


Understanding Bézier Extraction, Bézier Elements, and Hermite Elements in NURBS-Based Isogeometric Analysis

Christopher G. PROVATIDIS 

*Mechanical Design and Control Systems Division, School of Mechanical Engineering,
National Technical University of Athens, 15780 Zografos, Athens, Greece;
e-mail: cprovat@central.ntua.gr*

This paper discusses the current need for Bézier extraction in isogeometric analysis (IGA), and proposes a straightforward procedure to improve the accuracy of the numerical solution without increasing the number of B-spline elements. It shows that after knot insertion, where the shape and parameterization of the domain are preserved, the number of degrees of freedom (DOFs) leading to enhanced numerical accuracy of the numerical solution increases as well. Of particular interest is the fact that the control points implicitly introduced during Bézier extraction in IGA can be explicitly used to form Bézier elements with C^0 -continuity in several ways. Similarly, for any inner knot multiplicity less than the polynomial degree p , accuracy increases while maintaining the same number of B-spline elements. In conclusion, the set of the extracted Bézier or Hermite elements eventually leads to superior accuracy and performance compared to C^{p-1} -continuity. Nevertheless, if we consider a certain fixed number of DOFs for all three competing models (for $p = 3$), results show that the C^2 -continuous model is the most accurate, while the C^1 -continuous model (Hermite extraction) is more accurate than the C^0 -continuous model (Bézier extraction). The study includes six static and eigenvalue potential problems with known closed-form exact solution.

Keywords: Bézier extraction, isogeometric analysis, NURBS and finite elements, Helmholtz equation, error estimator.



Copyright © 2025 The Author(s).
Published by IPPT PAN. This work is licensed under the Creative Commons Attribution License
CC BY 4.0 (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The mechanical design of components and structures is greatly facilitated by computer simulation, which involves the numerical solution of partial differential equations (PDEs) governing the problem domain. This numerical solution may correspond to the so-called boundary-value problem (BVP) including transient

analysis (using initial conditions as well), or an eigenvalue (Sturm–Liouville) problem. One of the most commonly used computational methods is the finite element method (FEM) (Zienkiewicz [1], Bathe [2], among others).

As previously reviewed by Provatidis [3], the PhD thesis of Carl de Boor [4] (defended in 1966) promoted the use of spline approximation in computational methods. Furthermore, one of its most significant contribution was development of an efficient (recursive) algorithm for computing B-splines known as Cox–de Boor formula [5, 6]. Since then, there were many researchers who attempted to replace low-degree FEM using B-splines. A pioneering monograph by Böhmer [7], including computer programs written in ALGOL language and adopting the Cox–de Boor formulation, addressed the numerical solution of boundary-value and eigenvalue problems using Galerkin and collocation methods. Another remarkable contribution to the numerical solution of PDEs, focusing on the performance of B-spline finite elements, was made by Höllig in his book [8].

It is worth to mention that between the publication of the above mentioned works by mathematicians Böhmer (1974) and Höllig (2003), around year 1990, there were many attempts made by engineers to incorporate splines into the finite element analysis. Interestingly, a large number of them were published in the journal *Computers and Structures* (e.g., [9,10], among others). It is worth to mention that most of these papers (published in 1990s) were based on the very early definition of B-splines using cumbersome truncated powers documented in 1946 by Schoenberg [11]. However, in 1966, Curry and Schoenberg introduced the Curry–Schoenberg formulation [12], in which the inconvenient truncated powers of B-splines were successfully substituted by equivalent smooth, bell-shaped basis functions N_i 's associated with control points \mathbf{P} and generalized coefficients α_i 's, as we know them today [13,14]. In other words, although some mathematicians were already aware of the modern view of B-splines [7] (a perspective still used), most engineers at the time used the global approximation within the framework of the older B-spline formulation. Some of them implemented B-splines in conjunction with Coons patches (e.g., [15]) and other transfinite interpolations in single and multi-patch domains, using the same interpolation for both the geometry $x(\xi, \eta)$ and the physical quantity $u(\xi, \eta)$ [3,16, p. 8].

It is the IGA that has driven the engineers to implement the ‘modern’ view of B-splines and NURBS. Surprising, even today, many IGA-based papers begin with the definition of B-splines, a theory that has existed since 1966 (there is also an abstract paper by Curry–Schoenberg published as early as 1947 (see, [3, p. 90])), while it was Carl de Boor (along with Maurice Cox) who developed efficient algorithms for their numerical implementation. Since 2005, the standard FEM has been progressively replaced by IGA proposed by Hughes *et al.* [17]. In this approach, both the geometry $x(\xi, \eta)$ and the physical

variable $u(\xi, \eta)$ are represented using non uniform rational B-splines (NURBS), the standard global interpolation in computer-aided design (CAD) systems since 1990, and even earlier.

In more detail, all formulas known in standard FEM, including those referring to estimating of the stiffness and mass matrices, are also applicable to IGA. In brief, the whole structure (problem domain) is subdivided into many patches, with each patch approximated by a tensor-product NURBS. Therefore, bivariate and trivariate tensor-product NURBS are used for two- and three-dimensional problems, respectively [16].

Each direction of a patch is described by a knot vector Ξ , its associated polynomial degree p , and a set of control points P_i . Most of the control points may not belong to the actual surface, especially when the shape is curvilinear. The inner knots of the knot vector Ξ along with the end points (non-repeated) define the so-called breakpoints (breaks) which are also the extreme points of the associated NURBS elements. Within each element, numerical integration is performed using standard Gauss integration, though other dedicated integration schemes have also been studied (Hughes *et al.* [18], Auricchio *et al.* [19], Schillinger *et al.* [20]).

IGA-specialists know that isogeometric analysis can be implemented using a self-contained computer code, in which the only new issue (compared to standard FEM) is the computation of the basis functions and their derivatives, i.e., B-splines $N_i(\xi, \eta)$ or the closely related NURBS $R_i(\xi, \eta)$. Although N_i 's can be calculated using the previously mentioned de Boor iterative scheme (de Boor [6]), or even analytically, as shown in Sec. 2 for $p = 3$, there is a tendency to calculate them using the so-called Bézier extraction procedure instead. The reason is twofold. First, when properly programmed, the computation time may be somehow reduced (apart from the coefficients of the extraction operator, the basis functions are identical for all elements in the mesh as it is the case for classical finite elements; so there is no need to implement costly B-spline basis function evaluation routines). Second, data associated with Gauss points is previewed against potential errors and privacy assurance when third-party software handles the analysis module (i.e., computing the stiffness matrix and solving the equation system). More precisely, for each NURBS element 'e', the preprocessor generates Bézier extraction matrix \mathbf{C}_e , i.e., a square matrix of size $(p+1) \times (p+1)$ which is easily used by the software package (or module) responsible for further analysis. This approach is particularly important in the case of T-splines which were protected by a patent until recently (Scott *et al.* [21]).

This paper begins with the remark that Bézier extraction operator matrix \mathbf{C}_e , is theoretically derived through knot insertion, until the multiplicity of the inner knots matches equal to the polynomial degree, p . This knot insertion is related to a higher number of control points by preserving the shape

and the parameterization of the patch. Given the matrix \mathbf{C}_e for each NURBS element with control points \mathbf{P} , a set of new control points \mathbf{Q} as well as the new basis functions (e.g. Bernstein polynomials when the multiplicity equals p) and their derivatives can be easily calculated. Obviously, when updated basis functions are used, this procedure eventually leads to stiffness and mass matrices of a larger size. However, since the parameterization remains the same in both models, i.e., the initial NURBS elements and the new Bézier elements (both occupying the same area of the patch), the Jacobians can be calculated only once at the beginning, and thus computation time is reduced. Alternatively, the updated matrices associated with control points \mathbf{Q} can be calculated directly in an algebraic way by a quadratic form (i.e., without using the Gauss points), an approach will be explained later in this paper. Therefore, obtaining two or three different numerical solutions for the same elements, it becomes possible to determine those sub-patches with the highest relative error, and thus to establish an effective error estimator.

A critical question is whether it is better to apply the proposed methodology or to directly increase the breaks in the initial knot vector. This paper gives a definite answer by studying six typical steady-state and eigenvalue potential problems.

2. Basic theory

A B-spline is a piecewise polynomial of degree p defined over \tilde{n} breakpoints (also known as fixed or approximation points) $x_1, x_2, \dots, x_{\tilde{n}-1}, x_{\tilde{n}}$, referred to as ‘breaks’. The older definition of B-spline interpolation and approximation was introduced in 1946 by Schoenberg [11] using truncated polynomials. This expression was later replaced in 1966 by Curry and Schoenberg [12], who proposed using a set of bell-shaped basis functions $N_i(\xi)$, $i = 1, \dots, n$, called B-splines, and the associated coefficients α_i , $i = 1, \dots, n$. Note that in general we have $n > \tilde{n}$, as will be explained below. Therefore, in modern notation, a univariate function $u(\xi)$ is globally approximated as:

$$u(\xi) = \sum_{i=1}^n N_i(\xi) \alpha_i. \quad (1)$$

The numerical computation of the above basis functions $N_i(x)$ is facilitated by the algorithm introduced by de Boor [6]. The starting point of this algorithm is the knot vector:

$$\Xi = \left[\underbrace{x_1, \dots, x_1}_{(p+1)\text{-times}}, x_2, \dots, x_{\tilde{n}-1}, \underbrace{x_{\tilde{n}}, \dots, x_{\tilde{n}}}_{(p+1)\text{-times}} \right], \quad (2)$$

in which all the breaks $(x_1, x_2, \dots, x_{\tilde{n}-1}, x_{\tilde{n}})$ are included, but their ends are taken with multiplicity $\lambda = p + 1$. Therefore, the number of elements in the clamped knot vector Ξ is

$$m = \tilde{n} + 2p. \quad (3)$$

As proven elsewhere (Piegl and Tiller [13]), the number n of coefficients α_i is related to the number \tilde{n} of breaks by:

$$n = \tilde{n} + p - 1. \quad (4)$$

By eliminating the variable \tilde{n} between Eqs. (3) and (4), the relationship between the size of the knot vector Ξ and the number of coefficients α_i is:

$$m = n + p + 1. \quad (5)$$

According to de Boor [6], the basis functions are calculated recursively, starting with the piecewise constants (associated with $p = 0$, and thus the second subscript ‘0’):

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi \leq \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

For $p = 1, 2, 3, \dots$, and writing the knot vector from Eq. (2) as

$$\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$$

(where n is the number of control points, i.e., of the coefficients α_i), the B-spline functions are defined as:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \quad (7)$$

The implementation of Eq. (7) can be automatically performed in MATLAB[®] using the function `spcol`, while similar functions have been developed in open-source NURBS tools (for further information one may consult [3]). In addition, for $p = 3$, a closed-form analytical expression of the cubic basis function $N_{i,3}(\xi)$ is discussed next. For the general knot vector of Eq. (2), when $p = 3$, we delete the first two elements and the last two elements, and thus we construct the so-called **T**-mesh:

$$\mathbf{T}_{\text{mesh}} = \left[\underbrace{x_1, x_1}_{(2)\text{-times}}, x_2, \dots, x_{\tilde{n}-1}, \underbrace{x_{\tilde{n}}, x_{\tilde{n}}}_{(2)\text{-times}} \right]. \quad (8)$$

Based on the \mathbf{T} -mesh given by Eq. (8), we sweep from left to right, applying the well-known rule of thumb (two-left, one central, and two right), we construct the n following five-point sequences:

$$[x_1, x_1, x_1, x_1, x_2]_1, [x_1, x_1, x_1, x_2, x_3]_2, \dots, [x_{\tilde{n}-1}, x_{\tilde{n}}, x_{\tilde{n}}, x_{\tilde{n}}, x_{\tilde{n}}]_n. \quad (9)$$

Each of the local knot vectors involved in Eq. (9) produces one of the cubic B-spline functions $N_{i,3}$, where $i = 1, \dots, n$. By implementing Eq. (7) on the arbitrary set of five points $[x_1, x_2, x_3, x_4, x_5]$, we derive the following analytical expression:

$$N_{i,3}(x) = \begin{cases} \frac{(x-x_1)^3}{(x_2-x_1)(x_4-x_1)(x_3-x_1)}, & 0 < x \leq x_2, \\ \frac{(x-x_1)^2(x_3-x)}{(x_3-x_2)(x_4-x_1)(x_3-x_1)} \\ + \frac{(x_4-x)(x-x_1)(x-x_2)}{(x_3-x_2)(x_4-x_2)(x_4-x_1)} \\ + \frac{(x_5-x)(x-x_2)^2}{(x_3-x_2)(x_5-x_2)(x_4-x_2)}, & x_2 < x \leq x_3, \\ \frac{(x-x_1)(x_4-x)^2}{(x_4-x_3)(x_4-x_2)(x_4-x_1)} \\ + \frac{(x_5-x)(x_4-x)(x-x_2)}{(x_4-x_3)(x_5-x_2)(x_4-x_2)} \\ + \frac{(x_5-x)^2(x-x_3)}{(x_4-x_3)(x_5-x_3)(x_5-x_2)}, & x_3 < x \leq x_4, \\ \frac{(x_5-x)^3}{(x_5-x_4)(x_5-x_3)(x_5-x_2)}, & x_4 < x \leq x_5, \\ 0, & x \leq x_1 \quad \text{or} \quad x > x_5. \end{cases} \quad (10)$$

When dealing with curved boundaries, the above B-splines functions $N_i(\xi)$ are replaced by NURBS functions $R_i(\xi)$, which are simply obtained by using weights w_i associated with the control points P_i , as follows:

$$R_i(\xi) = \frac{w_i N_i(\xi)}{\sum_{j=1}^n w_j N_j(\xi)}. \quad (11)$$

As a result Eq. (1) is replaced by:

$$u(\xi) = \sum_{i=1}^n R_i(\xi)\alpha_i. \quad (12)$$

For the extension of Eq. (11) from 1D to 2D and 3D problems, the reader is referred to Cottrell *et al.* [16].

As was already mentioned in the Introduction section, in its original formulation (Cottrell *et al.* [16], Hughes *et al.* [17]) IGA was implemented in conjunction with the B-spline function (of C^{p-1} -continuity) given by Eq. (7). Typical open-source codes based on this approach in MATLAB[®] include those by Vuong *et al.* [22], De Falco *et al.* [23], Nguyen *et al.* [24]. For the sake of brevity, in the present paper, this approach will be labelled as ‘model 1a’.

Nevertheless, some researchers later claimed that the above-mentioned standard IGA (model 1a) requires substantial computation time to calculate the basis functions at the integration points and estimate the stiffness matrix (this issue is discussed at the end of Sec. 6). They also found it difficult to incorporate the NURBS basis functions in the workflow of an existing finite element code. To reduce computational effort and facilitate integration into several FEM codes, the Bézier extraction technique (model 1b) was proposed for NURBS (Borden *et al.* [25]) and at the same time for T-splines (Scott *et al.* [21]).

In brief, the main advantage of Bézier extraction (model 1b), compared to the original implementation of IGA (model 1a) [16, 17], is that, apart from the coefficients in the so-called extraction operator (matrix \mathbf{C}_e) of the ‘ e -th’ NURBS-element within a patch, the basis functions (i.e., Bernstein polynomials B_i , $i = 0, \dots, p$) are identical for all elements in the mesh, similar to the Lagrange polynomials in classical finite elements. Therefore, there is no need to implement B-spline function (N_i) evaluation routines which (as previously said) can be costly from a numerical perspective. But, the most important issue is probably that the owners of preprocessor software can protect their software from analysis partners.

Within the context of NURBS approximation of degree p (in conjunction with control points \mathbf{P}), after the computation of the extraction operator \mathbf{C}_e (for definitions, see Borden *et al.* [25]) the updated set of control points \mathbf{Q} (associated with Bernstein–Bézier polynomials) can be easily calculated by the linear formula $\mathbf{Q} = (\mathbf{C}_e)^t \mathbf{P}$, where the superscript ‘ t ’ stands for the transpose of the matrix \mathbf{C}_e . As a result, if the set of control points \mathbf{Q} is properly grouped, it can further define a set of Bézier elements of degree p with C^0 inter-element-continuity.

In this paper, it will be shown that the updated control points \mathbf{Q} , which are implicitly encountered in the above-mentioned Bézier extraction, can be

explicitly used to form Bézier elements of C^0 -continuity (called model 3) with superior accuracy. The model between model 1 and model 3, for piecewise cubic interpolation ($p = 3$) and C^1 -continuity, will be called model 2, and will be studied as well.

3. Bézier extraction, knot insertion, and updated basis functions

3.1. The state-of-the-art techniques in Bézier extraction and knot insertion

It is instructive to start with a one-dimensional problem. In this context, let us consider the initial knot vector Ξ_P :

$$\Xi_P = \{\xi_1, \xi_2, \dots, \xi_{m_P}\} \quad (13)$$

and let the n_P associated control points be

$$\mathbf{P} = \{P_1, P_2, \dots, P_{n_P}\} \quad (14)$$

with

$$n_P = m_P - (p + 1). \quad (15)$$

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

$$Q_{i+1} = (1 - a_{i+1})P_i + a_{i+1}P_{i+1}, \quad (16)$$

where

$$\alpha_{i+1} = \frac{\xi - \xi_{i+1}}{\xi_{i+1+p} - \xi_{i+1}}, \quad \text{for } k - p + 1 \leq i + 1 \leq k. \quad (17)$$

Equation (17) means that the control points (P_1, \dots, P_{k-p}) remain unchanged, and thus $Q_1 = P_1, \dots, Q_{k-p} = P_{k-p}$. Then, p new control points (i.e., Q_{k-p+1}, \dots, Q_k) are introduced according to Eq. (16). Finally, the control points $(P_{k+1}, \dots, P_{n_P})$ remain unchanged, so $(Q_{k+1} = P_{k+1}, \dots, Q_{n_P+1} = P_{n_P})$.

Applying Eq. (16) successively several times, the linear relationship between the new (\mathbf{Q}) and the old (\mathbf{P}) control points becomes:

$$\mathbf{Q} = \mathbf{T}^t \mathbf{P}, \quad (18)$$

where \mathbf{T} is the transformation matrix, produced by the successive implementation of Eq. (16).

Next, let the initial set of basis functions be:

$$\mathbf{N}_P = \{N_{P_1}, N_{P_2}, \dots, N_{P_{n_P}}\}, \quad (19)$$

and the new set of basis functions be:

$$\mathbf{N}_Q = \{N_{Q_1}, N_{Q_2}, \dots, N_{Q_{n_Q}}\}, \quad (20)$$

with $n_Q > n_P$.

Since the shape and the parametrization are the same (Piegl and Tiller [13]), we have:

$$\mathbf{x}(\xi, \eta) = \mathbf{N}_P^t \mathbf{P} = \mathbf{N}_Q^t \mathbf{Q}. \quad (21)$$

Substituting \mathbf{Q} from Eq. (18) into Eq. (21), we eventually obtain:

$$\mathbf{N}_P = \mathbf{T} \mathbf{N}_Q. \quad (22)$$

Equation (22) is of general nature and relates the initial (P) to the final (Q) set of basis functions after an arbitrary number of knot insertions. The initial set \mathbf{N}_P is characterized by C^{p-1} -continuity. For each knot insertion, the continuity at the knot under consideration reduces by one, and thus for knot multiplicity λ , the continuity becomes $C^{p-\lambda}$.

The so-called ‘Bézier extraction’ is a special case of the above general procedure, in which additional knots are successively inserted at all the inner knots of the initial knot vector Ξ_P (see Eq. (13)) until their multiplicity becomes equal to the polynomial degree p (i.e., $\lambda = p$). The choice $\lambda = p$ ensures the interelement continuity is C^0 (since $p - \lambda = 0$), which means that \mathbf{N}_Q becomes a set of Bernstein polynomials (Piegl and Tiller [13], Borden *et al.* [25]).

In summary, in ‘Bézier extraction’, the C^{p-1} -continuous B-spline functions \mathbf{N} of interest are calculated in terms of the C^0 -continuous Bernstein polynomials \mathbf{B} , according to the linear expression:

$$\mathbf{N} = \mathbf{C} \mathbf{B}, \quad (23)$$

where \mathbf{C} is the well-known Bézier extraction operator (Borden *et al.* [25]).

In 1D problems, with a knot vector Ξ consisting of m_P elements, following the structure of Eq. (2) or Eq. (13), the number n_P of control points is given by Eq. (15) while the number n_{ele} of B-spline elements is given by:

$$n_{\text{ele}} = n_P - p. \quad (24)$$

Therefore, the vector \mathbf{N} includes n_P basis functions elements N_i , while \mathbf{B} includes

$$(n_Q)_{C^0} = n_{\text{ele}} p + 1 \quad (25)$$

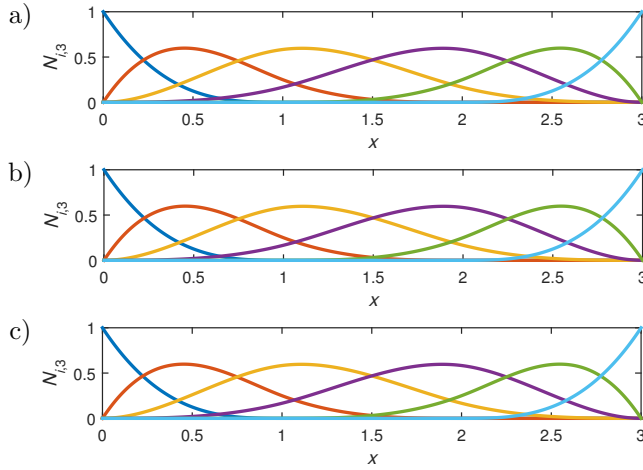


FIG. 2. Basis functions for: a) initial B-spline (C^2 -continuity), b) C^1 -continuity, and c) C^0 -continuity (after multiple knot insertions).

and

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{27}_2$$

Therefore, the total Bézier extraction operator is given as:

$$\mathbf{C} = \mathbf{T}_1 \cdot \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 7/12 & 2/3 & 1/3 & 1/6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/6 & 1/3 & 2/3 & 7/12 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{27}_3$$

The Bézier extraction matrix $\mathbf{C} = [c]_{ij}$, $i = 1, \dots, 6$, $j = 1, \dots, 10$ (of size 6×10) in Eq. (27)₃ works per element, as follows:

- The square submatrix of size 4×4 , between the extreme diagonal elements c_{11} and c_{44} , is applied to the B-spline element $e = 1$ (denoted by \mathbf{C}_1), having DOFs associated with the Bernstein polynomials (B_1, B_2, B_3, B_4) .

- The square submatrix of size 4×4 , between the extreme diagonal elements c_{24} and c_{57} , is applied to the B-spline element $e = 2$ (denoted by \mathbf{C}_2), having DOFs associated with the Bernstein polynomials (B_4, B_5, B_6, B_7).
- The square submatrix of size 4×4 , between the extreme diagonal elements c_{37} and $c_{6,10}$, is applied to the B-spline element $e = 3$ (denoted by \mathbf{C}_3), having DOFs associated with the Bernstein polynomials (B_7, B_8, B_9, B_{10}).

An alternative to the above-mentioned Bézier extraction, is to use only the transformation matrix \mathbf{T}_1 from Eq. (27)₁, and thus express the set of the six initial C^2 -continuous basis functions in terms of the eight C^1 -continuous basis functions shown in Fig. 2b. Inversely, the eight C^1 -continuous basis functions can be expressed in terms of the six C^2 -continuous basis functions through the element-wise inverse of the matrix \mathbf{T}_1 from Eq. (27)₁, in the sense of Eq. (22).

Below, we shall see that both transformation matrices, i.e., \mathbf{T}_1 and \mathbf{C} may be used to progressively increase the number of DOFs in the computational model, while at the same time preserving the shape and parametrization of object's shape.

3.2. Stiffness and mass matrix for multiplicity equal to p

In the original IGA, the element stiffness (\mathbf{K}_e) and mass (\mathbf{M}_e) matrices, each of size $n_P \times n_P$ (for potential problems), were calculated by the numerical integration of the B-spline functions N_i and their derivatives (at the Gauss points), using the recursive Eq. (7) in conjunction with the initial knot vector Ξ_P . Later, it was proposed to estimate the B-spline functions using Eq. (23), where the multiplicity was set equal to $\lambda = p$. Obviously, both procedures determine the same basis functions N_i , and thus the same element matrices ($\mathbf{K}_e, \mathbf{M}_e$) in magnitude and size ($p \times p$).

To show that the pair of matrices ($\mathbf{K}_e, \mathbf{M}_e$) corresponds to the initial state of C^{p-1} -continuity associated with the initial knot vector Ξ_P (given by Eq. (13)), we assign them an additional subscript, and thus denote them clearly as $[(\mathbf{K}_e)_P, (\mathbf{M}_e)_P]$. These element matrices eventually contribute to the assembly (stiffness and mass) matrices, each of size $n_P \times n_P$.

It should also be noted that, although the n_P basis functions of the usual IGA model (C^{p-1} -continuous) are calculated using a greater number of $n_Q = n_{\text{ele}} p + 1$ control points ($n_Q > n_P$) associated with the Bernstein polynomials (C^0 -continuous) in the usual Bézier extraction technique, in Subsec. 3.3 we will see that these DOFs can build a larger matrix of size $n_Q \times n_Q$ (for continuity C^0) or even smaller matrices (with a number of DOFs in-between: $n_P < n < n_Q$, for continuities C^{p-2}, \dots, C^1).

3.3. Stiffness and mass matrix for multiplicity less than or equal to p

When the multiplicity of the inner knots increases in any way (and thus the initial control points \mathbf{P} change to \mathbf{Q}), the updated knot vector Ξ_Q may be used to produce the new basis functions \mathbf{N}_Q , by implementing either of Eq. (7) or Eq. (23) at the previously used Gauss points. Although this procedure is possible and straightforward, an alternative algebraic procedure is given below.

For an arbitrary element ‘ e ’, Eq. (18) relates the initial control points to the final ones as follows:

$$\mathbf{P}_e = (\mathbf{T}_e^{-1})^t \mathbf{Q}_e. \quad (28)$$

Implementing the concept of IGA (i.e., using the same basis functions for the analysis as well), a similar relationship holds in both systems for DOFs (\mathbf{a}_P and \mathbf{a}_Q), as well, and thus the isoparametric analogue of Eq. (28) is:

$$\mathbf{a}_P = (\mathbf{T}_e^{-1})^t \mathbf{a}_Q. \quad (29)$$

Equation (29) depicts a linear relationship between the ‘local’ DOFs, \mathbf{a}_P , of the formulation (B-spline or NURBS with C^{p-1} -continuity) and the ‘global’ DOFs, \mathbf{a}_Q . The matrix \mathbf{T}_e depends on the applied multiplicity of inner knots as discussed below:

- If the multiplicity is $\lambda = 2$ (double inner knots), the continuity drops from C^{p-1} to C^{p-2} , and the transformation matrix \mathbf{T} has fewer columns than the Bézier extraction operator \mathbf{C} .
- If the multiplicity is $\lambda = 3$ (triple inner knots), the continuity drops from C^{p-1} to C^{p-3} . Therefore, in the special case of $p = 3$, by inserting inner knots twice ($\lambda = 3$) the set of basis functions \mathbf{N}_Q coincides with the Bernstein polynomials \mathbf{B} , which was discussed in Subsec. 3.2.

In both cases above, Eq. (29) relates the local DOFs (\mathbf{a}_P , C^{p-1} -continuity) with the global ones (\mathbf{a}_Q , $C^{p-\lambda}$ -continuity), through the transformation matrix $\mathbf{R} = (\mathbf{T}_e^{-1})^t$. As discussed elsewhere (e.g., Bathe [2]), this change of basis generally leads to the well-known quadratic form ($\mathbf{K}_{\text{global}} = \mathbf{R}^t \mathbf{K}_{\text{local}} \mathbf{R}$), and thus, in our case, the matrices can be written in terms of the local matrices as follows:

- For $\lambda = 2, \dots, p - 1$:

$$(\mathbf{K}_e)_Q = (\mathbf{T}_e^{-1})(\mathbf{K}_e)_P(\mathbf{T}_e^{-1})^t \quad \text{and} \quad (\mathbf{M}_e)_Q = (\mathbf{T}_e^{-1})(\mathbf{M}_e)_P(\mathbf{T}_e^{-1})^t. \quad (30)$$

- For $\lambda = p$:

$$(\mathbf{K}_e)_Q = (\mathbf{C}_e^{-1})(\mathbf{K}_e)_P(\mathbf{C}_e^{-1})^t \quad \text{and} \quad (\mathbf{M}_e)_Q = (\mathbf{C}_e^{-1})(\mathbf{M}_e)_P(\mathbf{C}_e^{-1})^t. \quad (31)$$

The points discussed above are the novel issue of this paper. In other words, having calculated the matrices $(\mathbf{K}_e)_P$ and $(\mathbf{M}_e)_P$ in the initial C^{p-1} -continuous

IGA model associated with n_P DOFs, either by implementing Eq. (7) or the Bézier extraction scheme, it is possible to employ Eq. (30) (using the transformation matrix \mathbf{T}_e from C^{p-1} to $C^{p-\lambda}$ with $2 \leq \lambda < p$) or Eq. (31) (using as transformation matrix the Bézier extraction operator \mathbf{C} with $\lambda = p$), and thus to analytically estimate the larger matrices $(\mathbf{K}_e)_Q$ and $(\mathbf{M}_e)_Q$ of the $C^{p-\lambda}$ -continuity, i.e., for $2 \leq \lambda \leq p$.

It is worth mentioning that, in general, the $C^{p-\lambda}$ model, applied to the initial set of n_{ele} elements, refers to:

$$(n_Q)_{C^{p-\lambda}} = n_P + (\lambda - 1)n_{\text{breaks,in}}, \quad 1 \leq \lambda \leq p. \quad (32)$$

DOFs, where $n_{\text{breaks,in}}$ is the number of inner knots in the initial knot vector Ξ_P given in Eq. (13). The special case of $p = 3$ is discussed below.

Cubic B-spline: In the case of $p = 3$ in conjunction with a knot vector including $n_{\text{breaks,in}}$ inner knots, the initial B-spline is of continuity C^2 ($\lambda = 1$) and includes n_P control points. After one knot insertion at all the inner knots, the continuity decreases to C^1 ($\lambda = 2$) and the number control points increase by $n_{\text{breaks,in}}$. After one more knot insertion at all the inner knots, the continuity becomes C^0 ($\lambda = 3$) and the final number of control points increases by $n_{\text{breaks,in}}$ more. In conclusion, the number of control points in each of the three 1D models is as follows:

$$(n_Q)_{p=3} = \begin{cases} n_P, & C^2\text{-continuity,} \\ n_P + n_{\text{breaks,in}}, & C^1\text{-continuity,} \\ n_P + 2n_{\text{breaks,in}}, & C^0\text{-continuity.} \end{cases} \quad (33)$$

4. IGA models

4.1. Model description

Let us consider that the knot vector includes $n_{\text{breaks,in}}$ inner knots. In this paper, three models will be tested as follows:

- Model 1: This refers to the original IGA associated with the initial knot vector Ξ_P , where the approximation within the domain (or patch) is C^{p-1} -continuous. The set of basis functions \mathbf{N}_P includes n_P elements, which are computed using either of Eq. (7) (model 1a) or Eq. (23) (model 1b).
- Model 2: This refers to knot insertion with multiplicity less than the polynomial degree p ($\lambda < p$). Therefore, the inter-element continuity is $C^{p-\lambda}$. For $p = 3$, this model deals with multiplicity $\lambda = 2$.
- Model 3: This refers to the assembly of Bézier elements coming from the extraction model associated with multiplicity $\lambda = p = 3$. Therefore, the inter-element continuity is C^0 .

4.2. Computational procedure

For a certain dimensionality (1D, 2D, or 3D) of the problem, it is possible to develop a single computer code per each dimensionality, which can handle all multiplicities, $\lambda = 1, \dots, p$, associated with a given knot vector Ξ_P and polynomial degree p . In other words, all the three models defined in Subsec. 4.1 can be executed with the same computer program, simply by changing the variable **multiplicity**, which corresponds to λ . A typical MATLAB[®] code, which computes and plots the basis functions $N_{i,p}(\xi)$ at uniform positions stored in the array **tau**, is given in Table 1.

TABLE 1. Typical MATLAB code for calculating and plotting the basis functions $N_{i,p}(\xi)$.

```

%% Calculate and plot the B-spline functions at 'npoints' points:
L = 3;                %domain length
p = 3;                %polynomial degree
k = p+1;              %polynomial order
multiplicity = 1;     %multiplicity (1,2,3)
nbrksIn = 2;          %number of uniform inner breakpoints
nele=nbrksIn+1;       %number of B-spline elements
nbrks = 2 + nbrksIn;  %total Nuber of BReakpoints
knots = augknt(linspace(0,L,nbrks),p+1,multiplicity);%knot sequence.
numknots = size(knots,2); %number of knots in the knot vector.
fprintf('Number of knots      (m) =%3i\n',numknots);
nctrlpoints = numknots - (p+1);
fprintf('Number of control points (n) =%3i\n',nctrlpoints);
% Basis functions at uniform points:
nseg=1000;            %number of segments
npoints=nseg+1;       %number of points to calculate the Ni,p's
tau = linspace(0,L,npoints);%points where Ni,p's are calculated
colmat=spcol(knots,k,brk2knt(tau,p)); %Ni,p till (p-1)-th derivative.
nrows=size(colmat,1); %number of rows in matrix 'colmat'.
Basis =colmat(1:p:nrows,:); %Matrix of B-splines at the 'npoints'.
Dbasis=colmat(2:p:nrows,:); %Matrix of B-splines derivatives.
plot(tau,Basis,'LineWidth',2)

```

Note that the above-mentioned computer code can be easily modified to calculate all the basis functions $N_{i,p}$ and their derivatives at the Gauss points, which will then be stored in the updated array **tau**. However, if we wish to save computer resources, we need to determine the element connectivity (or topology), which is usually specified by the well-known vector **IEN**. In one-dimensional problems, a set of n_{breaks} breakpoints (which include both the single ends and plus single inner knots), defines $n_{\text{ele}} = n_{\text{breaks}} - 1$ B-spline elements. In each of these n_{ele} elements there are $(p + 1)$ nonzero basis functions (compact support property). In more detail, taking $p = 3$ with C^2 -continuity, and running from left to right, within the first element the nonzero basis functions are (N_1, N_2, N_3, N_4) , within the second element are (N_2, N_3, N_4, N_5) , within the third are (N_3, N_4, N_5, N_6) , and so on.

To show what happens with different continuities, consider the knot vector $\Xi = [0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1]$ (i.e., for $n_{\text{ele}} = 3$, as shown in Fig. 1), for all the three continuities (C^2 , C^1 , and C^0) the IEN element topology vector is given in Table 2. Therein, one can observe that in each of the three elements under consideration, there are four (i.e., $p + 1$) nonzero basis functions, and this quantity is independent of continuity. For example, considering the second element, the C^2 -continuity demands the set (N_2, N_3, N_4 , and N_5) as previously mentioned, C^1 -continuity demands (N_3, N_4, N_5 , and N_6), while C^0 -continuity demands (N_4, N_5, N_6 , and N_7).

The estimation of the above-mentioned vector IEN can be easily programmed for 1D problems in MATLAB[®], as shown in the lower part of Table 2. For an assembly of n_{ele} B-spline elements, the input variable `multiplicity` stands for the multiplicity λ .

TABLE 2. Connectivity array IEN for several multiplicities ($p = 3$, $n_{\text{ele}} = 3$) in 1D problems.

Continuity	IEN(1)	IEN(2)	IEN(3)	IEN(4)
C^2	1	2	3	4
	2	3	4	5
	3	4	5	6
C^1	1	2	3	4
	3	4	5	6
	5	6	7	8
C^0	1	2	3	4
	4	5	6	7
	7	8	9	10

Typical MATLAB code for calculating and plotting the basis functions $N_{i,p}(\xi)$.

```

% Determination of IEN:
p=3;
nele=3;
if(multiplicity==1)      % C2-continuity
    for iel=1:nele
        IEN(iel,1:p+1)=iel:iel+p;
    end
elseif(multiplicity==2) % C1-continuity
    for iel=1:nele
        IEN(iel,1:p+1)= 2*(iel-1)+1:2*(iel-1)+1+p;
    end
elseif(multiplicity==3) % C0-continuity
    for iel=1:nele
        IEN(iel,1:p+1)=(iel-1)*p+1:((iel-1)*p+1)+p;
    end
else
    fprintf('*** MULTIPLICITY =%2i  GREATER THAN 3\n');
end

```


Therefore, using the element connectivity vector \mathbf{IEN} , for every Gauss point of the e -th element, we deal only with DOFs of the element stiffness matrix, i.e., those associated with $\mathbf{IEN}(\mathbf{e}, 1:4)$.

4.3. Alternative formulations of model 2 and model 3

4.3.1. C^0 -continuity. For both 1D- and 2D-problems addressed in this paper, four equivalent procedures have been developed to accomplish the proposed model 3, outlined below:

- *Procedure 3a* involves the MATLAB's function `spcol` in conjunction with inner knot multiplicity $\lambda = 3$ to determine the C^0 -continuous basis functions;
- *Procedure 3b* involves tensor product cubic Bézier elements, which are directly derived from the analytical expressions of Bernstein polynomials ($B_0 = (1 - \xi)^3$, $B_1 = 3(1 - \xi)^2\xi$, $B_2 = 3(1 - \xi)\xi^2$, $B_3 = \xi^3$);
- *Procedure 3c* is an algebraic scheme, which is based on a change of basis using Eqs. (30) and (31);
- *Procedure 3d* includes conventional tensor product cubic isoparametric Lagrange elements.

Note that the first three of them are coincident (i.e., the procedures 3a, 3b, and 3c lead to identical matrices); the last (procedure 3d) is equivalent (although, it differs in the matrices but leads to identical numerical solution). All four procedures constitute 'model 3'.

4.3.2. C^1 -continuity. In addition to the above two models (model 1 and model 3), the interim model 2 was applied in two equivalent formulations, as follows:

- Cubic B-spline with multiplicity $\lambda = 2$;
- Cubic Hermite elements with shape functions ($h_{00}(\xi) = (1 + 2\xi)(1 - \xi)^2$, $h_{10}(\xi) = \xi(1 - \xi)^2$, $h_{01}(\xi) = \xi^2(3 - 2\xi)$, $h_{11}(\xi) = \xi^2(\xi - 1)$), with DOFs: u , $\partial u/\partial x$, $\partial u/\partial y$, and $\partial^2 u/\partial x\partial y$.

5. Numerical results

In the static examples (i.e., the Laplace equation, $\nabla^2 u = 0$) within the domain Ω , the L_2 error norm (in %) was calculated according to the following formula:

$$L_2 = \left[\frac{\int (u_{\text{calculated}} - u_{\text{exact}})^2 d\Omega}{\int (u_{\text{exact}})^2 d\Omega} \right]^{1/2} \times 100, \quad (34)$$

where $u_{\text{calculated}}$ and u_{exact} are the calculated and exact values, respectively. Since the approximation is cubic, the integrand in the numerator of Eq. (34) is of degree 6, ensuring that accurate integration is achieved by a Gaussian scheme 4×4 within each B-spline (or NURBS) element.

Regarding eigenvalue problems with eigenvalues ω_m^2 , the error (in %) at the m -th mode was calculated according to the following formula:

$$E_m = \frac{\omega_{m,\text{calculated}}^2 - \omega_{m,\text{exact}}^2}{\omega_{\text{exact}}^2} \times 100. \quad (35)$$

5.1. One-dimensional problems

5.1.1. Problem 1. Consider the Laplace equation in spherical coordinates

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} = 0, \quad (36)$$

within a domain $[a, b]$ which represents an annulus of radial symmetry, with radii ($a = 1$, $b = 32$). The boundary conditions are of Dirichlet type: at the left (inner) end is ($u = 1000$ at $r = a$) while at the right (outer) end is ($u = 0$ at $r = b$). Find the L_2 -error (in %) implementing IGA for the polynomial degree $p = 3$ and several numbers of uniform B-spline elements, starting from $n_{\text{ele}} = 1$ till $n_{\text{ele}} = 16$, using all the three models (model 1: C^2 -continuity, model 2: C^1 -continuity, and model 3: C^0 -continuity).

Solution: The exact value for the temperature $u(r)$ is given by:

$$u_{\text{exact}}(r) = u(a) + \frac{u(b) - u(a)}{\ln\left(\frac{b}{a}\right)} \ln\left(\frac{r}{a}\right). \quad (37)$$

Figure 3 shows the obtained results of the error versus the number of DOFs. Note that the horizontal axis includes the number of DOFs (n_{DOFs}), and not the number of elements (n_{ele}). For example, for the extreme case of $n_{\text{ele}} = 16$, the number of DOFs becomes $n_{\text{DOFs}} = 19$ for C^2 -continuity, $n_{\text{DOFs}} = 34$ for C^1 -continuity, and $n_{\text{DOFs}} = 49$ for C^0 -continuity.

In contrast, for a fixed number of ($n_{\text{ele}} = 4$ to 16) cubic B-spline elements, the C^2 -continuous model 1 is less accurate than the C^1 -continuous model 2, and the latter is less accurate than the C^0 -continuous model 3, as shown in Table 3. One can also observe the increasing number of DOFs, as we move from model 1 to model 3.

Therefore, although the accuracy improves when the multiplicity λ of inner knots increases, for a certain given level of error L_2 , model 1 (C^2 -continuity) requires fewer DOFs (i.e., fewer B-spline elements) than what the other two models (i.e., model 2 (C^1 -continuity) and model 3 (C^0 -continuity)) need.

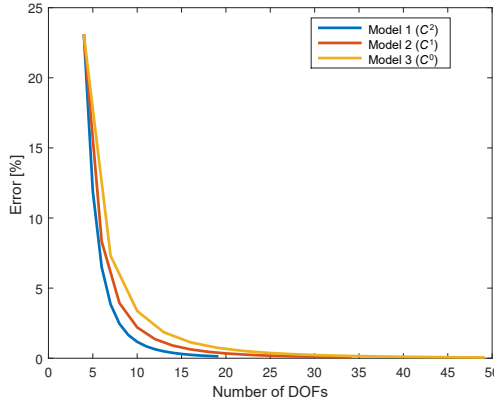


FIG. 3. Problem 1: Error (L_2 in %) of the calculated temperature.

TABLE 3. Errors (in %) for several numbers of elements and three multiplicities.

Number of elements (n_{ele})	Error					
	C^2 -continuity (model 1)		C^1 -continuity (model 2)		C^0 -continuity (model 3)	
	Error [%]	DOFs	Error [%]	DOFs	Error [%]	DOFs
4	3.85	7	2.20	10	1.85	13
6	1.66	9	0.91	14	0.75	19
8	0.85	11	0.46	18	0.37	25
10	0.49	13	0.26	22	0.21	31
16	0.14	19	0.07	34	0.06	49

5.1.2. Problem 2. Consider the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} = f(x), \tag{38}$$

where $f(x) = -\pi^2 \sin(\pi x)$, within the domain $[0, 1]$, under homogeneous Dirichlet boundary conditions ($u(0) = u(1) = 0$). Find the L_2 -error implementing IGA for the polynomial degree $p = 3$ and three uniform B-spline elements.

Solution: The exact value $u(x)$ is given by:

$$u(x) = \sin(\pi x). \tag{39}$$

Following the same procedure as in Problem 1, the obtained results are shown in Fig. 4, where one may observe the superiority of model 1.

In contrast, considering the number of elements as a constant, Table 4 shows that model 2 and model 3 provide progressively higher accuracy.

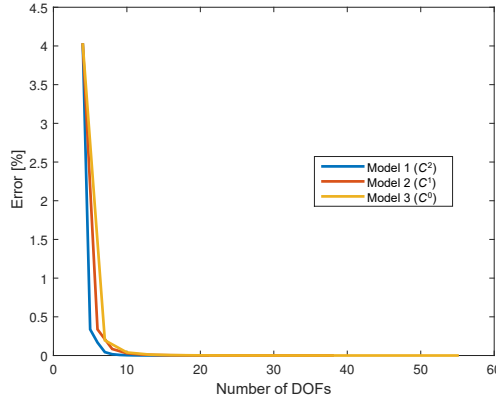


FIG. 4. Problem-2: Error (L_2 in %) of the calculated temperature.

TABLE 4. Problem 2: Errors (in %) for several numbers of elements and three multiplicities.

Number of elements (n_{ele})	Error					
	C^2 -continuity (model 1)		C^1 -continuity (model 2)		C^0 -continuity (model 3)	
	Error [%]	DOFs	Error [%]	DOFs	Error [%]	DOFs
1	4.0177214e+00	4	4.0177214e+00	4	4.0177214e+00	4
2	3.3781158e-01	5	3.3781158e-01	6	1.9630460e-01	7
3	1.6817612e-01	6	8.4363502e-02	8	3.9411722e-02	10
4	4.3986933e-02	7	2.9590614e-02	10	1.2541169e-02	13
5	1.6782693e-02	8	1.2777045e-02	12	5.1503829e-03	16
10	9.2572654e-04	13	8.6193427e-04	22	3.2302862e-04	31
15	1.7845774e-04	18	1.7282931e-04	32	6.3849528e-05	46
18	8.5543850e-05	21	8.3656972e-05	38	3.0796517e-05	55

5.1.3. Problem 3. Consider the wave equation

$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0, \quad (40)$$

in the interval $[0, L]$ with length $L = 3$ and wave velocity $c = 1$. For an elastic bar, we have $c = (E/\rho)^{1/2}$ (E the elastic modulus, ρ is the mass density). The boundary condition at the left end is of Dirichlet type ($u = 0$ at $x = 0$) while at the right end is of Neumann type ($\partial u/\partial x = 0$ at $x = L$). Find the eigenvalues implementing IGA for the polynomial degree $p = 3$ and an increasing number of uniform B-spline elements.

Solution: The exact eigenvalues $\lambda_i = \omega_i^2$ are given by:

$$\lambda_i = \frac{(2i-1)\pi^2 c^2}{4L^2}, \quad i = 1, 2, \dots \quad (41)$$

The case of $n_{\text{ele}} = 3$ (6 DOFs of which 2 at the ends) and $n_{\text{ele}} = 4$ (7 DOFs of which 2 at the ends) cubic B-spline elements, is shown in Figs. 5a and 5b, respectively. One may observe that, in each of these two cases and for any mode, model 1 (C^2 -continuous) is intensively less accurate than model 2 (C^1 -continuous) and it performs particularly worse than model 3 (C^0 -continuous).

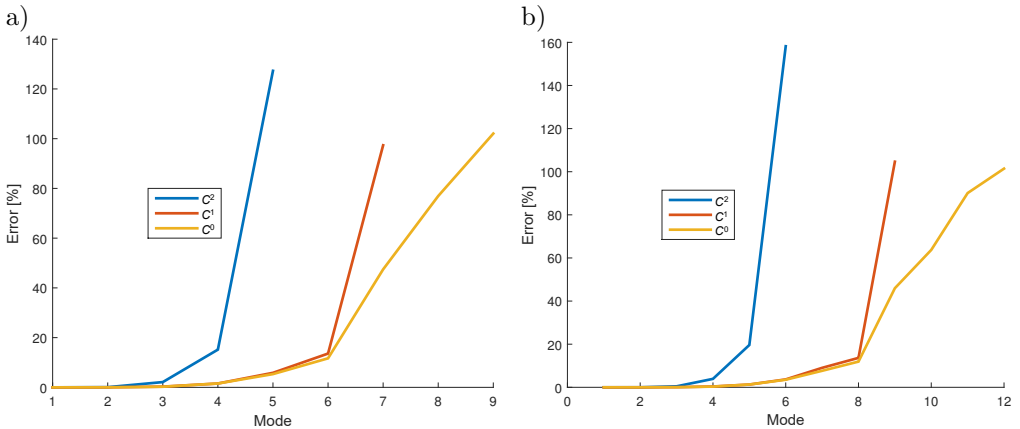


FIG. 5. Problem 3: Calculated eigenvalues of a semi-fixed bar for several modes and discretizations: a) $n_{\text{ele}} = 3$, b) $n_{\text{ele}} = 4$.

On the other hand, for a certain number of elements (n_{ele}), the number of involved DOFs increases when the continuity decreases, as clearly is depicted in Fig. 6. This observation is the reason why the comparison is reversed when considering the number of DOFs as the reference. Actually, if the error is plotted in terms of the DOFs involved in the eigenvalue extraction after imposing the boundary condition (i.e., deleting the first row and column), the comparison for the first and second eigenvalues is shown in Fig. 7.

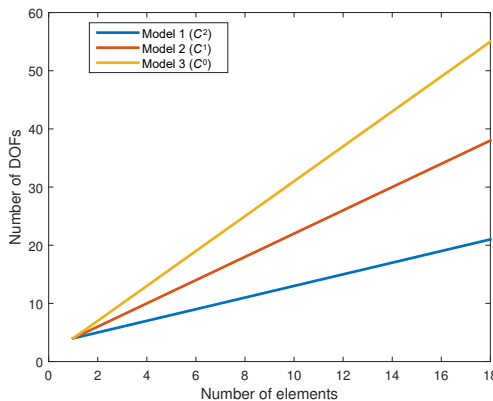


FIG. 6. Number of DOFs versus the number of B-spline elements for several knot continuities.

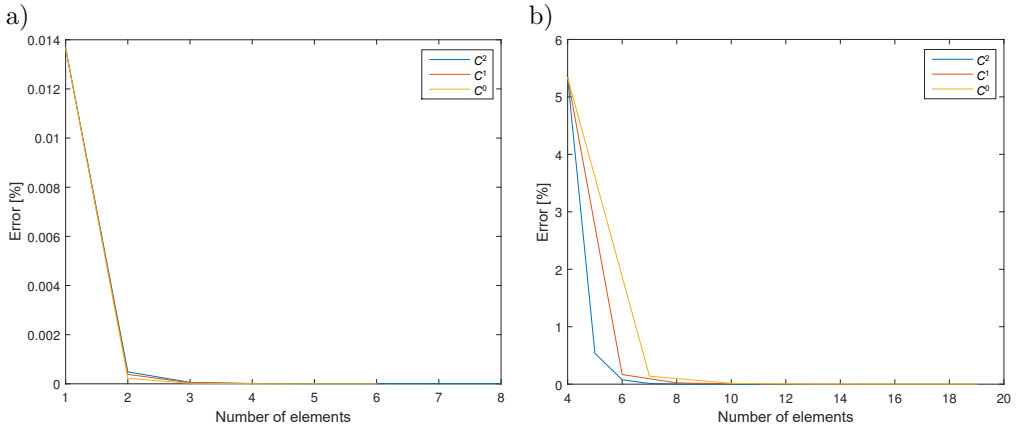


FIG. 7. Problem 3: Accuracy of calculated eigenvalues in terms of DOFs: a) mode 1, b) mode 2.

5.2. Two-dimensional problems

Two-dimensional problems are treated through the tensor product of the functional sets in the two directions.

5.2.1. Problem 4: Heat conduction in a rectangular plate. Consider a rectangular plate of dimensions $a \times b = 3 \times 12$ with uniform Dirichlet/Neumann boundary conditions across its thickness, as shown in Fig. 8.

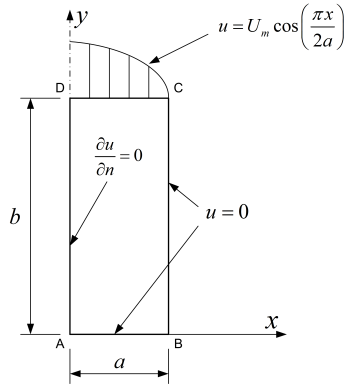


FIG. 8. Problem 4: Rectangular cavity under boundary conditions of Dirichlet and Neumann type.

The exact solution is given by:

$$u(x, y) = U_m \frac{\sinh\left(\frac{\pi y}{2a}\right)}{\sinh\left(\frac{\pi b}{2a}\right)} \cos\left(\frac{\pi x}{2a}\right), \quad (42)$$

where U_m is the maximum potential at the point $(x = 0, y = b)$.

Considering $p = 3$, we begin with $n_x \times n_y = 2 \times 5$ B-spline elements (40 control points and thus 40 DOFs, of which 22 are associated with the boundary), and we progressively increase the number of elements by one in each direction. The calculated error norm is shown in Fig. 9, where one can observe that for the same number of DOFs the error increases from C^2 - to C^0 -continuity.

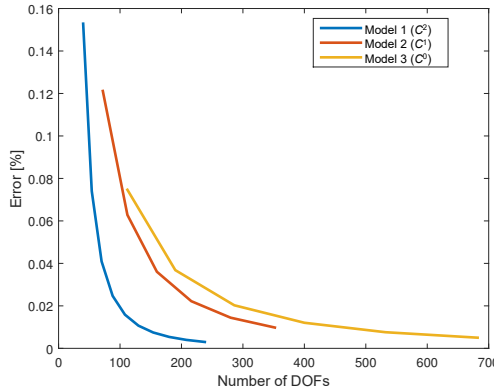


FIG. 9. Problem 4: Two-dimensional plate in heat conduction.

In contrast, for a given number of elements (n_{ele}) the number of DOFs increases from C^2 - to C^0 -continuity, while the error decreases. For example, for $n_{\text{ele}} = 2 \times 5 = 10$ B-spline elements, for model 1 the L_2 -error ($\lambda = 1$) is 0.1528%, for model 2 ($\lambda = 2$) is 0.1212%, whereas for model 3 ($\lambda = 3$) is 0.0747%. Taking as a reference the numerical solution of (the most accurate) model 3, the application of Eq. (34) with u_{exact} substituted by the numerical solution of model 3, leads to the estimated error 0.1123% (instead of the accurate absolute error 0.1528%). The distribution per element is shown in Fig. 10, where one can observe good agreement between the two references.

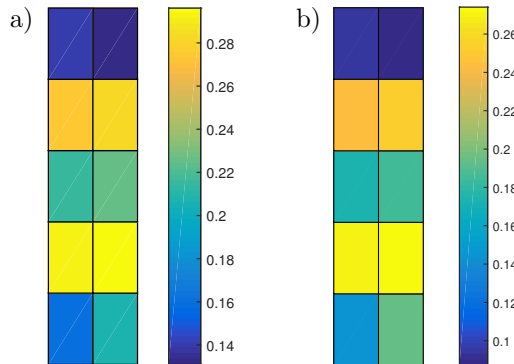


FIG. 10. Problem 4: L_2 -norm per element for 2×5 setup of cubic B-spline elements (model 1), with respect to: a) exact solution, and b) model 3 solution.

5.2.2. Problem 5: Acoustic cavity. Consider an acoustic cavity of size $a \times b = 2.5 \text{ m} \times 1.1 \text{ m}$, with normalized wave speed $c = 1 \text{ m/s}$, and rigid walls ($\partial u / \partial n = 0$). The governing wave equation is:

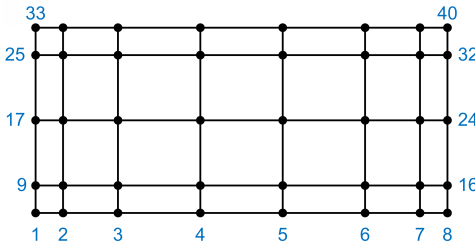
$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \nabla^2 u = 0. \quad (43)$$

We calculate the lowest 15 eigenvalues $\lambda_i = \omega_i^2$, $i = 1, 2, \dots, 15$. We recall that the exact eigenvalues are given by the formula:

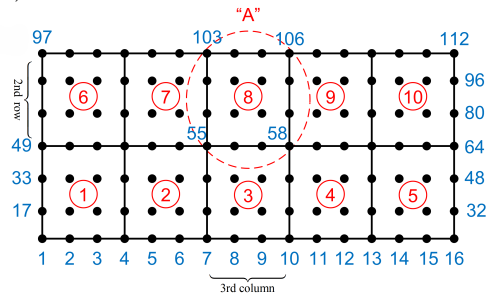
$$\omega_{mn}^2 = \lambda_{mn} = \pi^2 c^2 \left[\left(\frac{m}{a} \right)^2 + \left(\frac{n}{b} \right)^2 \right], \quad m, n = 0, 1, 2, \dots, \infty. \quad (44)$$

Solution: The initial isogeometric model (model 1 of C^2 -continuity) consists of 10 tensor-product cubic B-spline elements in a uniform 5×2 setup associated with the knot vectors $\Xi = [0, 0, 0, 0, 0.5, 1.0, 1.5, 2.0, 2.5, 2.5, 2.5, 2.5]$ and $\mathbf{H} = [0, 0, 0, 0, 0.55, 1.1, 1.1, 1.1, 1.1]$. The produced model 1 includes $8 \times 5 = 40$ control points (see, Fig. 11a). Therefore, each matrix for (\mathbf{K}, \mathbf{M}) comprises 40 rows and 40 columns, of which none is deleted when imposing the Neumann-type boundary condition (BC). The results of model 1 (15 lowest eigenvalues) are shown in Fig. 12 (blue line).

a)



b)



c)

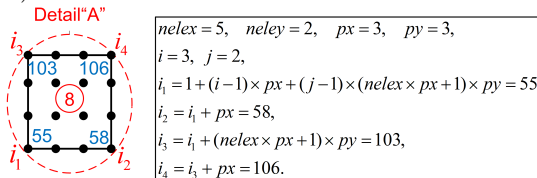


FIG. 11. Problem 5: Two-dimensional acoustic cavity of size $a \times b = 2.5 \times 1.1$ under BCs of Neumann type.

After executing Bézier decomposition, the totality of the new control points are rearranged, as shown in Fig. 11b, in which the total number increase from

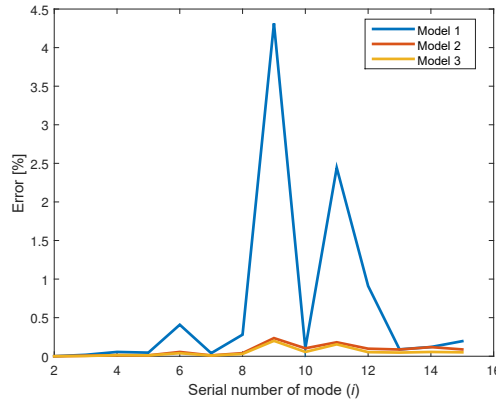


FIG. 12. Problem 5: Comparison between the three models.

40 to $16 \times 7 = 112$ (model 3 of C^0 -continuity). Now, each of the ten cubic Bézier elements consists of 16 control points arranged in a 4×4 setup. The extreme control points correspond to the true boundary of each element and they split each edge into three equal segments, similar to conventional rectangular finite elements.

For multiplicity $\lambda = 3$, the IEN matrix can be automatically computed in advance, based on the formulas illustrated in Fig. 11c, where the indices (i_1 , i_2 , i_3 , and i_4) denote the serial numbers of the control points at the corners of each NURBS element. In this convention, the Bézier elements are numbered sequentially starting from the lower left and increasing until the bottom edge is completed; then we continue from left to right along the row above the bottom layer, and so on (see the numbers inside the small red circles of Fig. 11b).

Although model 3 (based on C^0 -continuity) is superior to model 1 (C^2 -continuity), we have to admit that the former deals with 112 DOFs (of which 42 belong to the boundary) compared to 40 DOFs (of which 22 belong to the boundary) in the latter model.

Model 2, with double multiplicity $\lambda = 2$, leads to control points arranged in a setup of $12 \times 6 = 72$ points, which leads to 72 DOFs (of which 32 are associated with the boundary). Surprisingly, although this number of control points is close to the mean average between model 1 and model 3, the accuracy of model 2 is adequately high, as shown in Fig. 12.

5.2.3. Problem 6: Circular cavity. A circular acoustic cavity of a unit radius ($a = 1$) under Neumann boundary conditions is studied. The analytical solution is given by:

$$J'_m(ka) = 0, \quad m = 0, 1, 2, \dots, \quad (45)$$

where $J'_m(ka)$ is the first derivative of the Bessel function $J_m(ka)$ of the first kind and order m , and $k = \omega/c$ is the wavenumber. We aim to find the lowest seventeen eigenvalues using IGA.

Solution: The starting point is the well-known 9-point rational Bézier tensor product (a 3×3 setup, see Fig. 13a) for degree $p = 2$, the tensor-product knot vector $\Xi = \{0, 0, 0, 1, 1, 1\} \times \{0, 0, 0, 1, 1, 1\}$ and weights $w = \left\{1, \frac{1}{\sqrt{2}}, 1\right\} \times \left\{1, \frac{1}{\sqrt{2}}, 1\right\}$.

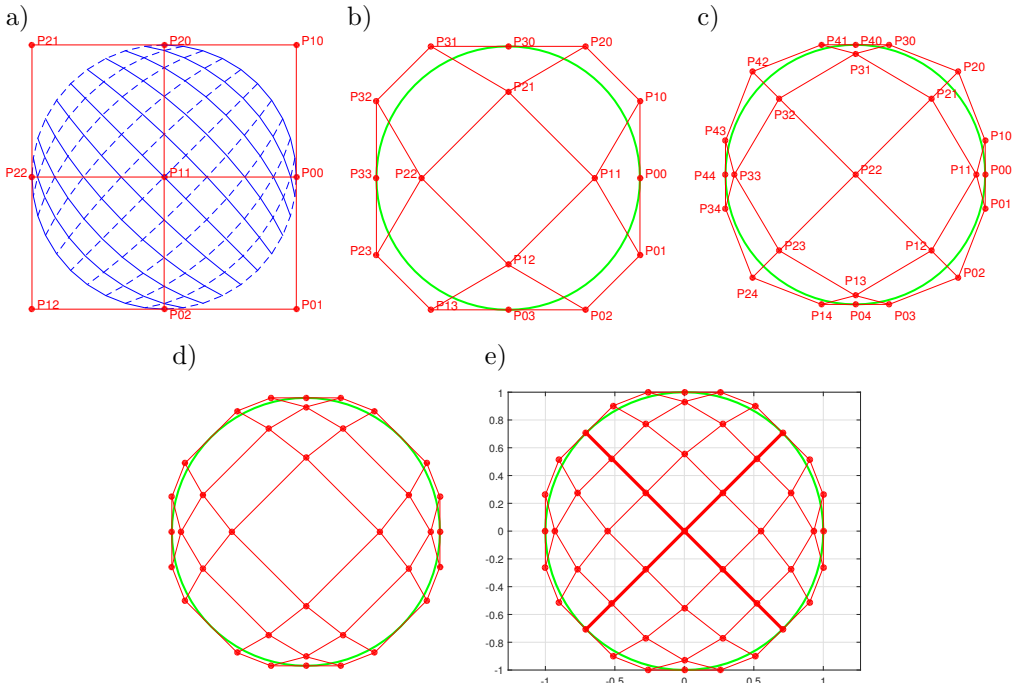


FIG. 13. Problem 6: B-spline elements (model 1) and Bézier elements (model 3): a) quadratic Bézier (9-points, $p = 2$), b) quadratic Bézier (16-points, $p = 3$), c) model 1: Cubic NURBS (25-points, $\lambda = 1$), d) model 2: Cubic NURBS (36-points, $\lambda = 2$), e) model 3: Cubic NURBS (49-points, $\lambda = 3$).

Then the degree is elevated to $p = 3$ and, as a result, the new set of the sixteen control points is arranged in a 4×4 setup (16 DOFs of which 12 are associated with the boundary), as shown in Fig. 13b, where the new knot vector per direction becomes $\Xi' = \{0, 0, 0, 0, 1, 1, 1, 1\} \times \{0, 0, 0, 0, 1, 1, 1, 1\}$.

Eventually, to produce a true NURBS patch that will accurately represent the circumference of the circle, an inner knot is inserted at the middle of each knot vector in either direction. Therefore, 25 control points are arranged in a 5×5 setup (25 DOFs of which 16 are associated with the bound-

ary), as shown in Fig. 13c, and therefore the final knot vector becomes $\Xi'' = \{0, 0, 0, 0, \frac{1}{2}, 1, 1, 1, 1\} \times \{0, 0, 0, 0, \frac{1}{2}, 1, 1, 1, 1\}$ while the weights are $\{1, 0.9024, 0.8047, 0.9024, 1\}$. The current 25-DOF model (model 1), in a 5×5 setup, as shown in Fig. 13c, is solved first in conjunction with standard NURBS-based IGA. This is accomplished once using the usual de Boor functions (`spcol` in MATLAB) and another using the equivalent Bézier extraction. Obviously, the results were found to be identical, and the associated errors are shown in the third column of Table 5.

TABLE 5. Problem 6: Eigenvalues of the circular cavity with hard walls.

Mode	Exact	Error [%]		
		Model 1 (25 DOFs)	Model 2 (36 DOFs)	Model 3 (49 DOFs)
1	0	–	–	–
2	3.389957716671888	0.43	0.07	0.07
3	3.389957716671888	0.43	0.07	0.07
4	9.328363213746355	0.34	0.34	0.32
5	9.328363213746355	3.75	0.47	0.47
6	14.681970642123899	1.13	1.13	0.96
7	17.649988519749648	11.85	2.20	1.99
8	17.649988519749648	11.85	2.20	1.99
9	28.276371248725660	13.68	3.45	3.25
10	28.276371248725660	98.64	3.45	3.25
11	28.424282047372301	97.60	3.01	3.01
12	28.424282047372301	157.67	13.09	8.97
13	41.160133480153071	140.69	12.39	12.39
14	41.160133480153071	159.07	20.19	15.31

Inserting a double knot at the middle $\xi = 1/2$ of each direction, we obtain a net of 36 control points (36 DOFs, of which 20 are associated with the boundary), as shown in Fig. 13d, which constitutes model 2. One may observe from the fourth column of Table 5, that model 2 is superior to model 1, of course having more DOFs (36 compared to 25).

Eventually, by considering all the 49 control points involved in the above-mentioned Bézier extraction, which is realized by increasing the multiplicity of the nine inner knots (in a 3×3 setup) to $\lambda = 3$, we obtain model 3 shown in Fig. 13e, with control points arranged in a 7×7 setup (49 DOFs of which 24 are associated with the boundary). This model is solved using four rational cubic Bézier elements (separated by the two perpendicular thick lines in red colour, as shown in Fig. 13e) and the relevant results are shown in the fifth column of Table 5.

Once again, one may observe that model 3 outperforms in accuracy by providing a constant setup of B-spline elements.

6. Discussion

The numerical results of the present paper show that, for a certain number of B-spline (or NURBS) elements (model 1) in the computational model, the introduction of a new set of control points produced after knot insertion, and the subsequent construction of the associated Hermite (model 2) or Bézier (model 3) elements leads to progressively higher accuracy of the numerical solution. As the knot multiplicity increases, both the number of DOFs increases, and the accuracy of the numerical solution improves. At first glance this seems reasonable, and is consistent with FEM experience; however, the relevant models (model 2 and model 3) are not simple enrichments of one other, but are closely related to the original IGA (model 1) as follows. Clearly, Eqs. (30) and (31) depict that by starting from the element stiffness (and mass) matrices of the original IGA (each of size 16×16 in potential problems, with C^2 -continuity when $p = 3$) and then performing a simple algebraic quadratic transformation on them (again transforming each element matrix to a new matrix of size 16×16), the corresponding matrices for the associated Hermite or Bézier elements are derived. Therefore, using the said quadratic form or performing the equivalent classical Gaussian integration with the updated set of basis functions within the same B-spline (or NURBS) elements (and maintaining the same Jacobians), one may easily derive the final matrix entries of the same element (of size 16×16) for Hermite or Bézier formulation. These, however, must be placed in a larger system matrix according to the connectivity vector IEN. Obviously, the higher accuracy in the associated Hermite or Bézier models facilitates the creation of an a posteriori error estimator, which can control the selective refinement of model 1.

Moreover, all six examples of the present paper show that for a certain number of DOFs, the original IGA (model 1 with C^2 -continuity) is more accurate than the formulation of Hermite elements (model 2 with C^1 -continuity) and the latter is more accurate than the Bézier element formulation (model 3 with C^0 -continuity). Therefore, model 2 and model 3 are not proposed as competitive alternatives to model 1 but rather as complementary methods for the above-mentioned a posterior error analysis.

In Subsec. 4.3.1 (with C^0 -continuity), we mentioned the application of four alternative procedures for implementing model 3. For example, regarding Problem 5, in all these four alternative procedures (3a÷3d) for model 3, the results were found to be identical (see Fig. 12, yellow line). One may observe that these methods outperform the model 1.

In Subsec. 4.3.2 (with C^1 -continuity), interestingly, while the stiffness and/or mass matrices in the two mentioned C^1 -continuous formulations appear to be different, the final numerical results were found to be identical. This happened in all the first five cases (from Problem 1 to Problem 5), where the problem was either 1D or the mesh was rectangular:

- Clearly, in 1D problems, each cubic B-spline element is replaced by a cubic Hermite element (with DOFs $u, \partial u/\partial x$, similar to a beam element). For example, in Fig. 2b the domain is spanned by eight B-spline functions associated with the four breaks at $\xi = 0, 1, 2, 3$ (and, obviously, eight control points). Equivalently, we can assign two DOFs at each of these four breaks, and thus we produce eight DOFs in total. These DOFs belong to three cubic Hermite elements, which successively span the subdomains: $[0, 1]$, $[1, 2]$, $[2, 3]$.
- In 2D problems, Hermitian elements of small size such as 4-node have been previously described in [26, p. 68], where they were criticized because – to retain C^1 -continuity – they should reduce to rectangles only. In this paper, for Problem 5 concerned with the rectangular acoustic cavity, we have six and three breaks in the x - and y -direction, respectively. For single knots, this corresponds to a mesh of $8 \times 5 = 40$ control points in total, while for double this results in $12 \times 6 = 72$ control points in total for IGA analysis. In contrast, considering the set of $5 \times 2 = 10$ uniform cubic Hermite finite elements, with $6 \times 3 = 18$ nodes and four DOFs per node, the total number of DOFs becomes again $18 \times 4 = 72$.
- It is worth mentioning that in the last case of a circular acoustic cavity (Problem 6), cubic Hermite finite elements were not applicable. In contrast, IGA was able to perform this kind of analysis, and the result (labelled as model 2) is shown in Table 5, as previously mentioned.

Before we comment on the performance of the models tested in the present paper (as discussed near the end of this section), it may be useful to mention the following useful computational issues. Any self-contained computer code implementing IGA, can be easily written in such a way that the desired multiplicity λ at inner knots is simply a variable in the crucial subroutine (or function) that calculates the basis functions N_i at the Gauss points (as shown in Table 1). A beginner can implement numerical integration for all the B-spline elements for each basis functions (B-splines), and then easily derive correct results for any multiplicity, simply by re-executing the computer code for a different λ . The only downside to this approach is that many useless numerical operations with zero terms will be performed (which is a waste of computing resources).

Regarding the required resources, numerous open-source B-spline and NURBS functions and libraries have emerged since 2000, the most reliable environment

is probably the function `spcol`, which have been available in MATLAB® since 1990. This function was re-written from older FORTRAN codes by Carl de Boor [14].

The next step regarding the improvement of the above computer code is to impose the so-called ‘local support’, which is very similar to the element topology (connectivity), well-known in standard finite element analysis, where banded matrices are built. Although any variable name could be attributed (e.g., `IX`, `elem`, etc.), to be consistent with some pioneering work, here we use the established name `IEN`. It is worthy to mention that, regardless of the multiplicity λ , for a given polynomial degree p there will be only $p + 1$ non-zero basis functions (per direction) within each B-spline element. This property referred to a ‘local support’, and `IEN` includes this element connectivity (as shown in Table 2).

From the above discussion it becomes evident that for a typical FEM-expert, no special knowledge is required to implement IGA, apart from a solid understanding of B-spline and NURBS (basis) functions, both based on knot vectors. This means that if the developers of the geometric model want to provide full information for further analysis (for example, to allow the independent approach of the analytical expression of cubic B-spline shown in Eq. (10)), they have to deliver the vector `IEN`, the coordinates and the weights of the control points, as well as the knot vector and the polynomial degree p , i.e., all the ‘secrets’ of their model. Of course, it would also be possible for the developers of the primary geometric model perform in-advance their calculations of the derivatives and the values of basis functions at the Gauss points and then provide them to their analysis partner. But, this would involve a large volume of data and could perhaps lead to a restriction in using their pre-calculated particular Gauss points.

The above difficulty in data transfer has been somehow overcome utilizing the Bézier extraction operator. This method provides compact information, transferred to any point of the patch, and thus not restricted to certain integration points. As previously said, the developers of the geometric model are not obligated to reveal all of their data to partners. In other words, as was clearly shown in the present paper, Bézier extraction is not absolutely necessary to perform IGA, but for these reasons it has become the standard in the commercial codes nowadays.

Having said this, in self-contained software, Bézier extraction can be further exploited to increase the accuracy of the IGA solution. In general, the numerical results of this paper show that for a certain fixed mesh of breakpoints (B-spline elements), as the multiplicity increases the accuracy improves. Unfortunately, when the multiplicity increases the number of DOFs increases as well, and thus we must solve a larger equations system. The critical issue is to determine what is better: to increase the multiplicity or to refine the element grid. Based on many

graphs presented in this paper, where the horizontal axis is the number of DOFs, as well as tables, it was found that for a given number of DOFs the C^2 -continuous B-spline is more accurate than the C^1 -continuous B-spline solution, and the latter is more accurate than the C^0 -continuous B-spline solution. Therefore, in principle, it is more effective to refine the mesh than simply increase the multiplicity for a certain standard number of B-spline elements.

On the other hand, for a certain number of B-spline elements, reducing the continuity by increasing the multiplicity up to the polynomial degree, is a useful procedure. This approach can act as an ‘exact’ solution, and thus serve as a simple error estimator that helps determine areas that need to be refined (see, Fig. 10).

A weak point of this article is that a CPU time comparison was intentionally not reported. In general, it was found that, in the six problems we studied, Bézier extraction (Eq. (23): model 1b) was substantially superior to the original IGA (Eq. (7): model 1a) only when the Bernstein polynomials and their derivatives were precomputed at the Gauss points; otherwise, the comparison is questionable.

On-going research indicates that the proposed technique is also applicable to T-splines (see [27]).

7. Conclusions

This paper tested the conjecture that increasing of the multiplicity at inner knots improves the accuracy of the IGA solution, for both eigenvalue and boundary-value problems. Actually, this happened for all the problems solved. One-knot insertion per inner knot reduced the inter-element continuity by one, leading to an improved numerical solution. Further knot insertion, one per knot, resulted in a further solution improvement. Nevertheless, for a certain number of DOFs, the accuracy of the IGA solution using single multiplicity was superior to that of double or triple multiplicities. Dealing with cubic B-splines, the first set of insertions was equivalent to cubic Hermite elements, while the second set of insertion was equivalent to Bézier elements or Lagrange elements (in the case of non-rational formulations) of the same degree. Additionally reducing continuity by increasing multiplicity, is a simple a posteriori error estimator, identifying the areas that need to be refined. In principle, the proposed methodology is also applicable to T-splines.

References

1. O.C. Zienkiewicz, *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.
2. K.J. Bathe, *Finite Element Procedures*, Prentice-Hall, New Jersey, 1996.

3. C.G. Provatidis, *Precursors of Isogeometric Analysis: Finite Elements, Boundary Elements and Collocation Methods*, Springer, Cham, 2019, <https://doi.org/10.1007/978-3-030-03889-2>.
4. C. De Boor, *The Method of Projections as Applied to the Numerical Solutions of Two Point Boundary Value Problems using Cubic Splines*, Ph.D. thesis, University of Michigan, 1966.
5. M.G. Cox, The numerical evaluation of B-splines, *International Journal of Mathematics and its Applications*, **10**(2): 134–149, 1972, <https://doi.org/10.1093/imamat/10.2.134>.
6. C. De Boor, On calculating with B-splines, *Journal of Approximation Theory*, **6**(1): 50–62, 1972, [https://doi.org/10.1016/0021-9045\(72\)90080-9](https://doi.org/10.1016/0021-9045(72)90080-9).
7. K. Böhmer, *Spline-Funktionen. Theorie und Anwendungen* [in German], Teubner Verlag, Stuttgart, 1974.
8. K. Höllig, *Finite Element Methods with B-splines*, SIAM, Philadelphia, 2003.
9. J. Kong, Y.K. Cheung, Application of the spline finite strip to the analysis of shear-deformable plates, *Computers & Structures*, **46**(6): 985–988, 1993, [https://doi.org/10.1016/0045-7949\(93\)90083-P](https://doi.org/10.1016/0045-7949(93)90083-P).
10. A.Y.T. Leung, F.T.K. Au, Spline finite elements for beam and plate, *Computers & Structures*, **31**(5): 717–729, 1990, [https://doi.org/10.1016/0045-7949\(90\)90100-G](https://doi.org/10.1016/0045-7949(90)90100-G).
11. I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. Part A. On the problem of smoothing or graduation. A first class of analytic approximation formulae, *Quarterly of Applied Mathematics*, **4**(1): 45–99, 1946.
12. H.B. Curry, I.J. Schoenberg, On Pólya frequency functions IV: The fundamental spline functions and their limits, *Journal d'Analyse Mathématique*, **17**(1): 71–107, 1966, <https://doi.org/10.1007/BF02788653>.
13. L. Piegl, W. Tiller, *The NURBS Book*, 2nd ed., Springer, Berlin, 1997.
14. C. De Boor, *A Practical Guide to Splines. Revised Edition*, Springer, New York, 2001.
15. C. Provatidis, A. Kanarachos, Performance of a macro-FEM approach using global interpolation (Coons') functions in axisymmetric potential problems, *Computers & Structures*, **79**(19): 1769–1779, 2001, [https://doi.org/10.1016/S0045-7949\(01\)00101-8](https://doi.org/10.1016/S0045-7949(01)00101-8).
16. J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Towards Integration of CAD and FEA*, Wiley, Chichester, 2009.
17. T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Computer Methods in Applied Mechanics and Engineering*, **194**(39–41): 4135–4195, 2005, <https://doi.org/10.1016/j.cma.2004.10.008>.
18. T.J.R. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering*, **199**(5–8): 301–313, 2010, <https://doi.org/10.1016/j.cma.2008.12.004>.
19. F. Auricchio, F. Calabrò, T.J.R. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering*, **249–252**: 15–27, 2012, <https://doi.org/10.1016/j.cma.2012.04.014>.

20. D. Schillinger, S.J. Hossain, T.J.R. Hughes, Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering*, **277**: 1–45, 2014, <https://doi.org/10.1016/j.cma.2014.04.008>.
21. M.A. Scott, M.J. Borden, C.V. Verhoosel, T.W. Sederberg, Isogeometric finite element data structures based on Bézier extraction of T-splines, *International Journal for Numerical Methods in Engineering*, **88**(2): 126–156, 2011, <https://doi.org/10.1002/nme.3167>.
22. A.V. Vuong, Ch. Heinrich, B. Simeon, ISOGAT: A 2D tutorial MATLAB code for isogeometric analysis, *Computer Aided Geometric Design*, **27**(8): 644–655, 2010, <https://doi.org/10.1016/j.cagd.2010.06.006>.
23. C. De Falco, A. Reali, R. Vázquez, GeoPDEs: A research tool for isogeometric analysis of PDEs, *Advances in Engineering Software*, **42**(12): 1020–1034, 2011, <https://doi.org/10.1016/j.advengsoft.2011.06.010>.
24. V.P. Nguyen, C. Anitescu, S.P.A. Bordas, T. Rabczuk, Isogeometric analysis: An overview and computer implementation aspects, *Mathematics and Computers in Simulation*, **117**: 89–116, 2015, <https://doi.org/10.1016/j.matcom.2015.05.008>.
25. M.J. Borden, M.A. Scott, J.A. Evans, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS, *International Journal for Numerical Methods in Engineering*, **87**(1–5): 15–47, 2011, <https://doi.org/10.1002/nme.2968>.
26. G.F. Carey, J.T. Oden, *Finite Elements: A Second Course*, Vol. 2., Prentice-Hall, Englewood Cliffs, NJ, 1983.
27. C. Provatidis, I. Dimitriou, Automatic handling of C^0 - G^0 continuous rational Bézier elements produced from T-splines through Bézier extraction, *Mathematics*, **13**(3): 377, 2025, <https://doi.org/10.3390/math13030377>.

*Received October 30, 2024; revised version March 13, 2025;
accepted March 13, 2025; published online April 3, 2025.*