J. Inst. Eng. India Ser. C (June 2019) 100(3):561–585
https://doi.org/10.1007/s40032-018-0462-6

ARTICLE OF PROFESSIONAL INTEREST

IGA: A Simplified Introduction and Implementation Details for Finite Element Users

Vishal Agrawal[1] · **Sachin S. Gautam**[1]

Received: 31 October 2017 / Accepted: 13 April 2018 / Published online: 14 May 2018

Abstract Isogeometric analysis (IGA) is a recently introduced technique that employs the Computer Aided Design (CAD) concept of Non-uniform Rational B-splines (NURBS) tool to bridge the substantial bottleneck between the CAD and finite element analysis (FEA) fields. The simplified transition of exact CAD models into the analysis alleviates the issues originating from geometrical discontinuities and thus, significantly reduces the design-to-analysis time in comparison to traditional FEA technique. Since its origination, the research in the field of IGA is accelerating and has been applied to various problems. However, the employment of CAD tools in the area of FEA invokes the need of adapting the existing implementation procedure for the framework of IGA. Also, the usage of IGA requires the in-depth knowledge of both the CAD and FEA fields. This can be overwhelming for a beginner in IGA. Hence, in this paper, a simplified introduction and implementation details for the incorporation of NURBS based IGA technique within the existing FEA code is presented. It is shown that with little modifications, the available standard code structure of FEA can be adapted for IGA. For the clear and concise explanation of these modifications, step-by-step implementation of a benchmark plate with a circular hole under the action of in-plane tension is included.

Keywords Linear elasticity · Finite element analysis · Isogeometric analysis · NURBS

✉ Sachin S. Gautam
ssg@iitg.ac.in

[1] Department of Mechanical Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India

List of symbols

$\bar{\phi}$	Total number of nodes in a finite element e
$\bar{\xi}, \bar{\eta}, \bar{\zeta}$	Coordinate points of the master space
$\mathbf{\Xi}, \mathbf{H}$	Knot vectors consisting the parametric coordinate points along the ξ and η directions
$\Omega, \tilde{\Omega}, \bar{\Omega}$	Domain of an element in the physical, parametric, and master spaces, respectively
$\mathbf{P}$	Array of control points
$\tilde{\phi}$	Mapping function transforming a NURBS element from the physical to the master space
ξ, η, ζ	Coordinate points of the parametric space
$B_{i,p}(\xi)$	An ith Bernstein basis function of order p defined at parametric point ξ
$L_{i,p}(\bar{\xi})$	An ith Lagrangian polynomial of order p defined at the integration point $\bar{\xi}$
n, m, l	Total numbers of basis functions along the ξ, η, and ζ parametric directions
n^e	Total number of nodes in a finite element e
n^e_{cp}	Total number of control points associated to an element e
$N_{i,p}(\xi)$	An ith B-spline basis function of order p defined at parametric point ξ
nel	Total number of elements
p, q, r	Order of one-dimensional basis functions along the ξ, η, and ζ directions
$R_{i,p}(\xi)$	An ith NURBS basis function of order p defined at the parametric point ξ
ϕ	Mapping function transforming an element from the physical to the master space

Introduction

State of the Art in IGA

Finite element analysis is considered to be one of the most popular numerical methods which are used to find the approximate solution of partial differential equations (PDEs). It has extensive applications in the fields of fluid mechanics, structural mechanics, contact mechanics and in many other scientific problems [1]. This technique utilizes the approximate form of the CAD geometries by discretizing them into a smaller sub-geometry called *elements* in FE community. Such geometrical approximations, which are inherent in the traditional FE, seriously affect the accuracy of the solution and lead to the various numerical errors during the analysis, e.g.,

- In the field of contact mechanics, FE based analysis of contact problems results in the faceted representation of the contact surface which leads to high jumps and artificial oscillations in the value of contact tractions [2].
- In the field of fluid mechanics, FE based representation of the geometry of aerodynamic bodies results in the non-physical entropy layers about the geometry shape which leads to the poor computation of the heat flux and pressure quantities [3].
- The thin shell problems are considered to be highly sensitive to the geometry approximations. It has been observed that the buckling load reduces by a considerable amount on increasing the magnitude of geometrical imperfections (see Fig. 1.3 in Ref. [4]).
- Higher order p-method based analysis of structural vibration problems causes significant errors in the frequency spectrum which diverge further on increasing the order of the FE polynomial [5].
- In the field of optimization, a tight coupling between the CAD-generated design model and the FE approximated analysis model is needed. However, FE based discretization of the design model, which differs much from the analysis model from the geometrical representation point of view, leads to the unrealistic optimal design model [6].

To minimize the geometrical discretization errors, various attempts have been made in the past. Kagan et al. [7] introduced a B-spline based FE approach that combines the geometrical design and the analysis systems into a common framework. Later, based on the subdivision of surface scheme Cirak et al. [8] presented a different paradigm for describing and modelling of the thin shells geometries in the FEA framework. Based on the motivation of combining the long available CAD systems with the simulation approaches, Hughes et al. [3] introduced a new numerical method, popularly known as Isogeometric analysis (IGA). In this technique, CAD-based NURBS described geometries are directly employed in the analysis framework without making any geometrical approximations like in FE. The fundamental concept of IGA is to utilize the NURBS basis functions not only for the construction or as a handler of the exact form of CAD geometries, but also as a tool that can be used for their mathematical analysis. This feature of IGA sets it apart from the traditional FEA technique. The coupling of CAD and analysis solvers into a unified framework reduces the burden of mesh regeneration and thus, minimizes the computational cost to a great extent (see Fig. 1.2 in Ref. [4]). Since the introduction of IGA technique [3], it has drawn the attention of many researchers and has been applied to a wide variety of problems, e.g.,

- Isogeometric based analysis of contact problems yields a smooth representation of the contact surfaces due to the inherent higher-order continuity of NURBS basis functions. With this, much more accurate results are obtained in comparison to the traditional FE approach [2, 9].
- Due to the inherent higher order continuity and exact representation of geometrical features in IGA, this technique has been shown to be advantageous in fluids [10] and fluid-structure interaction [11] problems.
- In comparison to the FE approach, IGA has proven to be extremely useful for the analysis of plate and shell problems [12].
- NURBS based IGA of structural vibration problems are shown to be advantageous than traditional FE higher order FE method. With this, the $k-$ refinement approach provides the robust and accurate frequency spectra in the wave propagation problems [5].
- In the case of optimization problems, a tight coupling between the design model and the analysis model is achieved as the design model is directly utilized in the analysis with IGA technique [6].

Besides the application of the IGA technology to the above-mentioned fields, this technique can also be applied to specific problems which have been solved with FE technology to attain the improved solution, e.g., see [13–15].

Apart from the inclusion of NURBS basis function for the modelling and analysis in IGA, other CAD tools such as T-spline [10], Polynomials splines over Hierarchical T-meshes (PHT-spline) [16], Locally Refined (LR) splines [17] and Hierarchical spline [18] have also been employed for the advancement of this technique. A technique based on Bézier extraction operator that transforms the isogeometric element in standard C^0-continuous element through their projection into Bézier element is devised in [19], which employs the IGA elements into finite element (FE) structure. However, the discussion in this paper is only limited to the NURBS basis functions for the sake of clarity. The reason for selecting this function as a basis is due to its widespread popularity in CAD field.

Current Status of IGA in Commercial Softwares

Due to the widespread application of IGA technology in different fields, few efforts have been made to integrate this technology (as a plug-in) with the commercial FEA software packages. Hartmann et al. [20] combined the NURBS-based isogeometric elements within the LS-DYNA package. In this, isogeometric NURBS elements are defined in the input data by declaring the necessary details for their generation. Later, Elguedj et al. [21] have provided a plug-in package 'AbqNURBS' that introduces the NURBS-based isogeometric element into a commercial software namely Abaqus. This package enables the integration of arbitrary isogeometric elements with the analysis framework of Abaqus using the user-defined material models. Recently, in Ref. [22], user elements within the software package Abaqus have been used for solving the one and two-dimensional higher-order gradient elasticity boundary value problems. However, in case of the usage of these commercial FEA packages for IGA of PDEs, the generation and visualization of NURBS defined geometries is neither user-friendly or direct nor fully-available. With these packages, other computing environments have to be additionally accessed for pre- and post-processing of geometries. For more details, the reader may refer to the above references.

Available Implementation Procedures of IGA and Their Shortcomings

For providing the implementation details of IGA technology in different computing environments various articles, as tutorial papers, have been presented. A monograph by Cottrell et al. [4] covers the in-detailed theory to thoroughly understand the transition procedure of NURBS described CAD geometries into the FEA technique and the basic concepts of this technology. However, the absence of isogeometric NURBS modelling toolbox with this monograph invokes the need to write the implementation codes from scratch, which might not be an easy-to-do task for a beginner in this technology. A tutorial Matlab® code, ISOGAT, for the NURBS based IGA is available in Ref. [23]. However, it is restricted to solving the two-dimensional elliptic diffusion-type problems. A procedure to integrate the IGA technology with an existing object-oriented finite element code architecture (written in C++ environment) is presented in Ref. [24]. Similarly, an object-oriented open-source C++ library based modules for the generation of geometry and simulation using the IGA of PDEs with a number of different CAD basis functions are developed in Ref. [25]. However, the incorporation of object-oriented code structure in [24, 25] makes the understanding of the computational procedure of primary variable less transparent to the user, which might complicate the debugging of the code. An excellent open source Matlab® implementation code for the IGA of one, two, and three-dimensional structural mechanics problems have been provided in [26]. The detailed application of B-spline based isogeometric analysis to one- and two-dimensional static structural and vibrational problems along with the implementation procedure is reported in [27]. A suite of free and open source research tool GeoPDEs, which serves as a starting tool for IGA users and is compatible with Octave, Matlab® programming languages, for the NURBS based isogeometric analysis of PDEs, have been included in Ref. [28]. However, this tool utilizes a different code-architecture than that used in the traditional FE codes, and the data is stored in the structured format. Hence, the tool might not be readily comprehensible by an FE user accustomed to traditional FE code architecture. Recently, a code framework PetIGA based on PETSc (Portable Extensible Toolkit for Scientific Computation), to achieve the high-performance with the IGA of PDEs is presented in [29]. However, for the utilization of any of these available implementation procedures (irrespective of their computational environment), it is assumed that the user should be well aware of the fundamentals of IGA and must be able to follow a large number of function libraries of available execution subroutines. Moreover, the usage of Bézier extraction operator technique, which combines the IGA technology with the traditional FEA framework, increases the computational cost substantially as an additional number of system equations has to be solved in comparison to traditional FEA approach.

Outline of Present Work

Based on the literature review presented above, it is found that not many papers are available which can quickly explain the IGA technology to a traditional FE user. Hence, the objective of this work is to present a simplified introduction and implementation procedure of IGA in the spirit of the traditional FE code architecture. In this presentation, it is shown that an existing code structure of FEA technique can be directly utilized for the implementation of IGA technology after making a few modifications. This eliminates the need to develop a new code architecture and eases the learning process of this technology. For the construction and handling of CAD geometries a free and open source NURBS modelling toolbox is used. The reasons behind choosing this toolbox are: (*a*) due to its wide accessibility as it can be used with the most common programming languages, e.g., Matlab® and Octave, and (*b*) it prevents the need for writing the NURBS modelling codes from scratch. The description of the presented implementation procedure is provided in a such a manner that an FE user can quickly know as to which subroutines or sections of traditional FE code structure has to be modified. To the best of the author's knowledge, no such implementation process has been covered in any literature. It might bear some similarity with the procedures reported in Refs. [26, 28]. However, as stated earlier, the code architecture that is used for the implementation of IGA technology with these procedures might not be an easily accomplished task for a traditional FE user, and a vast NURBS modelling functions libraries have to be handled. A simplified introduction, as well as the implementation of IGA technology in the existing framework of FEA technique, makes the present discussion distinct from the available approaches.

The paper is organized as follows: first, in "Preliminaries" section, a brief introduction to the FE and the IGA basis functions is provided. Then, in "Weak Formulation" section, the weak formulation of the linear elastic problem and the corresponding IGA based discretization is briefly discussed. Next, in "Structural difference between FEA and IGA codes" section, the flowchart of FEA and its modified form for IGA is presented in "Structural difference between FEA and IGA codes" section. In "A detailed stepwise implementation procedure for IGA" section, the stepwise implementation procedure of IGA based on the modification of FEA code structure is illustrated. Finally, "Conclusion" section, summerizes the paper. To outline the key points of the proposed implementation procedure a brief summary of each section is also included at their end.

Preliminaries

In FEA community, Lagrangian basis functions are particularly popular and are defined either in the physical space (space of x, y and z coordinates) or in the master space (space of ξ, η and ζ coordinates) [1, 30]. In IGA, Bernstein, B-spline, and NURBS basis functions are commonly employed for the modelling and discretization of CAD geometries and are defined in the parametric space directly [3, 4]. A short description of these functions is presented next.

Lagrangian Basis Function

A pth order one-dimensional Lagrangian function, $L_{i,p}(\xi)$, which passes through $p+1$ number of points is given by the following expression [1]:

$$L_{i,p}(\xi) = \prod_{k=1,k\neq i}^{p+1} \frac{\xi - \xi_k}{\xi_i - \xi_k}, \quad 1 \le k \le p+1 \quad \text{and} \quad -1 \le \xi \le 1 \tag{1}$$

These functions posses the non-negativity: $-1 \le L_{i,p}(\xi) \le 1$, Kronecker's delta: $L_{i,p}(\xi_j) = \delta_{ij}$, partition of unity: $\sum_{i=1}^{n} L_i = 1$, and linear independence: $\sum_{i=1}^{n} \alpha_i L_i = 0$ if $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$ properties. The major issues that are associated with the use of Lagrangian polynomial is due to its inability to satisfy the variation diminishing property and its inherited $-1 \le L_{i,p}(\xi) \le 1$ nature of the basis function. Due to these issues, the result obtained becomes highly sensitive to the order of the Lagrangian basis functions. This leads to oscillatory results or provides the non-smooth representation of the resulting fitting curves. This problem becomes even more prominent in case of higher order (i.e., $p \ge 3$) Lagrangian basis functions [3]. Additionally, these basis functions are restricted to only C^0-continuity across the inter-element boundary, which results in the non-smooth representation of the different derivative quantities, e.g., strains and stresses.

Bernstein Basis Functions

A pth order univariate (single variable) Bernstein polynomial, $B_{i,p}(\xi)(1 \le i \le p+1)$ is defined by the following recursive relation [19]:

$$B_{i,p}(\xi) = \frac{1}{2}(1-\xi)B_{i,p-1}(\xi) + \frac{1}{2}(1+\xi)B_{i-1,p-1}(\xi), \quad \xi \in [-1,1] \tag{2}$$

where $B_{1,0}(\xi) = 1$, and $B_{i,p}(\xi) \equiv 0$ if $i < 1$ or $i > p+1$. These basis functions posses following properties: partition of unity $\sum_{i=1}^{i=p+1} B_{i,p}(\xi) = 1$, Kronecker's delta property $B_{i,p}(\xi_j) = \delta_{ij}$, and are restricted to the C^0 inter-element continuity. However, unlike the Lagrangian polynomials these functions are always non-negative, i.e. $B_{i,p}(\xi) \ge 0 \forall \xi$. Moreover, these functions satisfy the variation diminishing property which leads to the monotonic improvement in the stability of results on increasing the order of the Bernstein basis functions [19].

Multi-variate Bernstein basis functions are defined by the tensor product of univariate functions. Let $B_{j,q}(\eta)$ and $B_{k,r}(\zeta)$ are the qth and rth order univariate Bernstein basis functions which are defined in the η and ζ directions, respectively. Then, bi-variate and tri-variate Bernstein basis functions are defined as [19]

$$B_{i,j}^{p,q}(\xi,\eta) = B_{i,p}(\xi)B_{j,q}(\eta), \tag{3}$$

$$B_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = B_{i,p}(\xi)B_{j,q}(\eta)B_{k,r}(\zeta) \tag{4}$$

Bézier described multi-dimensional geometries are constructed by the linear mapping of Bernstein basis functions from parametric to the physical space. A pth order Bézier curve is defined as:

$$\mathbf{C}(\bar{\xi}) = \sum_{i=1}^{p+1} B_{i,p}(\bar{\xi}) \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}^{\mathrm{T}} = \sum_{i=1}^{p+1} B_{i,p}(\bar{\xi}) \mathbf{P}_i \tag{5}$$

where $\mathbf{P}_i$ is a control point vector. Similarly, Bézier surfaces and solids are given through the tensor product of univariate basis functions and control points as [19]:

$$\mathbf{S}(\bar{\xi}, \bar{\eta}) = \sum_{i=1}^{p+1} \sum_{j=1}^{q+1} B_{i,j}^{p,q}(\bar{\xi}, \bar{\eta}) \mathbf{P}_{i,j} \tag{6}$$

$$\mathbf{V}(\bar{\xi}, \bar{\eta}, \bar{\zeta}) = \sum_{i=1}^{p+1} \sum_{j=1}^{q+1} \sum_{k=1}^{r+1} B_{i,j,k}^{p,q,r}(\bar{\xi}, \bar{\eta}, \bar{\zeta}) \mathbf{P}_{i,j,k} \tag{7}$$

where $\mathbf{P}_{i,j} = \mathbf{P}_i\mathbf{P}_j$ and $\mathbf{P}_{i,j,k} = \mathbf{P}_i\mathbf{P}_j\mathbf{P}_k$ are the control point net and control point volume arrays, respectively.

B-splines

B-splines are defined over a knot vector Ξ and are the linear combination of piecewise smooth basis functions. A knot vector Ξ is considered as the increasing arrangement of parametric space coordinates as [31]

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\} \tag{8}$$

where each knot entry,ξ_i, is a real number (i.e., $\xi_i \in \mathbb{R}$). The symbols p and n denotes the order and the total number of B-spline basis functions, respectively. Based on the difference between any two consecutive knots, a knot vector can be categorized either as a *uniform* or as a *non-uniform* vector. If the difference between any two consecutive knots remains fixed, it is said to be a uniform vector, otherwise a non-uniform vector. In CAD parlance a knot vector is also termed as a *patch*. If the first and the last knot entries of a knot vector are repeated by $p+1$ times, it is said to be an *open knot vector*, which is standard terminology in IGA. B-spline basis functions of order p are defined by the Cox-de Boor recursive relations as [31]:

$$\begin{aligned} &\text{for} \quad p = 0, \quad N_{i,0}(\xi) = \begin{cases} 1, & \text{if } \xi_i \le \xi < \xi_{i+1}, \\ 0, & \text{Otherwise.} \end{cases} \\ &\text{for} \quad p > 0, \quad N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1} + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1} \end{aligned} \tag{9}$$

As is with the Bernstein basis functions, the B-spline basis functions also exhibit: non-negativity $N_{i,p} \ge 0 \forall \xi$, partition of unity $\sum_{i=1}^{p+1} N_{i,p}(\xi) = 1$, linear independence, Kronecker's delta, and the variational diminishing properties. However, unlike the Bernstein basis, these functions are not restricted to the C^0 inter-element continuity as a pth order B-spline function $N_{i,p}(\xi)$ delivers the C^{p-1} continuity at the knot point ξ, if it is not repeated in the knot vector Ξ. The continuity can also be decreased to C^{p-1-k} if a knot is repeated by k times in the knot vector. The B-spline basis functions show the C^0-continuity at the boundary of a patch in an open knot vector as the end knot points (ξ_1 and ξ_{n+p+1}) are repeated by $p+1$ times. However, apart from the versatility of these bases over Lagrangian and Bernstein functions, one of the major limitation associated with the use of B-splines is its inability to accurately represent the circle, sphere and cylinder shaped geometries. On the other hand, NURBS, which are the rationale of these functions, overcome this limitation inherently and are described in the next subsection.

Multivariate B-spline basis functions are defined by the tensor product of the univariate functions. For example, the bivariate B-spline basis functions are defined as [31]:

$$N_{i,j}^{p,q}(\xi, \eta) = N_{i,p}(\xi) M_{j,q}(\eta) \tag{10}$$

and tri-variate functions as:

$$N_{i,j,k}^{p,q,r}(\xi, \eta, \zeta) = N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \tag{11}$$

where $M_{j,q}$ and $L_{k,r}$ are the qth and rth order B-spline basis functions that are defined in the η and ζ parametric directions, respectively. A pth order B-spline curve is defined as [31]

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) \mathbf{P_i} \tag{12}$$

For a given control point net $\mathbf{P}_{i,j}$ and control point volume array $\mathbf{P}_{i,j,k}$, B-spline surfaces and solids are defined through the tensor product of univariate B-spline basis functions in the following manner [31]

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,j}^{p,q}(\xi, \eta) \mathbf{P}_{i,j} \tag{13}$$

$$\mathbf{V}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_{i,j,k}^{p,q,r}(\xi, \eta, \zeta) \mathbf{P}_{i,j,k} \tag{14}$$

where m and l represent the total number of basis functions defined along the η and ζ parametric directions, respectively.

NURBS

NURBS are frequently employed in CAD and Computer Aided Manufacturing (CAM) industries as they offer great flexibility and accuracy in generation and representation of CAD geometries [3]. NURBS are considered as the generalization of B-splines. A univariate NURBS basis is defined by the rationale of weighted B-spline basis functions as [31]:

$$R_{i,p}(\xi) = \frac{w_i N_{i,p}(\xi)}{W(\xi)} \quad 1 \le i \le p+1 \tag{15}$$

where $w_i \ge 0$ denotes the weight value associated with control point vector $\mathbf{P}_i$, and weighted function $W(\xi) = \sum_{i=1}^{n_{cp}} w_i N_{i,p}(\xi)$ is the weighted linear combination of B-spline basis functions. Here, n_{cp} denotes the total number of NURBS control points. Since, NURBS are the rationale of B-spline basis function, they share the same properties of B-spline basis functions. Although, unlike the B-spline, NURBS are more flexible in nature and can represent the exact form of conic sections, e.g. circles, due to their association with the weight points.

Bi-variate and tri-variate NURBS basis are defined through the tensor product of univariate NURBS functions. Let $\mathbf{P}_{i,j}$, and $w_{i,j}$ are the known control point net and weight point arrays over the domain $\Omega(\xi,\eta)$, respectively where $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$. The knot vectors along the ξ, and η directions are $\boldsymbol{\Xi} = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ and $\mathbf{H} = \{\eta_1, \eta_2, \ldots, \eta_{m+q+1}\}$, respectively. Here, p and q represent the order of univariate basis functions while n and m denote the total number of control points along each direction, respectively. The bivariate NURBS functions are given by [31]

$$R_{i,j}^{p,q}(\xi,\eta) = R_{i,p}(\xi) R_{j,q}(\eta) = \frac{N_{i,p}(\xi) M_{j,q}(\eta) w_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) w_{i,j}} \tag{16}$$

Similarly, tri-variate NURBS functions are defined analogously and are given as [31]

$$\begin{aligned} R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) &= R_{i,p}(\xi) R_{j,q}(\eta) R_{k,r}(\zeta) \\ &= \frac{N_{i,p}(\xi) M_{j,q} L_{k,r}(\zeta) w_{i,j,k}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) w_{i,j,k}} \end{aligned} \tag{17}$$

It should be noted that these multivariate basis functions inherit all the properties of their ingredient univariate originators. The multi-dimensional NURBS defined geometries are constructed in the same fashion to the B-spline based geometry formation procedure. A NURBS curve is defined by the linear combination of univariate NURBS basis function $R_{i,p}(\xi)$ and P_i control points by the following expression:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n_{cp}} R_{i,p}(\xi) \mathbf{P}_i \tag{18}$$

NURBS surfaces and solids are constructed through the tensor product of the univariate basis functions as [31]

$$\text{NURBS surface:} \quad \mathbf{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\xi,\eta) \mathbf{P}_{i,j} \tag{19}$$

$$\text{NURBS solids:} \quad \mathbf{V}(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) \mathbf{P}_{i,j,k} \tag{20}$$

Summary

The properties of the above-mentioned basis functions are summarized in Table 1 to showcase their properties in a concise manner. Additionally, from the comparative point of view, a plot for the second order ($p = 2$) Lagrangian, Bernstein, B-spline, NURBS basis functions over the knot span $\xi \in [-1,1]$ is shown in Fig. 1 to visualize the arrangement and variation of these functions. It can be observed that all the basis functions except the Lagrangian are non-negative. B-spline basis functions seem identical to the Bernstein functions although they offer the tailorable C^{p-1-k} continuity across the boundary. Moreover, due to the grouping of NURBS with the additional weight points, they appear to be more flexible in comparison to B-spline. This explains the reason for choosing the NURBS as the tool for forming the basis in IGA.

Weak Formulation

Consider the domain Ω bounded by prescribed displacement, Γ_u, and tractions boundaries, Γ_t, such that the domain boundary $\Gamma = \Gamma_u \cup \Gamma_t$ and $\Gamma_u \cap \Gamma_t = \phi$. The governing equilibrium equation is given by the following mathematical expression [1, 4]:

$$\sigma_{ij,j} + f_i = 0 \quad \text{in } \Omega, \tag{21}$$

$$\mathbf{u} = \mathbf{u}_0 \quad \text{on } \Gamma_u, \tag{22}$$

$$\sigma_{ij} n_j = \bar{t}_i \quad \text{on } \Gamma_t, \tag{23}$$

where σ_{ij} is the Cauchy stress tensor, n_j is the unit normal to the surface, and f_i is the body force, respectively. From the Hooke's law, the stress tensor is given by [1]

$$\sigma_{ij} = C_{ijkl} \varepsilon_{kl} \tag{24}$$

where C_{ijkl} is a fourth order material constitutive tensor, and ε_{kl} is the index notation of strain tensor, which is defined as

$$\varepsilon_{kl} = \frac{1}{2}\left[u_{k,l} + u_{l,k}\right] \tag{25}$$

To determine the unique numerical solution out of a large set of solutions of governing equation of motion the displacement field $\mathbf{u}$ must satisfy the Eqs. (21) and (22). It is considered that S and ∂S are the trial and virtual solution

Table 1 Comparison of different properties of FEA and IGA basis functions

Property	Basis functions			
	Lagrangian	Bernstein	B-spline	NURBS
Non-negativity	Any sign	≥ 0	≥ 0	≥ 0
Inter element continuity	C^0	C^0	C^{p-1-k}	C^{p-1-k}
Partition of unity	Yes	Yes	Yes	Yes
Linear independence	Yes	Yes	Yes	Yes
Kronecker delta satisfied at the	Element boundary	Element boundary	Patch boundary	Patch boundary
Variation diminishing	No	Yes	Yes	Yes

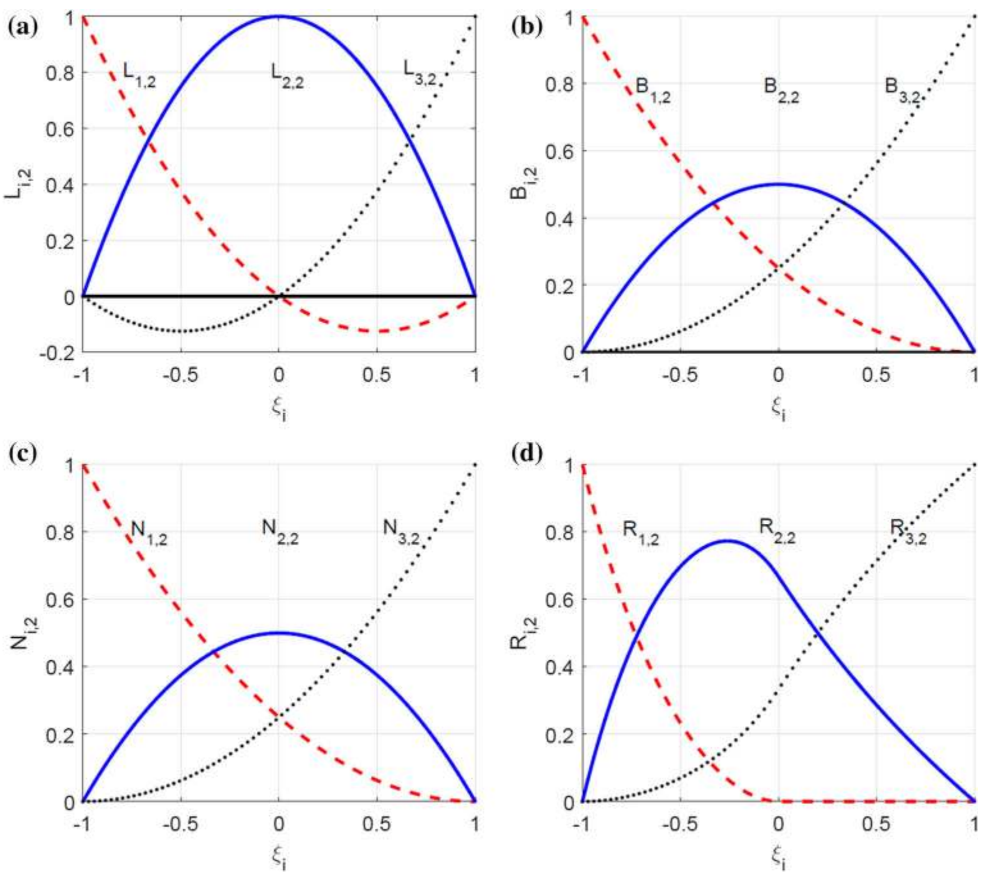

Fig. 1 Variation of quadratic: **a** Lagrangian, **b** Bernstein, **c** B-spline, and **d** NURBS basis functions over space ξ ranging from -1 to 1. The knot vector for B-spline and NURBS is $\boldsymbol{\Xi} = [-1, \ -1, \ -1, \ 1, \ 1, \ 1]$, and NURBS functions are associated with $w_1 = 0.5$, $w_2 = 0.6$ and $w_3 = 0.3$ weights. It should be noted that the variation of the NURBS functions will be identical to the variation of the B-spline basis functions for $w_1 = w_2 = w_3 = 1$

spaces, respectively, such that $\mathbf{u} \in S$ and $\delta\mathbf{u} \in \partial S$. The weak form is obtained by multiplying the Eq. (23) by the test function $\delta\mathbf{u}$ and then integrating by parts the stress term and using Eq. (24) as

$$\Pi_{\text{eq}} = \int_{\Omega} \varepsilon_{ij}(\mathbf{u}) C_{ijkl} \varepsilon_{kl}(\delta\mathbf{u}) \mathrm{d}\Omega - \int_{\Gamma_t} \bar{t}_i \delta u_i \mathrm{d}\Gamma - \int_{\Omega} f_i \delta u_i \mathrm{d}\Omega \tag{26}$$

Here, the virtual displacement $\delta\mathbf{u}$ represent the test or weight function. The solution of the weak form, Eq. (26), cannot often be evaluated directly through the analytical approaches. Thus, a numerical technique is followed in practice to find the approximate solution field. IGA incorporates the CAD modelling basis functions for the analysis and fetches results better than traditional FEA [4], but before outlining the details of IGA formulation, it is, first, instructive to consider the usual FE approach that will ease the understanding of variational formulation for IGA technique.

Finite Element Formulation

In general, finite element discretization technique is used to convert the weak form, Eq. (26), into a set of algebraic equations. The domain Ω is discretized into *nel* number of sub-domains Ω^e as: $\Omega = \sum_{e=1}^{nel} \Omega^e$. The weak form given by Eq. (26) can be written as summation over the sub-domains as [1]

$$\delta\Pi_{\text{eq}} \approx \sum_{e=1}^{nel} \delta\Pi_{\text{eq}}^e = \sum_{e=1}^{nel} \left[\int_{\Omega^e} \varepsilon_{ij}(\mathbf{u}^e) C_{ijkl} \varepsilon_{kl}(\delta\mathbf{u}^e) \mathrm{d}\Omega - \int_{\Gamma_t^e} t_i \delta u_i^e \mathrm{d}\Gamma - \int_{\Omega^e} f_i \delta u_i^e \mathrm{d}\Omega \right] \tag{27}$$

The geometry for a Lagrangian basis function described finite element is given as [1]

$$\mathbf{x}^e = \sum_{i=1}^{n^e} L_i \mathbf{x}_i \tag{28}$$

where $\mathbf{x}_i = [x_i\, y_i\, z_i]^{\mathrm{T}}$, and L_i represents the nodal variables and its corresponding Lagrangian basis, respectively and n^e is the number of nodes per element e. For the purpose of simplification, a shorter notation of multivariate FEA and IGA basis functions, $L_{i,j,k}^{p,q,r}(\bar{\xi}, \bar{\eta}, \bar{\zeta}) \rightarrow L_i$, is used in the remainder of this paper. Using the Galerkin formulation, the solution fields $\mathbf{u}$ and $\delta\mathbf{u}$ over a finite element is written [1]

$$\mathbf{u}^e = \sum_{i=1}^{n^e} L_i \mathbf{u}_i, \quad \delta\mathbf{u}^e = \sum_{i=1}^{n^e} L_i \delta\mathbf{u}_i \tag{29}$$

The strain-displacement matrix $\mathbf{B}$ for a three-dimensional Lagrangian element is given by [1]

$$\mathbf{B} = \begin{bmatrix} L_{1,x} & 0 & 0 & L_{2,x} & 0 & 0 & \cdots & L_{n^e,x} & 0 & 0 \\ 0 & L_{1,y} & 0 & 0 & L_{2,y} & 0 & \cdots & 0 & L_{n^e,y} & 0 \\ 0 & 0 & L_{1,z} & 0 & 0 & L_{2,z} & \cdots & 0 & 0 & L_{n^e,z} \\ L_{1,y} & L_{1,x} & 0 & L_{2,y} & L_{2,x} & 0 & \cdots & L_{n^e,y} & L_{n^e,x} & 0 \\ 0 & L_{1,z} & L_{1,y} & 0 & L_{2,z} & L_{2,y} & \cdots & 0 & L_{n^e,z} & L_{n^e,y} \\ L_{1,z} & 0 & L_{1,x} & L_{2,z} & 0 & L_{2,x} & \cdots & L_{n^e,z} & 0 & L_{n^e,x} \end{bmatrix} \tag{30}$$

By substituting Eqs. (29) and (30) in Eq. (26), and from the arbitrariness of virtual displacement $\delta\mathbf{u}$, the discretized weak form is obtained as

$$\sum_{e=1}^{nel} \left[\left(\int_{\Omega^e} \mathbf{B}^T \mathbf{C} \mathbf{B} \mathrm{d}\Omega \right) \mathbf{u} = \int_{\Gamma_t^e} \mathbf{L} \cdot t \mathrm{d}\Gamma + \int_{\Omega^e} \mathbf{L} \cdot f \mathrm{d}\Omega \right] \tag{31}$$

where $\mathbf{L}$ is the matrix of the Lagrangian basis functions. The definition of this matrix is given in "Appendix A.2". Based on the standard FEA notations, Eq. (31) can be cast into the following global matrix form

$$\mathbf{KU} = \mathbf{F} \tag{32}$$

where $\mathbf{K}$, $\mathbf{F}$, and $\mathbf{U}$ are the FE discretized global tangent matrix, external force vector, and displacement vector, respectively. This system of equation is solved for $\mathbf{U}$ after applying the essential boundary conditions. Stresses can then be determined using Eqs. (24) and (25).

IGA Formulation

The key feature of IGA which makes it distinct from traditional FEA is its manner of basis function incorporation and its ability to represent the exact form of geometries. The basis functions which are employed for modelling and discretization of CAD geometries are utilized in the numerical analysis. The space domain Ω, like in FEA, is discretized into a number of sub-domains Ω^e but with the CAD basis functions. The procedure of the discretization of the domain and the solution field can be understood from the following illustration. Let $R_{i,j,k}^{p,q,r}(\xi, \eta, \zeta) \rightarrow R_i$ be the multidimensional NURBS basis function used for the representation of an element Ω^e. The geometry of the NURBS discretized element Ω^e is given by the combination of the basis function R_i and the control points variables $\mathbf{P}_i$ as [4]

$$\mathbf{x}^e = \sum_{i=1}^{n_{cp}^e} R_i \mathbf{P}_i \tag{33}$$

where n_{cp}^e denotes the total number of control points in an element Ω^e. Using the Galerkin approach, the solution field is given by [4]

$$\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i \mathbf{u}_i, \quad \delta\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i \delta\mathbf{u}_i \tag{34}$$

where $\mathbf{u}_i$ and $\delta\mathbf{u}_i$ are the displacement and virtual displacement values at ith control point. The strain-displacement matrix $\mathbf{B}$ for a three-dimensional isogeometric element is given by [4]

$$\mathbf{B} = \begin{bmatrix} R_{1,x} & 0 & 0 & R_{2,x} & 0 & 0 & \cdots & R_{n_{cp}^e,x} & 0 & 0 \\ 0 & R_{1,y} & 0 & 0 & R_{2,y} & 0 & \cdots & 0 & R_{n_{cp}^e,y} & 0 \\ 0 & 0 & R_{1,z} & 0 & 0 & R_{2,z} & \cdots & 0 & 0 & R_{n_{cp}^e,z} \\ R_{1,y} & R_{1,x} & 0 & R_{2,y} & R_{2,x} & 0 & \cdots & R_{n_{cp}^e,y} & R_{n_{cp}^e,x} & 0 \\ 0 & R_{1,z} & R_{1,y} & 0 & R_{2,z} & R_{2,y} & \cdots & 0 & R_{n_{cp}^e,z} & R_{n_{cp}^e,y} \\ R_{1,z} & 0 & R_{1,x} & R_{2,z} & 0 & R_{2,x} & \cdots & R_{n_{cp}^e,z} & 0 & R_{n_{cp}^e,x} \end{bmatrix} \tag{35}$$

Substituting Eqs. (34) and (35) into Eq. (26), and from the arbitrariness of virtual displacement $\delta\mathbf{u}$ the discretized weak form is obtained as:

$$\sum_{e=1}^{nel}\left[\left(\int_{\Omega^e}\mathbf{B}^{\mathrm{T}}\mathbf{C}\mathbf{B}\,\mathrm{d}\Omega\right)\mathbf{u}=\int_{\Gamma_t^e}\mathbf{R}\cdot\boldsymbol{t}\,\mathrm{d}\Gamma+\int_{\Omega^e}\mathbf{R}\cdot\boldsymbol{f}\,\mathrm{d}\Omega\right] \tag{36}$$

where $\mathbf{R}$ is defined:

(a) for the boundary Γ^e as

$$\mathbf{R}=\begin{bmatrix} R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & 0 & \ldots & R_{n^e_{cp}}(\xi,\eta) & 0 \\ 0 & R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & \ldots & 0 & R_{n^e_{cp}}(\xi,\eta)\end{bmatrix}^{\mathrm{T}} \tag{37}$$

(b) and for the bulk Ω^e as

$$\mathbf{R}=\begin{bmatrix} R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & 0 & \ldots & R_{n^e_{cp}}(\xi,\eta,\zeta) & 0 \\ 0 & R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & \ldots & 0 & R_{n^e_{cp}}(\xi,\eta,\zeta)\end{bmatrix}^{\mathrm{T}} \tag{38}$$

As is for FEA, IGA discretized weak form, Eq. (36), can also be cast into the standard matrix form as

$$\sum_{e=1}^{nel}[\mathbf{K}^e\mathbf{U}^e=\mathbf{F}^e] \tag{39}$$

which in global form can be written as

$$\mathbf{K}\mathbf{U}=\mathbf{F} \tag{40}$$

where $\mathbf{K}^e$, $\mathbf{F}^e$, and $\mathbf{U}^e$ denote the NURBS described isogeometric element's stiffness matrix, external force vector, and displacement vector, respectively. It is important to note that for the case of IGA $\mathbf{U}$ corresponds to the displacement of the control points unlike in FEA where $\mathbf{U}$ corresponds to the nodal displacements. For the assembly of these locally computed quantities to their respective global part, i.e. $\mathbf{K}^e\rightarrow\mathbf{K}$ and $\mathbf{F}^e\rightarrow\mathbf{F}$, IGA follows the same FEA assembly procedure but utilizes the control point assembly array instead of nodal connectivity matrix. This assembly procedure is discussed in detail in "Assembly arrays" section.

Summary

From the above two formulations, it can be seen that the IGA based formulation of a boundary value problem yields the similar set of standard algebraic equations except for the change in the basis functions (from Lagrangian to NURBS).

Structural Difference Between FEA and IGA Codes

In this section, first, the code-architecture of a traditional FE code is presented. Then, based on the modifications within the FE code structure, a revised flow chart for the IGA technique is presented.

A standard flow chart of a typical FE code is divided into the three stages, e.g. pre-processor (I), processor (II), and post-processor (III) as shown in Fig. 2. The subroutines or sections of this chart, which need to be rewritten for the IGA, is also highlighted. A stepwise detailed description of modifications along with their respective implementations will be presented in the subsequent "A detailed stepwise implementation procedure for IGA" section. A revised flowchart, based on the modifications of featured subroutines in Fig. 2, for IGA is presented in Fig. 3.

Summary

In this section, it is illustrated that instead of building a new code structure for IGA the standard code architecture of FEA can directly be utilized by modifying a few subroutines, which are highlighted in Fig. 2.

A Detailed Stepwise Implementation Procedure for IGA

In this section, a step-by-step implementation procedure for IGA based on the modifications of the existing FE code structure, shown in Fig. 2, is presented. A collection of adaptable subroutines, which have been developed by Spink [32] and Spink et al. [33] based on the algorithms presented in Pigel and tiller [31], to solve general problems using IGA are utilized for the formation and handling of NURBS discretized geometries and basis functions. These subroutines can be used within any interpreted programming language, e.g., Matlab® and Octave. The overall modification and implementation procedure are sub-divided into the pre-processing, processing, and post-processing stages for the sake of simplicity. First, in "Changes within the preprocessing stage of analysis" section, the necessary modifications to be made in preprocessing stage of FE are discussed. Then, in "Changes within the processing stage" section, the highlighted subroutines, see Fig. 2, within the processing stage are revised for the setting of IGA. Finally, in "Post-processing in IGA" section, the necessary changes in post-processing stage of FEA are revised for the IGA. The reader may refer to [27] for understanding the IGA based implementation of one-dimensional structural bar problem. In this presentation, a standard infinite plate with the circular hole problem in

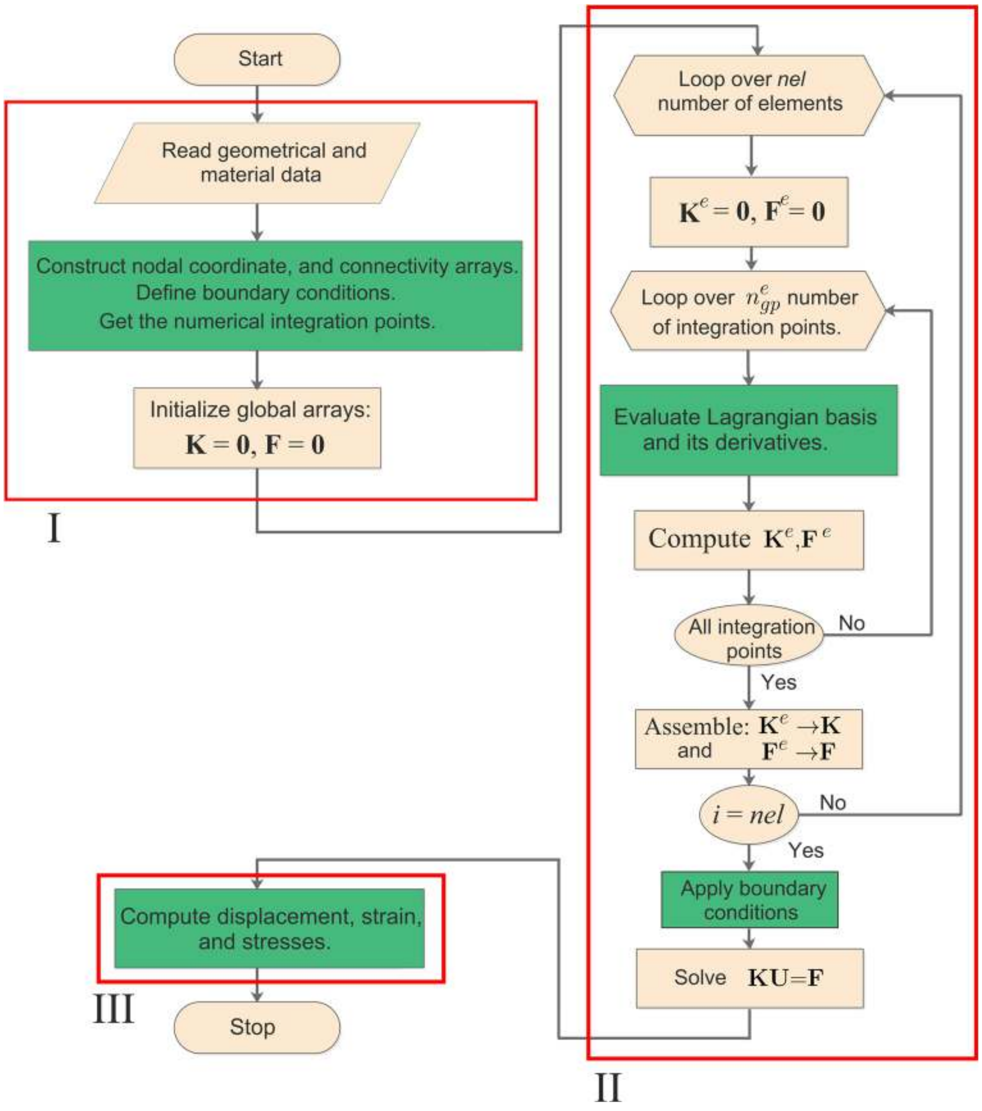
Start
Read geometrical and material data
Construct nodal coordinate, and connectivity arrays. Define boundary conditions. Get the numerical integration points.
Initialize global arrays: $\mathbf{K}=\mathbf{0}, \mathbf{F}=\mathbf{0}$
I
Loop over *nel* number of elements
$\mathbf{K}^e=\mathbf{0}, \mathbf{F}^e=\mathbf{0}$
Loop over n_{gp}^e number of integration points.
Evaluate Lagrangian basis and its derivatives.
Compute $\mathbf{K}^e, \mathbf{F}^e$
All integration points
No
Yes
Assemble: $\mathbf{K}^e \to \mathbf{K}$ and $\mathbf{F}^e \to \mathbf{F}$
$i = nel$
No
Yes
Apply boundary conditions
Solve $\mathbf{KU}=\mathbf{F}$
II
Compute displacement, strain, and stresses.
III
Stop

Fig. 2 A stage-wise standard flow chart of a typical FE code. Boxes titled 1, 2 and 3 represent the three stages of FE analysis, respectively. The highlighted boxes within each stage needs to be modified for IGA

tension, shown in Fig. 4, is taken for demonstrating the stepwise explanation of the proposed implementation procedure. The detailed description of the problem is provided in "Appendix A.1".

Changes Within the Preprocessing Stage of Analysis

In this section, subroutines accounting for the construction of geometry, connectivity arrays, and the boundary identification strategies for prescribed Dirichlet and Neumann conditions are revised within the framework of IGA.

Construction of NURBS Discretized Geometries

In IGA, apart from the geometrical details, e.g. height, length, width etc. of a CAD model, its parametric details such as knot vectors, the order of NURBS basis functions, and the control points are also needed to construct the exact form of a discretized geometry (as shown in "Preliminaries" section). This procedure can be understood as follows: Let for the given physical details of the geometry its parametric values such as the order of the NURBS basis functions, knot vectors, and control points are defined in each parametric directions such that the geometry retains its

original shape. Based on these input values, a subroutine '***nrbmak***' [34] is incorporated in place of the FEA geometry constructing subroutines to get the discretized form of the NURBS geometry. The format of this subroutine is

$$\mathbf{V}(\xi,\eta,\zeta) = \boldsymbol{nrbmak}(Control\,Points\,Array, \{knot\,Vector\,Arrays\}) \tag{41}$$

where $\mathbf{V}(\xi,\eta,\zeta)$ represents the NURBS described solid. For describing the application of this subroutine, let us consider the construction of geometry shown in Fig. 4. If the parametric details of a geometry are known it can be

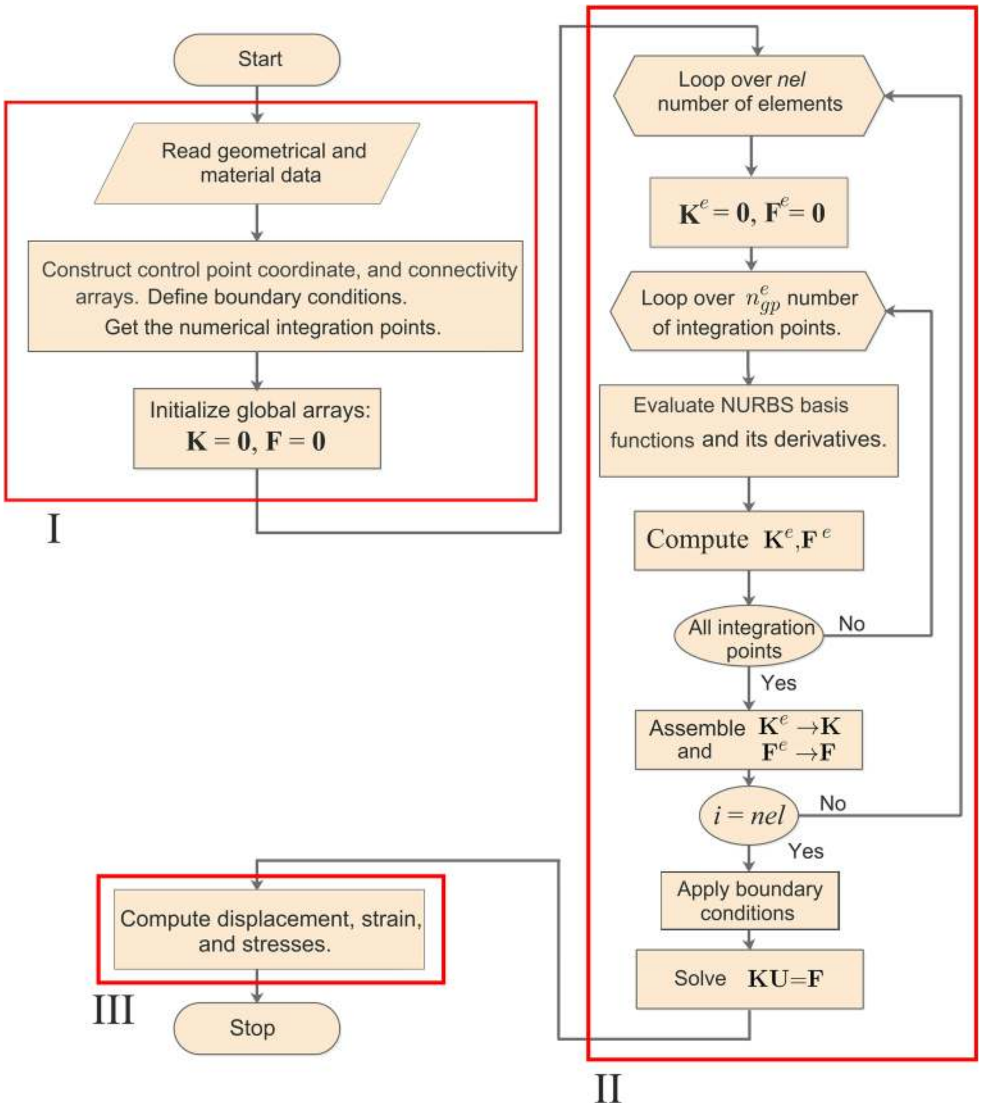
Start
Read geometrical and material data
Construct control point coordinate, and connectivity arrays. Define boundary conditions. Get the numerical integration points.
Initialize global arrays: $\mathbf{K}=\mathbf{0}, \mathbf{F}=\mathbf{0}$
I
Loop over nel number of elements
$\mathbf{K}^e = \mathbf{0}, \mathbf{F}^e = \mathbf{0}$
Loop over n^e_{gp} number of integration points.
Evaluate NURBS basis functions and its derivatives.
Compute $\mathbf{K}^e, \mathbf{F}^e$
All integration points
No
Yes
Assemble $\mathbf{K}^e \to \mathbf{K}$ and $\mathbf{F}^e \to \mathbf{F}$
$i = nel$
No
Yes
Apply boundary conditions
Solve $\mathbf{KU=F}$
II
Compute displacement, strain, and stresses.
III
Stop

Fig. 3 A flow chart for IGA. Boxed titled 1, 2 and 3 represent the three stages of analysis in the framework of IGA, respectively

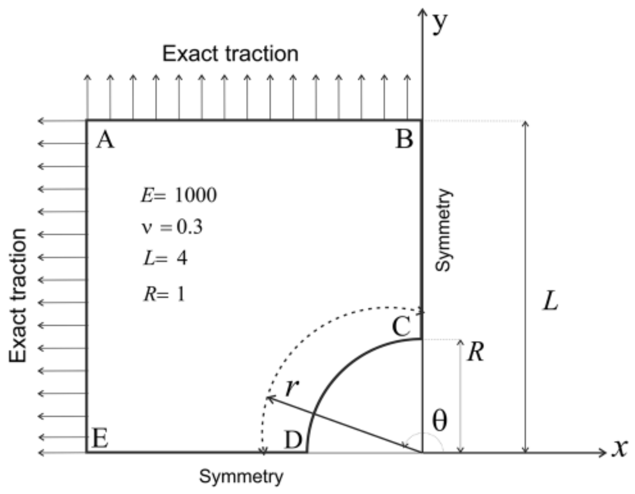

Fig. 4 Quarter model of an infinite plate with a circular hole that includes $u_y = 0$ (side ED), $u_x = 0$ (side BC), $t^{\mathrm{T}} = \sigma_{xx}$ (side AE) and $t^{\mathrm{T}} = \sigma_{yy}$ (side AB)

created in many different ways. A commercial CAD modelling software Rhino (www.rhino3d.com) can also be used for the construction of geometries. Although, for illustration, its simplest case, i.e., its coarsest form which incorporates two rational quadratic segments to ensure the C^1 continuity throughout the interior of a domain, is considered. Parametric quantities of quarter plate along the ξ and η directions are [3]:

(a) Order of univariate NURBS basis function:

$$p = 2 \text{ and, } q = 2$$

(b) Open knot vectors that divide the plate into two segments are

$$\boldsymbol{\Xi} = \{0,\quad 0,\quad 0,\quad 0.5\quad 1,\quad 1,\quad 1\} \quad \text{and,}$$
$$\mathbf{H} = \{0,\quad 0,\quad 0,\quad 1,\quad 1,\quad 1,\}$$

Here, the first and last knot entries are repeated by $p+1$ and $q+1$ times in $\boldsymbol{\Xi}$ and $\mathbf{H}$ knot vector, respectively to form an open knot vector.

(c) The control points array P along with the respective weight values are given in "Appendix B.1".

Based on these input information, the required geometry $\mathbf{S}(\xi,\eta)$ is constructed in the following fashion:

$$\mathbf{S}(\xi,\eta) = \boldsymbol{nrbmak}(P,\{\boldsymbol{\Xi},\mathbf{H}\}) \tag{42}$$

Then, subsequently, the '***nrbsctrlplot***' and '***nrbsctrlplot***' subroutines [34] are incorporated in the code structure for the visualization of required geometry $\mathbf{S}(\xi,\eta)$ by executing the following commands

$\boldsymbol{nrbsctrlplot}(\mathbf{S}(\xi,\eta))$

$\boldsymbol{nrbsctrlplot}(\mathbf{S}(\xi,\eta))$

The resulting plots from above commands are shown in Fig. 5a and b, respectively.

Assembly Arrays

In FEA technique, the domain Ω is first divided into a number of sub-domains Ω^e. Then, the elemental stiffness matrix $\mathbf{K}^e$ and the elemental force vector $\mathbf{F}^e$ are computed for each element Ω^e. These quantities are utilized to obtain the global stiffness matrix $\mathbf{K}$ and the global force vector $\mathbf{F}$, respectively with the help of local-to-global assembly arrays, e.g., nodal connectivity array. IGA follows the similar local-to-global assembly procedure to obtain a global set of algebraic equations but in a slightly different way. Two arrays: (1) control point assembly array and, (2) knot spans connectivity array is utilized for this purpose. These are discussed next

(1) **Control point assembly array** is built similar to the nodal assembly array in traditional FE procedure. It contains the global identity of the control points for all the elements in a row-wise manner, where each row stores the identity of control points participating in an element Ω^e. The total number of the control points participating in a three-dimensional element Ω^e is determined by the following relation:

$$n^e_{cp} = (p+1)(q+1)(r+1) \tag{43}$$

For the problem considered, a control point connectivity matrix is provided in "Appendix B.2". One may also utilize a generalized control point assembly array formation code, which is adapted from [4] and is presented in Listing 2 of "Appendix C". Further, to illustrate the assembly process within the framework of IGA, a code that shows the formation of global stiffness matrix using the assembly operation is also included in Listing 3 of "Appendix C".

(2) In IGA, apart from the control points connectivity matrix, an additional array called the **knot vector connectivity** is also needed to evaluate the NURBS basis functions and their respective derivatives in an element-wise manner. For this, the knot vector range of an element over which it is defined is needed. An example image of an element Ω^e which is defined over the $\left[\xi^e_i, \xi^e_{i+1}\right]$ and $\left[\eta^e_j, \eta^e_{j+1}\right]$ knot range along ξ and η direction is shown in Fig. 6. The procedure for defining the knot connectivity matrix for the plate problem is demonstrated in "Appendix B.2" based on which a generalized knot connectivity matrix can also be generated.

Boundary Conditions

In IGA, control points defining the boundaries Γ_u and Γ_t of the geometry $\mathbf{V}(\xi,\eta,\zeta)$ must be known for the enforcement of the Dirichlet and the Neumann boundary conditions. The control points defining these boundaries are detected in the same manner as in traditional FEA. The enforcement procedure of these conditions over the boundary control points is explained in the "Enforcement of the BCs in IGA" section.

Summary

In this section, it is illustrated that the preprocessing stage of FE code can be utilized directly for IGA if the geometry construction and connectivity arrays formation procedures are rewritten for the parametric details of NURBS basis functions, e.g., order of function, knot vector, and control point array along each parametric direction.

Changes Within the Processing Stage

In traditional FE procedure, the processing stage involves the computation of the elemental stiffness matrix, the external force vector, and the assembly of these arrays for the determination of solution field. The formation of these arrays requires the evaluation of FE basis function and its derivatives. The Gauss–Legendre quadrature rule is usually employed for the integration in Eq. (31), which involves the mapping of an element Ω^e from physical to master space. The processing stage of the FEA can be utilized for IGA in a straightforward manner if the evaluation procedure of FE basis function and its derivatives are replaced by their NURBS counterpart. Due to the incorporation of the NURBS basis functions, conventional mapping process

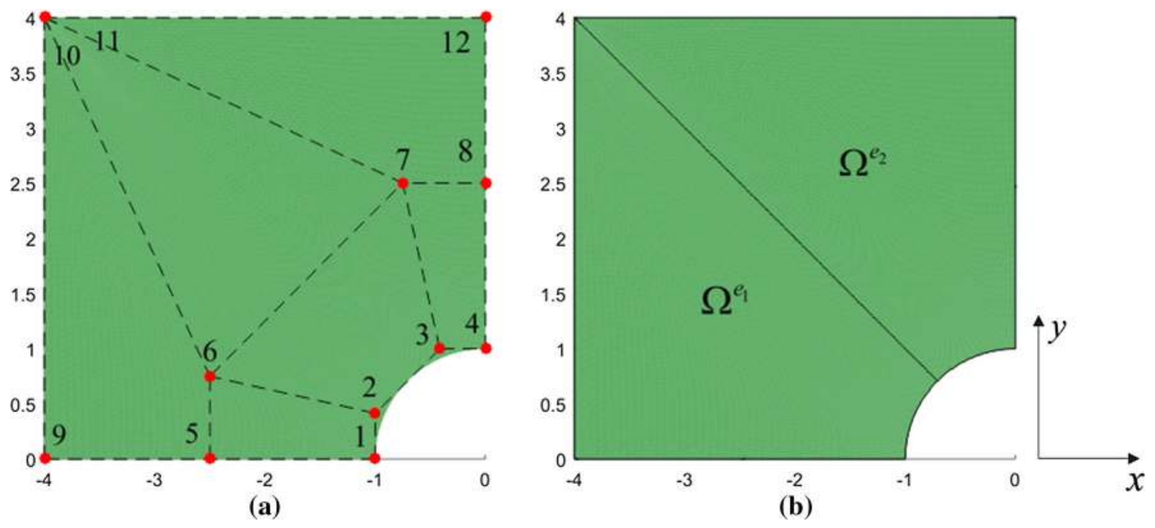

Fig. 5 Plot of NURBS discretized geometry **a** with $n_{cp} = 12$ number of control points, and **b** for $nel = 2$ number of elements. In (**a**) the dark (red) circles represent the control points whereas the shaded region is the modelled geometry

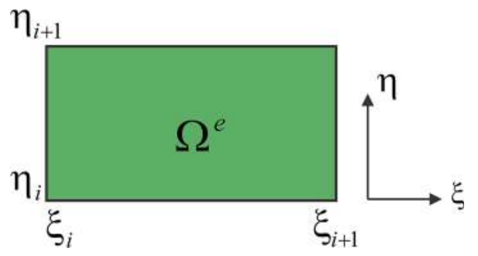

Fig. 6 Representation of the two-dimensional element Ω^e along with its knot spans

of FEA has to be redefined for IGA to compute the appropriate numerical integration of the NURBS discretized weak formulation.

The evaluation of NURBS basis functions and their respective derivatives, along with their corresponding mappings are first presented in "Evaluation of NURBS basis functions and their mapping in IGA" section. After that, the enforcement procedures for different boundary conditions (BCs) for NURBS defined geometry are included in the subsequent "Enforcement of the BCs in IGA" section.

Evaluation of NURBS Basis Functions and Their Mapping in IGA

In FEA, the Lagrangian basis functions L_i are defined either in the physical or the master space. The domain 'Ω' of the geometry is divided into a finite number elements Ω^e in the physical space. Then, each of these elements Ω^e are mapped to a regular shaped master element $\overline{\Omega^e}$ through their isoparametric mappings to facilitate the numerical integration. This procedure of mapping for a standard finite element, Ω^e, is shown in Fig. 7.

However, in the case of IGA, the NURBS basis functions are defined in the parametric space, see Eq. (15). The presence of the parametric space introduces the concept of an additional mapping which requires the transformation of NURBS described elements from the physical to the parametric space such that $\tilde{\phi} : \Omega^e \rightarrow \widetilde{\Omega^e}$. For the employment of the numerical integrations these elements are then mapped to regular shaped master element $\overline{\Omega^e}$ in the master space such that $\tilde{\phi} : \widetilde{\Omega^e} \rightarrow \overline{\Omega^e}$. The procedure of overall mapping for a NURBS defined element from physical to the master space is shown in Fig. 8.

For illustrating the implementation of the evaluation of NURBS basis functions and above explained mapping procedures, consider that an element is defined in the parametric space such that $\widetilde{\Omega}_e = [\xi_i, \xi_{i+1}] \otimes [\eta_j, \eta_{j+1}]$, see Fig. 6. In this regard, first, the NURBS basis functions and their respective derivatives are to be evaluated at ξ and η coordinates of element $\widetilde{\Omega}_e$. These coordinate values are calculated through the linear mapping from the master space to the parametric space by the following expressions:

$$\xi = \frac{1}{2}\left[(\xi_{i+1} - \xi_i)\bar{\xi} + (\xi_{i+1} + \xi_i)\right] \tag{44}$$

$$\eta = \frac{1}{2}\left[(\eta_{j+1} - \eta_j)\bar{\eta} + (\eta_{j+1} + \eta_j)\right] \tag{45}$$

where $\bar{\xi}$ and $\bar{\eta}$ are the known integration points in the master space and are obtained based on the integration rule, e.g., Gauss–Legendre quadrature rule.

Then, at these ξ and η coordinates of an element $\widetilde{\Omega^e}$, the NURBS basis functions and their first order derivatives are calculated with the help of the following subroutines [34]:

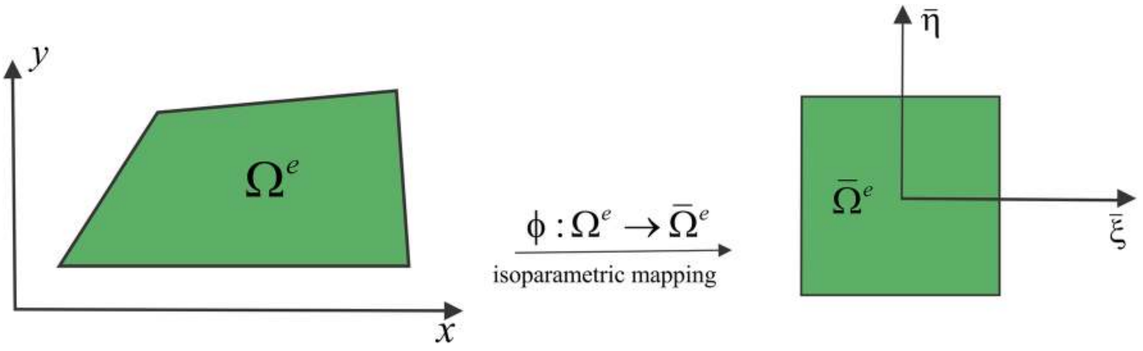

Fig. 7 Transformation of a finite element from the physical to the master space for the numerical integration in traditional FE

$$\bar{\mathbf{R}} = \boldsymbol{nrbbasisfun}(\{\xi, \eta\}, \mathbf{S}(\xi, \eta)) \tag{46}$$

$$\left[\frac{\partial \bar{\mathbf{R}}}{\partial \xi}, \frac{\partial \bar{\mathbf{R}}}{\partial \eta}\right] = \boldsymbol{nrbbasisfunder}(\{\xi, \eta\}, \mathbf{S}(\xi, \eta)) \tag{47}$$

where the matrix $\bar{\mathbf{R}}$ consists of n_{cp}^{e} number of non-zero NURBS basis functions, $R_i(\xi, \eta)$, at the ξ and η coordinate points of an element $\widetilde{\Omega}^e$. The description of NURBS function matrix $\bar{\mathbf{R}}$ is provided in "Appendix A.3" and the determinant of Jacobian matrix $\mathbf{J}_2$ which is required for the linear mapping is defined as

$$\mathbf{J}_2 = \frac{\partial \xi}{\partial \bar{\xi}} \frac{\partial \eta}{\partial \bar{\eta}} \tag{48}$$

$$|\mathbf{J}_2| = \left|\frac{1}{4}(\xi_{i+1} - \xi_i)(\eta_{j+1} - \eta_j)\right| \tag{49}$$

Thereafter, the Jacobian matrix $\mathbf{J}_1$ that enables the mapping of an element from the physical to the parametric space $\tilde{\phi} : \Omega^e \rightarrow \widetilde{\Omega^e}$ is computed as:

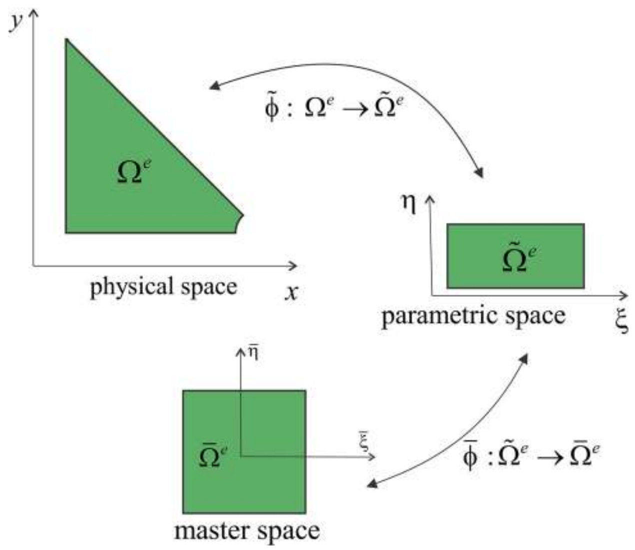

Fig. 8 Transformation of a physical element, Ω^e, to a master element, $\overline{\Omega^e}$, in IGA

$$\mathbf{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{50}$$

The components of $\mathbf{J}_1$ are calculated from Eq. (33) as

$$\frac{\partial x}{\partial \xi} = \sum_{k=1}^{n_{cp}^{e}} \frac{\partial R_k}{\partial \xi} x_i \tag{51}$$

$$\frac{\partial x}{\partial \eta} = \sum_{k=1}^{n_{cp}^{e}} \frac{\partial R_k}{\partial \eta} x_i \tag{52}$$

The mapping from the physical space to the master space $\left(\phi : \tilde{\phi} \cup \bar{\phi}\right)$ sets up the evaluation of NURBS basis functions and their respective derivatives. The procedure of overall mapping explained above can also be understood with the help of the following example in which the numerical integration of an arbitrary elemental quantity, e.g., $F(x, y)$, is computed over the physical space. To perform the numerical integration, the Gauss–Legendre integration rule is employed.

$$\begin{aligned}
\int_{\Omega} F(x, y) \mathrm{d}\Omega &= \sum_{e=1}^{nel} \int_{\Omega^e} F(x, y) \mathrm{d}\Omega \\
&= \sum_{e=1}^{nel} \int_{\widetilde{\Omega^e}} F(\xi, \eta) |J_1| \mathrm{d}\xi \mathrm{d}\eta, \quad \left(\tilde{\phi} : \Omega^e \rightarrow \widetilde{\Omega^e}\right) \\
&= \sum_{e=1}^{nel} \int_{\overline{\Omega^e}} F\left(\bar{\xi}, \bar{\eta}\right) |J_1||J_2| \mathrm{d}\bar{\xi} \mathrm{d}\bar{\eta}, \quad \left(\bar{\phi} : \widetilde{\Omega^e} \rightarrow \overline{\Omega^e}\right) \\
&= \sum_{e=1}^{nel} \int_{-1}^{1} \int_{-1}^{1} F\left(\bar{\xi}, \bar{\eta}\right) |J_1||J_2| \mathrm{d}\bar{\xi} d\bar{\eta} \\
&= \sum_{e=1}^{nel} \left[\sum_{i=1}^{n_{gp}^{e}} F\left(\bar{\xi}_i, \bar{\eta}_i\right) wp_i |J_1||J_2|\right]
\end{aligned}$$

where n_{gp}^{e} and wp_i denote the number of Gauss points and their weights, respectively in an element $\overline{\Omega^e}$. Once the global stiffness matrix and force vector are formed, the boundary conditions are applied to the global set of system equations, shown in the matrix form in Eq. (40), to determine the solution field. The procedure for enforcing the different BCs is presented next.

Enforcement of the BCs in IGA

In FEA, FE basis functions interpolate to all the nodes of the geometry due to their inherited C^0 continuity. This enables the FE basis functions to exhibit the Kronecker delta property at any of its boundary nodes. With this feature, the homogeneous, as well as non-homogeneous Dirichlet boundary conditions, are enforced directly at the nodes defining the boundaries. In IGA, the homogeneous Dirichlet and the Neumann boundary conditions are applied at the control points exactly as in the traditional FE approach as the NURBS basis functions also show C^0-continuity at the end of the open knot vector. However, in IGA, a particular treatment procedure for the imposition of inhomogeneous boundary condition is required, since the NURBS basis functions do not usually interpolate to all the boundary defining control points of the geometry due to their higher C^{p-1} continuity. The enforcement procedure for inhomogeneous Dirichlet boundary conditions (BCs) changes marginally for IGA. In this paper, the least square minimization method, which is adapted from [35], is used to explain the implementation procedure of the inhomogeneous boundary condition.

Let the displacement value $\bar{\mathbf{u}}(\xi)$ be prescribed at n number of ξ_i arbitrary points, which are scattered over the known displacement boundary Γ_u. The objective of least square method is to find the displacement quantity $u(\xi)$ that may be suitable for n^b_{cp} number of control points P_k over the boundary Γ_u such that it minimizes the following error quantity:

$$\begin{aligned} E &= \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{u}(\xi_i) - \bar{\mathbf{u}}(\xi_i)\|^2 \\ &= \sum_{i=1}^{n} \frac{1}{2} \left\| \sum_{k=1}^{n^b_{cp}} R_k(\xi_i) P_k - \bar{\mathbf{u}}(\xi_i) \right\|^2 \end{aligned} \tag{53}$$

where $R_k(\xi_i)$ is univariate NURBS basis function evaluated at ξ_i, $i = 1, 2, \ldots, n$ number of arbitrary control points, with the help of Eq. (15). The solution of Eq. (53) provides the $\mathbf{u}(\xi_i)$ displacement for n^b_{cp} number of the boundary defining control points, which are then applied to adjust the global system of equations in the same manner as in the traditional FEA to incorporate the inhomogeneous BCs. Other treatment methods can also be followed, see for Ref. [36]. It should be noted that apart from the above-explained modifications, the remaining code structure of the analysis stage of FEA is utilized in its original form for the implementation of IGA.

Summary

In this section, calculation of multi-variate NURBS basis functions and their corresponding derivatives within the framework of IGA along with the mappings is presented. Also, the enforcement procedures over the control points of NURBS defined boundaries is provided. The combinations of these modifications complete the processing stage for the proposed implementation of IGA within the FEA structure.

Post-processing in IGA

In FEA, post-processing is usually done for the visualization of the evaluated displacements, strains and stress fields. For this, the field is first computed is first computed at all the nodes of the discretized geometry. After that, the resulting values are plotted over either the undeformed or the deformed geometry for the visualization of the respective field. In IGA, this aspect does not differ much from the usual FEA's post-processing procedures [4]. The subroutines of the FEA stage, which need to be modified for the implementation of IGA are explained in this section. First, in "Visualization of the deformed geometry" section the implementation aspects for the visualization of deformed geometry are included, then in the "Plotting the displacement and the stress fields" section, the procedure of plotting the displacement and the stress fields is illustrated.

Visualization of the Deformed Geometry

In FEA, the nodal coordinates the deformed geometry are obtained by adding the nodal displacement field, **U**, to the nodal coordinates of the original geometry. These updated nodal values are then employed to plot the deformed shape. In IGA, an identical procedure is followed to determine the updated values of the control points, $[\mathbf{P}^{new}]$, of the deformed geometry. Then, the updated values of the control points, $[\mathbf{P}^{new}]$, is used to construct the deformed NURBS geometry. This is explained next. Let $[\mathbf{U}]$ be the control point displacement vector and $[\mathbf{P}] = [\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{n_{cp}}]^{\mathrm{T}}$ be the control points array which contains the initial coordinates of n_{cp} numbers of control points. The updated coordinates of the control points $[\mathbf{P}^{new}]$ for the deformed geometry are obtained as

$$[\mathbf{P}^{new}] = [\mathbf{P}] + [\mathbf{U}] \tag{54}$$

Then, the deformed configuration of the NURBS geometry is constructed by using the same '***nrbmak***'

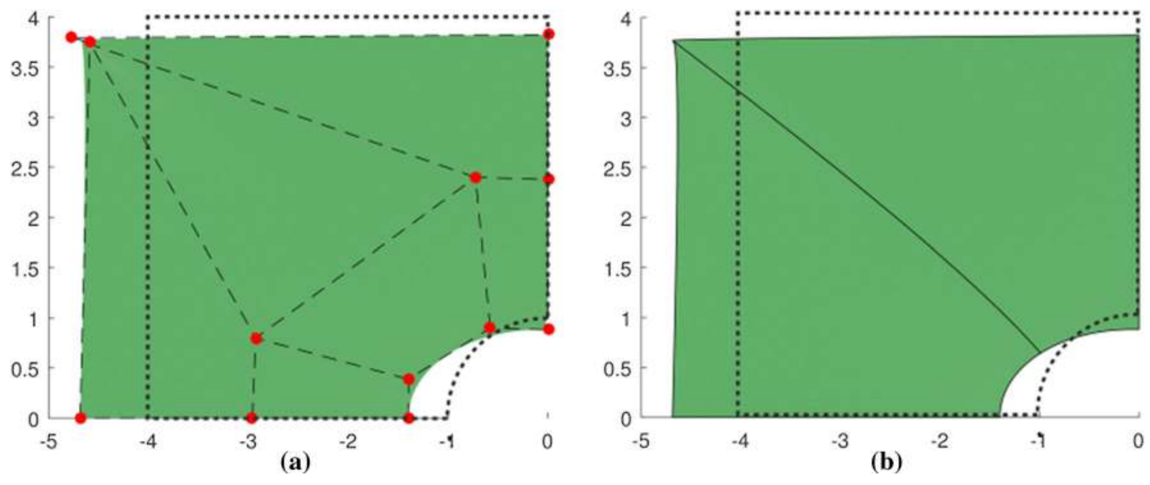

Fig. 9 Representation of deformed geometry: **a** with control points, **b** discretized form. Initial or the undeformed configuration is shown with bold dashed lines

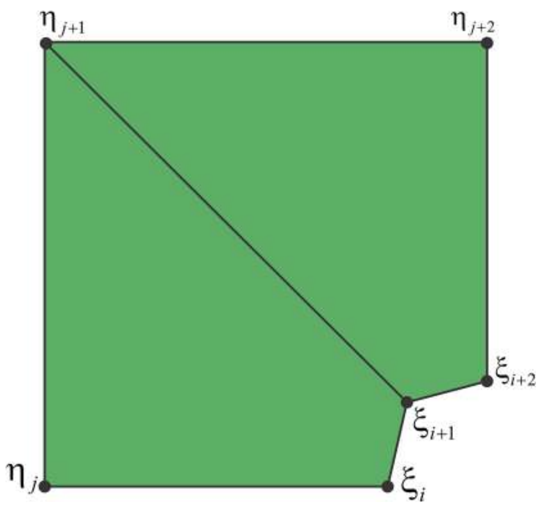

Fig. 10 Representation of the projection of NURBS elements (Q1) into the FEA like four noded quadrilateral elements, and the node coordinates are defined based on the unique value of knots in Ξ and **H** knot vectors

subroutine, which has been used to construct the initial NURBS geometry. The plot of deformed geometry along with its control points is obtained through '***nrbctrlplot***' subroutine as

$\boldsymbol{nrbctrlplot}(\mathbf{S}(\xi,\eta))$

and the plot of the discretized deformed geometry is obtained by

$\boldsymbol{nrbctrlplot}(\mathbf{S}(\xi,\eta))$

Based on these two subroutines, plots of the deformed configuration of the quarter plate with the circular hole along with the control points and the elements are shown in Fig. 9a and b, respectively. For the sake of visualization of deformation, the displacements are multiplied by a factor 1500.

Plotting the Displacement and the Stress Fields

In FEA, the values of the stains and stresses are obtained at the Gauss integration points of each element which are then interpolated at the element's nodes through various stress recovery methods, e.g., nodal averaging, least square and nodal projection method [1]. However, IGA incorporates the NURBS defined meshes that include the information of knots and control points of each element, which is not the case in standard FEA. In IGA, a NURBS mesh is first represented as a standard four-noded quadrilateral Lagrangian element (Q1) of FEA,[1] as shown in Fig. 10. The coordinates values of the nodes of projected NURBS element Q1 are assigned based on the $[\xi_i, \xi_{i+1}]$ and $[\eta_i, \eta_{i+1}]$ knot spans of an actual NURBS element, which is shown in Fig. 6.

To illustrate the procedure of forming the projected Q1 mesh from an implementation point of view, a code, which is written in the Matlab® environment, that can also be utilized with Octave or with similar programming languages, is included in Listing 4 of "Appendix C". Based on the computed value of displacements, as in Eq. (34), first, the strain is determined by substituting the values of displacement in Eq. (25). Then, using Eq. (24) the corresponding stresses at the nodes of the projected Q1 meshes are determined. It should be noted that in IGA the nodal averaging or stress recovery techniques are not necessary to obtain the continuous field due to the higher continuity of the NURBS basis functions. The computed stress along the nodal information of Q1 elements can either be exported to any commercial visualization programs, e.g., Paraview, HyperMesh or to a standard visualization Matlab® or Octave based subroutines for obtaining the plots of

[1] Herein, the procedure of projecting a NURBS mesh into a four-noded quadrilateral element is adopted from [26].

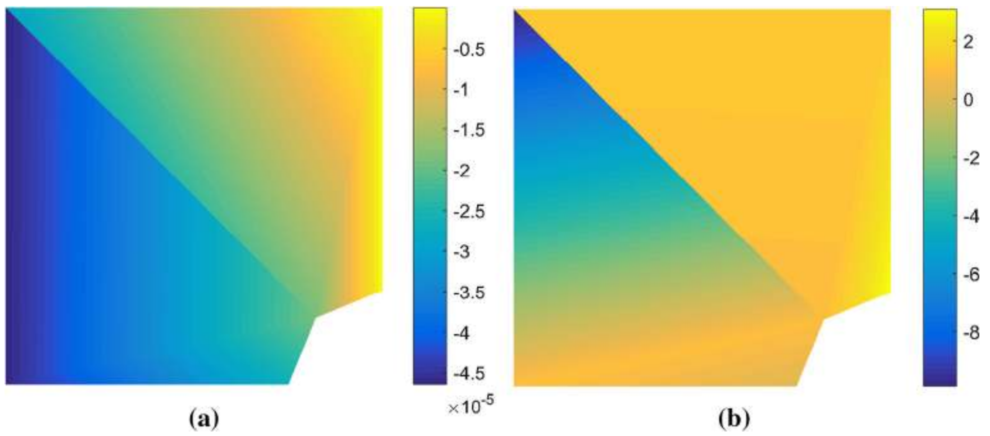

Fig. 11 The contour plot of (**a**) the displacement in mm and, **b** the IGA stress field in N/mm^2 along the x-direction over the projected Q1 meshes. **a** Displacement along the x-direction. **b** Stress along the x-direction

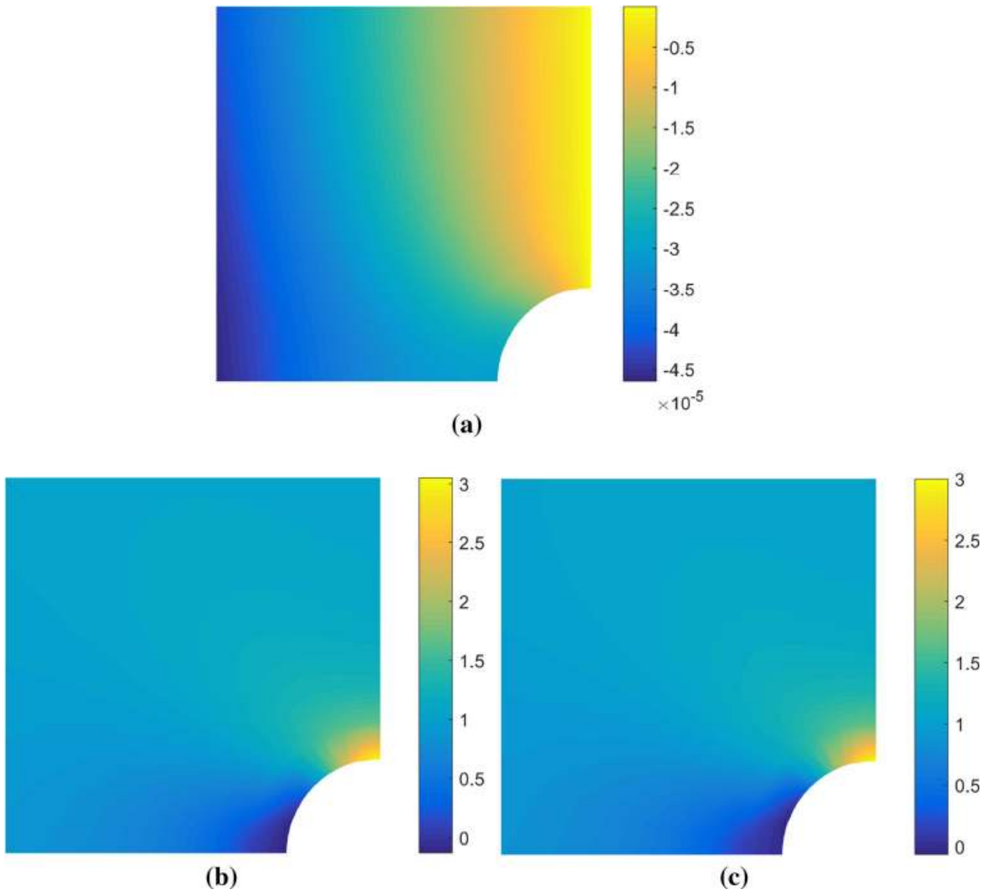

Fig. 12 The plot of the (**a**) displacement in mm, and **b** stress fields in N/mm^2 along the x-direction over refined Q1 meshes. In (**c**), the analytical stress field in N/mm^2 along the x-direction is shown

respective fields. In this work, Matlab® generated plots of the displacement and the stress fields along the x-direction are shown in Fig. 11a and b, respectively. However, it should be noted that the current discretization, as shown in Fig. 10, is too coarse to obtain the smooth displacement and stress fields. For this, a refined NURBS geometry that has a total of (128 = 16 × 8) elements is utilized to obtain the continuous plot of the displacement and stress fields. This is shown in Fig. 12a and b, respectively. It can be observed that the stress field, shown in Fig. 12b, matches well with the stress field obtained using the analytical solution shown in Fig. 12c. The expressions for the analytical stresses are given in "Appendix A.1".

Summary

It is shown that for post-processing the NURBS defined elements are first projected to the quadrilateral four-noded

FE elements. Then, the plot of the deformed geometry, displacement field, and the stress field can be obtained easily with the help of available NURBS subroutine functions.

Conclusion

In the present manuscript, a simplified tutorial for the implementation of NURBS based IGA within the existing FE code architecture is presented. A step-by-step explanation of the proposed tutorial is provided in clear and concise manner specifically for new IGA users or researchers who want to learn the implementation procedure of this technique. In this presentation, it is shown that the existing code structure of FEA can be adapted for the implementation of IGA by rewriting only few FE subroutines. It is, however, emphasized that the presented procedure avoids the need to write the application subroutines for IGA from scratch and thus, saves a considerable amount of time. To aid the understanding of the practical implementation of IGA, a plate with the circular hole under the action of in-plane tension is included. Comparison between the IGA and the FE techniques are made throughout this paper, and the major differences between these two techniques are summarized in Table 2 of "Appendix D". The inclusion of T-spline basis functions in the current framework, which are robust and the generalized form of NURBS basis functions, will be considered in the future work.

Acknowledgement The authors are grateful to the SERB, DST for supporting this research under project SR/FTP/ETA-0008/2014.

Appendix A

A.1 Problem Description

An infinite plate (L = 8 mm and W = 8 mm) with a circular hole subjected to in-plane traction is shown in Fig. 13. Due to the symmetries of the geometric boundary conditions and the loading about the origin only a quarter of the problem is considered, see Fig. 4. The material properties are taken as: Young's modulus E = 10^3 N/mm^2 and Poisson's ratio $\nu = 0.3$. The plane stress conditions are assumed. The exact solution for the in-plane stresses are given by [26]:

$$\sigma_{xx}(r,\theta) = 1 - \frac{R^2}{r^2}\left(\frac{3}{2}\cos 2\theta + \cos 4\theta\right) + \frac{3}{2}\frac{R^4}{r^4}\cos 4\theta$$
$$\sigma_{yy}(r,\theta) = -\frac{R^2}{r^2}\left(\frac{1}{2}\cos 2\theta - \cos 4\theta\right) - \frac{3}{2}\frac{R^4}{r^4}\cos 4\theta$$
$$\sigma_{xy}(r,\theta) = -\frac{R^2}{r^2}\left(\frac{1}{2}\sin 2\theta + \sin 4\theta\right) + \frac{3}{2}\frac{R^4}{r^4}\sin 4\theta$$

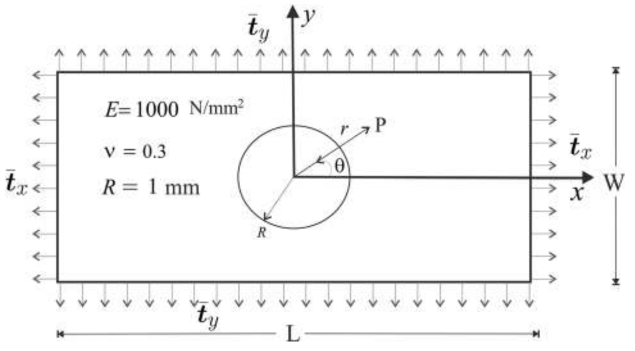

Fig. 13 An infinite plate a circular hole under the application of in-plane tractions

where r and θ are shown in Fig. 13.

A.2 Form of the Lagrangian Basis Function Matrix

The Lagrangian basis function matrix **L**, in Eq. (31) is defined for the boundary Γ^e as

$$L = \begin{bmatrix} L_1(\bar{\xi},\bar{\eta}) & 0 & L_2(\bar{\xi},\bar{\eta}) & 0 & \dots & L_{n_n^e}(\bar{\xi},\bar{\eta}) & 0 \\ 0 & L_1(\bar{\xi},\bar{\eta}) & 0 & L_2(\bar{\xi},\bar{\eta}) & \dots & 0 & L_{n_n^e(\bar{\xi},\bar{\eta})} \end{bmatrix}^T$$

and for the bulk Ω^e as

$$L = \begin{bmatrix} L_1(\bar{\xi},\bar{\eta},\bar{\zeta}) & 0 & L_2(\bar{\xi},\bar{\eta},\bar{\zeta}) & 0 & \dots & L_{n^e}(\bar{\xi},\bar{\eta},\bar{\zeta}) & 0 \\ 0 & L_1(\bar{\xi},\bar{\eta},\bar{\zeta}) & 0 & L_2(\bar{\xi},\bar{\eta},\bar{\zeta}) & \dots & 0 & L_{n^e}(\bar{\xi},\bar{\eta},\bar{\zeta}) \end{bmatrix}^T$$

where n_n^e, and n^e denote the number of nodes on the boundary, and within the bulk of the element $\mathbf{\Omega}^e$.

A.3 Form of the Matrices Employed in IGA

The NURBS basis function matrix $\bar{\mathbf{R}}$, shown in Eqs. (46) and (47), is defined as:

$$\bar{\mathbf{R}} = \left[R_1(\xi,\eta,\zeta) \quad R_2(\xi,\eta,\zeta) \quad \dots \quad R_{n_{cp}^e}(\xi,\eta,\zeta) \right] \tag{55}$$

where, $R_i(\xi,\eta,\zeta)$ represents the tensor product of univariate NURBS basis function at ith control point. For the plate with a circular hole problem considered in this work, the order of the basis function is chosen as $p = 2$ and $q = 2$. Therefore, the total number of active control points and the basis functions in an element are $= (p+1)\times(q+1) = 9$. The NURBS basis functions' matrix and the strain-displacement matrix for an element $\tilde{\mathbf{\Omega}}^e$ are given by:

$$\bar{\mathbf{R}} = [R_1(\xi,\eta) \quad R_2(\xi,\eta) \quad \dots \quad R_9(\xi,\eta)],$$

and

$$\mathbf{B} = \begin{bmatrix} R_{1,x}(\xi,\eta) & 0 & R_{2,x}(\xi,\eta) & 0 & \dots & R_{9,x}(\xi,\eta) & 0 \\ 0 & R_{1,y}(\xi,\eta) & 0 & R_{2,y}(\xi,\eta) & \dots & 0 & R_{9,y}(\xi,\eta) \\ R_{1,y}(\xi,\eta) & R_{1,x}(\xi,\eta) & R_{2,y}(\xi,\eta) & R_{2,x}(\xi,\eta) & \dots & R_{9,y}(\xi,\eta) & R_{9,x}(\xi,\eta) \end{bmatrix} \tag{56}$$

The matrix $\mathbf{R}$, which includes the boundary defining univariate basis functions for the plate problem is given by

$$\mathbf{R} = \begin{bmatrix} R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & 0 & \dots & R_9(\xi,\eta) & 0 \\ 0 & R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & \dots & 0 & R_9(\xi,\eta) \end{bmatrix}^{\mathrm{T}} \tag{57}$$

Appendix B

B.1 Parametric Details for the Plate Problem

The control points and weight values for the quarter of plate can be defined as follows. The number of control points along the ξ direction is given by [31]: $n_{cp}(\xi) =$ total number of knots in $\boldsymbol{\Xi} - (\mathrm{p}+1) = 7 - (2+1) = 4$. Similarly, the number of control points along the η direction is given by [31]: $m_{cp}(\eta) =$ total number of knots in $\mathbf{H} - (q+1) = 6 - (2+1) = 3$. Thus, a total number of control points n_{cp} for the coarse mesh shown in Fig. 5a is given by $n_{cp} = n_{cp}(\xi) \times m_{cp}(\eta) = 4 \times 3 = 12$. The coordinate and the weight values of a control point are stored in vector $\mathbf{P}_i = [x_i \quad y_i \quad z_i \quad w_i]^{\mathrm{T}}$, where $i = 1, 2, \ldots, n_{cp}$. Since, the plate with a hole geometry has $n_{cp}(\xi) = 4$, and $m_{cp}(\eta) = 3$ number of control points along the ξ, and η directions, hence, the control point array $\mathbf{P}$ for the geometry shown in Fig. 5, is given by

$$\begin{aligned} \mathbf{P}(:,:,1) &= \begin{bmatrix} -1 & -0.853553 & -0.353553 & 0 \\ 0 & 0.353553 & 0.853553 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0.853553 & 0.853553 & 1 \end{bmatrix} \\ \mathbf{P}(:,:,2) &= \begin{bmatrix} -2.5 & -2.5 & -0.75 & 0 \\ 0 & 0.75 & 2.5 & 2.5 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ \mathbf{P}(:,:,3) &= \begin{bmatrix} -4 & -4 & -4 & 0 \\ 0 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned} \tag{58}$$

B.2 Formation of the Assembly Arrays

B.2.1 Formation of the Control Point Assembly Array

The control points that participate in an element for the plate geometry, Eq. (43), is given by $n_{cp}^e = (p+1)(q+1) = 9$. The control point connectivity array, denoted by ConnectivityCP, for the coarse mesh, which includes two elements as shown in Fig. 5b, is given by

$$\mathrm{ConnectivityCP} = \begin{bmatrix} 1 & 2 & 3 & 5 & 6 & 7 & 9 & 10 & 11 \\ 2 & 3 & 4 & 6 & 7 & 8 & 10 & 11 & 12 \end{bmatrix} \tag{59}$$

A generalized code for forming the control point connectivity matrix for any two-dimensional geometry, which can be used within Matlab®, Octave, or with any other similar programming platform, is included in Listing 2. Using this code, the assembly array, given by Eq. (59), is formed.

B.2.2 Formation of the Knot Vector Connectivity Matrix

The knot vector connectivity matrix consists of the knot spans for all the elements in a row-wise manner. The knot span ranges along the ξ, η, and ζ directions which are associated with an element are stored in the columns of that element's respective row. Therefore, before forming the knot vector connectivity matrix for a NURBS discretized geometry, first, the knot span arrays for all the elements along each parametric direction need to be defined. To understand this, let us consider the knot vector connectivity matrix forming procedure for the plate with a circular hole problem. This geometry is discretized with 2 elements along the ξ and 1 element along η direction. Therefore, the total number of elements *nel* is $2 \times 1 = 2$. Hence, the knot span array along the ξ direction, denoted by SpanRangU, has rows equal to the total number of elements along the ξ direction, i.e., 2. The explicit form for SpanRangU is given by

$$\mathrm{SpanRangU} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 1 \end{bmatrix} \tag{60}$$

Similarly, the knot span array along η direction, denoted by SpanRangV, has rows equal to the total number of elements along the η direction, i.e., 1. The explicit form for SpanRangV is given by

$$\mathrm{SpanRangV} = [0 \quad 1] \tag{61}$$

The knot vector connectivity matrix, denoted by knotConnectivity, for the plate with the circular hole problem is given by

$$\mathrm{knotConnectivity} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \tag{62}$$

where the first and second columns of the array store the global number of knot span ranges (or the row number of SpanRangeU and SpanRangeV arrays) of each element along the ξ and η directions respectively. A generalized code for forming the knot vector connectivity matrix for any two-dimensional geometry is included in Listing 2.

Appendix C

Listing 1 Construction of the geometry for the plate with a hole problem

```
% 1. Physical details fo the geometry are:
L      = 4;                  % Length/Height of plate in mm
R      = 1;                  % Radius of hole in mm, see Fig. 4


% 2. Parametric details of the geometry are:
KnotU = [0 0 0 0.5 1 1 1];   % Knot vector defining the coarset mesh
                             % along the xi direction.
KnotV = [0 0 0 1 1 1];       % Knot vector defining the coarset mesh
                             % along the eta direction.
w      = 0.5*(1+1/sqrt(2));  % Weight points associated to the quarter
                             % of the hole defining control points.

% 3. Control point matrix for the coarsest mesh is given as, Eq. (58):
CParray(:,:,1) = [-R   -R*w           -0.414214*w  0;
                  0    0.414214*w      R*w         R;
                  0    0               0           0;
                  1    w               w           1];
CParray(:,:,2) = [-2.5  -2.5          -0.75        0;
                  0     0.75           2.5         2.5;
                  0     0              0           0;
                  1     1              1           1];
CParray(:,:,3) = [-L  -L             -L          0;
                  0   L               L          L;
                  0   0               0          0;
                  1   1               1          1];

% 4. Construct the NURBS described goemetry.
geometry = nrbmak(CParray,{KnotU, KnotV});  % Eq. (41)
```

Listing 2: Construction of the assembly arrays

```matlab
%-------------------------------------
% 1. Formation of the control point assembly array
%-------------------------------------
function [ConnectivityCP]= CPassembly_arrays(n,p,m,q)
% Input:   n— no of control points along xi direction
%          p— order of basis function along xi direction
%          m— no of control points along eta direction
%          q— order of basis function along eta direction


nel              = (n—p)*(m—q);    % Total no of elements in the mesh
ncp              = n*m;            % Total no of control points
necp             = (p+1)*(q+1);    % Total no. of CPs in a element
ConnectivityCP    = zeros(nen,nel); % Control point assembly array

A = 0;                              % Counter variable
e = 0;                              % Counte variable
for j = 1:m                         % Loop over total CPs along eta
    for i = 1:n                     % Loop over total CPs along xi
        A = A+1;
        if i >= (p+1) && j >= (q+1)
            e = e+1;
            for jloc = 0:q                  % Loop over order eta
                for iloc = 0:p              % Loop over order xi
                    B = A — jloc*n — iloc;      % Assigning global no.
                    b = jloc*(p+1) + iloc + 1; % Assigning local no.
                    ConnectivityCP(b,e) = B;    % Connectivity array
                end
            end
        end
    end
end
ConnectivityCP = sort(ConnectivityCP)';
end

%-------------------------------------
% 2. Formation of the knot vector connectivity array
%-------------------------------------
nelU              = (n—p)           ; % No. of element along xi
nelV              = (m—q)           ; % No. of element along eta
nel               = nelU*nelV       ; % Total no. of elements
knotConnectivity = zeros(nel,2)     ; % Knot connet. array
SpanRangU         = zeros(noElemU,2); % Knot connet. array in xi
SpanRangV         = zeros(noElemV,2); % Knot connet. array in eta
temp              = 1               ; % Temporary variable
for i=1:nelV                          % Loop over elements in eta
    for j=1:nelU                      % Loop over elements in xi
        knotConnectivity(temp,:)  = [j, i];                 % Eq. (62)
        SpanRangU(j,:)            = [unqU(j), unqU(j+1)];   % Eq. (60)
        SpanRangV(i,:)            = [unqV(i), unqV(i+1)];   % Eq. (61)
        temp                      = temp+1;
    end
end
```

Listing 3: Generation of the global stiffness matrix

```
KG           = zeros(2*ncp, 2*ncp); % Initialize the global [K] matrix
FG           = zeros(2*ncp, 1)    ; % Initialize the global [F] vector
CnnCPe       = sort([2* ConnectivityCP-1,  2*ConnectivityCP]);
noGPs        = 2*(p+1)*(q+1)      ; % Total no. of Gauss points
[GPwt, Gpnt] = Gauss_rule(noGPs)  ; % Gauss-points and weights


for i = 1:nel                       % Loop over total no. of elements
    Ke      = zeros(2*necp,2*necp); % Initialize local matrix Ke
    B       = zeros(3,2*necp)     ; % Initialize strain-disp matrix B
    elU     = SpanRangU(knotConnectivity(i,1),:);  % Knot span in xi
    elV     = SpanRangV(knotConnectivity(i,2),:);  % Knot span in eta

    for j = 1: noGPs                % Loop over total no. Gauss-points
        gpnt             = Gpnt(j,:);          % jth Gauss-point
        wg               = GPwt(j,:);          % jth Gauss weight
        xitilda          = gpnt(1);            % Parametric cord in xi
        etatilda         = gpnt(2);            % Parametric cord in et
        xi               = 0.5*((elU(2)-elU(1))*xitilda + ...
                           (elU(2)+ elU(1)));           % Eq. (44)
        eta              = 0.5*((elV(2)-elV(1))*etatilda + ...
                           (elV(2)+ elV(1)));           % Eq. (45)
        dxi_dxitilda     = 0.5*(elU(2)-elU(1));
        deta_detatilda   = 0.5*(elV(2)-elV(1));
        J2               = dxi_dxitilda*deta_detatilda; % Eq. (49)

        %---Evaluation of NURBS basis functions derivatives----%
        [dRedxi, dRedeta]  = nrbbasisfunder({xi, eta}, geometry);%Eq 47

        %---Jacobian Matrix J1----%
        jacob      = [geometry.coefs(1,ConnectivityCP(:,i)); ...
             geometry.coefs(2,ConnectivityC     P(:,i))]*[dRedxi',
                dRedeta'];
        J1         = det(jacob);                       % Eq. (50)
        invJ1      = inv(jacob);                       % Inverse of J1

        %---NurbsBasisFunctions Derivative w.r.t. x & y-------%
        dRdx        = [dRdxi' dRdeta']* invJ1(:,1);
        dRdy        = [dRdxi' dRdeta']* invJ1(:,2);

        %-----Strain matrix B-----------%              % Eq. (56)
        for i2=1:necp                                  % Loop over necp
            j1        = (2*i2-1);
            j2        = 2*i2;
            B(1,j1)   = dRedx(i2);                     % 1st row of B
            B(2,j2)   = dRedy(i2);                     % 2nd row of B
            B(3,j1)   = dRedy(i2);                     % 3rd row of B
            B(3,j2)   = dRedx(i2);                     % 3rd row of B
        end

        % Forming the local stiffness matrix Ke
        Ke = Ke+ B'*C*B*J1*J2*GPwt;
    end

    % Assembly from the local-to-global level where CnCPe corresponds
       to the global row and column number of the gloabl stiffness
       matrix.
    KG(CnnCPe(i,:), CnnCPe(i,:))= KG(CnnCPe(i,:), CnnCPe(i,:)) + Ke;
end
```

Listing 4: Generation of the projected Q1 element shown in Fig. 10 (adopted from [26])

```
 1  CPsX      = geoemetry.coefs(1,:)';     % Get the control points along xi
 2  CPsY      = geometry.coefs(2,:)';      % Get the control points along et
 3  noCPsX    = length(CPsX);              % Get the no. of CPs along xi
 4  noCPsY    = length(CPsY);              % Get the no. of CPs along eta
 5
 6  noKnotU   = length(KnotU);             % No. of knots in xi
 7  noKnotV   = length(KnotV);             % No. of knots in eta
 8
 9  CParray1 = CParry(:,:)';               % Transpose of the CParray
10  Dimn      = size(CParray1, 2);         % Dimension of the CParray
11
12  Node      = zeros(noKnotU*noKnotV,2); % Initial Node array
13  count     = 1;                         % Counter variable
14
15  % building nodes for the projected Q1 mesh
16  for i=1:noKnotV                        % Loop over knots in eta
17      eta = KnotV(i);                        % Get knot i from KnotV
18      for j=1:noKnotU                        % Loop over knots in xi
19          xi = KnotU(j);                     % Get knot j from KnotU
20          temp           = Node_Coords(noCPsX-1, p, KnotU, noCPsY-1,...
21                             q, KnotV, CParray1, Dimn, xi, eta);
22                                             % See, below to find
                                                  Node_Coords function
23          Node(count,1) = temp(1)/temp(3); % Store knot in xi
24          Node(count,2) = temp(2)/temp(3); % Store knot in eta
25          count          = count + 1;        % Update counter var.
26      end
27  end
28
29
30  % Building the projected Q1 mesh
31  Q1_mesh   = zeros(nelU*nelV,4);            % Initialize the mesh array
32  for i = 1: nelV                            % Loop over element in eta
33      for j = 1: nelU                        % Loop over element in xi
34          Q1_mesh( (nelU*(i-1)+ j), :) = [nelU*(i-1)+j+(i-1),...
35          nelU*(i-1)+j+1+(i-1), nelU*(i-1)+j+1+(i-1)+noKnotU, ...
36          nelU*(i-1)+j+(i-1)+noKnotU];       % Generating Q1_mesh
37      end
38  end
39
40
41  %------------------------------
42  % Node_coords function which is used for construction of Q1_mesh, see
43  % line 20 of this Listing
44  %------------------------------
45  function S = Node_Coords(n1, p1, U1, m1, q1, V1, P1, dim1, u1, v1)
46  uspan = findspan(n1,p1,u1,U1);          % Find span of knot u along xi
47  vspan = findspan(m1,q1,v1,V1);          % Find span of knot v along eta
48  Nu     = basisfun(uspan1,u1,p1,U1);     % Compute basis fun. at uspan
49  Nv     = basisfun(vspan1,v1,q1,V1);     % Compute basis fun. at vspan
50  uind = uspan-p1;                        % Find no. CPs along xi in e
51  S     = zeros(1,dim1);                  % Initiazie the CPs nodal array
52  for l=0:q1                              % Loop over order along eta
53      temp = zeros(1,dim1);               % Initialize a temp variable
54      vind = vspan-q1+l;                  % Find no. CPs along eta in e
55      for i=0:p1                          % Loop over order along xi
56          CP   = P(uind+i+1 + vind*(n1+1),:); % Get proj nodes in an elem
57          temp = temp + Nu(i+1)*CP;       % Update the temp for an elem
58      end
59      S = S + Nv(l+1)*temp;               % Proj node array for an elem
60  end
```

Appendix D

See Table 2.

Table 2 Differences between the IGA and FEA techniques [3, 4]

Features	IGA	FEA
Geometrical approximations	Exact geometry is utilized in analysis	Approximated form of geometry is used
Construction of discretized geometry	Combination of control points array and knot vectors	Node points array
Basis	CAD functions, most popularly NURBS basis	Lagrangian and Hermite polynomials
Non-negativity	Always positive basis functions	Not necessarily always positive
Interpolation	NURBS basis do not interpolate control points	FE polynomials interpolate nodes
Continuity at inter-element boundary	C^{p-1-k} at a knot point, $k = 0$ if knot is not repeated	C^0, always fixed
Variation diminishing property	A monotone fit is obtained for discontinuous data	Fitting curve oscillates
Quantity of interest, e.g., displacement, strain, or stress fields	At control points	At nodes
Neumann and homogeneous Dirichlet BCs	Enforced directly at control points	Enforced at nodes
Inhomogeneous Dirichlet BCs	Special treatment procedure is required	Enforced directly at nodes
Computation of stress or strain fields	Directly at the control points	A recovery technique is used

References

1. O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 7th edn. (Butterworth-Heinemann, Oxford, 2013)
2. L. De Lorenzis, P. Wriggers, T.J.R. Hughes, Isogeometric contact: a review. GAMM Mitteilungen **37**(1), 85–123 (2014)
3. T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. **194**(39–41), 4135–4195 (2005)
4. J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA* (Wiley, New York, 2009)
5. J.A. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations. Comput. Methods Appl. Mech. Eng. **195**(41–43), 5257–5296 (2006)
6. W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization. Comput. Methods Appl. Mech. Eng. **197**(33–40), 2976–2988 (2008)
7. P. Kagan, A. Fischer, P.Z. Bar-Yoseph, New B-spline finite element approachfor geometrical design and mechanical analysis. Int. J. Numer. Meth. Eng. **41**(3), 435–458 (1998)
8. F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. Int. J. Numer. Meth. Eng. **47**(12), 2039–2072 (2000)
9. C.J. Corbett, R.A. Sauer, NURBS-enriched contact finite elements. Comput. Methods Appl. Mech. Eng. **275**, 55–75 (2014)
10. Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method. J. Comput. Phys. **229**(9), 3402–3414 (2010)
11. Y. Bazilevs, V.M. Calo, T.J.R. Hughes, Y. Zhang, Isogeometric fluid-structure interaction: theory, algorithms, and computations. Comput. Mech. **43**(1), 3–37 (2008)
12. D.J. Benson, Y. Bazilevs, E. De Luycker, M.C. Hsu, M. Scott, T.J.R. Hughes, T. Belytschko, A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM. Int. J. Numer. Methods Eng. **83**(6), 765–785 (2010)
13. S.A. Kalam, V.K. Rayavarapu, R.J. Ginka, Impact behaviour of soft body projectiles. J. Inst. Eng. (India): Ser. C **99**(1), 33–44 (2017)
14. A. Saha, A. Das, A. Karmakar, First-ply failure analysis of delaminated rotating composite conical shells: a finite element approach. J. Inst. Eng. (India): Ser. C (2017). https://doi.org/10.1007/s40032017-0373-y
15. H. Pathak, A. Singh, I.V. Singh, Numerical simulation of 3D thermo-elastic fatigue crack growth problems using coupled FE-EFG approach. J. Inst. Eng. (India): Ser. C **98**(3), 295–312 (2017)
16. P. Wang, J. Xu, J. Deng, F. Chen, Adaptive isogeometric analysis using rational PHT-splines. Comput. Aided Des. **43**(11), 1438–1448 (2011)
17. T. Dokken, T. Lyche, K.F. Pettersen, Polynomial splines over locally refined box-partitions. Comput. Aided Geom. Des. **30**(3), 331–356 (2013)
18. A.V. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis. Comput. Methods Appl. Mech. Eng. **200**(49–52), 3554–3567 (2011)
19. M.J. Borden, M.A. Scott, J.A. Evans, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS. Int. J. Numer. Meth. Eng. **87**(1–5), 15–47 (2011)
20. S. Hartmann, D.J. Benson, and D. Lorenz, About Isogeometric analysis and the new NURBS-based Finite Elements in LS-DYNA, in 8th European LS-DYNA User Conference, 2016

21. A. D. T. Elguedj and H. A. Akhras. An implementation of NURBS based Isogeometric anlysis for Abaqus. (2012) http://abqnurbs.insa-lyon.fr. Accessed 21 Nov 2017
22. S. Khakalo, J. Niiranen, Isogeometric analysis of higher-order gradient elasticity by user elements of a commercial finite element software. Comput. Aided Des. **82**, 154–169 (2017)
23. A.V. Vuong, C. Heinrich, B. Simeon, ISOGAT: a 2D tutorial Matlab code for isogeometric analysis. Comput. Aided Geom. Des. **27**(8), 644–655 (2010)
24. D. Rypl, B. Patzák, From the finite element analysis to the isogeometric analysis in an object oriented computing environment. Adv. Eng. Softw. **44**(1), 116–125 (2012)
25. B. Jüttler, U. Langer, A. Mantzaflaris, S.E. Moore, W. Zulehner, Geometry + simulation modules: implementing isogeometric analysis. PAMM **14**(1), 96–962 (2014)
26. V.P. Nguyen, C. Anitescu, S.P.A. Bordas, T. Rabczuk, Isogeometric analysis: an overview and computer implementation aspects. Math. Comput. Simul. **117**, 89–116 (2015)
27. S. Gondegaon, H.K. Voruganti, Static structural and modal analysis using isogeometric analysis. J. Theor. Appl. Mech. **46**(4), 36–75 (2016)
28. C. de Falco, A. Reali, R. Vázquez, GeoPDEs: a research tool for isogeometricanalysis of PDEs. Adv. Eng. Softw. **42**(12), 1020–1034 (2011)
29. L. Dalcin, N. Collier, P. Vignal, A.M.A. Côrtes, V.M. Calo, PetIGA: a framework for high-performance isogeometric analysis. Comput. Methods Appl. Mech. Eng. **308**, 151–181 (2016)
30. U. S. Dixit. Finite element methods for engineers. CENGAGE Learning Asia, 2009
31. L. Piegl, W. Tiller, *The NURBS book. Monographs in Visual Communication* (Springer, Berlin, 2012)
32. M. Spink. NURBS toolbox by D. M. Spink. (2010) http://in.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink/. Accessed 12 May 2015
33. M. Spink, D. Claxton, F. de Carlo, and R. Vázquez. NURBS Toolbox. (2015) http://octave.sourceforge.net/nurbs. Accessed 21 Nov 2015
34. R. Vázquez, A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0. Comput. Math Appl. **72**(3), 523–554 (2016)
35. E. De Luycker, D.J. Benson, T. Belytschko, Y. Bazilevs, M.C. Hsu, X-FEM in isogeometric analysis for linear fracture mechanics. Int. J. Numer. Meth. Eng. **87**(6), 541–565 (2011)
36. V.P. Nguyen, P. Kerfriden, M. Brino, S.P.A. Bordas, E. Bonisoli, Nitsche's method for two and three dimensional NURBS patch coupling. Comput. Mech. **53**(6), 1163–1182 (2014)