Isogeometric Transfinite Elements: A Unified B-Spline Framework for Arbitrary Node Layouts

Christopher G. Provatidis *

Posted Date: 27 November 2025

doi: 10.20944/preprints202511.2141.v1

Keywords: transfinite elements; B-spline functions; unstructured grids; partition of unity; isogeometric analysis; Coons-patch elements; Gauß–Lobatto–Legendre (GLL) points; finite element method

 doi:10.20944/preprints202511.2141.v1

Article

Isogeometric Transfinite Elements: A Unified B-Spline Framework for Arbitrary Node Layouts

Christopher Provatidis

National Technical University of Athens, 9 Iroon Polytechniou Str., 15780 Athens, Greece; cprovat@mail.ntua.gr

Abstract

This paper presents a unified framework for constructing partially unstructured B-spline transfinite finite elements with arbitrary nodal distributions. Three novel distinct classes of elements are investigated and compared with older single Coons-patch elements. 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. For all three 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 transfinite element. 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; unstructured grids; partition of unity; isogeometric analysis; Coons-patch elements; Gauß–Lobatto–Legendre (GLL) points; 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 breakpoint spans [8], they were initially regarded more as integration cells than as B-spline

 doi:10.20944/preprints202511.2141.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 in the form of Lagrange polynomials. The univariate trial functions defined along the mesh lines (often referred to as *stations*) are commonly selected as *piecewise linear*, *piecewise Hermite*, or *cubic splines*. Note that, since spline theory had not yet been standardized at the time, the term *cubic splines* was used to denote a globally smooth cubic curve composed of several cubic Hermite segments, rather than a spline space defined in the modern B-spline sense.

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.

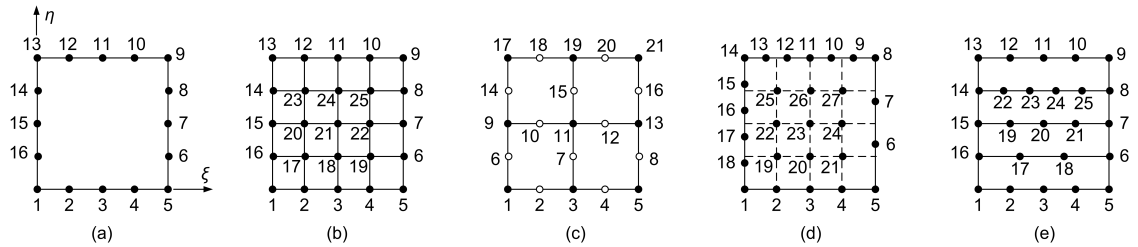


Figure 1. (**a**) Coons element. (**b**) Tensor-product element. (**c**) Classical transfinite element. (**d**) Arbitrary-noded boundary with inner tensor-product element. (**e**) One-directional element.

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 [17]:

- 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 a 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, while the rationale behind the tensor-product B-spline is relatively easy to grasp, the same cannot be

 doi:10.20944/preprints202511.2141.v1

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 Cox-de Boor 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. For instance, at the intersection point of a ξ-isoline with an η-isoline, the associated coefficient α is not uniquely defined. Addressing this incompatibility is the central objective of the present study.

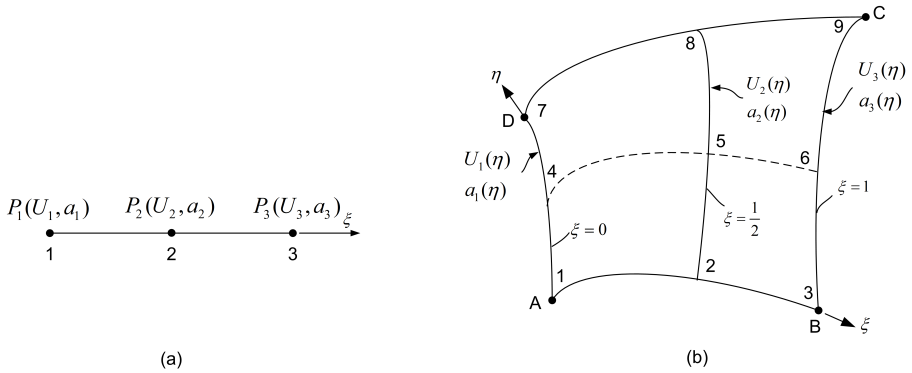


Figure 2. (**a**) One-dimensional curve. (**b**) Two-dimensional patch.

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/preprints202511.2141.v1

The paper 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 study is to validate the approach of treating the discrete coefficients as a bivariate function $a(\xi, \eta)$.

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 presents the classical transfinite element formulation, rewritten in a B-spline framework. Section 4 describes 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 discusses software-related aspects. Section 7 provides numerical results for all models. Section 8 briefly refers to a possible implementation of the collocation method as an alternative. Section 9 offers a detailed discussion, and Section 10 presents the conclusions. The paper is supplemented by four appendices.

2. Transfinite Interpolation Using B-Splines

2.1. Terminology

In classical computer-aided geometric design (CAGD) [18], 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.

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.

For instance, considering the 21-node classical transfinite element (Figure 1c), the distinction between blending functions and trial functions is illustrated in Figure 3. Specifically, perpendicular to the ξ-direction, there are three stations with ending nodes: 1–17, 3–19, and 5–21, respectively; hence, the corresponding blending functions $E_1(\xi)$, $E_2(\xi)$, and $E_3(\xi)$ are Lagrange polynomials of degree 2. Similarly, perpendicular to the η-direction, there are three stations with ending nodes: 1–5, 9–13, and 17–21, respectively; thus, the associated blending functions $E_1(\eta)$, $E_2(\eta)$, and $E_3(\eta)$ are also Lagrange polynomials of degree 2. Furthermore, considering a typical station, such as the middle horizontal one—comprising nodes 9–10–11–12–13—the function $U(\xi, \frac{1}{2})$ across these five nodes can be interpolated using a set of five Lagrange polynomials in ξ of degree 4. These five functions constitute the aforementioned trial functions $\tilde{B}_i(\xi)$, $i = 1, \ldots, 5$. In general, each station, whether horizontal or vertical, is interpolated by a distinct set of trial functions.

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.

 doi:10.20944/preprints202511.2141.v1

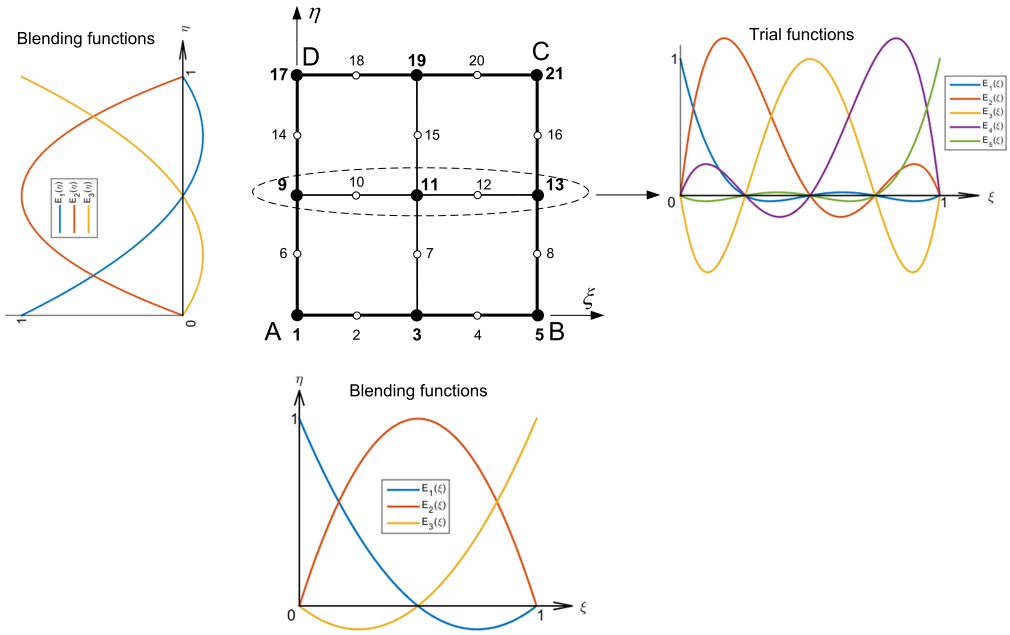


Figure 3. Transfinite element with the blending functions and a typical set of trial functions.

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 stations.

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:

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

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

 doi:10.20944/preprints202511.2141.v1

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, $P_{\tilde{\xi}}\{U\}$ is written simply as $P_{\tilde{\xi}}$, and similarly for the others.

2.3. Extension of the Projectors Using Bernstein Polynomials

The classical interpretation of the well-known projectors ($P_{\tilde{\xi}}$, P_{η}, and $P_{\tilde{\xi}\eta}$) 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 [19], 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 $P_{\tilde{\xi}}$; 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 $\tilde{E}_3(\xi)=B_{3,2}(\xi)$:

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

The above equations can be extended from one to two dimensions, beginning with the projector $P_{\tilde{\xi}}$, 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:

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

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

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

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 (Eq. (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:

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

 doi:10.20944/preprints202511.2141.v1

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 4d. 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 $\xi = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1$.

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 vertical station values. Similarly, in the η-direction, the blending functions may be Lagrange or Bernstein polynomials of degree 3, associated with four horizontal station values.

Alternatively, when employing cubic B-splines, the ξ-direction can be represented using the global 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 global 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.

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 Eq. (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:

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

 doi:10.20944/preprints202511.2141.v1

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 4d—are composed of three terms, as exemplified by Eq. (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 Eq. (13).

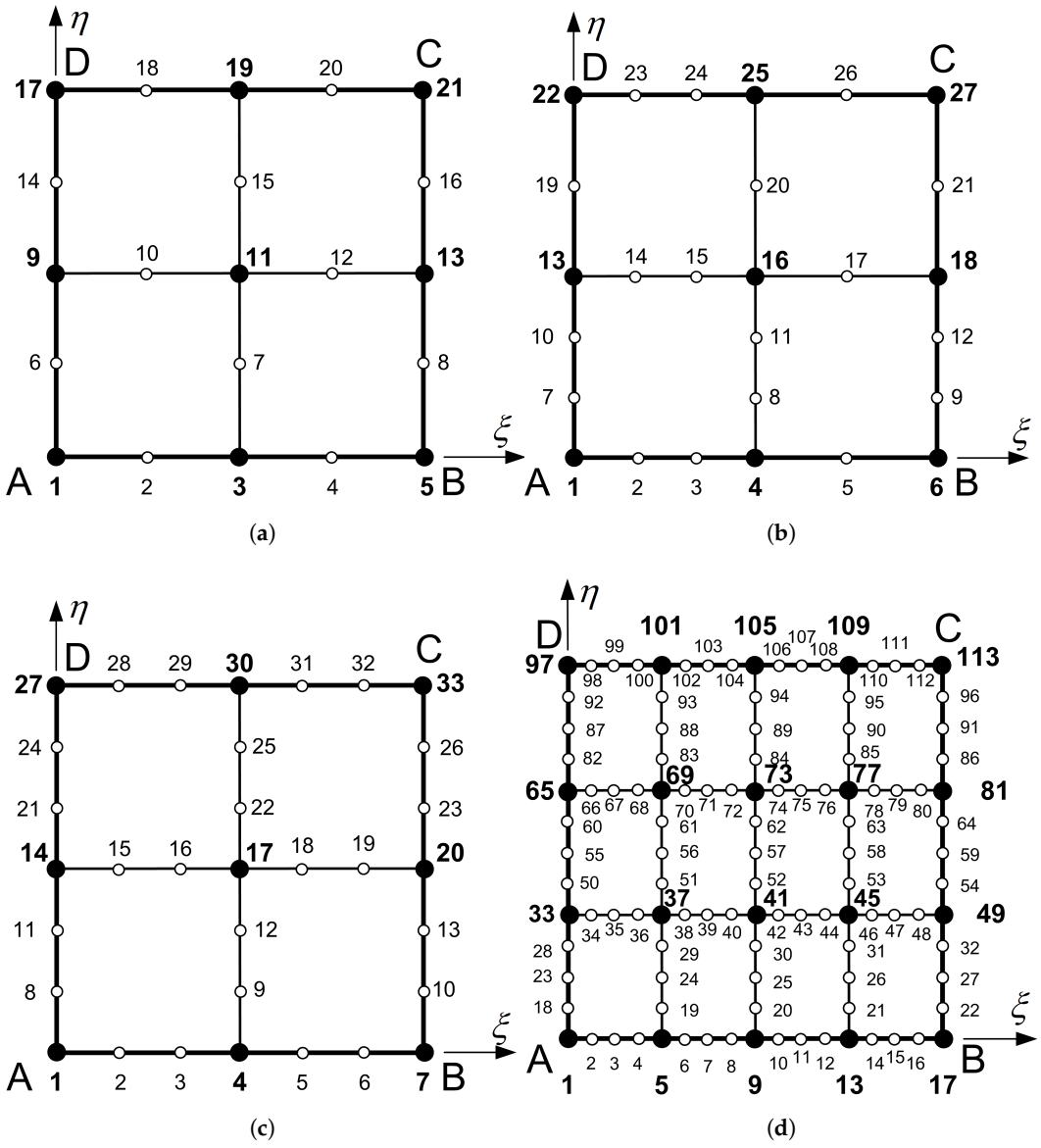


Figure 4. A collection of classical transfinite elements: (**a**) 21-node element. (**b**) 27-node element. (**c**) 33-node element. (**d**) 113-node element.

 doi:10.20944/preprints202511.2141.v1

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 4d):

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 5a. Notably, elevated function values are observed along the vertical edges, indicating localized influence. Furthermore, the constructed basis satisfies the partition of unity property:

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

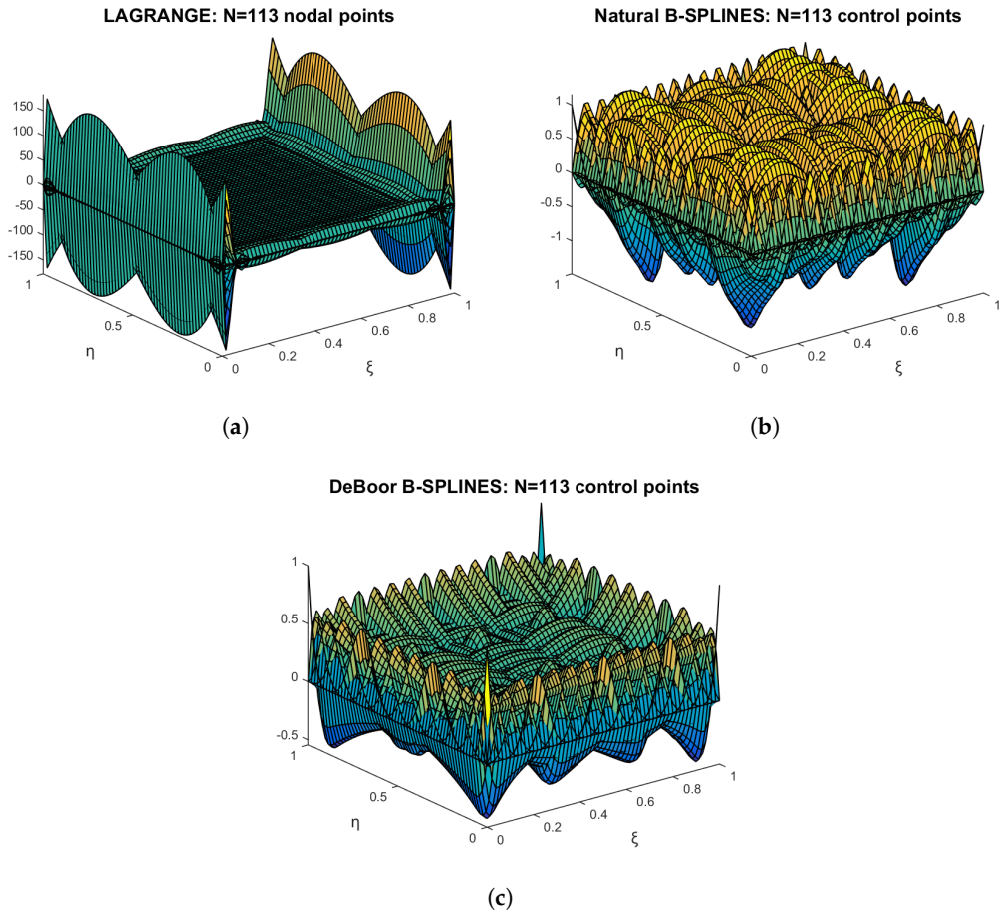


Figure 5. Bivariate basis functions for the 113-node transfinite element, using Lagrange polynomials as blending functions, while trial functions are: (**a**) Lagrange polynomials; (**b**) natural cubic cardinal B-splines; (**c**) de Boor cubic B-splines.

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

 doi:10.20944/preprints202511.2141.v1

values U (i.e., they are of $[0,1]$-type). This choice effectively reproduces the behavior of the Lagrange polynomials used in Model 1 and thus eliminates the need for the 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 [20], 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 5b. These basis functions have been verified to satisfy the partition of unity property:

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

exhibit unity as their upper bound, and satisfy the Kronecker-delta property ($\psi_i(\xi_j,\eta_j) = \delta_{ij}$).

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 Appendix 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 4d 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 5c.

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.

 doi:10.20944/preprints202511.2141.v1

4.1. Unidirectional Multi-Layer Transfinite Lagrange and Bernstein Elements

The unidirectional analog of Eq. (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:

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

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.

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 6. Let us focus on the 12-node element (Figure 6a). Replacing Eq. (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:

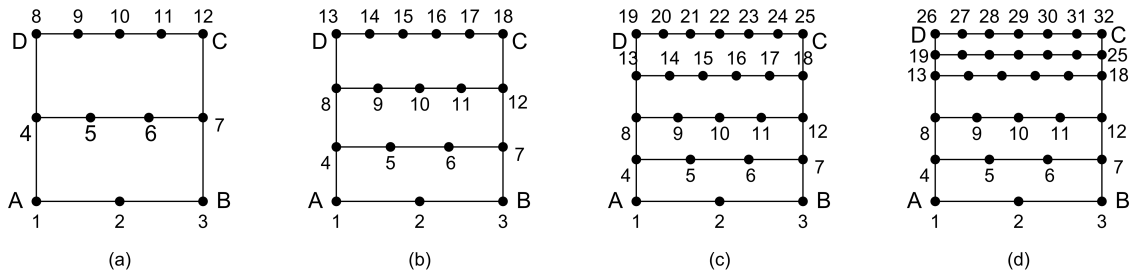


Figure 6. Transitional elements: (**a**) 12-node; (**b**) 18-node; (**c**) 25-node; (**d**) 32-node element.

 doi:10.20944/preprints202511.2141.v1

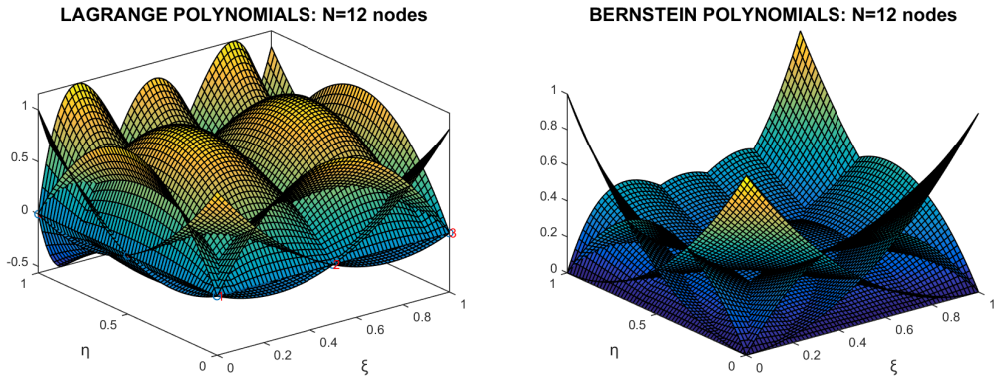
$$
\begin{aligned}
\phi_1(\xi,\eta) &= L_{1,2}(\xi)L_{1,2}(\eta),\\
\phi_2(\xi,\eta) &= L_{2,2}(\xi)L_{1,2}(\eta),\\
\phi_3(\xi,\eta) &= L_{3,2}(\xi)L_{1,2}(\eta),\\
&- - - - -\\
\phi_4(\xi,\eta) &= L_{1,3}(\xi)L_{2,2}(\eta),\\
\phi_5(\xi,\eta) &= L_{2,3}(\xi)L_{2,2}(\eta),\\
\phi_6(\xi,\eta) &= L_{3,3}(\xi)L_{2,2}(\eta),\\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

Next, replacing Lagrange polynomials with their Bernstein counterparts, or alternatively applying the projector defined by the analog of Eq. (18) (including three terms associated with the three layers in Figure 6a) 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:

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

The shape functions $\phi_i(\xi,\eta)$ based on Lagrange polynomials (Eq. (19)) are illustrated in Figure 7a, while the basis functions $\psi_i(\xi,\eta)$ corresponding to Bernstein polynomials (Eq. (20)) are shown in Figure 7b. This element is not provided for B-spline interpolation, as it contains too few nodes to support such a representation.

 doi:10.20944/preprints202511.2141.v1



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

Figure 7. 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 6b. 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, $B_{i,3}(\xi), i = 1,2,3,4$.
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 $[0,0,0,0,\frac{2}{5},\frac{3}{5},1,1,1,1]$.

 doi:10.20944/preprints202511.2141.v1

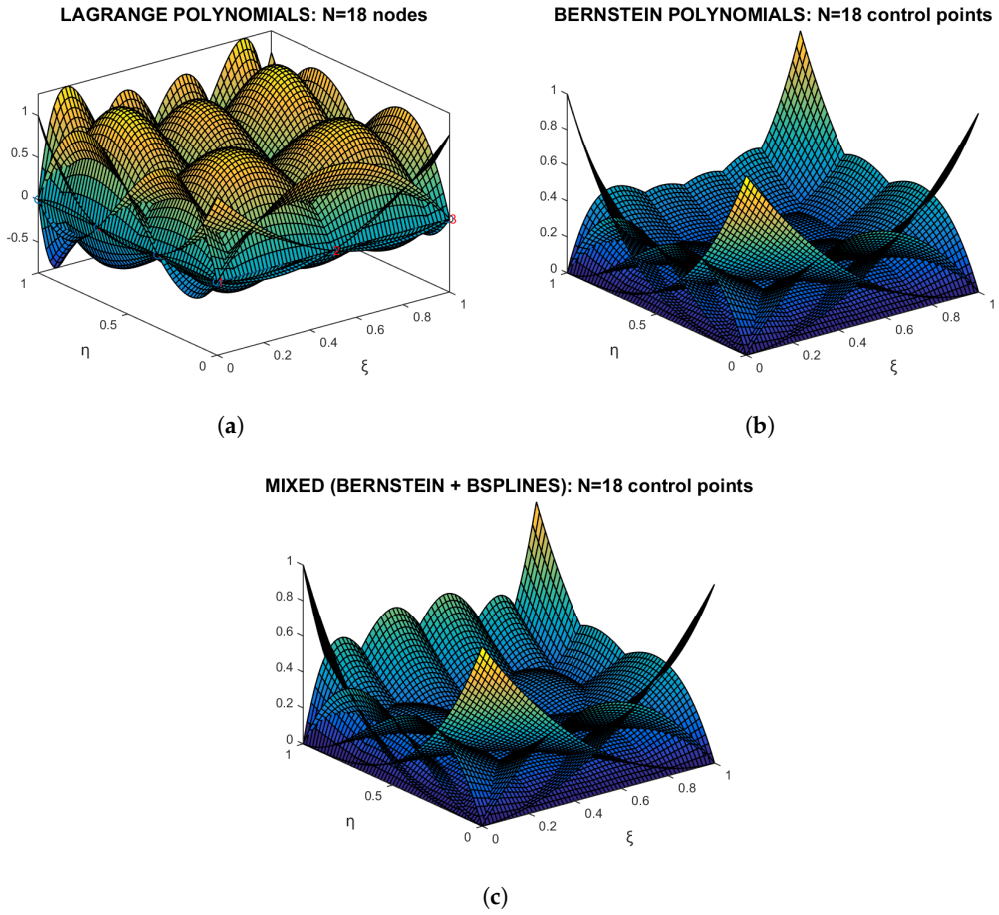
Writing a projector similar to that of Eq. (18), in conjunction with four parallel layers (stations) at $\eta = 0, \frac{1}{3}, \frac{2}{3}, 1$, we obtain the following basis functions:

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

Obviously, the same form as Eq. (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 8.

Regarding the estimation of the stiffness matrix of the 18-node transfinite element in Figure 6b, 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).

 doi:10.20944/preprints202511.2141.v1



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

(**c**)

Figure 8. Basis functions for the 18-node transfinite element: (**a**) Lagrange polynomials; (**b**) Bernstein polynomials; (**c**) Mixed, Bernstein polynomials and B-splines.

Note: When the number of parallel sections increases to 5 (and thus the number of nodes becomes 25, as shown in Figure 6c), 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 6d), results and discussion are given in Sect. 7.1.2.

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 [16,17]. In the 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 illustrative example, we consider a 27-node transfinite element defined over the patch $ABCD$, whose four edges are uniformly partitioned into 4, 3, 6, and 5 segments, respectively, as shown in Figure 9. For this particular edge configuration, it is possible to construct at least three distinct formulations of the transfinite element, each based on different combinations of blending functions and trial (interpolating) functions, as outlined below.

 doi:10.20944/preprints202511.2141.v1

- 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 9), 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.

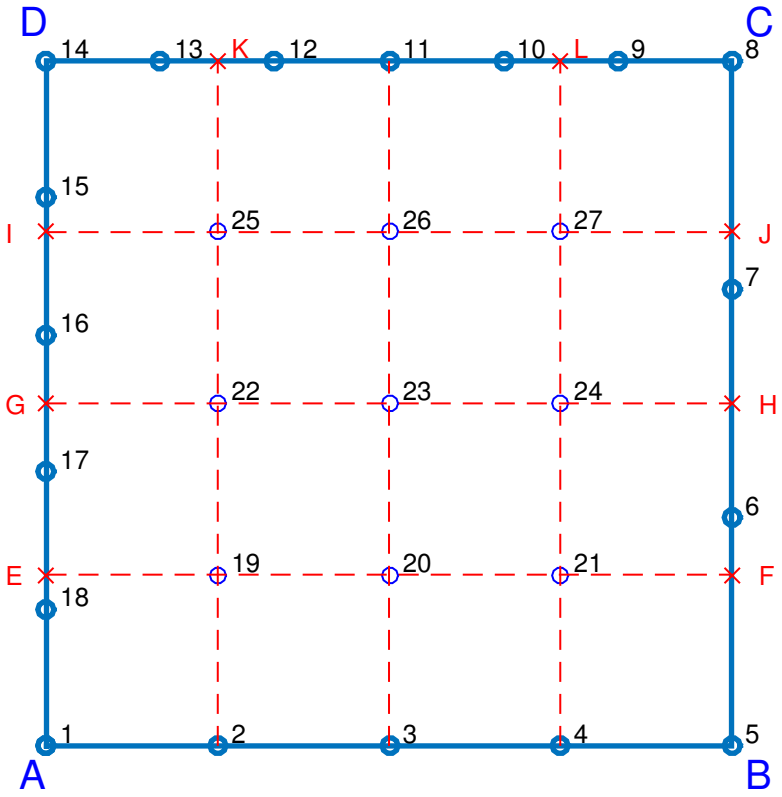


Figure 9. 27-node transfinite element.

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.

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

 doi:10.20944/preprints202511.2141.v1

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

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/preprints202511.2141.v1

the substitution of Eq. (22) to Eq. (25) into the Boolean sum given by Eq. (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/preprints202511.2141.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 Eq. (26) and described by Eq. (27), is shown in Figure 10. It has also been verified that the functional set $\{\psi_j(\xi,\eta)\}$ satisfies the partition of unity property, i.e.,

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

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 Eq. (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).

 doi:10.20944/preprints202511.2141.v1

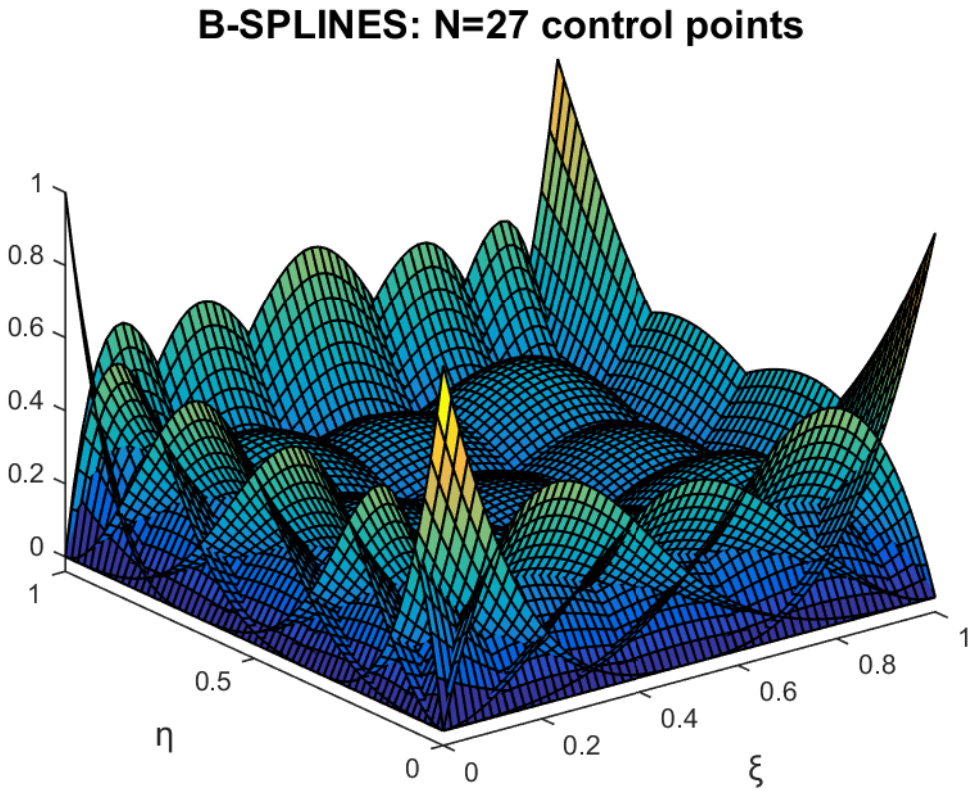


Figure 10. 27-DOF B-spline transfinite element.

6. 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 [21].

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.

6.1. 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 [19, p. 26]. Additionally, a tensor-product implementation of uniform Lagrange polynomials, named `shape_full_lagrange`, is provided in [19, 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 Appendix B. Gaussian integration was performed using the subroutine `lgwt` (see, [22]).

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

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

 doi:10.20944/preprints202511.2141.v1

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 Eq. (19)—the complete subroutine is provided in Appendix C.

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

6.2. 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. 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:

$$L_2 = \frac{\int_\Omega (U_{\text{calculated}} - U_{\text{exact}})^2 \, d\Omega}{\int_\Omega (U_{\text{exact}})^2 \, d\Omega} \times 100\,(\%)\,. \tag{29}$$

The third example concerns an eigenvalue problem, for which details on the error are provided in the dedicated subsection 7.3. Finally, the fourth example is a patch test in plane elasticity (Sect. 7.4).

7.1. Example 1

A square plate ABCD is subjected to steady-state heat conduction, with boundary conditions illustrated in Figure 11. The exact solution is given by:

$$U(x,y) = U_m \frac{\sinh\left(\frac{\pi y}{2a}\right)}{\sinh\left(\frac{\pi b}{2a}\right)} \cos\left(\frac{\pi x}{2a}\right). \tag{30}$$

 doi:10.20944/preprints202511.2141.v1

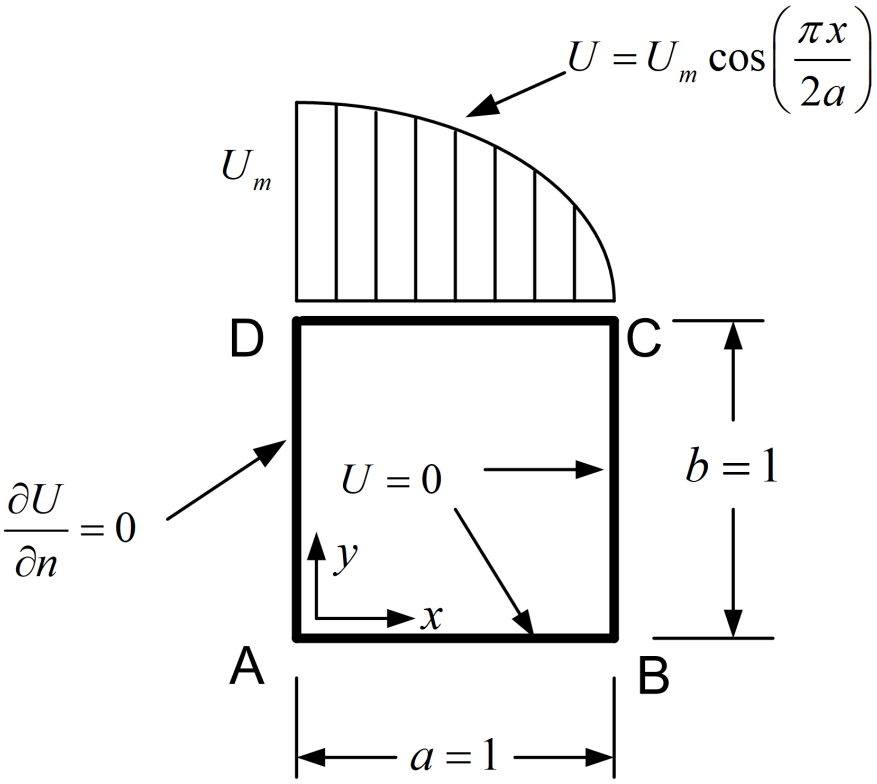


Figure 11. Geometry and boundary conditions.

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

- Classical B-spline transfinite element (21-27-33-113 nodes; see Figure 4).
- 12-node element (see Figure 6a).
- 18-node element (see Figure 6b).
- 25-node element (see Figure 6c).
- 27-node element (see Figure 9).

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:

7.1.1. 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 4) 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 4

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}

More precisely, for the particular case of the 113-degree-of-freedom element shown in Figure 4d, we have:

- 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: $L_2 =$

 doi:10.20944/preprints202511.2141.v1

1.9140×10^{-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.

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

These elements belong to the *second class* studied in the present paper (Figure 6). 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.

The 32-node element (shown in Figure 6d) 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 $\eta = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \frac{7}{8}, 1$. 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 $L_2 = 0.0173\%$.

Table 2. Errors (L_2 in %) for the Unidirectional Elements shown in Figure 6.

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: $\Xi = [0, 0, 0, 0, \frac{1}{10}, \frac{3}{10}, \frac{6}{10}, 1, 1, 1, 1]$.

7.1.3. Arbitrary-Noded 27-Node Transfinite Element

This element belongs to the *third class* examined in the present study (Sect. 5.1, Figure 9). 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 9

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 corner 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.

 doi:10.20944/preprints202511.2141.v1

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. 7.2.1.

7.1.4. Coons-Patch Elements

This represents a type of element in addition to the three 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 6a–d (with 10, 13, 16, and 18 nodes, respectively), and further detailed in Figure 12, the corresponding numerical results are reported in the left half of Table 4. The results indicate that, with standard *linear* blending functions, $E_1(\xi) = 1 - \xi$, $E_2(\xi) = \xi$ (first set), 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.

For completeness, in addition to the abovementioned standard linear blending functions, we also examined 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). In contrast to the above first set, the second set—although heuristic in nature and inspired by the boundary conditions imposed on the top edge—leads rapidly to the exact solution (shown in the right half of Table 4).

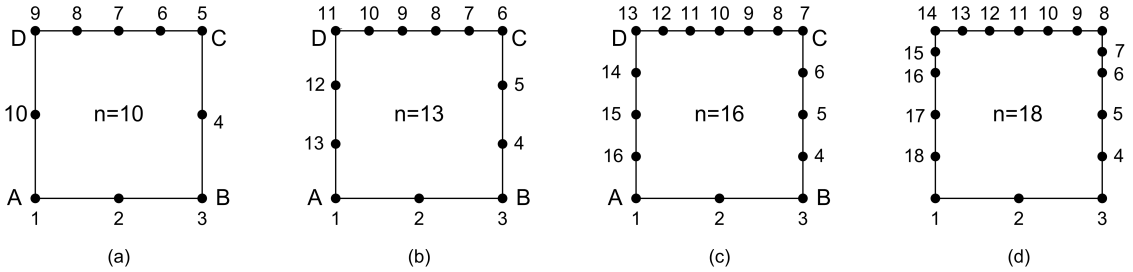


Figure 12. Coons elements.

Table 4. 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}

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.

 doi:10.20944/preprints202511.2141.v1

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.

7.2. 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 13).

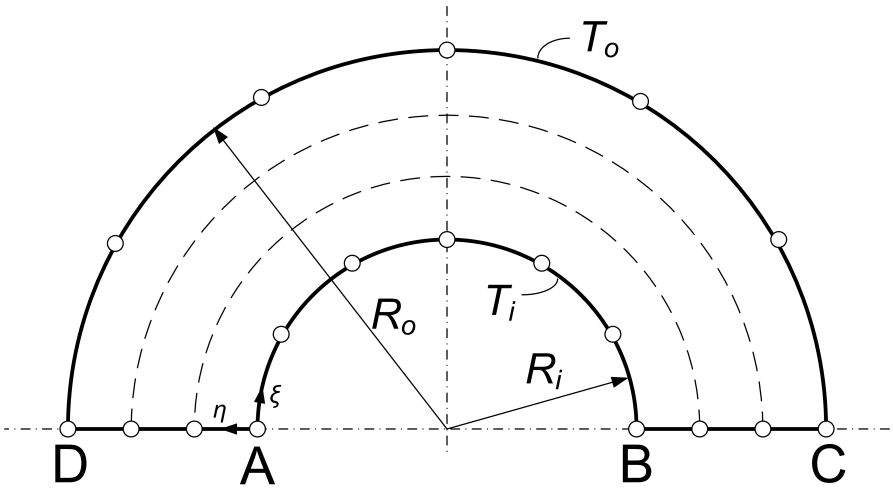


Figure 13. Half-Annulus ($n_\xi = 6, n_\eta = 3$).

According to [23], the steady-state temperature distribution in the radial direction is given by:

$$T(r) = T_i + \frac{(T_o - T_i)}{\log\left(\frac{R_o}{R_i}\right)} \log\left(\frac{r}{R_i}\right) \tag{31}$$

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 macroelements based on cardinal natural cubic B-splines [8], formulated via truncated power series (i.e., not the procedure shown in Appendix 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. 7.2.1) and progressively introduce internal nodes (Sect. 7.2.2).

7.2.1. 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,

 doi:10.20944/preprints202511.2141.v1

(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 [24–26].

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., $n_{c,\xi} = n_\xi + 3$) [21].

The parameterization of the patch is illustrated in Figure 13:
- 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 (T_i, T_o, respectively)
- 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 total degrees of freedom includes $2(n_\xi + n_\eta)$ nodal values, T_i, all located on the boundary of the patch, of which $2(n_\eta - 1)$ are unknown. 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 14 (blue line).

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 14 (red line), are identical whether the old-fashioned methodology of Ref. [8] (based on truncated powers) or the procedure of Appendix A is used—namely, the Curry-Schoenberg formulation [24], computed via the Cox-de Boor algorithm [25,26], with vanishing second derivatives at the ends of each edge.

 doi:10.20944/preprints202511.2141.v1

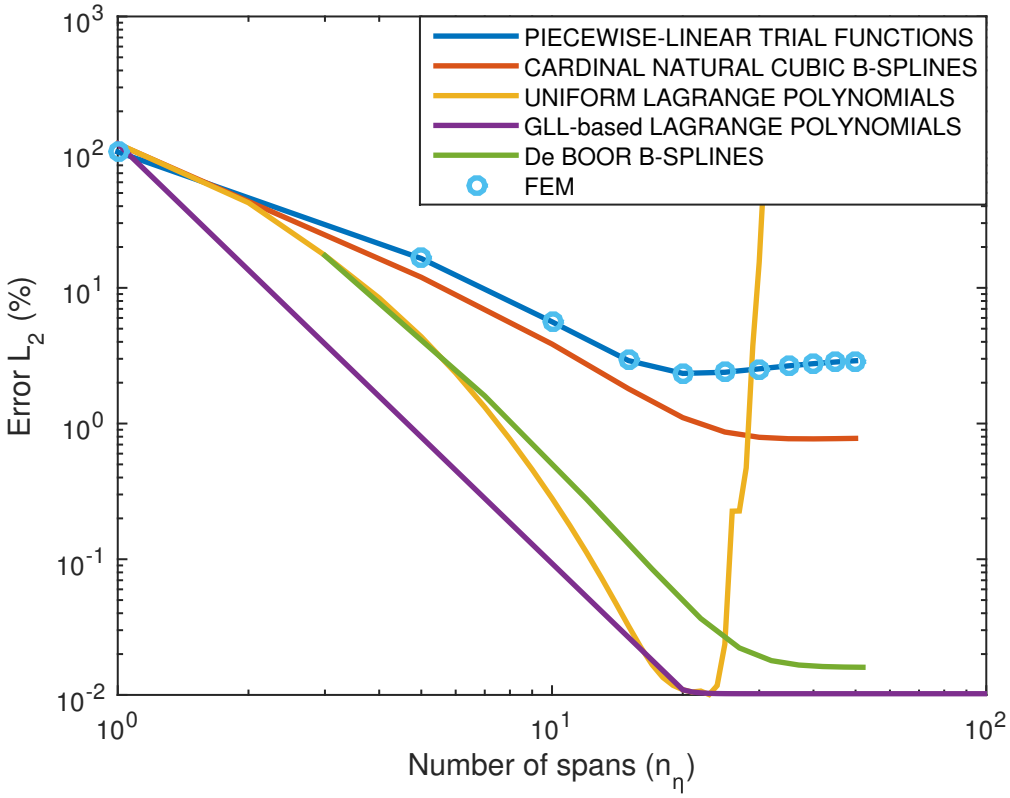


Figure 14. Convergence quality of numerical solution for the circular ring.

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 14 (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* $\Xi = [0,0,0,0,\xi_1,\ldots,\xi_{n_\xi},1,1,1,1]$, 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 Appendix 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) [24] and implemented via the recursive Cox–de Boor algorithm (1972) [25,26]. Note that, since the *linear* blending functions used in Coons

 doi:10.20944/preprints202511.2141.v1

interpolation are identical across all formulations (Lagrange, Bernstein, and B-splines), the B-spline version of the Coons element follows directly.

Model C5-I: 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 13 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 14 (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 C5-II: Bernstein polynomials. A variation of Model C5-I is to retain the same linear blending functions while replacing the *uniform* Lagrange trial functions of degree (p,q) with their corresponding *Bernstein* polynomials. For the numerical examples considered, both formulations produce identical results, and a theoretical justification is provided in Ref. [17]. Briefly, along each edge of the patch $ABCD$, the univariate Bernstein polynomials are linear combinations of the respective Lagrange polynomials; therefore, the same relation holds for the bivariate basis functions obtained by tensor products. As a consequence, the stiffness matrix of the Bernstein-based system is a quadratic form of the stiffness matrix obtained with Lagrange polynomials. The only practical difference is that Lagrange polynomials yield cardinal basis functions, whereas Bernstein polynomials lead to non-cardinal—yet equally complete—trial functions.

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 [27], one of the best choices is the set of *Gauß–Lobatto–Legendre* (GLL) points, which are defined as follows:

$$\xi_i^{\text{GLL}} = \begin{cases} -1 & \text{for } i = 1 \\ \hat{\xi}_i & \text{for } i = 2,3,\ldots,p \\ +1 & \text{for } i = p+1\,, \end{cases} \tag{32}$$

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:

$$Lo_{p-1}(\xi) = \frac{dL_p(\xi)}{d\xi}\,. \tag{33}$$

Practically, for any given degree p, the $(p+1)$ GLL points $\xi_i^{\text{GLL}} \in [-1,1]$, defined in Eq. (32), 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/preprints202511.2141.v1

$L_2 = 0.0102053\%$ (Figure 13). 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 14 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 model.
2. Piecewise-linear Coons interpolation model 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 model is characterized by a smaller error than the above piecewise-linear model and FEM.
4. Uniform Lagrange polynomials model leads to a smaller error but, after the value $n_\eta = 21$, diverge.
5. Cox-de Boor cubic B-splines model closely follows 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 model based on GLL points rapidly converges 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 (e.g., rational Bézier of degree $p = 2$).

7.2.2. Transfinite Elements

A. Classical Transfinite Elements

Each of the four classical transfinite elements shown in Figure 4 is mapped onto the entire half-annulus depicted in Figure 13. 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:

$$r = R_i + \eta(R_o - R_i), \tag{34}$$

$$\theta = (1-\xi)\,\pi\,. \tag{35}$$

 doi:10.20944/preprints202511.2141.v1

The numerical results are reported in Table 5, where convergence behavior can be observed. As in Example 1, the same result was obtained when using Bernstein polynomials.

Table 5. 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 14 (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 5), 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. 7.2.1. 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 5).

7.3. 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 partially Dirichlet ($p = 0$ at $y = 0$: bottom edge) and Neumann boundary conditions ($\partial p/\partial n = 0$) elsewhere. The governing wave equation is

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \nabla^2 p = 0, \tag{36}$$

where p denotes the acoustic pressure. According to Courant and Hilbert [28], the exact eigenvectors are given by:

$$p_{mn} = \cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{(2n+1)\pi y}{2b}\right), \quad m, n = 0, 1, \ldots, \infty, \tag{37}$$

whence the exact eigenvalues are:

$$\lambda_{mn}^{\text{exact}} = \omega_{mn}^2 = \pi^2 c^2\left(\frac{m^2}{a^2} + \frac{(2n+1)^2}{(2b)^2}\right), \quad m, n = 0, 1, \ldots, \infty. \tag{38}$$

The application of the Galerkin procedure to Eq. (36) for the eigenvalue problem, leads to the following matrix formulation:

$$[\mathbf{M}]\{\ddot{\mathbf{p}}\} + [\mathbf{K}]\{\mathbf{p}\} = 0, \tag{39}$$

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

$$m_{ij} = \frac{1}{c^2}\int_\Omega N_i N_j \, d\Omega, \quad k_{ij} = \int_\Omega \nabla N_i \cdot \nabla N_j \, d\Omega \tag{40}$$

 doi:10.20944/preprints202511.2141.v1

The eigenvalues λ_{mn} are obtained by solving the characteristic equation:

$$\det\left(\mathbf{K}_f - \lambda \mathbf{M}_f\right) = 0 \tag{41}$$

where $\mathbf{K}_f$ and $\mathbf{M}_f$ represent the stiffness and mass matrices, respectively, after applying Dirichlet boundary conditions.

Next, three classes of transfinite elements are studied. For all modes the error (in percent) was calculated by the formula:

$$E_{mn} = \frac{\lambda_{mn}^{\text{calculated}} - \lambda_{mn}^{\text{exact}}}{\lambda_{mn}^{\text{exact}}} \times 100(\%) \tag{42}$$

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

For the four classical transfinite elements in Figure 4 in conjunction with Cox-de Boor B-splines (for both the blending and trial functions), Table 6 shows the accuracy of the computed eigenvalues. All modal values converge to the reference solution. Note that the 113-node element values are scaled by $\times 10^{-3}$.

Table 6. Errors (E_{mn} in %) of calculated eigenvalues using classical transfinite elements (Figure 4).

MODEL	Mode-1	Mode-2	Mode-3	Mode-4	Mode-5	Mode-6	Mode-7
21-node	0.0005	0.0334	0.1028	13.5363	9.6868	27.8023	18.6680
27-node	0.0001	0.0108	0.6582	0.1905	0.0755	0.5610	2.7985
33-node	0.0001	0.0098	0.0907	2.3633	0.0160	0.5051	2.7342
113-node	0.0000 $\times 10^{-3}$	0.0004 $\times 10^{-3}$	0.0208 $\times 10^{-3}$	0.7636 $\times 10^{-3}$	0.0371 $\times 10^{-3}$	0.1873 $\times 10^{-3}$	0.8891 $\times 10^{-3}$

7.3.2. Unidirectional (Multi-Layer) 12-18-25-32-Node Elements

Regarding the blending functions, for the 12-node element (with three horizontal layers) we employed B-splines of degree $p = 2$ (quadratic Bernstein), whereas in all other cases we used standard B-splines of degree $p = 3$.

All trial functions along the horizontal layers were taken to be Cox–de Boor B-splines. More precisely, the bottom edge 1–2–3 was represented using degree-$p = 2$ B-splines (although all associated DOFs were eliminated due to Dirichlet boundary conditions), while all remaining layers were interpolated using degree-$p = 3$ B-splines.

For the first seven modes, the computed eigenvalues are listed in Table 7.

Table 7. Errors (E_{mn} in %) of calculated eigenvalues using unidirectional (multi-layer) transfinite elements (Figure 6).

MODEL	Mode-1	Mode-2	Mode-3	Mode-4	Mode-5	Mode-6	Mode-7
12-node	0.7522	0.4482	7.5494	63.6512	53.5242	46.8740	43.1462
18-node	0.0137	0.0203	1.2620	18.9535	7.8167	4.9200	12.4160
25-node	0.0005	0.0074	0.5855	10.0763	0.5376	0.5007	2.2968
32-node	0.0001	0.0040	0.3422	5.4634	0.0755	0.0730	0.9024

7.3.3. Coons-Patch Elements

The entire acoustic cavity is now modelled using a single B-spline Coons-patch element, with 10, 13, 16, and 18 boundary nodes, as shown in Figure 12a–d. The blending functions were linear, i.e., $E_0(\xi) = 1 - \xi$ and $E_1(\xi) = \xi$. In all models considered, the bottom edge AB consists of only three nodes (1–2–3) and was therefore approximated with quadratic trial functions using the knot vector ($p_{AB} = 2$, $\Xi = [0, 0, 0, 1, 1, 1]$).

Except for the vertical edges in the coarsest model (12 nodes, Figure 12a), which also contain only three nodes and thus required quadratic approximation, all remaining boundary curves were

 doi:10.20944/preprints202511.2141.v1

interpolated using cubic B-splines. The results for the first seven modes are listed in Table 8. One may observe that, aside from the first and fifth eigenvalues—which converge rapidly—all others tend to values higher than the exact solution.

Table 8. Errors (E_{mn} in %) of calculated eigenvalues using single Coons-patch macroelements.

MODEL	Mode-1	Mode-2	Mode-3	Mode-4	Mode-5	Mode-6	Mode-7
10-node	0.7522	0.7573	5.3454	63.6512	54.7545	49.5222	108.5697
13-node	0.0137	0.3075	5.3376	3.9774	5.3346	6.6324	124.0532
16-node	0.0005	0.3010	5.3341	4.8844	0.5376	2.2317	12.8249
18-node	0.0001	0.3008	5.3341	4.8837	0.0663	1.7983	12.8249

7.3.4. Finite Element Solution

For the sake of completeness, in addition to the preceding results of the proposed models, we also provide in Table 9 the numerical solution obtained using conventional bilinear rectangular finite elements (FEM solution). The computational mesh of the FEM model is uniform and matches the maximum number of nodes (per direction) used in the previous models. For reference, we also include the exact mode shapes (m, n) together with their corresponding eigenvalues.

Table 9. Errors (E_{mn} in %) of calculated eigenvalues using conventional FEM.

			FEM: Error (in %)			
MODE	**(m,n)**	ω^2_{mn}	**5** × **3** (15 DOF)	**6** × **4** (24 DOF)	**7** × **5** (35 DOF)	**21** × **11** (231 DOF)
1	(0,0)	2.039174	20.6150	11.3761	7.0307	1.1640
2	(1,0)	3.618311	43.9549	18.0698	9.9169	1.3161
3	(2,0)	8.355721	43.8271	18.2452	11.0758	1.1431
4	(3,0)	16.251405	46.7519	28.8735	19.9321	1.8626
5	(0,1)	18.352570	45.2297	23.8842	14.0259	2.5834
6	(1,1)	19.931707	50.7517	30.8681	19.5464	3.4274
7	(2,1)	24.669117	34.6244	35.1754	21.8653	3.0007

Overall, the convergence performance of all models for the three lowest modes is illustrated in Figure 15. One may observe that:

1. The class of classical transfinite interpolation (TFI) elements exhibits the highest convergence rate.
2. The Coons element (labelled COONS) is sufficiently accurate for the first mode, but for higher modes it experiences difficulty converging toward the exact solution.
3. The unidirectional (multi-layer) elements (labelled LAYER) provide adequate accuracy, but their convergence rate is lower than that of the TFI formulation.
4. The FEM solution, based on a bilinear formulation, displays the lowest convergence rate.

 doi:10.20944/preprints202511.2141.v1

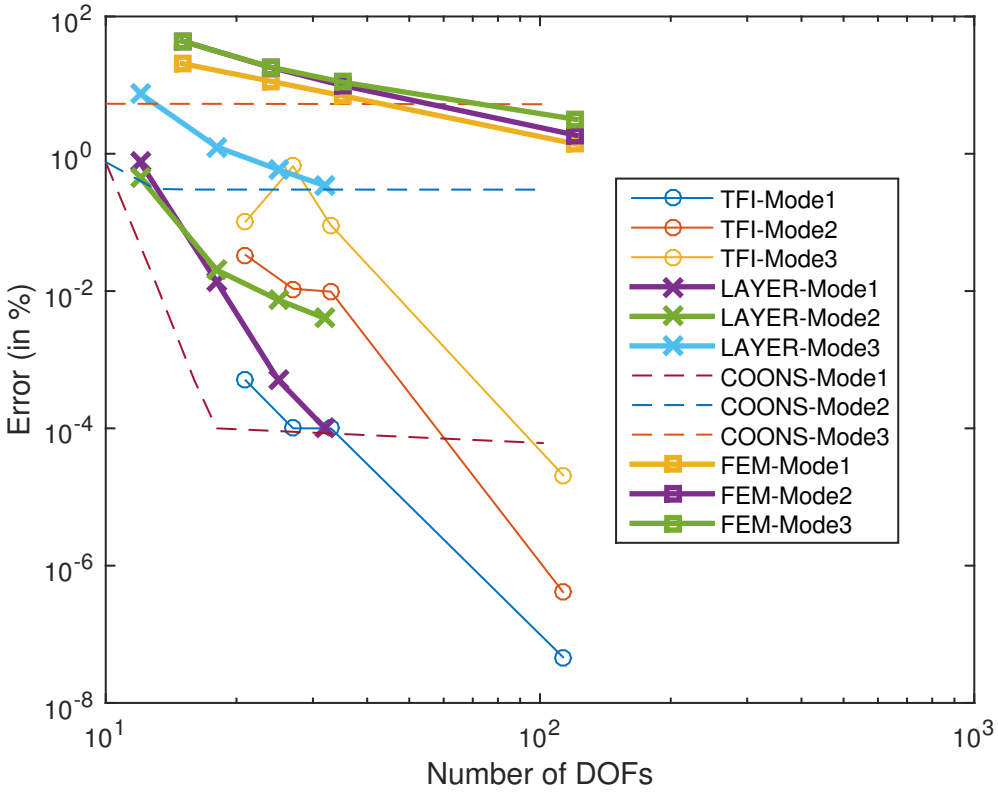


Figure 15. Convergence quality of numerical solutions for the rectangular acoustic cavity.

7.4. Example 4

Patch test in plane elasticity: Among several patch tests [29], 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 $0\le\xi,\eta\le 1$. For the 12-node element (Figure 6a), the implementation of Lagrange polynomials (Eq. (19)) or Bernstein polynomials (Eq. (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 6a) 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 (Eq. (19)) and Bernstein (Eq. (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 6a), 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 $\Xi'=[0,0,0,0,\frac{1}{2},1,1,1,1]$.

 doi:10.20944/preprints202511.2141.v1

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

8. 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 [30] and [31], 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 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 [21], 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 [32–34]—remains an open area for future investigation.

9. 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 three distinct classes, ranging from classical transfinite elements, such as those of Gordon and Hall [5], to more advanced unstructured constructs. 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

 doi:10.20944/preprints202511.2141.v1

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 Appendix 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. [35] for further details, and also refer to Model C6 around Eq. (32)).

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. [36].

An early attempt to adapt transfinite interpolation for the accurate representation of conic sections and quadrics was presented in Ref. [37], 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 4d. Specifically, the existence of a transformation matrix between the two sets of basis functions was revealed [17].

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].

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

We now refer to these three 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.

 doi:10.20944/preprints202511.2141.v1

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. 7.2). 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 [38] and GeoPDEs [39].

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 [40] 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 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 [41].

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

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

 doi:10.20944/preprints202511.2141.v1

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 capability to handle unstructured grids, and (ii) the straightforward joining of patches. These features have recently attracted considerable interest from the research community (e.g., [42–44]).

In summary, this study advances the theory and practice of transfinite interpolation by integrating Bernstein polynomials, B-spline, and NURBS constructions within a unified framework capable of handling classical elements as well as structured, semi-structured, and irregular patches. Although fully unstructured B-splines on curved domains and T-meshes were not explored here, practical strategies were demonstrated to achieve consistent parametrizations and accurate geometric representations within the proposed element classes. Future work includes extending the framework to three-dimensional patches, developing systematic procedures for interior control-point placement in complex geometries, and adapting the formulations for plate and shell applications. Owing to its flexibility, modularity, and compatibility with standard spline technologies, the approach provides a robust foundation for further developments in both academic research and industrial applications.

10. Conclusions

This work has shown that classical transfinite finite elements can be systematically reformulated using Bernstein polynomials and B-splines, yielding a unified isogeometric framework capable of handling both structured and partially unstructured node layouts. By expressing the blending functions and trial functions in B-spline form, and by reinterpreting the generalized coefficients as samples of an underlying continuous univariate function evaluated independently at each station, the proposed formulation enables flexible control of horizontal and vertical subdivisions while preserving key properties such as partition of unity. Numerical studies—including two potential-flow problems, an eigenvalue acoustic example, and a patch test in plane elasticity—demonstrate robust convergence for both uniform and nonuniform configurations. While the current study focuses on two-dimensional surface patches, the framework provides a solid basis for future extensions to plate and shell formulations, as well as fully three-dimensional transfinite isogeometric elements, through appropriate generalizations of the control-point and blending-function constructions.

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 endpoints. 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

 doi:10.20944/preprints202511.2141.v1

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:

$$\mathbf{U}_{\text{brk}} = \begin{bmatrix} U_1 \\ \vdots \\ U_n \\ U''_1 \\ U''_n \end{bmatrix} \tag{A4}$$

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

The inversion of Eq. (A3) implies

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

Equation (A1) is unfolded as:

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

Substituting Eq. (A6) into Eq. (A7), we receive:

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

The term within the parentheses of Eq. (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.

 doi:10.20944/preprints202511.2141.v1

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 $(N_1, \ldots, N_n)$, 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.

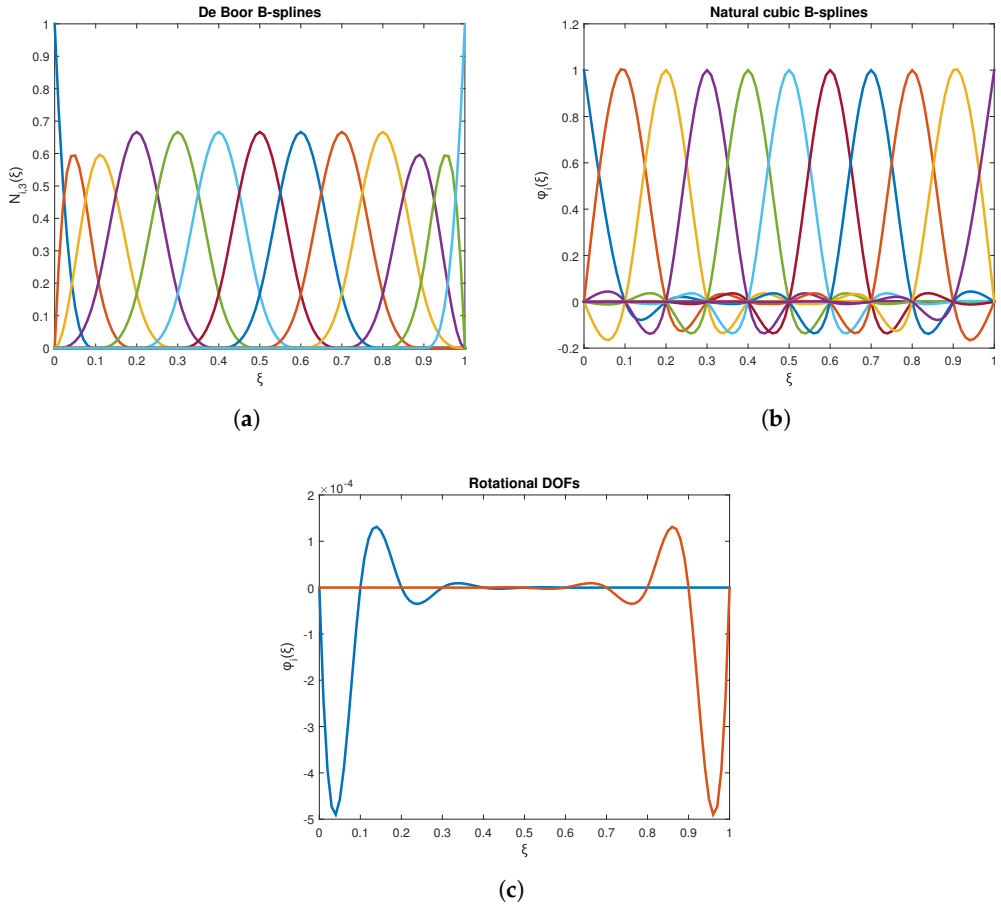


Figure A1. Basis functions for 11 uniform breaks: (**a**) de Boor B-splines, (**b**) Natural cubic B-splines, (**c**) Basis functions associated with the two rotational DOFs at the ends.

Obviously, the set of de Boor—more precisely, Curry-Schoenberg (1966) [24]—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/preprints202511.2141.v1

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

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

$$\mathbf{x}(\xi) = \sum_{i=1}^{n+1} B_{i,n}(\xi)\mathbf{P}_i, \quad 0 \le \xi \le 1, \tag{A9}$$

with the Bernstein polynomials given by

$$B_{i,n}(\xi) = \frac{n!}{(i-1)!(n-i+1)!}\xi^{i-1}(1-\xi)^{n-i+1}, \quad i = 1,\ldots,n+1 \tag{A10}$$

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:

$$U(\xi) = \sum_{i=1}^{n+1} B_{i,n}(\xi)\alpha_i, \quad 0 \le \xi \le 1\,. \tag{A11}$$

 doi:10.20944/preprints202511.2141.v1

To avoid the computation of factorials of Eq. (A10) involved in Eq. (A9), 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 A2. 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. (A9) and (A10).

Table A2. 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
colmat = spcol(knots, p+1, brk2knt(tau, nset));
imax = size(colmat,1);

Basis  = colmat(1:nset:imax, :);           % Bernstein polynomials B_i
dBasis = colmat(2:nset:imax, :);           % Bernstein derivatives B_i'
end
```

Appendix C. 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 Eq. (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
                |                                           |
                |                                           |
                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, Eq.(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);
```

 doi:10.20944/preprints202511.2141.v1


```
dL1x=dBasis(1); dL2x=dBasis(2); dL3x=dBasis(3); dL4x=dBasis(4);
%---ASSOCIATED BIVARIATE BASIS FUNCTIONS (4-7) [see, Eq.(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, Eq.(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
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 D. Main Program

Below is the main program for the solution of Example-1 (Figure 11) using the 12-node transfinite element shown in Figure 6a. 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 C).
`SHAPE_Peta_12_BSPLINES`	Basis functions using univariate B-splines, as blending and trial functions.
`lgwt`	Gauss points and associated weights (see Ref. [22]).

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

 doi:10.20944/preprints202511.2141.v1

`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 C).
`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
        |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
```

 doi:10.20944/preprints202511.2141.v1


```
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;
% 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;
```

 doi:10.20944/preprints202511.2141.v1


```
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)';
%% 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;
```

 doi:10.20944/preprints202511.2141.v1


```
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. Provatidis C. Transfinite elements using Bernstein polynomials. Axioms 2025;14(6):433. https://doi.org/10.3390/axioms14060433.
18. Farin G. Curves and Surfaces for CAGD. Morgan Kaufmann-Elsevier. USA, 2002. https://doi.org/10.1016/B978-1-55860-737-8.X5000-5.
19. 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.
20. 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.
21. DeBoor C, A Practical Guide to Splines, Revised ed., Springer: New York, 2001.
22. 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.
23. Carslaw HS, Jaeger JC. Conduction of Heat in Solids. 2nd ed. Oxford: Oxford University Press; 1969.

 doi:10.20944/preprints202511.2141.v1

24. 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
25. 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
26. 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
27. 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.
28. Courant R, Hilbert D. Methods of mathematical physics (1st English ed., Vol. I). New York: InterScience (translated and revised from the German original); 1966.
29. 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.
30. 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.
31. 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.
32. 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.
33. 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.
34. 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. https://doi.org/10.2478/ijmce-2025-0019.
35. 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.
36. 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.
37. 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.
38. 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
39. 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
40. 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
41. 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.
42. 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.

 doi:10.20944/preprints202511.2141.v1

43. 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.
44. Sangalli G, Takacs T, Vázquez R. Unstructured spline spaces for isogeometric analysis based on spline manifolds. Computer Aided Geometric Design 2016;47:61–82. https://doi.org/10.1016/j.cagd.2016.05.004.