is sometimes *similar*. In the particular case of Example 2, in which the solution does not depend on the polar angle θ (axisymmetric problem with no angular dependence), the numerical solution of the Coons-patch element with piecewise-linear trial functions *coincides* with that of the FEM solution using bilinear 4-node elements—provided the same mesh density is used. The result is shown in Figure 20 (blue line).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 October 2025 doi:10.20944/preprints202510.1500.v1

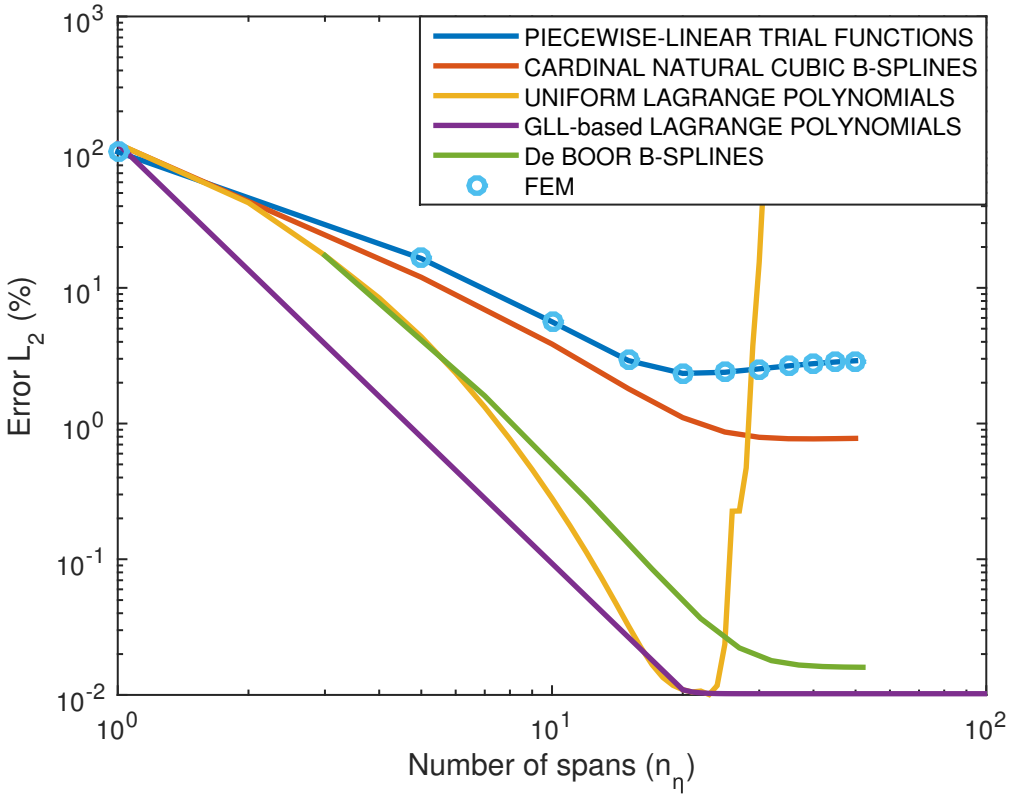


Figure 20. Convergence quality of numerical solution for the circular ring.

Model C2: Cardinal Natural Cubic B-Spline. This model briefly reproduces part of Ref. [8], in which *cardinal natural cubic B-splines* were previously used. Before the imposition of boundary conditions (BCs), the number of nodes is $n_{\text{tot}} = 2(n_\xi + n_\eta)$. Since the $2(n_\xi + 1)$ nodal points on edges AB and DC are restricted under Dirichlet-type BCs, the number of free degrees of freedom becomes $n_{\text{free\_dofs}} = 2(n_\eta - 1)$. The vector of unknowns includes only *nodal values*, T_i, which was the primary reason for adopting this model in the mid-1980s. Numerical integration is performed within the $n_\xi n_\eta$ cells created by nodal breakpoints in both directions. Given the piecewise polynomial degrees $p = q = 3$, the number of Gauss points per integration cell is 6×6 for curvilinear patches (and 4×4 for rectangular ones). The results, shown in Figure 20 (red line), are identical whether the old-fashioned methodology of Ref. [8] (based on truncated powers) or the procedure of A is used—namely, the Curry-Schoenberg formulation [29], computed via the Cox-de Boor algorithm [30,31], with vanishing second derivatives at the ends of each edge.

Model C3: Truncated Power Series. This model extends the previous Model C2 on the same single Coons-patch element, but now additionally incorporates rotational degrees of freedom (DOFs) associated with the second derivatives at the ends of each edge—as illustrated, for example, in Figure A1c. In this case, although an edge such as AB is discretized into n_ξ breakpoint spans (yielding $n_\xi + 1$ breakpoints), the number of associated control points along that edge becomes $n_\xi + 3$. Consequently, after imposing Dirichlet boundary conditions, the final number of free DOFs becomes $n'_{\text{free\_dofs}} = 2(n_\eta + 1)$. In this quantity, eight additional 'rotational' DOFs are considered, though four of them—those along edges AB and DC under Dirichlet BCs—are eliminated. The vector of unknowns includes both nodal values, T_i, and the four curvatures $T(\eta)''$ at the ends of edges AD and BC. The layout and number of integration cells remain the same as in Model C2, defined by the breakpoints. The corresponding results are presented in Figure 20 (green line).

Model C4: Cox–de Boor B-spline. This model continues to use the same Coons-patch element. Let $\xi_1 = 0, \xi_2, \ldots, \xi_{n_\xi+1} = 1$ be the parameters associated with the breakpoints along edge AB. For a piecewise-cubic polynomial, we construct the well-known *knot vector* $\Xi = [0, 0, 0, 0, \xi_1, \ldots, \xi_{n_\xi}, 1, 1, 1, 1]$,

 doi:10.20944/preprints202510.1500.v1

which defines $n_\xi + 3 = n + 2$ basis functions: $N_{1,3}, \ldots, N_{n+2,3}$, where $n = n_\xi + 1$ is the number of breaks along the edge, following the notation of A. The number of DOFs, both before and after applying boundary conditions, is the same as in Model C3, but here they consist of generalized coefficients α_i rather than nodal values. The integration cells are also the same, now with the additional property of *local support*. Despite this difference, the results are *identical*. In other words, although Model C3 lacks compact support (and deals with DOFs: T_i and T''_{end}) while Model C4 possesses it (and deals with DOFs: a_i), the two functional sets are *equivalent*, as confirmed by spline theory and also sustained by numerous test cases. It should be noted that most literature discussing Coons patches in conjunction with B-splines and NURBS refers to the present Model C4, as it is based on the modern spline formulation introduced by Curry and Schoenberg (1966) [29] and implemented via the recursive Cox–de Boor algorithm (1972) [30,31]. Note that, since the *linear* blending functions used in Coons interpolation are identical across all formulations (Lagrange, Bernstein, and B-splines), the B-spline version of the Coons element follows directly.

Model C5: Uniform Lagrange polynomials. This model continues with Coons elements, using *uniform* global Lagrange polynomials as trial functions along each entire edge. Each bivariate function $N_i(\xi, \eta)$ is continuous, influences the entire quadrilateral patch, and is associated with the nodal value T_i. Let p and q denote the polynomial degrees in the ξ- and η-directions, respectively. It can be readily verified that the integrand of each entry in the stiffness matrix, $k_{ij} = \int_\Omega \nabla N_i \nabla N_j \det J \, d\Omega$, is of degree $4p - 1$ per direction—specifically, $2p$ from $\nabla N_i \nabla N_j$ and $2p - 1$ from $\det J$. Consequently, for a rectangular patch $ABCD$, the stiffness matrix and error norm are estimated using a global scheme of $(p + 1) \times (q + 1)$ Gauss points, whereas curvilinear patches such as that in Figure 19 require $2p \times 2q$ Gauss points spanning the entire patch. Due to this, the terms "patch" and "Coons element" are used interchangeably. Closely related terminology such as "macroelement" or "Coons-patch element" has also appeared in previous literature [9]. From the convergence diagram in Figure 20 (orange color), it becomes evident that increasing the polynomial degree p in the η-direction steadily improves numerical accuracy up to $p_{\text{cr}} = 21$, beyond which the solution becomes unstable.

Model C6: Non-uniform Lagrange polynomials. The sixth model studies Coons elements, using *non-uniform* global Lagrange polynomials as trial functions along each entire edge. Following the practice of spectral methods by Karniadakis and Sherwin [32], one of the best choices is the set of *Gauß–Lobatto–Legendre* (GLL) points, which are defined as follows:

$$\xi_i^{\text{GLL}} = \begin{cases} -1 & \text{for } i = 1 \\ \hat{\xi}_i & \text{for } i = 2, 3, \ldots, p \\ +1 & \text{for } i = p + 1\,, \end{cases} \tag{50}$$

where $\hat{\xi}_i$ denotes the roots of the Lobatto polynomial $Lo_{p-1}(\xi)$ of order $p - 1$, which is defined as the first derivative of the Legendre polynomial $L_p(\xi)$ of order p:

$$Lo_{p-1}(\xi) = \frac{dL_p(\xi)}{d\xi}\,. \tag{51}$$

Practically, for any given degree p, the $(p + 1)$ GLL points $\xi_i^{\text{GLL}} \in [-1, 1]$, defined in Equation (50), can be numerically determined by setting `i = p - 1` in the following MATLAB command:

```
roots = vpasolve((legendreP(i,x) - legendreP(i+2,x)) == 0);
```

Based on the above nodal points (images of the GLL points), we can easily construct non-uniform Lagrange polynomials and use them as trial functions for each edge of the quadrilateral patch $ABCD$.

In our case, we kept the uniform subdivision of the circular arcs at $n_\xi = 6$, and adopted GLL-based nodal points along the edges BC and AD. The polynomial degree varied from $p_{\min} = 1$ to $p_{\max} = 99$. It was found that, up to degree $p_{\text{cr}} = 21$, the results are practically the same in both formulations—i.e., the uniform and non-uniform (GLL-based). By further increasing the polynomial degree up to $p_{\max} = 99$, the non-uniform formulation yielded a monotonically decreasing error norm, converging to the value

 doi:10.20944/preprints202510.1500.v1

$L_2 = 0.0102053\%$ (Figure 19). This remaining error is attributed to the discretization of the circular arcs, which are based on seven nodal points per arc (i.e., six uniform nodal spans: $n_\xi = 6$).

Remark: Let us divide the interval $\xi \in [0, 1]$ into n segments (i.e., $n + 1$ nodal points) in the ξ-direction. Regarding global interpolation, one possibility is to introduce $(n + 1)$ Lagrange polynomials $L_i(\xi)$ of degree $p = n$, which leads to integrands in the stiffness and mass matrix entries

$$k_{ij} = \int L_i'(\xi)\, L_j'(\xi)\, d\xi, \quad m_{ij} = \int L_i(\xi)\, L_j(\xi)\, d\xi$$

of degree $p_{\text{integrand}} = 2n$; therefore, the required number of Gauss points for the whole interval is $n_g^{\text{Lagrange}} = n + 1$.

On the other hand, cubic B-spline interpolation over the aforementioned n elements leads to matrix element integrands of degree six, and thus requires four Gauss points per element; this results in a total of $n_g^{\text{B-spline}} = 4n$ Gauss points.

Of course, the stiffness and mass matrices in the Lagrange formulation will be of size $(n + 1) \times (n + 1)$, whereas in the B-spline formulation they will be of size $(n + 3) \times (n + 3)$ (because $n + 1$ breakpoints give $n + 3$ control points). In any case, the B-spline formulation requires more Gauss points than the Lagrange formulation.

Model C7: Finite Element Method (FEM). Based on the pair (n_ξ, n_η), the FEM model using $(n_\xi n_\eta)$ bilinear 4-node elements consists of $(n_\xi + 1) \times (n_\eta + 1)$ nodal points. In the general case in which the solution does not follow the Coons interpolation (like Example-1), the model of piecewise-linear Coons-patch element—with $2(n_\xi + n_\eta)$ nodes—differs from the FEM solution. In other cases such as in Example-2, the piecewise-linear based Coons element is equivalent to the FEM model, and thus both models have the same numerical error L_2.

Overall, the results shown in Figure 20 suggest:

1. All models converge to different values. This fact is mainly due to the incapability of accurately representing the circular arc (for $n_\xi = 6$). Below we start from the less accurate model and end with the most accurate.
2. Piecewise-linear Coons interpolation coincides with the FEM solution. This is because Example-2 is an axisymmetric problem with no angular dependence.
3. The cardinal natural cubic B-spline is characterized by a smaller error than the above piecewise-linear model and FEM.
4. Uniform Lagrange polynomials lead to a smaller error but, after the value $n_\eta = 21$, diverge.
5. De Boor cubic B-splines closely follow the accuracy of uniform Lagrange polynomials up to $n_\eta = 5$, then become less accurate until $n_\eta \approx 25$, and eventually converge to the value $L_2 = 0.0160\%$.
6. The non-uniform Lagrange polynomials based on GLL points rapidly converge to the accurate solution. A small error ($L_2 = 0.0102\%$) remains, due to the incapability of accurately representing the circular arc using $n_\xi = 6$ spans.

In all the above models, it is anticipated that the error norm L_2 will be decreased when the circular arcs are represented through more nodal points or NURBS.

8.2.2. Transfinite Elements

A. Classical Transfinite Elements

Each of the four classical transfinite elements shown in Figure 3 is mapped onto the entire half-annulus depicted in Figure 19. Given the parametric coordinates ξ and η of the nodal points, the corresponding polar coordinates r and θ in the physical domain are obtained as:

$$r = R_i + \eta(R_o - R_i), \tag{52}$$

$$\theta = (1 - \xi)\, \pi\,. \tag{53}$$

 doi:10.20944/preprints202510.1500.v1

The numerical results are reported in Table 7, where convergence behavior can be observed. As in Example 1, the same result was obtained when using Bernstein polynomials.

Table 7. Errors (L_2 in %) of classical transfinite elements using Lagrange or Bernstein polynomials (half-annulus).

21-node	27-node	33-node	113-node
8.5839	4.3548	2.0773	0.1127

Moreover, it is instructive to compare the above results with those obtained for the previously mentioned Model C5 (uniform Lagrange or Bernstein polynomials for a single Coons element) under similar conditions, i.e., with $n_\eta = 4, 5, 6, 12$ subdivisions along the radial direction.

From Figure 20 (orange curve), the corresponding errors are 8.5851%, 4.3559%, 2.3578%, and 0.1127%, respectively.

It is evident that, in Example 2, the aforementioned accuracy of the Coons model is very close to that of the classical transfinite elements (Table 7), primarily due to the independence of the solution from the polar angle θ.

To some extent, the small differences can be attributed to the different discretization of the semi-circular edges in the transfinite patch. It is likely that, if NURBS were employed, these differences would be further reduced.

The aforementioned similarity between classical transfinite elements and the Coons element is observed for all models discussed in Sect. 8.2.1. For instance, both the 113-node transfinite element (with 56 DOFs on the boundary) and its counterpart Coons element (model C4, based on Cox–de Boor cubic B-splines with a total of 56 DOFs) yield the same error norm, $L_2 = 0.2777\%$. This value is approximately 2.5 times larger than the error obtained when using Lagrange polynomials as trial functions (see $L_2 = 0.1127\%$ in Table 7).

B. T-spline Elements

The half annulus shown in Figure 19 was discretized using a uniform tensor-product of 7×7 breakpoints, resulting in a grid of 9×9 control points (81 DOFs), as illustrated in Figure 21a. More specifically, a tensor-product of Cox–de Boor cubic B-splines ($p = 3$) was constructed using a uniform knot vector

$$\Xi = [0, 0, 0, 0, 1/6, 2/6, 3/6, 4/6, 5/6, 1, 1, 1, 1].$$

The parametric space $(\xi, \eta) \in [0, 1] \times [0, 1]$ was uniformly divided into a grid of 9×9 test points (ξ_i, η_i), for $i = 1, \ldots, 81$, and a corresponding uniform mesh of test points $\mathbf{x}_{\text{test}}$ was generated in the physical domain.

 doi:10.20944/preprints202510.1500.v1

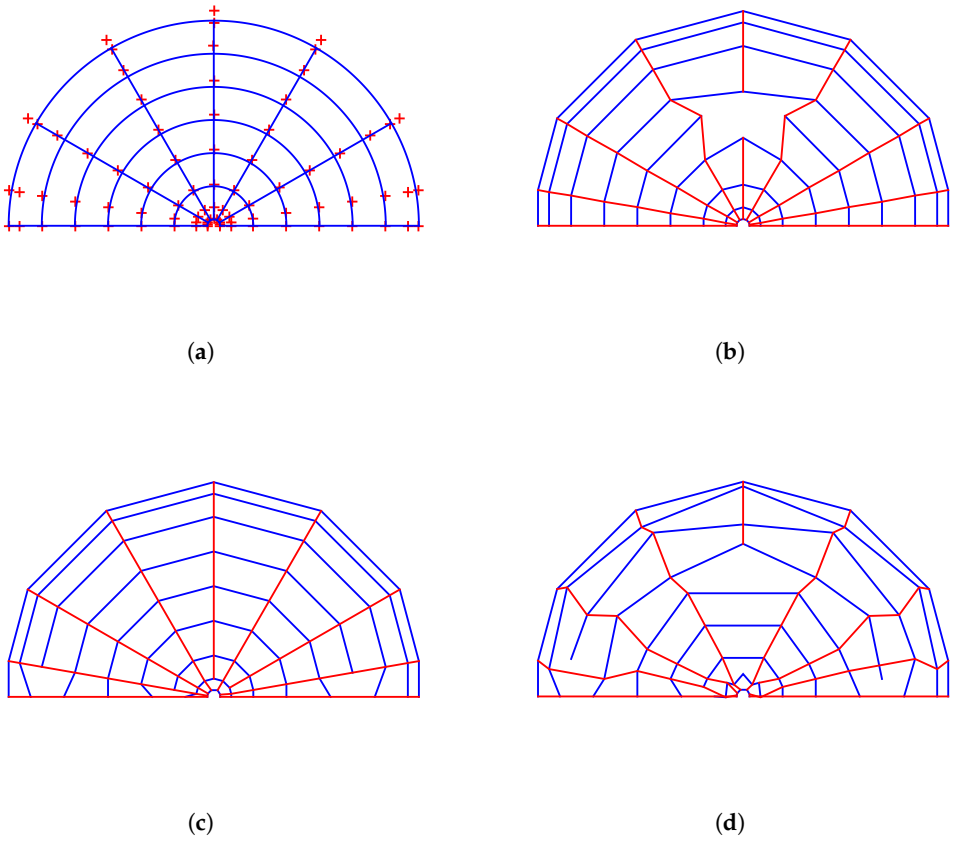


Figure 21. Tensor-product and T-mesh control points of half-annulus: (**a**) Tensor product (81 DOFs): B-spline elements and control points (×). (**b**) T-mesh (80 DOFs). (**c**) T-mesh (76 DOFs). (**d**) T-mesh (73 DOFs).

The system of 81 equations for $(k, l = 1, \ldots, 9)$:

$$\mathbf{x}_{\text{test}}(\xi_k, \eta_l) = \sum_{i=1}^{9} \sum_{j=1}^{9} N_{i,p}(\xi_k) N_{j,p}(\eta_l) \mathbf{x}_c^{ij}, \quad k = 1, \ldots, 9;\ l = 1, \ldots, 9\,, \tag{54}$$

determines the positions of the 81 control points $\mathbf{x}_c$, illustrated by red crosses in Figure 21a. In this way, the B-spline curve (with unit weights), corresponding to each circular arc, passes through nine uniformly distributed points along the arc. Consequently, a small deviation from the exact representation of the arc occurs between these points. Extending this formulation to NURBS is a straightforward task, but it is left to the interested reader.

It was found that the above B-spline tensor product yields the same numerical result, i.e., $L_2 = 1.0494\%$, as that obtained using the corresponding Coons-patch element with the same number of breakpoint spans (see Sect. 8.2.1, Model C4). As previously mentioned, this outcome arises because the problem is axisymmetric and exhibits no angular dependence.

Starting from the abovementioned B-spline tensor product with 81 control points, we proceed as follows. First, we remove the central node, resulting in a T-mesh with 80 nodes (see Figure 14 in the parameter space, and Figure 21b in the physical space). Second, we retain the central node but eliminate five boundary nodes, yielding a model with 76 nodes (see Figure 10b in the reference square, and Figure 21c in the physical domain). Third, we further remove three internal nodes, ultimately producing a T-mesh with 73 nodes (see Figure 10c in parametric space and Figure 21d for the physical annular domain). For all these cases, the results are presented in Table 8, where it can be observed that as the number of removed nodes increases, the error also increases.

 doi:10.20944/preprints202510.1500.v1

Table 8. Errors and condition numbers for proposed T-spline elements (Half annulus).

	81-node	**80-node**	**76-node**	**73-node**
$\mathbf{L_2}$ (in %)	1.0494	1.0521	1.3590	1.7091
Condition number	72.6	51.8	71.0	47.5

For completeness, Table 9 also presents the numerical solution based on the conventional normalized T-spline formulation [18] (Equation (48)). It can be observed that as the number of removed nodes increases, the error also increases.

Table 9. Errors and condition numbers for conventional normalized T-spline (Half annulus).

	81-node	**80-node**	**76-node**	**73-node**
$\mathbf{L_2}$ (in %)	1.0494	1.0264	1.3428	1.6424
Condition number	72.6	54.4	71.2	64.9

8.3. Example 3

Eigenvalues of acoustic cavity: This example deals with the natural frequencies of a rectangular acoustic cavity of size $a \times b = 2.5 \times 1.1$, under Neumann boundary conditions ($\partial p/\partial n = 0$). The governing wave equation is

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \nabla^2 p = 0, \tag{55}$$

where p denotes the acoustic pressure. According to Courant and Hilbert [33], the exact eigenvalues are given by:

$$\lambda_{mn} = \omega_{mn}^2 = \pi^2 c^2 \left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right), \quad m, n = 0, 1, \ldots, \infty \tag{56}$$

The application of the Galerkin procedure to Equation (55) for the eigenvalue problem, leads to the following matrix formulation:

$$[\mathbf{M}]\{\ddot{\mathbf{p}}\} + [\mathbf{K}]\{\mathbf{p}\} = 0\,, \tag{57}$$

where the entries of mass matrix ($[\mathbf{M}]$) and stiffness matrix ($[\mathbf{K}]$) are given by:

$$m_{ij} = \frac{1}{c^2}\int_\Omega N_i N_j \, d\Omega, \quad k_{ij} = \int_\Omega \nabla N_i \cdot \nabla N_j \, d\Omega \tag{58}$$

The computational mesh of 73 nodes was that of Figure 10c, adapted from unit square to a rectangle of dimensions 2.5×1.1. The first eigenvalue (Mode-1) was found equal to zero (modes $m = n = 0$; $\lambda_{00} = 0$) as should be, whereas for the next modes the error (in percent) was calculated by the formula:

$$E_{mn} = \frac{\lambda_{mn}^{\text{calculated}} - \lambda_{mn}^{\text{exact}}}{\lambda_{mn}^{\text{exact}}} \times 100(\%) \tag{59}$$

Using the proposed formulation of the T-spline (according to Sect. 6) and also that of the conventional T-spline (Equation (48))–both of them associated with the 73-node element that is illustrated in Figure 10c–the errors of calculated eigenvalues are shown in Table 10. One may observe that the proposed T-spline outperforms.

Table 10. Errors (E_{mn} in %) of calculated eigenvalues using a single 73-node T-spline element (Rectangular cavity).

MODEL	**Mode-2**	**Mode-3**	**Mode-4**	**Mode-5**	**Mode-6**	**Mode-7**	**Mode-8**
Proposed	0.0002	2.3244	0.0002	0.0003	0.1028	0.5887	0.0657
Conventional	0.0267	2.1646	0.0002	0.0091	1.2588	0.5946	0.5847

 doi:10.20944/preprints202510.1500.v1

For the sake of completeness, in addition we present the results obtained using the B-spline tensor-product of 81 control points (Figure 10a) for degree $p = 3$, and also the assembly of 64 uniform bilinear elements (81 nodes). The results are shown in Table 11, where one may observe that the tensor product B-spline model of 81 nodes is slightly better than both T-spline models of 73 nodes (with the exception of Mode-3 where it outperforms). In addition, all B-spline and T-spline models substantially outperformed compared with the conventional bilinear FEM model of 81 nodes.

Table 11. Errors (E_{mn} in %) of calculated eigenvalues using tensor-product and conventional FEM (81 nodes).

MODEL	Mode-2	Mode-3	Mode-4	Mode-5	Mode-6	Mode-7	Mode-8
Tensor-product	0.0001	0.0054	0.0001	0.0001	0.1021	0.0024	0.0649
FEM	1.2916	5.2387	1.2916	1.2916	4.9061	9.9833	8.0973

8.4. Example 4

Patch test in plane elasticity: Among several patch tests [34], we present here the case of a unit square under plane stress, with Dirichlet boundary conditions $u(\xi, \eta) = 0$, $v(\xi, \eta) = \eta$, where $0 \leq \xi, \eta \leq 1$. For the 12-node element (Figure 5a), the implementation of Lagrange polynomials (Equation (19)) or Bernstein polynomials (Equation (20)) yields exactly the theoretical values at the internal nodes 5 and 6 (i.e., horizontal displacements $u = 0$, and vertical displacements $v = 0.5$). Moreover, for elastic modulus $E = 1$, the stress throughout the element coincides exactly with the theoretical values, namely

$$\sigma_{xx} = \frac{\nu E}{1 - \nu^2} = 0.3297, \quad \sigma_{yy} = \frac{E}{1 - \nu^2} = 1.0989, \quad \tau_{xy} = 0\,.$$

Similarly, we tested the case where node 2 (Figure 5a) was fully supported, while nodes 1 and 3 were constrained to roll horizontally. The top edge 8–9–10–11–12 was uniformly loaded in tension, and the vertical edges were free-free. In this setting, both the Lagrange (Equation (19)) and Bernstein (Equation (20)) polynomials yielded the theoretical values, i.e., $\sigma_{xx} = 0$, $\sigma_{yy} = 1$, and $\tau_{xy} = 0$.

Regarding the implementation of B-splines on the 12-node element (Figure 5a), the same accurate result as above was obtained, despite its hybrid nature. One illustrative example is the following. The *blending* function along the vertical (η) direction was taken as a Bernstein polynomial of degree $p_y = 2$, since a cubic B-spline is not applicable. The *trial* functions were chosen as follows:

1. On the bottom layer (consisting of 3 nodes: 1–2–3), a non-cardinal Bernstein polynomial of degree $p_x = 2$ was used.
2. On the middle layer (consisting of 4 nodes: 4–5–6–7), a Bernstein polynomial of degree $p'_x = 3$ was employed. Alternatively, one could adopt a quadratic B-spline ($p''_x = 2$) with knot vector $\Xi = [0, 0, 0, \frac{1}{2}, 1, 1, 1]$.
3. On the top layer (consisting of 5 nodes: 8–9–10–11–12), a cubic B-spline ($p_x = 3$) was used with knot vector $\Xi' = [0, 0, 0, 0, \frac{1}{2}, 1, 1, 1, 1]$.

Further results will be reported in a future paper, dedicated to elasticity problems.

9. Extension to Collocation Method

Retaining the bivariate global shape (or basis) functions of the transfinite elements discussed in the previous sections, the Galerkin method can be readily replaced by the Collocation Method. This alternative has been explored in a series of studies since 2007 (see [9, Chapter 11] for an overview, including IGA-based papers since 2010), with representative applications to potential and elasticity problems presented in [35] and [36], respectively.

In the context of transfinite elements using Lagrange polynomials for both blending and univariate trial functions, collocation points can be selected as either Gauss or Chebyshev points, defined globally over the entire patch. This formulation offers the flexibility to choose a number of collocation points equal to the number of unknowns. Notably, it has been demonstrated that the computed eigenvalues

 doi:10.20944/preprints202510.1500.v1

remain unchanged whether a *uniform* nodal mesh is used in conjunction with Gauss/Chebyshev collocation points, or a *non-uniform* mesh is constructed by placing nodes at the mapped roots of Legendre or Chebyshev polynomials and applying nodal collocation directly.

In contrast, B-spline-based collocation requires a standard elementwise distribution of collocation points within each B-spline element across the patch. In general, this results in a number of collocation points exceeding the number of unknowns, thereby necessitating a least-squares solution procedure. Alternatively, following the pioneering work of de Boor [26], Greville or Demko abscissae can be globally selected as collocation points across the entire patch.

An additional challenge inherent to the collocation method is the imposition of mixed boundary conditions, where segments of the boundary are governed by Dirichlet conditions while others are subject to Neumann conditions. This complicates the consistent enforcement of boundary constraints. The optimal handling of this issue—along with the assessment of the performance of alternative basis functions such as trigonometric or Hermite B-splines [37–39]—remains an open area for future investigation.

10. Discussion

This paper demonstrates that transfinite elements are not limited to Lagrange polynomials but can also be formulated using B-splines, which may serve as both blending and trial functions. In general, all three formulations—Lagrange polynomials, Bernstein polynomials, and B-splines—yield basis functions with identical closed-form expressions. Consequently, the term *macroelement* has been adopted to describe the unified framework represented by a collection of sixty papers documented in Ref. [9]. Within this framework, the entire patch can be treated as a single macroelement, regardless of the specific type of blending or trial functions employed. This paper supports the conjecture that the classical blending functions used in Coons–Gordon patches can be replaced by Bernstein polynomials or B-splines, thereby enabling the seamless transfer of established techniques into the isogeometric analysis (IGA) domain.

The use of B-splines is inherently characterized by their local support, meaning that within each integration cell—defined between adjacent breakpoints—only a limited number of basis functions are non-zero. However, in this paper, we did not focus on bandwidth optimization; rather, our primary objective was to establish a proof of concept demonstrating that B-splines are fully applicable within the framework of transfinite elements. This applicability was verified across at least four distinct classes, ranging from classical transfinite elements, such as those of Gordon and Hall [5], to more advanced constructs like T-splines. For completeness, the Coons–patch macroelement was studied as well.

The outcome of the present study is the culmination of a long-term effort and accumulated experience that dates back to mid-1984. At that time, the author was co-supervising a PhD thesis—which was ultimately never defended—and subsequently a Master's thesis in 1988, both conducted within the CAD/CAE group of the Mechanical Engineering Department at the National Technical University of Athens. Further historical context is provided in Ref. [9, Chapter 14]. The group's first publication appeared in 1989 [8]; however, due to various circumstances, the second publication—part of the collection mentioned in the first paragraph of this section—was delayed until 1999 (see Ref. [9]).

Motivated by concerns related to the Runge phenomenon, the group initially developed a Coons macroelement based on cardinal natural cubic B-splines defined along each edge of the Coons patch (explained in A). Considerable time and deliberation were required before adopting Lagrange polynomials, which ultimately proved to offer excellent performance, even when constructed using uniformly spaced nodal points. Based on our experience, we propose that potential and elasticity problems can be effectively addressed using Gordon's transfinite interpolation, with polynomial degrees of up to 8 and 10, respectively. Today, we also advocate for the use of spectral methods, which mitigate the excessive end-point oscillations often encountered in high-degree polynomial approximations (see Ref. [40] for further details, and also refer to Model C6 around Equation (50)).

 doi:10.20944/preprints202510.1500.v1

The deeper motivation behind this research stems from our conviction that, since isogeometric analysis (IGA) operates over entire patches, it is fundamentally aligned with the general principles of transfinite interpolation. This perspective suggests that IGA is not an entirely distinct methodology, but rather a natural evolution or extension of transfinite concepts. Preliminary reflections on this connection were outlined in our earlier monograph (see Ref. [9, pp. 216 and 555], among others), while a more recent review paper is due to Provatidis et al. [41].

An early attempt to adapt transfinite interpolation for the accurate representation of conic sections and quadrics was presented in Ref. [42], where weights were determined via nonlinear optimization. In that study, it was—almost intuitively—found that a direct substitution of Lagrange polynomials with their Bernstein counterparts produced nearly identical and highly accurate numerical results. This observation further supported the idea that transfinite formulations are inherently compatible with the foundational principles of isogeometric analysis (IGA).

Since then, significant progress has been made:

First, it was proven that the substitution of Lagrange polynomials with Bernstein polynomials of the same degree is fully compatible in both Coons elements and classical transfinite elements, such as the 113-node element shown in Figure 3. Specifically, the existence of a transformation matrix between the two sets of basis functions was revealed [20].

Second, it was shown that elements with structured interior nodes but irregular boundary node distributions do not admit the aforementioned transformation matrix. Nevertheless, replacing the Lagrange polynomials with Bernstein polynomials of the same degree yields slightly improved results. This observation was explained by extending the formulation of the transfinite interpolation formula with respect to the choice of blending functions and trial functions [16].

Third, a mechanism was identified that enables the application of the transfinite interpolation formula to T-meshes, but only with Lagrange [21] and Bernstein [20] polynomials. Notably, it was revealed that this can be achieved by enforcing constraints both through the transfinite interpolation formula and with the aid of a tensor product defined on the background mesh.

The present paper completes the aforementioned efforts by demonstrating the use of B-spline interpolation both as blending functions and as trial functions, treating the entire patch as a single transfinite element. This approach is illustrated across four classes of elements: classical transfinite elements, one-directional elements with nodes arranged in parallel layers, elements with structured interiors but irregular boundaries, and, finally, T-splines.

We now refer to these four classes:

First class: The classical 21-, 27-, 33-, and 113-node transfinite elements were successfully treated using B-spline interpolation, together with integration cells automatically determined from the mesh density along the horizontal and vertical stations containing the nodal points.

Second class: The unidirectional transfinite element is also handled automatically by considering a single subdivision in the vertical direction, while in the horizontal direction—along which the layers of nodes are aligned—the union of knot spans is readily determined using the MATLAB function `unique`.

Third class: The transfinite element with a structured interior arranged in a tensor-product pattern but with arbitrarily distributed boundary nodes extends the previous approach by considering stations in both directions. The advantage of using the MATLAB function `unique` to identify knot spans for numerical integration of the stiffness matrix remains applicable.

Fourth class: Regarding the class of transfinite elements known as T-splines—possibly first considered in Ref. [6] using Hermite splines—prior studies have addressed the enforcement of constraints at the missing nodes within the T-mesh. Within this framework, Lagrange polynomials can be systematically treated on each segment containing a missing node, with an analogous procedure applicable to Bernstein polynomials. Notably, in both cases, the constraint formulation—linking the state of an incomplete station to that of a complete station—remains identical [16,20].

The present study extends this approach by employing concepts from computer-aided geometric design (CAGD), specifically one-dimensional knot insertion theory, to impose analogous constraints.

 doi:10.20944/preprints202510.1500.v1

For boundaries featuring missing nodes, the transformation matrix is readily derived (cf. B), given the independence of each edge from the remainder of the patch; consequently, knot insertion on the curve representing the edge is well-defined and effective.

In this work, each station is treated as an individual curve with associated data, which leads to the assumption that the same type of transformation matrix applies uniformly to all horizontal and vertical stations (cf. B). Furthermore, at vertices corresponding to the intersection of two stations, the mean average of the contributing stations was employed. These assumptions, previously validated for both Lagrange and Bernstein polynomials [20,21], form the foundation of the present methodology.

The advantage of the proposed transfinite element—compared to the conventional T-spline method [18,19] (nowadays of great interest, e.g., [43,44])—is that it *a priori* satisfies the partition of unity property, thus eliminating the need for normalization of the resulting basis functions. In fact, as noted at the end of Sect. 8.1, the proposed methodology produces basis functions that are closer to the raw (unnormalized) set than to the normalized one. Nevertheless, the error norm obtained using the proposed method for the solution of Laplace equation is of the same order of magnitude with the conventional normalized T-spline, but the numerical solution is characterized by a smaller condition number. In the solution of the eigenvalue problem, the proposed T-spline outperforms. A potential disadvantage, however, is that the proposed method does not *a priori* guarantee the initial parameterization of object's shape under analysis.

In all computations reported here, when the patch is a rectangle of size $L_x \times L_y$, the control points were chosen so that at the vertices of the parametric unit square, the physical coordinates satisfy $x(\xi,\eta) = L_x\xi$ and $y(\xi,\eta) = L_y\eta$, which leads to a constant Jacobian everywhere. A similar technique—using the mapping between uniform test points on both the reference unit square and the true domain—was applied for the half-annulus of Example 2 (Sect. 8.2). Therefore, the implemented computer codes perform equally well for curvilinear patches as well.

The treatment of circular (or possibly elliptic) parts is better accomplished using NURBS with predefined control points and associated weights, rather than the inverse engineering approach applied here for estimating control points that accurately represent these curves only at distinct uniform positions. However, this is a straightforward task that may be carried out by the interested reader to observe firsthand the improvement resulting from this modification.

Although the present paper provides self-contained MATLAB codes, it is not difficult to incorporate the proposed formulations—Bernstein, B-spline, and the NURBS version—into popular academic software such as IGAFEM [45] and GeoPDEs [46].

The proposed procedure leads to the construction of basis functions that can be directly applied to the estimation of mass and stiffness matrices for the numerical solution of potential and elasticity problems in the continuum. Both static, dynamic, and coupled problems (e.g., thermoelasticity, fluid–structure interaction) can be solved by following standard FEM/IGA algorithms. A common feature of these problems is that only C^0 continuity is required between the proposed elements in an assembly. In contrast, for shells and plates requiring C^1 continuity, the formulation presented in this paper must be extended. A brief summary of past research, along with ongoing work, is presented in the following three paragraphs.

The original report by Coons [7] introduced two interpolation formulas for the function $w(\xi,\eta)$ over a curvilinear patch. The first formula is based on prescribed values of w along the four boundaries of the patch; this case was thoroughly discussed in the present paper. The second formula involves prescribed values of w and its partial derivatives $\partial w/\partial n$ on the four edges of the patch. Based on this second interpolation formula, Provatidis and Angelidis [47] constructed a rectangular Coons-patch macroelement (with nodes located only on the boundary) suitable for thin plate-bending analysis. Following Coons [7], the four blending functions per direction were chosen as cubic Hermite polynomials (two cardinal and two non-cardinal, as in beam bending), while the trial functions were Hermite polynomials of degree $(2n-1)$, with n denoting the number of nodes along an edge. This element was tested in six examples with simply supported or clamped edges, subjected to concentrated

 doi:10.20944/preprints202510.1500.v1

or distributed loads. Despite using only boundary information, in most cases the deflection error was reported to be less than 0.2%. Furthermore, it was shown that the lowest-order version of this transfinite element—using cubic Hermite polynomials as both blending and trial functions with two nodes per edge ($n = 2$)—reduces to the well-known Bogner–Fox–Schmit (BFS) element [48].

In a later study ([9, Chapter 9]), it was demonstrated that by selecting the same degree of Hermite polynomials for both blending and trial functions, transfinite interpolation degenerates to conventional tensor-product Hermitian interpolation. In this case, the results for both static and eigenvalue problems were found to be exceptionally accurate. By analogy to the present paper, where the treatment of internal nodes deviating from the tensor-product structure has been addressed, Hermite-based transfinite plate-bending elements with slightly unstructured interiors (such as the one shown in Figure 1e) can be readily constructed in a similar manner, although no published report currently exists. Despite this flexibility, Hermite interpolation cannot naturally accommodate non-rectangular plate elements, which motivates the use of a different approach, as discussed in the next paragraph.

To overcome the limitations of Hermite polynomials and to accommodate smooth curvilinear patches such as circular plates, the use of B-splines and NURBS has been proposed [9, pp. 407–412]. The central idea is to express the deflection as a tensor product of B-splines (or NURBS), namely

$$w(\xi, \eta) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,n}(\xi)\, N_{j,m}(\eta)\, a_{ij},$$

where a_{ij} denote generalized coefficients. This formulation has been shown to provide excellent accuracy for both static and dynamic problems. Nevertheless, within this framework, the treatment of non-tensor-product B-spline/NURBS elements using non-cardinal blending functions (either B-splines or NURBS) remains an active area of research. From an industrial perspective, the important aspects of the proposed formulation that favor its practical adoption are: (i) the partition of unity property without the need for normalization, (ii) the capability to handle unstructured grids, and (iii) the straightforward joining of patches. Moreover, after the analytical determination of the basis functions, the proposed method was found to be ten to twenty times faster than the conventional T-spline; however, this improvement may be related to the choice of B-spline subroutines (e.g., MATLAB function `spcol`) or to specific programming implementations. These features have recently attracted considerable interest from the research community (e.g., [49–51]).

This study is also connected to recent advances in multi-material and isogeometric-inspired topology optimization, where some remarkable contributions include [52–56].

In summary, the present study advances the theory and implementation of transfinite interpolation by integrating B-spline and NURBS-based formulations within a unified framework that accommodates a wide range of element classes, including classical transfinite elements, structured and irregular patches, and T-splines. By leveraging concepts from CAGD and knot insertion theory, the methodology ensures compatibility with isogeometric principles while preserving desirable properties such as partition of unity. Although the initial parameterization is not inherently guaranteed, practical strategies have been demonstrated to achieve consistent mappings and accurate geometric representations. Future work may focus on extending this framework to three-dimensional patches, refining control point estimation for complex geometries, and integrating the proposed formulations into broader isogeometric analysis pipelines. The flexibility and modularity of the approach make it a promising candidate for further exploration in both academic and industrial settings.

11. Conclusions

This work demonstrates that classical transfinite finite elements can be reformulated using Bernstein polynomials and B-splines, enabling a unified framework for both structured and partially unstructured grids, including unidirectional layouts and T-meshes. By replacing Lagrange polynomials with Bernstein or B-spline bases and expressing transfinite interpolation projectors in B-spline form, the method allows independent control of horizontal and vertical stations and facilitates the imposition

 doi:10.20944/preprints202510.1500.v1

of linear constraints along patch boundaries and internal mesh-lines using established knot insertion techniques. Two complementary strategies—transfinite interpolation and tensor-product background grids—lead to automatic construction of blending functions that satisfy the Partition of Unity Property without normalization. The approach further simplifies the system by eliminating secondary degrees of freedom in favor of primary ones, resulting in a flexible and computationally efficient formulation. Convergence studies in two examples of potential problems, eigenvalue acoustic analysis in a third example, and a patch test in plane elasticity have shown that the proposed approach performs robustly for both uniform and non-uniform grids. The proposed formulation can be extended to plate-bending and shell analysis as well as to three-dimensional problems.

Funding: "This research received no external funding"

Data Availability Statement: Available upon request.

Conflicts of Interest: "The authors declare no conflicts of interest."

Appendix A. Natural Cubic B-Splines

Let us consider the interval $\xi \in [0,1]$, which consists of n single breakpoints (simply 'breaks'): $[\xi_1, \ldots, \xi_n]$. If the breaks are treated as nodal points, Lagrange polynomials $L_1(\xi), \ldots, L_n(\xi)$ of degree $(n-1)$ can be constructed, associated with nodal values $(U_1, U_2, \ldots, U_n)$. On the other hand, when using a cubic spline with polynomial degree $p = 3$, the same n breaks generate $n_{\text{ctrl}} = n + 2$ control points, associated with coefficients $(a_1, \ldots, a_n, a_{n+1}, a_{n+2})$. In other words, the number of coefficients exceeds the number of breaks by 2. This is because, in addition to the n nodal values $(U_1, \ldots, U_n)$, two extra degrees of freedom appear at the ends of the interval $[0,1]$, typically corresponding to the second derivatives $U''(0)$ and $U''(1)$.

To reduce the number of degrees of freedom from $n+2$ to n, thereby matching the number of breaks, many studies have imposed vanishing curvature (i.e., zero second derivative) at the end-points. This introduces two additional equations, resulting in what are known as 'natural' cubic B-splines—analogous to a beam in bending with no moments applied at its ends. Although these concepts were originally developed using the power series formulation of B-splines (Schoenberg, 1946), a more accessible construction method for natural cubic B-splines is presented below.

First, consider the knot vector $\Xi = [0,0,0,0 = \xi_1, \ldots, \xi_{n-1}, \xi_n = 1,1,1,1]$. Using this, we construct the classical cubic B-splines $N_{i,3}(\xi)$ for $i = 1, \ldots, n+2$, which satisfy the relationship:

$$U(\xi) = \sum_{i=1}^{n+2} N_{i,3}(\xi) a_i \,. \tag{A1}$$

Equation (A1) is collocated at the n breaks. Moreover, the second derivative is given as

$$U''(\xi) = \sum_{i=1}^{n+2} N''_{i,3}(\xi) a_i \,. \tag{A2}$$

The totality of $n+2$ equations is written in matrix form as

$$\mathbf{U}_{\text{brk}} = \mathbf{A}\mathbf{a} \,, \tag{A3}$$

where:

 doi:10.20944/preprints202510.1500.v1

$$\mathbf{U}_{\text{brk}} = \begin{bmatrix} U_1 \\ \vdots \\ U_n \\ U_1'' \\ U_n'' \end{bmatrix} \tag{A4}$$

$$\mathbf{A} = \begin{bmatrix} N_{1,3}(\xi_1) & N_{2,3}(\xi_1) \dots & N_{n+2,3}(\xi_1) \\ N_{1,3}(\xi_2) & N_{2,3}(\xi_2) \dots & N_{n+2,3}(\xi_2) \\ \vdots & \vdots & \vdots \\ N_{1,3}(\xi_n) & N_{2,3}(\xi_n) \dots & N_{n+2,3}(\xi_n) \\ N_{1,3}''(\xi_1) & N_{2,3}''(\xi_1) \dots & N_{n+2,3}''(\xi_1) \\ N_{1,3}''(\xi_n) & N_{2,3}''(\xi_n) \dots & N_{n+2,3}''(\xi_n) \end{bmatrix} \tag{A5}$$

The inversion of Equation (A3) implies

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{U}_{brk}\,, \tag{A6}$$

Equation (A1) is unfolded as:

$$U(\xi) = [N_{1,3}(\xi) \dots N_{n+2,3}(\xi)]\mathbf{a}\,. \tag{A7}$$

Substituting Equation (A6) into Equation (A7), we receive:

$$U(\xi) = \left([N_{1,3}(\xi) \dots N_{n+2,3}(\xi)]\mathbf{A}^{-1}\right)\mathbf{U}_{brk}\,. \tag{A8}$$

The term within the parentheses of Equation (A8) is evidently a row vector containing the natural cubic B-splines (shape functions). These shape functions are directly associated with the nodal values represented by the column vector $\mathbf{U}_{\text{brk}}$.

At this point, we distinguish between two modeling options:

1. **Including end curvatures:** If the curvatures (second derivatives) at the boundaries of the interval $[0, 1]$ are retained, the formulation involves $n + 2$ shape functions and correspondingly $n + 2$ degrees of freedom. These consist of the n nodal values shown in Figure A1b, along with the two boundary curvatures illustrated in Figure A1c.
2. **Natural B-spline assumption:** If the boundary curvatures are assumed to vanish—consistent with the natural B-spline formulation—the last two shape functions (out of the $n + 2$) are effectively multiplied by zero and thus do not contribute to the solution. Consequently, the number of active shape functions reduces to n, as depicted in Figure A1b. It is worth noting that this reduced functional set, comprising n functions $(N_1, \dots, N_n)$, satisfies the partition of unity property, ensuring rigid-body consistency. A MATLAB® implementation for $n = 11$ is provided in Table A1, which also generates the graphical results shown in Figure A1.

 doi:10.20944/preprints202510.1500.v1

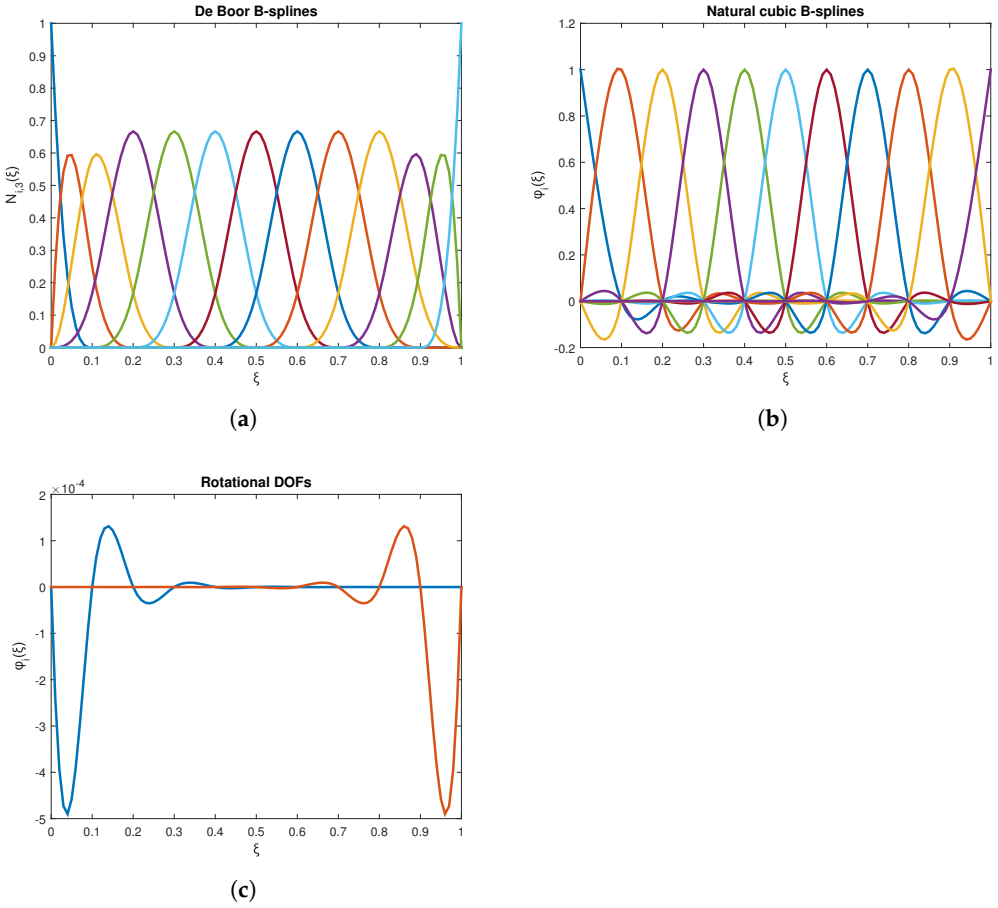


Figure A1. Basis functions for 11 uniform breaks: (**a**) de Boor B-splines, (**b**) Natural cubic B-splines, (**c**) Basis functions associated with the two rotational DOFs at the ends.

Obviously, the set of de Boor—more precisely, Curry-Schoenberg (1966) [29]—cubic B-splines depicted in Figure A1a is equivalent to the complete set of natural cubic B-splines shown in Figure A1b, augmented by the two basis functions associated with the boundary curvatures illustrated in Figure A1c. This equivalence highlights a critical distinction: earlier analyses that relied solely on natural cubic B-splines, often supplemented by power series expansions (e.g., [8]), inherently lack the full representational capacity of the complete de Boor spline basis. As a result, such approaches may yield solutions of inferior fidelity compared to those constructed using the full set of de Boor B-splines.

 doi:10.20944/preprints202510.1500.v1

Table A1. Typical MATLAB code for calculating and plotting cubic B-splines $N_{i,p}(\xi)$ and natural B-splines N_i.

```
%% PROGRAM TO DERIVE NATURAL CUBIC BSPLINES
% In the interval [0,L], a number of internal breakpoints (nbrksIn),
% the polynomial degree (p), and the multiplicity is given:
L = 1;              % length of interval
p = 3;              % polynomial degree
multiplicity = 1;   % Number of knots per inner breakpoint
nbrksIn = 9;        % Number of inner breakpoints
nbrks = 2 + nbrksIn; % TOTAL number of breakpoints (2 ends + inner)
nodes = nbrks;
knots = augknt(linspace(0,L,nodes),p+1,multiplicity); % knot sequence
numknots = size(knots,2); % Number of knots
fprintf('Number of knots = %3i\n', numknots);
nctrlpoints = numknots - (p+1);
fprintf('Number of control points = %3i\n', nctrlpoints);

% Points where the basis functions will be calculated:
Ndivisions = 100;
tau = linspace(0,L,Ndivisions+1);

%% B-Splines
nstep=p;
colmat1=spcol(knots,p+1,brk2knt(tau,nstep));
[i1,i2]=size(colmat1);
Basis1 = colmat1(1:nstep:i1,:);

% Plot basis functions
plot(tau,Basis1,'LineWidth',2)
xlabel('cs')
ylabel('Ni,p')
title(['P = ', int2str(p)])
legend('N1p','N2p','N3p','N4p','N5p','N6p','N7p','Location','best')

%% NATURAL BSPLINES
% MATRIX [A]
for i=1:nbrks
tau0 = knots(p+i);
A(i,1:nctrlpoints) = spcol(knots,p+1,brk2knt(tau0,1));
end
tau1=0; tau2=1;
basis=spcol(knots,p+1,brk2knt(tau1,3)); A(nbrks+1,1:nctrlpoints)=basis(3,:);
basis=spcol(knots,p+1,brk2knt(tau2,3)); A(nbrks+2,1:nctrlpoints)=basis(3,:);
Ainv = inv(A);

% Calculate and plot natural B-spline
N=[];
for i=1:length(tau)
x=tau(i);
lineN=spcol(knots,p+1,brk2knt(x,1));
for j=1:nbrks+2
N(i,j) = lineN * Ainv(1:nbrks+2,j);
end
end
figure(2)
plot(tau,N(:,1:nbrks),'LineWidth',2)
figure(3)
plot(tau,N(:,nbrks+1:nbrks+2),'LineWidth',2)
```

Appendix B. Knot Insertion in One-Dimension

Let us consider the initial knot vector Ξ_P in the interval $[0,1]$:

$$\Xi_P = \{\xi_1, \xi_2, \ldots, \xi_{m_P}\}, \tag{A9}$$

while let the n_P associated control points be

$$\mathbf{P} = \{P_1, P_2, \ldots, P_{n_P}\} \tag{A10}$$

with

$$n_P = m_P - (p+1). \tag{A11}$$

In general, if we insert a knot at point ξ with $\xi \in [\xi_k, \xi_{k+1})$, the updated control points will be given by the linear relationship:

$$Q_{i+1} = (1 - a_{i+1})P_i + a_{i+1}P_{i+1}, \tag{A12}$$

 doi:10.20944/preprints202510.1500.v1

where

$$\alpha_{i+1} = \frac{\xi - \xi_{i+1}}{\xi_{i+1+p} - \xi_{i+1}}, \quad \text{for} \quad k - p + 1 \leq i + 1 \leq k\,. \tag{A13}$$

Equation (A13) means that the control points $(P_1, \ldots, P_{k-p})$ remain as are, and thus $Q_1 = P_1, \ldots, Q_{k-p} = P_{k-p}$. Then, p new control points (i.e., $Q_{k-p+1}, \ldots, Q_k$) are introduced according to Equation (A12). At the end, the control points $(P_{k+1} \ldots P_{n_P})$ remain as are, and thus $(Q_{k+1} = P_k, \ldots, Q_{n_P+1} = P_{n_P})$.

Applying Equation (A12) successively for some times, the linear relationship between the new (**Q**) and the old (**P**) control point becomes:

$$\mathbf{Q} = \mathbf{T}^t \mathbf{P}\,. \tag{A14}$$

where **T** is the transformation matrix, produced by the successive implementation of Equation (A12).

In the context of IGA, since the basic assumption of isoparametric property is adopted, the relationship between the new ($\mathbf{a}_Q$) and the old ($\mathbf{a}_P$) coefficients will be:

$$\mathbf{a}_Q = \mathbf{T}^t \mathbf{a}_P\,. \tag{A15}$$

On the other hand, let the initial set of basis functions be

$$\mathbf{N}_P = \left\{ N_{P_1}, N_{P_2}, \ldots, N_{P_{n_P}} \right\}, \tag{A16}$$

while the new set of basis functions be

$$\mathbf{N}_Q = \left\{ N_{Q_1}, N_{Q_2}, \ldots, N_{Q_{n_Q}} \right\}, \tag{A17}$$

with $n_Q > n_P$.

Since the shape and the parameterization is the same (Piegl and Tiller [57]), we have

$$\mathbf{x}(\xi, \eta) = \mathbf{N}_P^t \mathbf{P} = \mathbf{N}_Q^t \mathbf{Q}\,. \tag{A18}$$

Substituting **Q** from Equation (A14) into Equation (A18), we eventually obtain:

$$\mathbf{N}_P = \mathbf{T}\mathbf{N}_Q\,. \tag{A19}$$

Equation (A19) is of general nature and relates the initial (P) with the final (Q) set of basis functions, for an arbitrary number of knot insertions. The initial set, $\mathbf{N}_P$, is characterized by C^{p-1}-continuity. For each knot insertion, the continuity at the knot under consideration reduces by one, and thus for knot multiplicity λ, the continuity becomes $C^{p-\lambda}$.

Appendix C. Transformation Matrix Along an Edge or an Isoline

Each mesh-line (station) is determined by its local knot vector `knots`, which is always defined in the parametric space ranging from zero to unity (i.e., $0 \leq \xi \leq 1$ or $0 \leq \eta \leq 1$). Given the initial set of control points **P**, the updated control points **Q** and the corresponding transformation matrix **T**—after knot insertion at $\xi = \bar{\xi}$—are computed according to Equation (A12) through Equation (A14). A MATLAB® implementation of this procedure, valid for any polynomial degree p, is provided in Table A2.

 doi:10.20944/preprints202510.1500.v1

Table A2. Transformation matrix along an edge or an isoline.

```
function [ T, Q ] = Matrix_Knot_Insertion( knots, p, P, cs_insert )
%   knots     = knot sequence
%   p         = polynomial degree
%   P         = control points (before)
%   Q         = control points (after)
%   cs_insert = inserted value in knot vector
%-------------------------------------------------------------------------
        k = find(knots <= cs_insert, 1, 'last');
        %---
        iplus1_min = k-p+1; % minimum index 'i+1'
        iplus1_max = k;     % maximum index 'i+1'
        nctrl = length(knots) - p - 1;    % number of control points
        %---
        T = zeros(nctrl+1,nctrl);
        %---Q Remain as are:
        for index = 1:k-p
                Q(index) = P(index);
                T(index,index) = 1;
        end
        for i = iplus1_min:iplus1_max
                a = (cs_insert - knots(i)) / (knots(i+p) - knots(i));
                %---New-Q:
                Q(i) = (1-a)*P(i-1) + a*P(i);
                T(i,i-1) = (1-a);
                T(i,i)   = a;
                %---
        end
        %---Q Remain as are:
        for index = k+1:nctrl+1
                Q(index) = P(index-1);
                T(index,index-1) = 1;  %size = (nctrl+1) * nctrl
        end
        %---
        T = T';   %transpose: size = nctrl * (nctrl+1).
        %---
        end %end-of-function
```

The input to the function `Matrix_Knot_Insertion` includes the knot vector `knots`, the polynomial degree `p`, the control points `P`, and the insertion value `cs_insert` corresponding to the parametric coordinate ξ. The function returns the transformation matrix `T` and the updated control points `Q`. If only the transformation matrix $\mathbf{T}$ is required, the input array `P` may be assigned dummy values (e.g., all entries set to unity), as it does not affect the computation of `T`.

Numerical application: Considering the incomplete knot vector `knot`$=[0,0,0,0,\frac{1}{6},\frac{2}{6},\frac{4}{6},1,1,1,1]$ (i.e., the entry $\frac{3}{6}$ and and $\frac{5}{6}$ are missing), after performing two successive calls of the function `Matrix_Knot_Insertion`, in which we insert the knots at $\xi = \frac{3}{6}$ and $\xi = \frac{5}{6}$, consecutively, the resulting transformation matrix (of size 7×9) is found to be:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3/4 & 3/5 & 3/20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2/5 & 53/80 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3/16 & 3/4 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1 \end{bmatrix} \tag{A20}$$

 doi:10.20944/preprints202510.1500.v1

One may observe that the sum of the entries of each column in matrix **T** equals unity, which means that the partition of unity property is fulfilled.

Appendix D. Symbolic Derivation of Basis Functions

All the constraints produced by knot insertion are manually introduced in the following MATLAB script, given in Table A3:

Table A3. Symbolic determination of T-spline basis functions.

```
syms x y E1x E2x E3x E4x E5x E6x E7x E8x E9x ...
E1y E2y E3y E4y E5y E6y E7y E8y E9y ...
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 ...
A21 A22 A23 A24 A25 A26 A27 A28 A29 A30 A31 A32 A33 A34 A35 ...
A36 A37 A38 A39 A40 A41 A42 A43 A44 A45 A46 A47 A48 A49 A50 ...
A51 A52 A53 A54 A55 A56 A57 A58 A59 A60 A61 A62 A63 A64 A65 ...
A66 A67 A68 A69 A70 A71 A72 A73 A74 A75 A76 A77 A78 A79 A80
%% ---INTERNAL NODES
A40tp = 1/4*A39 + 3/4*A40;
A81x = 1/2*A40 + 1/2*A41;
A41tp = 3/4*A41 + 1/4*A42;
A32tp = 1/4*A23 + 3/4*A32;
A81y = 1/2*A32 + 1/2*A49;
A49tp = 3/4*A49 + 1/4*A58;
A81 = (A81x + A81y)/2;
% Define Axy
Axy = (E1x*E1y)*A1 + (E2x*E1y)*A2 +(E3x*E1y)*A3 + (E4x*E1y)*A4 ...
+ (E5x*E1y)*A5 +(E6x*E1y)*A6 + (E7x*E1y)*A7+ (E8x*E1y)*A8 +(E9x*E1y)*A9 ...
+ (E1x*E2y)*A10 + (E2x*E2y)*A11+(E3x*E2y)*A12+ (E4x*E2y)*A13 ...
+ (E5x*E2y)*A14 + (E6x*E2y)*A15+(E7x*E2y)*A16+ (E8x*E2y)*A17+(E9x*E2y)*A18 ...
+ (E1x*E3y)*A19 + (E2x*E3y)*A20+(E3x*E3y)*A21+ (E4x*E3y)*A22 ...
+ (E5x*E3y)*A23 + (E6x*E3y)*A24+(E7x*E3y)*A25+ (E8x*E3y)*A26+(E9x*E3y)*A27 ...
+ (E1x*E4y)*A28 + (E2x*E4y)*A29+(E3x*E4y)*A30+ (E4x*E4y)*A31 ...
+ (E5x*E4y)*A32tp+(E6x*E4y)*A33+(E7x*E4y)*A34+ (E8x*E4y)*A35+(E9x*E4y)*A36 ...
+ (E1x*E5y)*A37+ (E2x*E5y)*A38+ (E3x*E5y)*A39+ (E4x*E5y)*A40tp ...
+ (E5x*E5y)*A81+ (E6x*E5y)*A41tp+(E7x*E5y)*A42+ (E8x*E5y)*A43+(E9x*E5y)*A44 ...
+ (E1x*E6y)*A45+ (E2x*E6y)*A46+ (E3x*E6y)*A47+ (E4x*E6y)*A48 ...
+ (E5x*E6y)*A49tp+ (E6x*E6y)*A50+(E7x*E6y)*A51+ (E8x*E6y)*A52+(E9x*E6y)*A53 ...
+ (E1x*E7y)*A54+ (E2x*E7y)*A55+ (E3x*E7y)*A56+ (E4x*E7y)*A57 ...
+ (E5x*E7y)*A58+ (E6x*E7y)*A59+ (E7x*E7y)*A60+ (E8x*E7y)*A61+(E9x*E7y)*A62 ...
+ (E1x*E8y)*A63+ (E2x*E8y)*A64+ (E3x*E8y)*A65+ (E4x*E8y)*A66 ...
+ (E5x*E8y)*A67+ (E6x*E8y)*A68+ (E7x*E8y)*A69+ (E8x*E8y)*A70+(E9x*E8y)*A71 ...
+ (E1x*E9y)*A72+ (E2x*E9y)*A73+ (E3x*E9y)*A74+ (E4x*E9y)*A75 ...
+ (E5x*E9y)*A76+ (E6x*E9y)*A77+ (E7x*E9y)*A78+ (E8x*E9y)*A79+(E9x*E9y)*A80;
% Collect coefficients of A1 to A80
vars = [A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, ...
A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, A32, ...
A33, A34, A35, A36, A37, A38, A39, A40, A41, A42, A43, A44, A45, A46, A47, A48, A49, A50, ...
A51, A52, A53, A54, A55, A56, A57, A58, A59, A60, A61, A62, A63, A64, A65, ...
A66, A67, A68, A69, A70, A71, A72, A73, A74, A75, A76, A77, A78, A79, A80];
[coefficients, terms] = coeffs(Axy, vars);
% Initialize a symbolic row vector
coeff_vector = sym(zeros(1, length(vars)));
% Store symbolic coefficients
for i = 1:length(vars)
idx = find(terms == vars(i));
if ~isempty(idx)
coeff_vector(i) = coefficients(idx); % Store the coefficient
else
coeff_vector(i) = 0; % If no coefficient exists, assign 0
end
end
% Convert to numeric using vpa, if needed
coeff_vector_numeric = vpa(coeff_vector);
% Display the results
disp('Row vector of symbolic coefficients:');
disp(coeff_vector);
disp('Row vector of numeric coefficients:');
disp(coeff_vector_numeric);
% Display coefficients
for i = 1:length(vars)
fprintf('Coefficient of %s: %s\n', char(vars(i)), char(coefficients(find(terms == vars(i)))));
end
return
```

 doi:10.20944/preprints202510.1500.v1

Appendix E. Bernstein Polynomials and Derivatives

It is well known that Bézier interpolation of a curve or a univariate function has the following form

$$\mathbf{x}(\xi) = \sum_{i=1}^{n+1} B_{i,n}(\xi)\mathbf{P}_i, \quad 0 \le \xi \le 1, \tag{A21}$$

with the Bernstein polynomials given by

$$B_{i,n}(\xi) = \frac{n!}{(i-1)!(n-i+1)!}\xi^{i-1}(1-\xi)^{n-i+1}, \quad i = 1,\ldots,n+1 \tag{A22}$$

whereas $\mathbf{P}_i$ are the $(n+1)$ control points. In case of a univariate function $U(\xi)$, the control points are substituted by the generalized coefficients α_i, as follows:

$$U(\xi) = \sum_{i=1}^{n+1} B_{i,n}(\xi)\alpha_i, \quad 0 \le \xi \le 1\,. \tag{A23}$$

To avoid the computation of factorials of Equation (A22) involved in Equation (A21), the values $B_{i,n}(\xi)$ and the first derivative $dB_{i,n}(\xi)/d\xi$ can be alternatively determined using the B-spline technique. The complete in-house function `Bernstein` is shown in Table A4. It automatically generates the knot vector `knots` associated with the interval $[0,1]$, sets the ends with multiplicity $(n+1)$, calls the MATLAB built-in function `spcol`, and finally stores all the values of $B_{i,n}$ into the array `Basis`, and all the derivatives in array `dBasis`. These two arrays refer to all sites of vector `tau` (e.g., at Gauss points). The input variable `p` is equivalent to the polynomial degree n shown in Eqs. (A21) and (A22).

Table A4. Computation of Bernstein polynomials and derivatives.

```
function [ Basis, dBasis ] = Bernstein( tau, p )
%BERNSTEIN polynomials and Derivatives within [0,1], at 'tau' sites.
multiplicity=1;                 %multiplicity
knots=augknt([0 1],p+1,multiplicity);
nset=2;                 %number of sets: 1)Bernstein, 2)first derivative
colmat=spcol(knots,p+1,brk2knt(tau,nset)); %array including all results
imax=size(colmat,1);
Basis =colmat(1:nset:imax,:);   %Bernstein polynomials (Bi)
dBasis=colmat(2:nset:imax,:);   %Bernstein derivatives (Bi')
end
```

Appendix F. Shape Functions of 12-Node Element

When Bernstein polynomials are used, the bivariate basis functions of the 12-node transfinite element of three layers are given by Equation (20). The MATLAB script, which exits the shape functions, the partial derivatives and the determinant of the Jaxobian matrix at the Gauss points is given below:

```
function [SHP,XSJ] = SHAPE_Pht_12_BEZIER(xcor,ycor,cs,ht)
%% TRANSFINITE ELEMENT, 12
nodes, interboundaries at xi,eta=0,1/2,1.
%{
% 10 bounday nodes + 1*1 = 11 total nodes
                13        14       15       16      17      18
              D o-------o--------o--------o-------o------o   C
                |                                          |
                |                     10                   |
                8 o          o 9        o        o 11       o 12
                |                                          |
                |                                          |
                4 o          o 5              o 6          o 7
                |                                          |
                |                                          |
```

 doi:10.20944/preprints202510.1500.v1


```
                o---------------------o------------------o--->X
                A 1                   2                  3 B
                \%}
%% BLENDING FUNCTIONS IN Y-dir:
py=2; tauy=ht;
[ Basis, dBasis ] = Bernstein( tauy, py );      %Bernstein plynomials
E1y=Basis(1);    E2y=Basis(2);   E3y=Basis(3);  %Blending functions
dE1y=dBasis(1); dE2y=dBasis(2); dE3y=dBasis(3); %Derivatives of >>
%-------------------------------------------------------------------------
NEL = 12; SHP(1:3,1:NEL)=0;
%---LAYER#1:
px=2; taux=cs;
[ Basis, dBasis ] = Bernstein( taux, px );  % Bernstein polynomials
L1x=Basis(1);    L2x=Basis(2);   L3x=Basis(3);
dL1x=dBasis(1); dL2x=dBasis(2); dL3x=dBasis(3);
%---ASSOCIATED BIVARIATE BASIS FUNCTIONS (1-3) [see, Equation(20)]:
SHP(3,1) = L1x* E1y;    %psi_1
SHP(1,1) =dL1x* E1y;    %d(psi_1)/dxi;
SHP(2,1) =  L1x*dE1y;   %d(psi_1)/deta;
SHP(3,2) = L2x* E1y;    %psi_2
SHP(1,2) =dL2x* E1y;   %d(psi_2)/dxi;
SHP(2,2) = L2x*dE1y;   %d(psi_2)/deta;
SHP(3,3) = L3x* E1y;    %psi_3
SHP(1,3) =dL3x* E1y;    %d(psi_3)/dxi;
SHP(2,3) = L3x*dE1y;    %d(psi_3)/deta;
%---LAYER No.2: HT=1/2:
px=3; taux=cs;
[ Basis, dBasis ] = Bernstein( taux, px );
L1x=Basis(1);    L2x=Basis(2);   L3x=Basis(3);   L4x= Basis(4);
dL1x=dBasis(1); dL2x=dBasis(2); dL3x=dBasis(3); dL4x=dBasis(4);
%---ASSOCIATED BIVARIATE BASIS FUNCTIONS (4-7) [see, Equation(20)]:
SHP(3,4) = L1x* E2y; SHP(1,4) =dL1x* E2y; SHP(2,4) = L1x* dE2y;
SHP(3,5) = L2x* E2y; SHP(1,5) =dL2x* E2y; SHP(2,5) = L2x* dE2y;
SHP(3,6) = L3x* E2y; SHP(1,6) =dL3x* E2y; SHP(2,6) = L3x* dE2y;
SHP(3,7) = L4x* E2y; SHP(1,7) =dL4x* E2y; SHP(2,7) = L4x* dE2y;
%---LAYER No.3: HT=1:
px=4; taux=cs;
[ Basis, dBasis ] = Bernstein( taux, px );
L1x=Basis(1);    L2x=Basis(2);   L3x=Basis(3);   L4x= Basis(4);  L5x= Basis(5);
dL1x=dBasis(1); dL2x=dBasis(2); dL3x=dBasis(3); dL4x=dBasis(4); dL5x=dBasis(5);
%---ASSOCIATED BIVARIATE BASIS FUNCTIONS (8-12) [see, Equation(20)]:
SHP(3,8)  = L1x* E3y; SHP(1,8)  = dL1x* E3y; SHP(2,8)  = L1x* dE3y;
SHP(3,9)  = L2x* E3y; SHP(1,9)  = dL2x* E3y; SHP(2,9)  = L2x* dE3y;
SHP(3,10) = L3x* E3y; SHP(1,10) = dL3x* E3y; SHP(2,10) = L3x* dE3y;
SHP(3,11) = L4x* E3y; SHP(1,11) = dL4x* E3y; SHP(2,11) = L4x* dE3y;
SHP(3,12) = L5x* E3y; SHP(1,12) = dL5x* E3y; SHP(2,12) = L5x* dE3y;
%-----CONSTRUCT JACOBIAN:
XL(1,1:NEL)=xcor(1:NEL);
XL(2,1:NEL)=ycor(1:NEL);
for I=1:2
for J=1:2
XS(I,J)=0.0;
for K=1:NEL
XS(I,J)=XS(I,J)+XL(I,K)*SHP(J,K);
end
```

 doi:10.20944/preprints202510.1500.v1


```
end
end
%---INVERSE OF JACOBIAN:
XSJ=XS(1,1)*XS(2,2)-XS(1,2)*XS(2,1);
SX(1,1)=XS(2,2)/XSJ;
SX(2,2)=XS(1,1)/XSJ;
SX(1,2)=-XS(1,2)/XSJ;
SX(2,1)=-XS(2,1)/XSJ;
% C-----FORM GLOBAL DERIVATIVES
for I=1:NEL
TP       = SHP(1,I)*SX(1,1)+SHP(2,I)*SX(2,1);
SHP(2,I) = SHP(1,I)*SX(1,2)+SHP(2,I)*SX(2,2);
SHP(1,I) = TP;
end
end     %end-of-subroutine
```

Appendix G. Main Program

Below is the main program for the solution of Example-1 (Figure 16) using the 12-node transfinite element shown in Figure 5a. It calls the four following subroutines,

`SHAPE_Peta_12_LAGRANGE`	Shape functions using univariate Lagrange polynomials, as blending and trial functions.
`SHAPE_Peta_12_BEZIER`	Basis functions using univariate Bernstein-Bézier polynomials, as blending and trial functions (cited in Appendix F).
`SHAPE_Peta_12_BSPLINES`	Basis functions using univariate B-splines, as blending and trial functions.
`lgwt`	Gauss points and associated weights (see Ref. [27]).

The main integer variables and several real arrays used by the program, together with their meaning are given below.

`area`	Is the calculated patch area using Lagrange (L), Bernstein-Bézier (B) and B-spline (C) interpolation.
`Coeffs`	Calculated coefficients (a_i) using Lagrange (L), Bernstein-Bézier (B) and B-spline (C) interpolation.
`gi`	Location of Gauss points ($-1 \leq r, s \leq 1$).
`L2percent`	L_2-error norm (in %) of the numerical solution $U(\xi, \eta)$.
`NEL`	Is the number of Nodes on the ELement.
`ncellx`	Is the number of integration cells in ξ-direction.
`ncelly`	Is the number of integration cells in η-direction.
`ngauss`	Is the number of Gauss points per direction.
`ome`	Weights associated with Gauss points.
`SHPL`	Shape functions and partial derivatives for Lagrange polynomials.
`SHPB`	Shape functions and partial derivatives for Bernstein-Bézier polynomials (see, Appendix F).
`SHPC`	Shape functions and partial derivatives for B-splines.
`x,y,XL`	Cartesian coordinates (x, y) of nodal points or breakpoints.
`xi,eta`	Parameters $(\xi, \eta) \in [0,1] \times [0,1]$.
`XLb`	Cartesian coordinates of control points for Bernstein-Bézier polynomials.
`XLc`	Cartesian coordinates of control points for B-splines.
`XSJ`	Determinant of Jacobian matrix.

To solve a different problem with a more complex mesh, a preprocessor has to be added with the purpose of replacing the variables `x,y` at the beginning of the main program.

```
%*********************************************************************
%% Solves the BVP for 12-node Transfinite Element
%*********************************************************************
%{
     D  o---------o---------o--------o--------o C      Layer#3
```

 doi:10.20944/preprints202510.1500.v1


```
        |8         9          10         11      12|
        |                                          |
        4 o    ---    o 5      ---     o 6    ---     o 7    Layer#2
        |                                         |
        |                                         |
        o-------------------o-----------------o        Layer#1
        A 1                   2                 3  B
        %}
%-------------------------------------------------------------------------
clear all; clc
%*************************************************************************
%%                                PRE-PROCESSOR
%*************************************************************************
%---CARTESIAN COORDINATES (UNIT SQUARE: xmax=1, ymax=1):
NEL = 12;   %number of element nodes
x(1:3) =linspace(0,1,3); y(1:3)=0.0;     % Layer#1: uniform
x(4:7) =linspace(0,1,4); y(4:7)=1/2;     % Layer#2: uniform
x(8:12)=linspace(0,1,5); y(8:12)=1;      % Layer#1: uniform
% Store both coordinates in the single array XL:
XL(1,1:NEL)=x(1:NEL);      XL(2,1:NEL)=y(1:NEL); %store in one array
figure(1)
plot(x,y,'o-')      % Plot Cartesian coordinates (nodes or breakpoints)
for i=1:NEL
text(x(i)+0.02,y(i)+0.02,int2str(i),'Color','red') % node numbering
end
%% ADJUST CONTROL POINTS SO THAT TO PRODUCE UNIFORM IMAGES:
for i=1:NEL
xi=x(i); eta=y(i);
[SHPB,~] = SHAPE_Peta_12_BEZIER(x,y,xi,eta);            %Bezier
Bmat(i,1:NEL)=SHPB(3,1:NEL);
[SHPC,XSJdummy] = SHAPE_Peta_12_BSPLINES(x,y,xi,eta);   %B-splines
Cmat(i,1:NEL)=SHPC(3,1:NEL);
end
% Control points (CTRLs) for BERNSTEIN-BEZIER approximation:
XLb(1,1:NEL) = Bmat \ XL(1,1:NEL)'; %control pts (x-coordinate)
XLb(2,1:NEL) = Bmat \ XL(2,1:NEL)'; %control pts (y-coordinate)
% Control points (CTRLs) for B-SPLINE approximation:
XLc(1,1:NEL) = Cmat \ XL(1,1:NEL)'; %control pts (x-coordinate)
XLc(2,1:NEL) = Cmat \ XL(2,1:NEL)'; %control pts (y-coordinate)
%Finalize the plot:
hold on
plot(XLb(1,1:NEL), XLb(2,1:NEL),'b+')   %Plot BERNSTEIN-BEZIER CTRLs
hold on
plot(XLc(1,1:NEL), XLc(2,1:NEL),'rx')   %Plot B-SPLINE CTRLs
% For each layer, store location of breakpoints (or nodes) into a vector:
Layer1=[0,1/2,1];           % Breakpoints (BRKs) of Layer#1
Layer2=[0 1/3 2/3 1];       % BRKs of Layer#2
Layer3=[0 1/4 2/4 3/4 1];   % BRKs of Layer#3
AllValues = [Layer1, Layer2, Layer3];    % All possible BRKS, together.
Ubrkx = unique(AllValues);  % Determine Unique values
ncellx=length(Ubrkx)-1;     % number of cells in horizontal direction
ncelly=2;                   % number of cells in vertical direction
%*************************************************************************
%%                                ANALYSIS
%*************************************************************************
%---Usual Gauss Points (GPTs) per direction in each integration cell:
ngauss = 04;    % For rectangle: (p+1) GPTs per direction are required.
[gi,ome]=lgwt(ngauss,-1,1); % Gauss points and weights
% Initialize Stiffness and Patch area: L=Lagrange, B=Bernstein, C=B-spline
KL=zeros(NEL,NEL); KB=zeros(NEL,NEL); KC=zeros(NEL,NEL); % stiffness
areaL=0; areaB=0; areaC=0;                               % patch area
dy=1/ncelly;    % vertical distance betwen successive layers
xmax=1; ymax=1; % edge lengths of quadrilateral patch
% Loop over all integration cells
for jcelly=1:ncelly
ym=(jcelly-1)*dy+dy/2; etam=ym/ymax;
for icellx=1:ncellx
dx=Ubrkx(icellx+1)-Ubrkx(icellx);
Jloc=1/4*dx*dy; % local det(Jacobian)
xm=(Ubrkx(icellx)+Ubrkx(icellx+1))/2; xim=xm/xmax;
for igau=1:ngauss
for jgau=1:ngauss
xsj=1;  %square [0,1]x[0,1]
xi=xim+gi(igau)*dx/2;
eta=etam+gi(jgau)*dy/2;
```

 doi:10.20944/preprints202510.1500.v1


```
% For current Gauss point (xi,eta), find Shape Functions:
[SHPL,XSJL] = SHAPE_Peta_12_LAGRANGE( x, y, xi, eta );%Lagrange
[SHPB,XSJB] = SHAPE_Peta_12_BEZIER(XLb(1,1:NEL),XLb(2,1:NEL),xi,eta);%Bezier
[SHPC,XSJC] = SHAPE_Peta_12_BSPLINES(XLc(1,1:NEL),XLc(2,1:NEL),xi,eta);%B-splines, p=3
% Weighted Jacobian term:
dvolL=XSJL*ome(igau)*ome(jgau)*Jloc; %volume element (Lagrange)
dvolB=XSJB*ome(igau)*ome(jgau)*Jloc; %volume element (Bernstein)
dvolC=XSJC*ome(igau)*ome(jgau)*Jloc; %volume element (B-spline)
% Patch area using (L=Lagrange, B=Bernstein, C=B-spline):
areaL=areaL+dvolL;
areaB=areaB+dvolB;
areaC=areaC+dvolC;
% Stiffness matrices in three fomulations (L, B, C):
for i=1:NEL
for j=1:NEL
%---LAGRANGE:
KL(i,j)=KL(i,j) + (SHPL(1,i)*SHPL(1,j) + SHPL(2,i)*SHPL(2,j))*dvolL;
%---BEZIER-BERNSTEIN:
KB(i,j)=KB(i,j) + (SHPB(1,i)*SHPB(1,j) + SHPB(2,i)*SHPB(2,j))*dvolB;
%---B-SPLINES:
KC(i,j)=KC(i,j) + (SHPC(1,i)*SHPC(1,j) + SHPC(2,i)*SHPC(2,j))*dvolC;
end
end
end
end
end
end
fprintf('\nAREAS: areaL = %12.5e  areaB = %12.5e  areaC = %12.5e\n',areaL,areaB,areaC);
%% IMPOSE BOUNDARY CONDITIONS
BC(1:NEL)=0;
BC(1:3)=1;                  %edge AB (bottom)
BC(3)=1;BC(7)=1;BC(12)=1;   %edge BC (right vertical)
BC(8:12)=1;                 %edge DC (top)
%-------------------------------------------------------------------------
% LAGRANGE POLYNOMIALS (index 'L'):
fiL(1:NEL)=0;
edofTop=[8 9 10 11 12];
Xtop=x(edofTop);
fiL(edofTop)=1000*cos(pi*Xtop/2/1);
%-------------------------------------------------------------------------
% BERNSTEIN POLYNOMIALS (index 'B'):
fiB(1:NEL)=0;
rhsB(1:NEL)=0;
TopU(1:5)=1000*cos(pi*Xtop/2/1);
px=4;
knotsx=augknt([0 1],px+1,1);
taux=Xtop;
Bmatrix=spcol(knotsx,px+1,brk2knt(taux,1));
nctrlx=size(Bmatrix,2);
fiB(edofTop)=Bmatrix \ TopU';
%-------------------------------------------------------------------------
% B-SPLINES (index 'C'):
fiC(1:NEL)=0;
rhsC(1:NEL)=0;
TopU(1:5)=1000*cos(pi*Xtop/2/1);
px=3;
knotsx=[0 0 0 0 1/2 1 1 1 1];
taux=Xtop;
Cmatrix=spcol(knotsx,px+1,brk2knt(taux,1));
nctrlx=size(Cmatrix,2);
fiC(edofTop)=Cmatrix \ TopU';
%-------------------------------------------------------------------------
%% DIVIDE DOFs IN TWO PARTS:
free_dofs  = find(BC==0);
fixed_dofs = find(BC==1);
%% SOLVE FOR LAGRANGE POLYNOMIALS:
rhsL(1:NEL)=0;
rhsL(free_dofs)=-KL(free_dofs,fixed_dofs)*fiL(fixed_dofs)';
UnodalL(fixed_dofs)=fiL(fixed_dofs);
UnodalL(free_dofs) = KL(free_dofs,free_dofs) \ rhsL(free_dofs)';
%% SOLVE FOR BERNSTEIN POLYNOMIALS:
rhsB(1:NEL)=0;
rhsB(free_dofs)=-KB(free_dofs,fixed_dofs)*fiB(fixed_dofs)';
CoeffsB(fixed_dofs)= fiB(fixed_dofs);
CoeffsB(free_dofs) = KB(free_dofs,free_dofs) \ rhsB(free_dofs)';
```

 doi:10.20944/preprints202510.1500.v1


```
%% SOLVE FOR B-SPLINES:
rhsC(1:NEL)=0;
rhsC(free_dofs)=-KC(free_dofs,fixed_dofs)*fiC(fixed_dofs)';
CoeffsC(fixed_dofs)= fiC(fixed_dofs);
CoeffsC(free_dofs) = KC(free_dofs,free_dofs) \ rhsC(free_dofs)';
%**************************************************************************
%%                                POST-PROCESSOR
%**************************************************************************
%% CALCULATE THE ERROR IN THE DOMAIN
areaL=0;        areaB=0;        areaC=0;
numeratorL=0;   numeratorB=0;   numeratorC=0;
denominatorL=0; denominatorB=0; denominatorC=0;
for jcelly=1:ncelly
ym=(jcelly-1)*dy+dy/2; etam=ym/ymax;
for icellx=1:ncellx
dx=Ubrkx(icellx+1)-Ubrkx(icellx);
Jloc=1/4*dx*dy;
xm=(Ubrkx(icellx)+Ubrkx(icellx+1))/2; xim=xm/xmax;
for igau=1:ngauss
for jgau=1:ngauss
xi=xim+gi(igau)*dx/2; eta=etam+gi(jgau)*dy/2; %parameters
% Shape functions:
[SHPL, XSJL] = SHAPE_Peta_12_LAGRANGE( x,y,xi,eta ); %Lagrange
[SHPB, XSJB] = SHAPE_Peta_12_BEZIER( XLb(1,1:NEL), XLb(2,1:NEL), xi, eta );  %Bezier
[SHPC, XSJC] = SHAPE_Peta_12_BSPLINES( XLc(1,1:NEL), XLc(2,1:NEL), xi, eta );  %Bezier
% Calculated values at Gauss points:
UcalculatedL = SHPL(3,:)*UnodalL';
UcalculatedB = SHPB(3,:)*CoeffsB';
UcalculatedC = SHPC(3,:)*CoeffsC';
% Cartesian coordinates of Gauss points:
xgpt=SHPL(3,:)*XL(1,:)'; ygpt=SHPL(3,:)*XL(2,:)';
Uexact=1000*sinh(pi*ygpt/2/1)/sinh(pi*1/2/1) * cos(pi*xgpt/2/1);
% Volume elements
dvolL=XSJL*ome(igau)*ome(jgau)*Jloc;
dvolB=XSJB*ome(igau)*ome(jgau)*Jloc;
dvolC=XSJC*ome(igau)*ome(jgau)*Jloc;
% Lagrange polynomials (L):
areaL=areaL+dvolL;
numeratorL  =numeratorL   + (UcalculatedL-Uexact)^2*dvolL;
denominatorL=denominatorL + (0-Uexact)^2*dvolL;
% Bernstein-Bezier polynomals (B):
areaB=areaB+dvolB;
numeratorB  =numeratorB   + (UcalculatedB-Uexact)^2*dvolB;
denominatorB=denominatorB + (0-Uexact)^2*dvolB;
% B-splines (C):
areaC=areaC+dvolC;
numeratorC  =numeratorC   + (UcalculatedC-Uexact)^2*dvolC;
denominatorC=denominatorC + (0-Uexact)^2*dvolC;
end
end
end
end
L2percentL = sqrt(numeratorL/denominatorL)*100;
fprintf('\nLAGRANGE : Area = %12.5e  L2-error norm(percent) = %12.5e\n',areaL,L2percentL);
L2percentB = sqrt(numeratorB/denominatorB)*100;
fprintf('BERNSTEIN: Area = %12.5e  L2-error norm(percent) = %12.5e\n',areaB,L2percentB);
L2percentC = sqrt(numeratorC/denominatorC)*100;
fprintf('B-SPLINES: Area = %12.5e  L2-error norm(percent) = %12.5e\n',areaC,L2percentC);
return
```

References

1. Rayleigh L. Scientific Papers, Vol. II, Cambridge University Press, Cambridge, 1900.
2. Ritz W. Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik, (On a new method for the solution of certain variational problems of mathematical physics). Journal für reine und angewandte Mathematik 1909;135:1–61.
3. Zienkiewicz OC. *The Finite Element Method*, 3rd ed. London: McGraw-Hill; 1977.
4. Bathe KJ. *Finite Element Procedures*, 2nd ed. New Jersey: Prentice-Hall; 1996.
5. Gordon WJ, Hall CA. Transfinite element methods: Blending-function interpolation over arbitrary curved element domains. Numerische Mathematik 1973;21:109–129. https://doi.org/10.1007/BF01436298.

6. Birkhoff, G; Cavendish, J.C.; Gordon, W.J. Multivariate Approximation by Locally Blended Univariate Interpolants. Proc. Nat. Acad. Sci. USA 1974;71(9):3423–3425. https://doi.org/10.1073/pnas.71.9.3423.
7. Coons SA. Surfaces for computer-aided design of space forms. MIT (1967).
8. Kanarachos A, Deriziotis D. On the solution of Laplace and wave propagation problems using "C-elements". Finite elements in Analysis and Design 1989;5(2):97–109. https://doi.org/10.1016/0168-874X(89)90015-2.
9. Provatidis C.G. Precursors of isogeometric analysis: Finite elements, Boundary elements, and Collocation methods. Springer, Cham, 2019. https://doi.org/10.1007/978-3-030-03889-2.
10. Cottrell JA, Hughes TJR, Bazilevs Y. Isogeometric analysis: towards integration of CAD and FEA, Wiley, Chichester, 2009.
11. Scholz E, Altenbach J, Kompatible übergangselemente für lokale Netzverfeinerungen bei 2D- und 3D-Finite-Elemente-Modellen. Tech. Mech. 1985;6:72–78.
12. Altenbach J, Scholz E. Ableitung von Formfunktionen für finite Standard- und Übergangselemente Grundlage der gemischten Interpolation. Tech. Mech. 1987;8:18–30.
13. Weinberg K, Gabbert U. Adaptive local-global analysis by pNh transition elements. Tech. Mech. 1999;19:115–126.
14. Weinberg K, Gabbert U. An adaptive pNh-technique for global-local finite element analysis. Eng. Comput. 2002;19:485–500. https://doi.org/10.1108/02644400210435825.
15. Duczek S, Saputra AA, Gravenkamp H. High Order Transition Elements: The xNy-Element Concept–Part I: Statics. Comput. Methods Appl. Mech. Eng. 2020;362:112833. https://doi.org/10.1016/j.cma.2020.112833.
16. Provatidis C. Transfinite patches for isogeometric analysis. Mathematics 2025;13(3):35. https://doi.org/10.3390/math13030335.
17. Sederberg, T.W.; Zheng, J.; Bakenov, A.; Nasri, A. T-splines and T-NURCCs. ACM transactions on graphics (TOG) 2003;22(3):477–484. https://doi.org/10.1145/882262.882295.
18. Bazilevs, Y.; Calo, V.M.; Cottrell, J.A.; Evans, J.A.; Hughes, T.J.R.; Lipton, S.; Scott, M.A.; Sederberg, T.W. Isogeometric analysis using T-splines. Computer Methods in Applied Mechanics and Engineering 2010;199(5–8):229–263. https://doi.org/10.1016/j.cma.2009.02.036.
19. Dörfel, M.R.; Jüttler, B.; Simeon, B. Adaptive isogeometric analysis by local h-refinement with T-splines. Computer Methods in Applied Mechanics and Engineering 2010;199(5–8):264-275. https://doi.org/10.1016/j.cma.2008.07.012.
20. Provatidis C. Transfinite elements using Bernstein polynomials. Axioms 2025;14(6):433. https://doi.org/10.3390/axioms14060433.
21. Provatidis C, Eisenträger S. Macroelement analysis in T-patches using Lagrange polynomials. Mathematics 2025;13(9):1498. https://doi.org/10.3390/math13091498.
22. Farin G. Curves and Surfaces for CAGD. Morgan Kaufmann-Elsevier. USA, 2002. https://doi.org/10.1016/B978-1-55860-737-8.X5000-5.
23. Provatidis C. Bézier versus Lagrange polynomials-based finite element analysis of 2-D potential problems. Advances in Engineering Software 2014;73:22–34. https://doi.org/10.1016/j.advengsoft.2014.02.006.
24. Farouki, R., and Hinds, J. A hierarchy of geometric forms. IEEE Computer Graphics and Applications 1985;5(5):51–78. https://doi.org/10.1109/MCG.1985.276393.
25. Cai Z, Wang W, Du X, Zhao G. A new method for converting trimmed NURBS surface to T-spline surface for isogeometric analysis. AIP Conference Proceedings. Vol. 1834. No. 1. AIP Publishing, 2017. https://doi.org/10.1063/1.4981625.
26. DeBoor C, A Practical Guide to Splines, Revised ed., Springer: New York, 2001.
27. Greg von Winckel (2025). Legendre-Gauss Quadrature Weights and Nodes (https://www.mathworks.com/matlabcentral/fileexchange/4540-legendre-gauss-quadrature-weights-and-nodes), MATLAB Central File Exchange. Recovered August 6, 2025.
28. Carslaw HS, Jaeger JC. Conduction of Heat in Solids. 2nd ed. Oxford: Oxford University Press; 1969.
29. Curry HB, Schoenberg IJ. On Pólya Frequency Functions IV: The Fundamental spline functions and their limits. Journal d'Analyse Mathématique 1966;17(1):71–107. https://doi.org/10.1007/BF02788653
30. Cox MG. The Numerical Evaluation of B-Splines. IMA Journal of Applied Mathematics 1972;10(2):134–149. https://doi.org/10.1093/imamat/10.2.134
31. De Boor C. On Calculating with B-Splines. Journal of Approximation Theory 1972;6(1):50–62. https://doi.org/10.1016/0021-9045(72)90080-9
32. Karniadakis GE, Sherwin SJ. Spectral/ Hp Element Methods for Computational Fluid Dynamics. Oxford: Oxford Science Publications; 2005. http://dx.doi.org/10.1093/acprof:oso/9780198528692.001.0001.

 doi:10.20944/preprints202510.1500.v1

33. Courant R, Hilbert D. Methods of mathematical physics (1st English ed., Vol. I). New York: InterScience (translated and revised from the German original); 1966.
34. Felippa CA, Haugen B, Militello C. From the individual element test to finite element templates: Evolution of the patch test. International Journal for Numerical Methods in Engineering 1995:38(2):199–229. http://dx.doi.org/10.1002/nme.1620380204.
35. Provatidis CG., Lumped mass acoustic and membrane eigenanalysis using the global collocation method. Cogent Engineering 2014;1(1), Article: 981366. https://doi.org/10.1080/23311916.2014.981366.
36. Provatidis CG, CAD-based collocation eigenanalysis of 2-D elastic structures, Computers and Structures 2017,182:55–73. http://dx.doi.org/10.1016/j.compstruc.2016.11.007.
37. Yağmurlu NM, Karakaş AS, A novel perspective for simulations of the MEW equation by trigonometric cubic B-spline collocation method based on Rubin-Graves type linearization. Computational Methods for Differential Equations 2022;10(4):1046–1058. https://doi.org/10.22034/cmde.2021.47358.1981.
38. Kutluay S, Yağmurlu NM, Karakaş AS, A novel perspective for simulations of the Modified Equal-Width Wave equation by cubic Hermite B-spline collocation method. Wave Motion 2024;129:103342. https://doi.org/10.1016/j.wavemoti.2024.103342.
39. Kutluay S, Yagmurlu NM, Karakaş AS, A robust septic hermite collocation technique for dirichlet boundary condition Heat conduction equation. International Journal of Mathematics and Computer in Engineering 2025;3(2):253–266.
40. Eisenträger S, Eisenträger J, Gravenkamp H, Provatidis CG. High order transition elements: The xNy-element concept, Part II: Dynamics. Computer Methods in Applied Mechanics and Engineering 2021;387:114145. https://doi.org/10.1016/j.cma.2021.114145. https://doi.org/10.2478/ijmce-2025-0019.
41. Provatidis C, Sevilla R, Schillinger D, Eisenträger S. Revisiting Transfinite Elements: Unifying Element Formulations for IGA, SEM, NEFEM, p-FEM and h-FEM. Archives of Computational Methods in Engineering 2025 (in press). https://doi.org/10.1007/s11831-025-10351-3.
42. Provatidis CG. Non-rational and rational transfinite interpolation using Bernstein polynomials. International Journal of Computational Geometry and Applications 2022;32(1-2):55–89. https://doi.org/10.1142/S0218195922500030.
43. Zepeng Wen, Qiong Pan, Xiaoya Zhai, Hongmei Kang, Falai Chen, Adaptive isogeometric topology optimization of shell structures based on PHT-splines, Computers & Structures 2024;305:107565. https://doi.org/10.1016/j.compstruc.2024.107565.
44. Chenk K, Qiu C, Liu Z, Tan J. A high-precision T-spline blade modeling method with 2D and 3D knot adaptive refinement. Advances in Engineering Software 2024;194:103684. https://doi.org/10.1016/j.advengsoft.2024.103684.
45. Nguyen VP, Anitescu C, Bordas SP, Rabczuk T. Isogeometric analysis: an overview and computer implementation aspects. Mathematics and Computers in Simulation 2015;117:89-116. https://doi.org/10.1016/j.matcom.2015.05.008
46. Garau EM, Vázquez R. Algorithms for the implementation of adaptive isogeometric methods using hierarchical B-splines. Applied Numerical Mathematics 2018;123:58-87. https://doi.org/10.1016/j.apnum.2017.08.006
47. Provatidis CG, Angelidis DI. Performance of Coons' macroelements in plate bending analysis. International Journal for Computational Methods in Engineering Science and Mechanics 2014;15(2):110–125. https://doi.org/10.1080/15502287.2013.874057
48. Bogner FK, Fox RL, Schmit LA. The generation of inter-element-compatible stiffness and mass matrices by the use of interpolation formulas. In Proceedings of the Conference on Matrix Methods in Structural Mechanics (1965), pp. 397–444.
49. Thomas DC, Engvall L, Schmidt SK, Tew K, Scott MA. U-splines: Splines over unstructured meshes. Computer Methods in Applied Mechanics and Engineering 2022;401:115515. https://doi.org/10.1016/j.cma.2022.115515.
50. Wei X, Li X, Qian K, Hughes TJR, Zhang YJ, Casquero H. Analysis-suitable unstructured T-splines: Multiple extraordinary points per face. Computer Methods in Applied Mechanics and Engineering 2022;391:114494. https://doi.org/10.1016/j.cma.2021.114494.
51. Sangalli G, Takacs T, Vázquez R. Unstructured spline spaces for isogeometric analysis based on spline manifolds. Computer Aided Geometric Design 2016;47:61–82.
52. Banh, TT, Lieu QX, Kang J, Ju Y, Shin S, Lee D. A novel robust stress-based multimaterial topology optimization model for structural stability framework using refined adaptive continuation method. Engineering with Computers 2024;40(2):677–713. https://doi.org/10.1007/s00366-023-01829-4

53. Nguyen MN, Lee D. Design of the multiphase material structures with mass, stiffness, stress, and dynamic criteria via a modified ordered SIMP topology optimization. Advances in Engineering Software 2024;189:103592. https://doi.org/10.1016/j.advengsoft.2023.103592
54. Banh TT, Lieu QX, Nguyen SH, Lee D. Stress-driven design of incompressible multi-materials under frequency constraints. International Journal of Mechanical Sciences 2024;277:109416. https://doi.org/10.1016/j.ijmecsci.2024.109416
55. Nguyen MN, Hoang VN, Lee D. Multiscale topology optimization with stress, buckling and dynamic constraints using adaptive geometric components. Thin-Walled Structures 2023;183:111405. https://doi.org/10.1016/j.tws.2022.110405
56. Banh TT, Lee D. Comprehensive polygonal topology optimization for triplet thermo-mechanical-pressure multi-material systems. Engineering with Computers 2024;40(5):3295–3317. https://doi.org/10.1007/s00366-024-01982-4. https://doi.org/10.1016/j.cagd.2016.05.004.
57. Piegl L, Tiller W. The NURBS Book, 2nd ed. Berlin; New York: Springer; 1997. ISBN: 978-3-642-59223-2. https://link.springer.com/book/10.1007/978-3-642-59223-2
58. Farin, G. *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. Morgan Kaufmann: San Francisco, USA, 2002.
59. Piegl, L; Tiller, W. *The NURBS Book*, 2nd ed. Springer: Berlin, Germany, 1997.
60. Hoschek, J.; Lasser, D. *Fundamentals of computer aided geometric design*, 2nd ed. A.K. Peters: Wellesley, Mass, 1996.
61. Hughes, T. J. R.; Cottrell, J. A.; Bazilevs, Y. (2005). Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* **2005**;*194*(39–41), 4135–4195. https://doi.org/10.1016/j.cma.2004.10.008
62. Desiderio, L.; D'Inverno, G.A.; Sampoli, M.L.; Sestini, A. Hierarchical matrices for 3D Helmholtz problems in the multi-patch IgA-BEM setting. Engineering with Computers **2025**;*41*, 2021–2042. https://doi.org/10.1007/s00366-025-02144-w.
63. Schillinger, D.; Ruthala, P. K.; Nguyen, L. H. Lagrange extraction and projection for NURBS basis functions: A direct link between isogeometric and standard nodal finite element formulations. *International Journal for Numerical Methods in Engineering* **2016**;*108*(6), 515–534. https://doi.org/10.1002/nme.5216.
64. Liu, B.; Xing, Y.; Wang, Z.; Lu, X.; Sun, H. Non-uniform rational Lagrange functions and its applications to isogeometric analysis of in-plane and flexural vibration of thin plates. *Computer Methods in Applied Mechanics and Engineering* **2017**; *321*, 173–208. https://doi.org/10.1016/j.cma.2017.04.007.
65. Lu, L. Z. Adaptive Polynomial Approximation to Circular Arcs. *Applied Mechanics and Materials* **2011**; *50*, 678–682. https://doi.org/10.4028/www.scientific.net/AMM.50-51.678
66. Vavpetič, A. Optimal parametric interpolants of circular arcs. *Computer Aided Geometric Design* **2020**; *80*, 101891. https://doi.org/10.1016/j.cagd.2020.101891.
67. Vavpetič, A.; Žagar, E. A general framework for the optimal approximation of circular arcs by parametric polynomial curves. *Journal of Computational and Applied Mathematics* **2019**; *345*, 146–158. https://doi.org/10.1016/j.cam.2018.06.020.
68. Vavpetič, A.; Žagar, E. On optimal polynomial geometric interpolation of circular arcs according to the Hausdorff distance. *Journal of Computational and Applied Mathematics* **2021**; *392*, 113491. https://doi.org/10.1016/j.cam.2021.113491.
69. De Boor, C. *A Practical Guide to Splines*, rev ed. Applied mathematical sciences. Springer: New York, 2001.
70. Patera, A.T. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics* **1984**; *54*(3), 468–488. https://doi.org/10.1016/0021-9991(84)90128-1.
71. Pozrikidis, C. *Introduction to Finite and Spectral Element Methods Using MATLAB*, 2nd ed. CRC Press: Boca-Raton, 2014.
72. Eisenträger, S.; Eisenträger, J.; Gravenkamp, H.; Provatidis, C.G. High order transition elements: The xNy-element concept-part II: Dynamics. *Computer Methods in Applied Mechanics and Engineering* **2021**;*387* (Dec. 2021), 114145. https://doi.org/10.1016/j.cma.2021.114145.
73. Provatidis, C.G. *Precursors of Isogeometric Analysis: Finite Elements, Boundary Elements, and Collocation Methods*. Solid Mechanics and Its Applications. Springer International Publishing: Cham, 2019. https://doi.org/10.1007/978-3-030-03889-2
74. Piegl, L.; Tiller, W. A menagerie of rational B-spline circles. *IEEE Computer Graphics and Applications* **1989**; *9*(5), 48–56. https://doi.org/10.1109/38.35537

 doi:10.20944/preprints202510.1500.v1

75. Provatidis C. How accurately can spherical caps be represented by rational quadratic polynomials? WSEAS Transactions on Computers **2021**;*20*, 139–146. https://doi.org/10.37394/23201.2021.20.17 (https://wseas.com/journals/articles.php?id=273).
76. Cobb, J.E. Tiling the Sphere with Rational Bézier Patches, TR UUCS-88-009, 1988. Download from: https://core.ac.uk/download/pdf/276277507.pdf (see also, Letter to the editor, *Computer Aided Geometric Design* **1989**; *6*(1)).
77. Levenberg, K. A Method for the Solution of Certain Problems in Least-Squares. *Quarterly Applied Mathematics* **1944**;*2*(2):164–168. https://doi.org/10.1090/qam/10666.
78. Marquardt, D. An Algorithm for Least-squares Estimation of Nonlinear Parameters. *SIAM Journal Applied Mathematics* **1963**;*11*(2):431–441. https://doi.org/10.1137/0111030.
79. Carslaw, H.S.; Jaeger, J.C. *Conduction of Heat in Solids*, 2nd ed. Oxford: Oxford University Press, 1969.