Isogeometric analysis: an overview and computer implementation aspects

Vinh Phu Nguyen[a,1,*], Stéphane P.A. Bordas[a,2], Timon Rabczuk[b,3]

[a] *School of Engineering, Institute of Mechanics and Advanced Materials, Cardiff University, Queen's Buildings, The Parade, Cardiff CF24 3AA*

[b] *Institute of Structural Mechanics, Bauhaus-Universität Weimar, Marienstraße 15 99423 Weimar*

Abstract

Isogeometric analysis (IGA) represents a recently developed technology in computational mechanics that offers the possibility of integrating methods for analysis and Computer Aided Design (CAD) into a single, unified process. The implications to practical engineering design scenarios are profound, since the time taken from design to analysis is greatly reduced, leading to dramatic gains in efficiency. The tight coupling of CAD and analysis within IGA requires knowledge from both fields and it is one of the goals of the present paper to outline much of the commonly used notation. In this manuscript, through a clear and simple Matlab® implementation, we present an introduction to IGA applied to the Finite Element (FE) method and related computer implementation aspects. Furthermore, implementation of the extended IGA which incorporates enrichment functions through the partition of unity method (PUM) is also presented, where several examples for both two-dimensional and three-dimensional fracture are illustrated. The open source Matlab® code which accompanies the present paper can be applied to one, two and three-dimensional problems for linear elasticity, linear elastic fracture mechanics, structural mechanics (beams/plates/shells including large displacements and rotations) and Poisson problems with or without enrichment. The Bézier extraction concept that allows FE analysis to be performed efficiently on T-spline geometries is also incorporated. The article includes a summary of recent trends and developments within the field of IGA.

Keywords: isogeometric analysis, linear elasticity, Matlab®, NURBS, finite elements, partition of unity, enrichment, 3D cracks, boundary conditions, CAD, large displacements and rotations, shells

*Corresponding author

[1] nguyenpv@cardiff.ac.uk

[2] stephane.bordas@alum.northwestern.edu

[3] timon.rabczuk@uni-weimar.de

1. Introduction

1.1. Underlying concepts of isogeometric analysis

The predominant technology that is used by CAD to represent complex geometries is the Non-Uniform Rational B-spline (NURBS). This allows certain geometries to be represented exactly that are only approximated by polynomial functions, including conic and circular sections. There is a vast array of literature focused on NURBS (e.g. [1], [2]) and as a result of several decades of research, many efficient computer algorithms exist for their fast evaluation and refinement. The key concept outlined by Hughes et al. [3] was to employ NURBS not only as a geometry discretisation technology, but also as a discretisation tool for analysis, attributing such methods to the field of 'Isogeometric Analysis' (IGA). Since this seminal paper, a monograph dedicated entirely to IGA has been published [4] and applications can now be found in several fields including structural mechanics, solid mechanics, fluid mechanics and contact mechanics. We give in this section an overview of some of these recent developments while outlining the benefits and present shortcomings of IGA. It should be emphasized that the idea of using CAD technologies in finite elements dates back at least to [5, 6] where B-splines were used as shape functions in FEM. In addition, similar methods which adopt subdivision surfaces have been used to model shells [7]. We also review some of the recent attempts at simplifying the CAD-FEA integration by separating boundary and domain discretisations.

1.2. Applications

In contact formulations using conventional geometry discretisations, the presence of faceted surfaces can lead to jumps and oscillations in traction responses unless very fine meshes are used. The benefits of using NURBS over such an approach are evident, since smooth contact surface are obtained, leading to more physically accurate contact stresses. Recent work in this area includes [8, 9, 10, 11, 12].

IGA has also shown advantages over traditional approaches in the context of optimisation problems [13, 14, 15, 16] where the tight coupling with CAD models offers an extremely attractive approach for industrial applications. Another attractive class of methods include those that require only a boundary discretisation, creating a truly direct coupling with CAD. Isogeometric boundary element methods for elastostatic analysis were presented in [17, 18], demonstrating that mesh generation can be completely circumvented by using CAD discretisations for analysis.

Shell and plate problems are another field where IGA has demonstrated compelling benefits over conventional approaches [19, 20, 21, 22, 23, 24, 25]. The smoothness of the NURBS basis functions allows for a straightforward

construction of plate/shell elements. Particularly for thin shells, rotation-free formulations can be easily constructed [20, 26]. Note that for multi-patch NURBS surfaces, rotation-free IGA elements require special treatment at patch boundaries where the basis functions are found to be C^0 continuous. Furthermore, isogeometric plate/shell elements exhibit much less pronounced shear-locking compared to standard FE plate/shell elements. Elements with smooth boundaries such as circular and cylindrical elements were successfully constructed using the IGA concept [27, 28].

The smoothness of NURBS basis functions is attractive for analysis of fluids [29, 30, 31] and for fluid-structure interaction problems [32, 33]. In addition, due to the ease of constructing high order continuous basis functions, IGA has been used with great success in solving PDEs that incorporate fourth order (or higher) derivatives of the field variable such as the Hill-Cahnard equation [34], explicit gradient damage models [35] and gradient elasticity [36]. The high order NURBS basis has also found potential applications in the Kohn-Sham equation for electronic structure modeling of semiconducting materials [37].

NURBS provide advantageous properties for structural vibration problems [38, 39, 40, 41] where k-refinement (unique to IGA) has been shown to provide more robust and accurate frequency spectra than typical higher-order FE p-methods. Particularly, the optical branches of frequency spectra, which have been identified as contributors to Gibbs phenomena in wave propagation problems (and the cause of rapid degradation of higher modes in the p-version of FEM), are eliminated. However when lumped mass matrices were used, the accuracy is limited to second order for any basis order. High order isogeometric lumped mass matrices are not yet available. The mathematical properties of IGA were studied in detail by Evans et al.[42].

The isogeometric concept has also spread to the field of meshfree methods such as [43, 44] in which spline-based meshfree methods were presented. Industrial applications of IGA have been presented in [45, 18] along with applications in experimental mechanics [46] where NURBS-based DIC (Digital Image Correlation) was shown to outperform standard FE DIC.

1.3. Shortcomings of NURBS and alternative geometry discretisations

NURBS are ubiquitous in CAD but are known to exhibit major shortcomings from a computational geometry standpoint. Perhaps the greatest difficulty encountered is the inability of NURBS to produce watertight geometries, often complicating mesh generation. From an analysis perspective, the tensor product structure of NURBS proves to be inefficient, caused by the global nature of refinement operations. In turn, this leads to inefficient error estimation

and adaptivity algorithms. One solution which has gathered momentum from both the computational geometry and analysis communities is the use of T-splines [47] which overcome the limitations of NURBS while retaining the familiar structure of NURBS algorithms. T-splines correct the deficiencies of NURBS by creating a single patch, watertight geometry which can be locally refined and coarsened. Buffa et al. [48] note that linear independence of the T-spline basis functions is not guaranteed on generic T-meshes leading to the definition of analysis-suitable T-splines [49], a mildly restricted subset of T-splines which meet the demands of both design and analysis. Utilisation of T-splines in an IGA framework has been illustrated in [50, 51], and by adopting a Bézier extraction process, Scott et al.[52] showed that T-splines can be incorporated efficiently into existing FE codes.

Alternatives to T-splines include polycube splines [53], PHT-splines [54] and LR-splines [55]. PHT-splines (polynomial spline over hierarchical T-meshes) have been extended to rational splines and applied in [56, 57] to problems in elasticity for continua and thin structures. Adaptive refinement with PHT-splines is particularly simple. Although T-splines allow for local adaptive refinement, the complexity of knot insertion under adaptive refinement is complex, particularly in 3D. However, we note that research is currently being pursued on hierarchical T-spline refinement algorithms that address this issue.

Another direction of IGA research includes hierarchical B-splines [58, 59, 60] and unstructured Powell-Sabin splines [61]. The hierarchical B-splines finite cell method [59] furnishes a seamless CAD-FEA integration for very complex geometries. We refer also to [62] for IGA combined with finite element based local refinement capabilities. Different subdivision surface techniques (Catmull-Clark, Loop) have also been utilized for solid and shell modeling [63, 64].

In computer aided geometric design, patching multiple NURBS parameterizations to form complex topologies is far from trivial if certain continuity requirements are to be maintained. Trimming techniques provide a promising alternative for representing complex NURBS domains. In [65], a trimmed surface based analysis framework has been proposed where NURBS-enhanced FEM [66, 67] was applied to define a suitable integration domain within parameter space. In a recent contribution [68], the authors presented an alternative method to handle trimmed NURBS geometries.

1.4. Discontinuities and fracture

IGA has been applied to cohesive fracture [69], outlining a framework for modeling debonding along material interfaces using NURBS and propagating cohesive cracks using T-splines. The method relies upon the ability to specify the continuity of NURBS and T-splines through a process known as knot insertion. As a variation of the eXtended Finite

Element Method (XFEM) [70], IGA was applied to Linear Elastic Fracture Mechanics (LEFM) using the partition of unity method (PUM) to capture two dimensional strong discontinuities and crack tip singularities efficiently [71, 72]. The method is usually referred to as XIGA (eXtended IGA). In [73] an explicit isogeometric enrichment technique was proposed for modeling material interfaces and cracks exactly. Note that this method is contrary to PUM-based enrichment methods which define cracks implicitly.

A phase field model for dynamic fracture was presented in [74] using adaptive T-spline refinement to provide an effective method for simulating fracture in three dimensions. In [75] high order B-splines were adopted to efficiently model delamination of composite specimens and in [76], an isogeometric framework for two and three dimensional delamination analysis of composite laminates was presented where the authors showed that using IGA can significantly reduce the usually time consuming pre-processing step in generating FE meshes (solid elements and cohesive interface elements) for delamination computations. A continuum description of fracture using explicit gradient damage models was also studied using NURBS [35].

1.5. Alternatives to IGA

Other techniques which integrate CAD and analysis include the use of subdivision surfaces to model shells [7], NURBS-enhanced finite elements [66, 67] and NURBS for BEM shape optimisation [77]. Immersed boundary methods [78] (and references therein), the finite cell method [59, 79] and the structured XFEM [80, 81] are yet other alternatives which aim to combine analysis and design technologies. In general, it is found that for CAD and analysis technologies to work seamlessly together, the underlying discretisation must either be directly compatibile or easily converted between the two.

IGA has offered significant advances towards the goal of a unified design and analysis framework, but much research is still needed before this goal is realised. There are several indications of the future promise of IGA for industrial design but ultimately, the litmus test of success for IGA will be whether the approach is widely adopted by industry.

1.6. Computational aspects

Some major computational aspects of IGA which have been studied so far include (i) locking issues, (ii) sensitivity to mesh distortion, (iii) impact of high continuity of NURBS on direct solvers, (iv) collocation methods, (v) competing demands of analysis and computational geometry discretisations, (vi) construction of trivariate solids from given bivariate surface representations and (vii) optimal quadrature rules.

(i) Although the smoothness of NURBS basis functions reduces to some extent the locking phenomena for constrained problems such as incompressible media, thin-walled structures, NURBS-based FEs are not locking free [82, 83, 84, 85, 86]. Existing locking removal techniques in standard FEMs were successfully adapted to IGA such as the Discrete Strain Gap method [84], the F/B-bar method [83, 85], the enhanced assumed strain method [86] and mixed formulations [87].

(ii) The effect of mesh distortion on the performance of IGA for solid mechanics was discussed in [88] in which it was found that higher-order NURBS functions are able to somewhat alleviate the impact of the distortions.

(iii) The high order continuity offered by NURBS has a negative impact on the performance of direct solvers as pointed out in [89]. The authors found that for a fixed number of unknowns and basis degree, a higher degree of continuity drastically increases the CPU time and RAM needed to solve the problem when using a direct solver.

(iv) In an attempt to compete with low order FEs with one-point quadrature that are extensively used in industrial applications, isogeometric collocation methods were developed [90, 91].

(v) Due to the fact that meshes in an isogeometric framework are defined by the parametrisation of the object of interest, the quality of the geometry parametrisation plays an important role in ensuring mesh quality. This issue has, however, been addressed by only a few researchers [92, 93, 94, 95, 96]. In particular, in [94], the authors proposed the concept of "analysis-aware geometry modeling".

(vi) In CAD, solids are defined as boundary surfaces in which the interior is not explicitly modeled. In FEA, a solid representation is necessary and therefore, the transition from CAD solids to FEA solids demands a step in which the CAD representations are converted to solid FEA representations. Initial developments have been reported in [97, 98, 99, 100, 101, 102]. Note that in this regard, the isogeometric boundary element method (IGABEM) can be considered a truly isogeometric method [17, 18, 103, 104] since BEM analysis requires only the definition of a boundary discretisation, completely defined by CAD.

(vii) Gaussian quadrature is not optimal for IGA. Research is currently focussed on optimal integration techniques such as that in [105, 106] in which (nearly) optimal quadrature rules have been presented.

1.7. Available implementations

Some implementation aspects of IGA were reported in [4] and more recently, an open source IGA Matlab® code was described in [107] with a restriction to 2D scalar PDEs. An excellent open source IGA code written in Matlab® is given in [108]. Incorporating IGA within an object-oriented C++ FE code was discussed in [109]. Implementation details for enriched formulations within an IGA framework are reported in [110] using commercial FE software. An IGA BEM code written in Matlab® was presented in [104]. Isogeometric analysis was also incorporated into FEAP [111, 87]. A high performance IGA code was given in [112] which is based on PETSc, the Portable, Extensible Toolkit for Scientific Computation. The Python programming language has also been adopted to implement IGA e.g., [113, 114].

1.8. Contributions and outline

In this paper, we present in detail an isogeometric (Bubnov-)Galerkin finite element method applied to two- and three-dimensional elasto-static solid/structural mechanics problems and traction-free crack problems. The discussion is confined to NURBS for the sake of simplicity. Although no new fundamental findings are presented the contribution of the paper are

- an overview of IGA, applications and recent developments are presented;
- implementation details for 1D/2D/3D solid mechanics and rotation-free plate elements are provided;
- implementation for 2D and 3D XIGA for traction-free cracks;
- visualization techniques for (X)IGA are discussed;
- techniques to impose Dirichlet boundary conditions including the penalty method, the Lagrange multipliers method, the least square projection method are implemented;
- different techniques to model discontinuities in the context of IGA are discussed;

where some implementations are not available in the literature.

The paper is structured as follows: Section 2 outlines B-spline and NURBS technology used to construct curves, surfaces and solids; the use of NURBS for discretisation within a finite element framework is treated in Section 3; implementation of an isogeometric finite element method for two-dimensional elasticity is described in Section 4 and an

extended isogeometric formulation based on the concept of the PUM is the subject of Section 5. A detailed description of our IGA Matlab code is given in Section 6 and Application of IGA to structural mechanics problems is presented in Section 7. Numerical examples including 2D/3D fracture mechanics and large deformation shell problems are given in Section 8.

1.9. Notation

We use lowercase indices to indicate a local index and uppercase indices to indicate a global index. We denote d_p and d_s as the number of parametric directions and spatial directions respectively. **Boldfont** is used to indicate matrices and vectors where the number of components is implied.

2. A brief introduction to B-splines/NURBS

As a precursor to Section 3, an understanding of the discretisation technology which underpins IGA is essential, since the beneficial properties which are found through analysis emanate directly from the underlying CAD basis functions. We give a brief outline of parametric functions, then state the terminology and basis function definitions which allow curves, surfaces and solids to be represented by B-splines and NURBS. The algorithms which define B-splines and NURBS in the present Matlab® code are based on those given in [1] in which explicit implementations are given for all algorithms.

2.1. Parametric representation of geometry

What is fundamental to all the discretisation technology used in the present paper (and the majority of technology in the CAGD community), is the representation of geometry through *parametric functions*. These define a mapping from a given parameter to the desired geometry. We can imagine that as we move through the parameter domain, the parametric function 'sweeps' out the desired shape. For example, if we consider the case of a circle of radius 1, the implicit form of this equation is given by

$$x^2 + y^2 = 1 \tag{1}$$

or alternatively, if we define a mapping $f : [0, 2\pi] \to \mathbb{R}^2$, the parametric form of the same circle is given by

$$f(t) = (\cos t, \sin t)\,. \tag{2}$$

This is found to be much more conducive for graphical implementation. To see this, consider the task of plotting the circle through both Eq. (1) and Eq. (2). In the case of the latter, by simply determining t at a discrete set of points in the interval $[0, 2\pi]$, the desired result is obtained. In addition, many algorithms which perform geometrical transformations become much simpler when parametric functions are used. Both B-splines and NURBS are based on parametric functions.

2.2. B-splines

The demands of Computer Aided Geometrical Design place certain requirements on the types of discretisation that can be used to represent geometrical objects where, for example, in the case of car design, surfaces of G^2 continuity[4] are required to avoid unnatural reflections. Additional requirements include local control of geometrical features, the ability to apply refinement algorithms and numerical stability of high order curves. It is found that representations of objects using Bézier curves or Lagrange polynomials do not meet many of these requirements and alternatives are sought. The CAGD community have settled on the use of B-spline-based technology since it is found to provide many of the desired properties for interactive geometrical design, realised through the properties of the underlying B-spline basis functions.

B-splines can be considered as a mapping from *parametric space* $\hat{\Omega} \subset \mathbb{R}^{d_p}$ to *physical space* $\Omega \subset \mathbb{R}^{d_s}$ [5]. In this sense, a B-spline can be considered to 'sweep' out a curve, surface or volume as we move through the range of parameter values. In the present work we define coordinates in parameter space as $\boldsymbol{\xi} = (\xi, \eta, \zeta) = (\xi^1, \xi^2, \xi^3)$ and coordinates in physical space as $\mathbf{x} = (x, y, z) = (x^1, x^2, x^3)$. These are simplified accordingly in the case of one- and two-dimensional problems. What remains is to determine the particular form of the mapping $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$.

To construct a B-spline, a *knot vector* must be specified and is defined as an ordered set of increasing parameter values $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$, $\xi_i \leq \xi_{i+1}$ where ξ_i is the ith knot, n is the number of basis functions and p is the polynomial order. The knot vector divides the parametric space into intervals usually referred to as *knot spans* with the number of coincident knots for a particular knot value referred to as a knot with a certain *multiplicity* k. That is, a knot has a multiplicity k if it is repeated k times in the knot vector. Most commonly, *open* knot vectors are used where the first and last knots have a multiplicity $k = p + 1$. Appendix A outlines alternative knot vector notations

[4] Appendix B outlines the differences between C^k and G^k continuity

[5] Further discussion of these spaces in given in Section 3.1.

which remove redundant information, but it can be assumed that the above knot vector notation is used throughout the present work.

An important property of B-splines formed from open knot vectors is that they are interpolatory at their start and end points, greatly facilitating the imposition of boundary conditions for analysis. In geometrical terms, this corresponds to a B-spline that coincides with its start and end control points. A *uniform knot vector* is associated to evenly distributed knots. Otherwise it is classified as a *non-uniform knot vector*.

Knot vectors are not commonly used by CAD designers, and in most CAD software the ability to modify knot vectors is not provided. It is much more common to tailor the geometry through modification of the polynomial order, control points and weightings.

2.2.1. B-spline basis functions

Given a knot vector Ξ, the associated set of B-spline basis functions $\{N_{i,p}\}_{i=1}^{n}$ are defined recursively by the Cox-de-Boor formula, starting with the zeroth order basis function ($p = 0$)

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and for a polynomial order $p \geq 1$

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{4}$$

in which fractions of the form $0/0$ are defined as zero.

Some salient properties of B-spline basis functions include: (1) they constitute a partition of unity, (2) each basis function is non-negative over the entire parametric domain, (3) they are linearly independent, (4) the support of a B-spline basis function $N_{i,p}$ is given by $p + 1$ knot spans denoted by $[\xi_i, \xi_{i+p+1}]$, (5) they exhibit C^{p-m_i} continuity across knot ξ_i where m_i is the multiplicity of knot ξ_i and (6) B-spline basis functions in general are not interpolatory, equivalent to saying that the Kronecker-delta property is not guaranteed at control points. This last point requires careful treatment when imposing non-homogeneous Dirichlet boundary conditions, and is discussed later in Section 4.2.

To demonstrate the effect of alternative knot vectors, Fig. 1 illustrates a set of quadratic ($p = 2$) B-spline basis

functions for a uniform knot vector. Fig. 2 illustrates a corresponding set of basis functions for an open, non-uniform knot vector. Of particular note is the interpolatory nature of the basis function at each end of the interval created through an open knot vector, and the reduced continuity at $\xi = 4$ due to the presence of the location of a repeated knot where C^0 continuity is attained. Elsewhere, the functions are C^1 continuous (C^{p-1}). The ability to control continuity by means of knot insertion is particularly useful for modeling discontinuities such as cracks or material interfaces [69, 76].

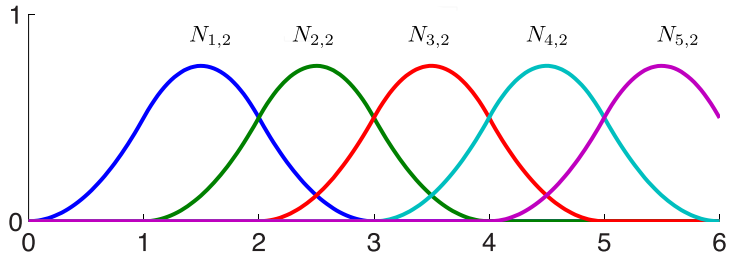


Figure 1: Quadratic B-spline basis functions defined for the uniform knot vector $\Xi = \{0, 1, 2, 3, 4, 5, 6\}$.

During implementation, the derivatives of B-splines are required in both CAD and analysis to compute quantities such as tangent and normal vectors and field variable derivatives. The first derivative of a B-spline basis function is computed recursively from lower order basis functions as

$$\frac{d}{d\xi}N_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i}N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}}N_{i+1,p-1}(\xi). \tag{5}$$

Derivatives of higher order can be found in [1].

2.3. Refinement algorithms

A feature which is essential for both computational geometry and analysis is the ability to successively refine discretisations to allow intricate geometries to be modelled or to capture rapid variations in the field solution. A

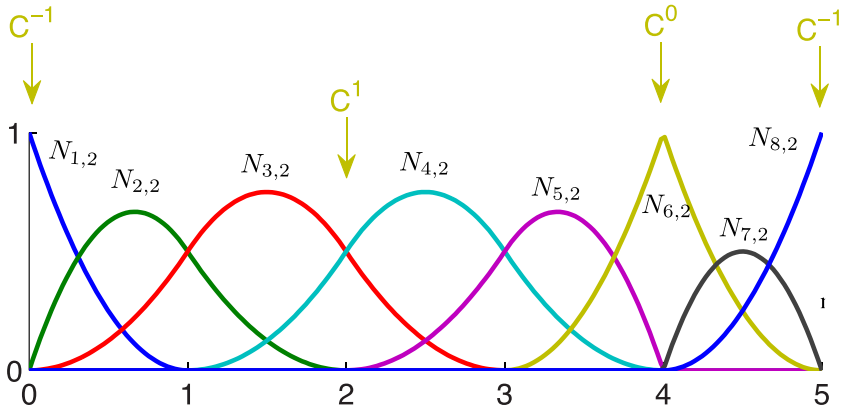


Figure 2: Quadratic B-spline basis functions defined for the open, non-uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$. Note the flexibility in the construction of basis functions with varying degrees of regularity.

significant advantage of adopting B-spline-based functions as a discretisation tool is the ability to apply a variety of refinement algorithms in a simple manner. We restrict ourselves to a brief discussion of the available algorithms, with a more detailed treatment given in [4].

B-spline refinement algorithms in the context of isogeometric analysis can be assigned to one of three types:

1. **Knot insertion** is the process whereby additional knots are inserted into the knot vectors thereby creating additional knot intervals or elements, in the context of analysis. This is directly analogous to h-refinement seen in the conventional FEM.

2. **Degree elevation** is the process of raising the order of the underlying basis, directly analogous to p-refinement in the conventional FEM.

3. **k-refinement** is unique to IGA and consists of a combined process of degree elevation and knot insertion. These processes are not commutative and therefore the order in which these refinements are applied will change the final basis. k-refinement first applies degree elevation proceeded by knot insertion, offering a reduction in degrees of freedom over its counterpart [3].

2.4. B-spline curves

Given a set of B-spline basis functions $\{N_{A,p}\}_{A=1}^{n}$ and a set of *control points* $\{\mathbf{P}_A\}_{A=1}^{n}$ where $\mathbf{P}_A \in \mathbb{R}^{d_s}$, we can construct a piecewise-polynomial B-spline curve as

$$\mathbf{C}(\xi) = \sum_{A=1}^{n} \mathbf{P}_A N_{A,p}(\xi). \tag{6}$$

An example quadratic B-spline curve is shown in Fig. 3 constructed from a uniform open knot vector. Its associated *control polygon*, constructed by piecewise linear interpolation of the control points is also shown. A B-spline curve inherits all of the continuity properties of its underlying basis and, as mentioned previously, the use of open knots ensures that the first and last points are interpolatory. Note that the tangent to the curve at the first control point is defined by the first two control points. This property is exploited in rotation-free element formulations for thin-walled structures (section 7).

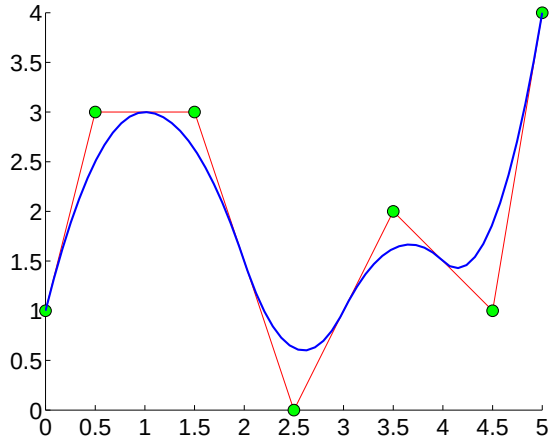


Figure 3: A quadratic ($p = 2$) B-spline curve with a uniform open knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}$. Control points are denoted by filled circles and the control polygon is denoted by the red line. Note that the curve is at least C^1 continuous everywhere except at the extremities where, due to the presence of repeated knots, C^{-1} continuity is obtained.

2.5. B-spline surfaces and volumes

Given two knot vectors for each parametric direction $\Xi^1 = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ and $\Xi^2 = \{\eta_1, \eta_2, \ldots, \eta_{m+q+1}\}$, and a control net $\mathbf{P}_{i,j} \in \mathbb{R}^{d_s}$, a tensor-product B-spline surface is defined as

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) \mathbf{P}_{i,j}, \tag{7}$$

where $N_{i,p}(\xi)$ and $M_{j,q}(\eta)$ are the univariate B-spline basis functions of order p and q corresponding to knot vectors Ξ^1 and Ξ^2, respectively. Defining a global index as

$$A = n(j-1) + i, \tag{8}$$

Eq. (7) can be rewritten in a more compact form as

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{A=1}^{n \times m} \mathbf{P}_A N_A^{p,q}(\boldsymbol{\xi}), \tag{9}$$

in which $N_A^{p,q}$ is a bivariate B-spline basis function defined as $N_A^{p,q}(\boldsymbol{\xi}) = N_{i,p}(\xi) M_{j,q}(\eta)$. Figs. 4 illustrates an example bicubic B-spline basis function.

The extension to B-spline volumes is straightforward, where a trivariate basis is formed through a tensor product of B-spline basis functions as

$$\mathbf{V}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \mathbf{P}_{i,j,k}, \tag{10}$$

or, by defining a global index A through

$$A = (n \times m)(k-1) + n(j-1) + i, \tag{11}$$

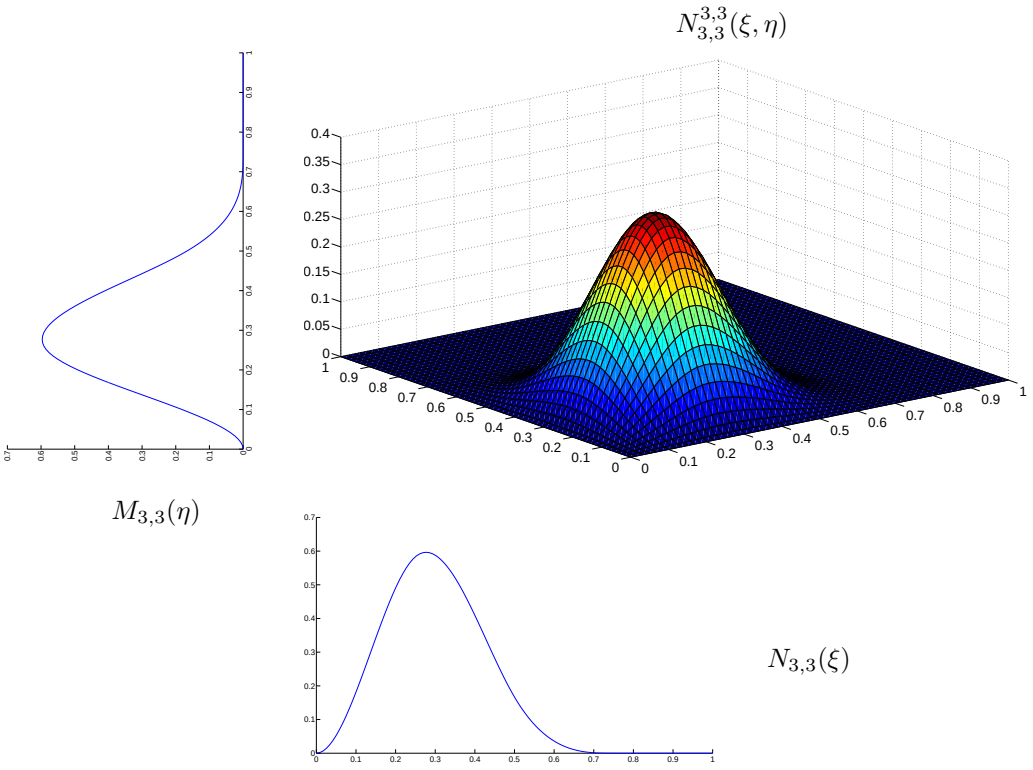


Figure 4: A bivariate cubic B-spline basis function for knot vectors $\Xi^1 = \Xi^2 = \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}$.

a simplified form of Eq. (10) can be written as

$$\mathbf{V}(\boldsymbol{\xi}) = \sum_{A=1}^{n\times m\times l} \mathbf{P}_A N_A^{p,q,r}(\boldsymbol{\xi}). \tag{12}$$

To compute derivatives of B-spline surfaces and volumes, the chain rule is applied to (5) with a detailed treatment of this topic provided in [1].

2.6. NURBS

B-splines are convenient for free-form modelling, but they lack the ability to exactly represent some simple shapes such as circles and ellipsoids. This is why today, the *de facto* standard technology in CAD is a generalisation of B-splines referred to as NURBS (Non-Uniform Rational B-Splines). NURBS are formed through rational functions of B-splines, forming a superset of B-splines. They inherit all the favourable properties of B-splines and are favoured over their counterpart due to their ability to form exact representations of conic sections such as spheres, ellipsoids, paraboloids and hyperboloids. In addition, there exist efficient algorithms for their evaluation and refinement.

2.6.1. NURBS basis functions

NURBS basis functions are defined as

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi)w_i}{W(\xi)} = \frac{N_{i,p}(\xi)w_i}{\sum_{\hat{i}=1}^{n} N_{\hat{i},p}(\xi)w_{\hat{i}}} \tag{13}$$

where $\{N_{i,p}\}_{i=1}^{n}$ is the set of B-spline basis functions of order p and $\{w_i\}_{i=1}^{n}, w_i > 0$ is the set of NURBS weights. Selecting appropriate weights permits the description of many different types of curves including polynomials and circular arcs. For the special case in which all weights are equal, the NURBS basis reduces to the B-spline basis. NURBS weights for certain simple geometries are given in [1], but in general, weights are user-defined through CAD packages such as Rhino [115].

The first derivative of a NURBS basis function $R_{i,p}$ is computed using the quotient rule as

$$\frac{d}{d\xi}R_{i,p}(\xi) = w_i\frac{N'_{i,p}(\xi)W(\xi) - N_{i,p}(\xi)W'(\xi)}{W(\xi)^2}, \tag{14}$$

where $N'_{i,p}(\xi) \equiv \frac{d}{d\xi} N_{i,p}(\xi)$ and

$$W'(\xi) = \sum_{\hat{i}=1}^{n} N'_{\hat{i},p}(\xi) w_{\hat{i}}. \tag{15}$$

2.6.2. NURBS curves, surfaces and volumes

In a similar fashion to B-spline curves, the NURBS curve associated with a set of control points and weights $\{\mathbf{P}_A, w_A\}_{A=1}^{n}$ and basis functions $\{R_{A,p}\}_{A=1}^{n}$ is defined as

$$\mathbf{C}(\xi) = \sum_{A=1}^{n} \mathbf{P}_A R_{A,p}(\xi). \tag{16}$$

NURBS surfaces are constructed from a linear combination of bivariate NURBS basis functions, control points $\mathbf{P}_{i,j} \in \mathbb{R}^{d_s}$ and weights $w_{i,j} > 0$ as

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{P}_{i,j} R_{i,j}^{p,q}(\xi, \eta), \tag{17}$$

where the bivariate NURBS basis functions are defined as

$$R_{i,j}^{p,q}(\xi, \eta) = \frac{N_i(\xi) M_j(\eta) w_{i,j}}{\sum_{\hat{i}=1}^{n} \sum_{\hat{j}=1}^{m} N_{\hat{i}}(\xi) M_{\hat{j}}(\eta) w_{\hat{i},\hat{j}}}. \tag{18}$$

Alternatively, using the mapping defined by Eq. (8), Eq. (17) can be written more succinctly as

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{A=1}^{n \times m} \mathbf{P}_A R_A^{p,q}(\boldsymbol{\xi}). \tag{19}$$

NURBS volumes are constructed from control points $\mathbf{P}_{i,j,k} \in \mathbb{R}^{d_s}$, weights $w_{i,j,k} > 0$ as

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

where the trivariate NURBS basis functions $R_{i,j,k}^{p,q,r}$ are given by

$$R_{i,j,k}^{p,q,r}(\xi, \eta, \zeta) = \frac{N_i(\xi) M_j(\eta) P_k(\zeta) w_{i,j,k}}{\sum_{\hat{i}=1}^{n} \sum_{\hat{j}=1}^{m} \sum_{\hat{k}=1}^{l} N_{\hat{i}}(\xi) M_{\hat{j}}(\eta) P_{\hat{k}}(\zeta) w_{\hat{i},\hat{j},\hat{k}}}. \tag{21}$$

Using the mapping given by Eq. (11), Eq. (20) can also be written as

$$\mathbf{V}(\boldsymbol{\xi}) = \sum_{A=1}^{n\times m\times l} \mathbf{P}_A R_A^{p,q,r}(\boldsymbol{\xi}) \tag{22}$$

Remark 2.1. As mentioned in Section 1.6, CAD representations are usually composed of surface models or boundary-representations. Trivariate discretisations defined by (22) are not normally explicitly given, and therefore some pre-processing is required before domain based numerical methods such as the FEM can be applied. Except for the case of simple cases such as extruded-surface models and swept models, this task is far from trivial. This is currently an open research topic in IGA.

3. NURBS as a basis for analysis: isogeometric finite element formulation

Our attention now focusses on the use of B-splines and NURBS as a discretisation tool for analysis, outlining the core concepts of isogeometric analysis. In this section the important spaces and mappings are defined, followed by the isogeometric FEM formulation in which we use NURBS as a basis for analysis. The discussion is made more concrete through a one-dimensional example.

3.1. Relevant spaces

Familiarity must be gained with the spaces that are commonplace in isogeometric analysis and the relationships that exist between each. Those that are considered presently in the context of B-splines and NURBS include: index, parameter, physical and parent space.

3.1.1. Index space

Index space is formed through the specified knot vectors by giving each knot value a distinct coordinate, regardless of whether the knot is repeated or not. As an example, consider a NURBS patch defined through bivariate NURBS basis functions with knot vectors $\Xi^1 = \{0,0,0,1,2,3,3,3\}$, $\Xi^2 = \{0,0,1,1\}$ in each of the parametric directions ξ, η respectively. This will form the index space as illustrated in Fig. 5 where the presence of repeated knots leads to several regions of zero parametric area. Index space is often used during implementation[6], discarding elements of non-zero

[6]T-splines being one notable example, with the local basis function mesh directly analogous to index space

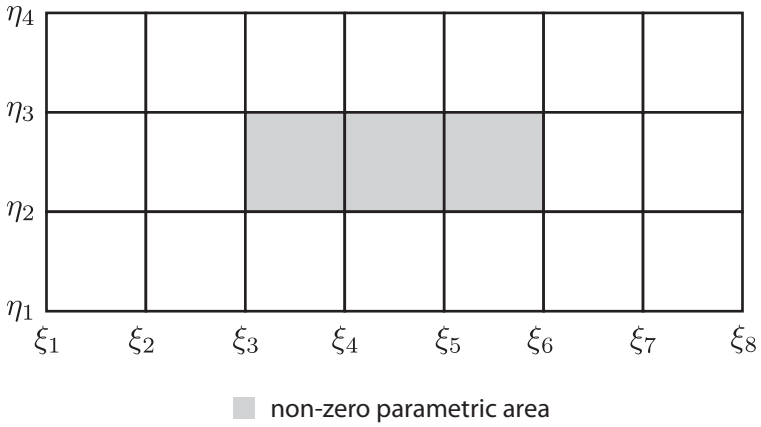


Figure 5: Creation of index space from knot vectors $\Xi^1 = \{0, 0, 0, 1, 2, 3, 3, 3\}$, $\Xi^2 = \{0, 0, 1, 1\}$ with non-zero parametric area highlighted.

parametric area, but in the present work we choose to only consider those elements that have a non-zero parametric area, obviating the need for index space.

3.1.2. Parametric space

Parametric space (sometimes referred to as the 'pre-image' of the NURBS mapping) is formed by considering only the non-zero intervals between knot values. For the knot vectors considered previously, the parametric space is illustrated in Fig. 6 which can subsequently be reduced to a unit square through appropriate normalisation. All parametric spaces can be reduced to a unit interval ($d_p = 1$), square ($d_p = 2$) or cube ($d_p = 3$) in this manner. We define the parametric space as $\hat{\Omega} \subset \mathbb{R}^{d_p}$ with a associated set of parametric coordinates $\boldsymbol{\xi} = (\xi, \eta, \zeta) = (\xi^1, \xi^2, \xi^3) \in \hat{\Omega}$ ($d_p = 3$). If normalisation is performed, $\hat{\Omega} = [0, 1]^{d_p}$.

Fig. 6 also reveals that regions bounded by knot lines with non-zero parametric area lead to a natural definition of element domains. More formally, a set $\mathcal{S} \subset \Xi$ of unique knot values can be defined as

$$\mathcal{S} = \{\xi_1, \xi_2, \cdots, \xi_{n_s}\} \quad \xi_i \neq \xi_{i+1} \text{ for } 1 \leq i \leq n_s - 1 \tag{23}$$

where n_s is the number of unique knot values. This is generalised to $\mathcal{S}^i \subset \Xi^i$ which represents the unique knot values

for each parametric direction $i = 1, 2, \ldots d_p$. Elements can now be defined in the general multivariate case as

$$\begin{aligned} \hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_j, \eta_{j+1}] \otimes [\zeta_k, \zeta_{k+1}] \quad & 1 \leq i \leq n_s^1 - 1, \\ & 1 \leq j \leq n_s^2 - 1, \\ & 1 \leq k \leq n_s^3 - 1, \\ & \xi_i \in \mathcal{S}^1,\ \eta_j \in \mathcal{S}^2,\ \zeta_k \in \mathcal{S}^3 \end{aligned} \tag{24}$$

where n_s^1, n_s^2 and n_s^3 represent the number unique knots in the ξ,η and ζ parametric directions respectively. This leads to a natural numbering scheme for elements over a patch as

$$e = k(n_s^2 - 1)(n_s^1 - 1) + j(n_s^1 - 1) + i. \tag{25}$$

Eq. (24) and (25) can be simplied accordingly for $d_p = 1, 2$.

3.1.3. Physical space

The B-spline and NURBS mappings of Eqs. (9) and (17) transform coordinates in parameter space to physical space $\Omega \subset \mathbb{R}^{d_s}$. For three-dimensional domains, we associate a coordinate system $\mathbf{x} = (x, y, z) = (x^1, x^2, x^3)$ for physical space, which appropriate modifications for one- and two-dimensional problems. Fig. 7 illustrates a NURBS mapping for the parametric space shown in Fig. 6 for an arbitrary set of control points and weights. The control grid (which defines the connectivity between control points) is also shown. The non-interpolatory nature of control points in the interior of the domain is evident, and represents a notable difference over conventional Lagrangian meshes.

3.1.4. Parent space

The previous three spaces are inherent to B-splines and NURBS, but for analysis to be performed we require the definition of an additional space, commonly referred to as parent space $\tilde{\Omega} = [-1, 1]^{d_p}$. This is required for the use of numerical integration routines which are often defined over the interval $[-1, 1]$. Parent space coordinates are denoted as $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}, \tilde{\eta}, \tilde{\zeta}) = (\tilde{\xi}^1, \tilde{\xi}^2, \tilde{\xi}^3)$ with corresponding simplifications for $d_p = 1, 2$.

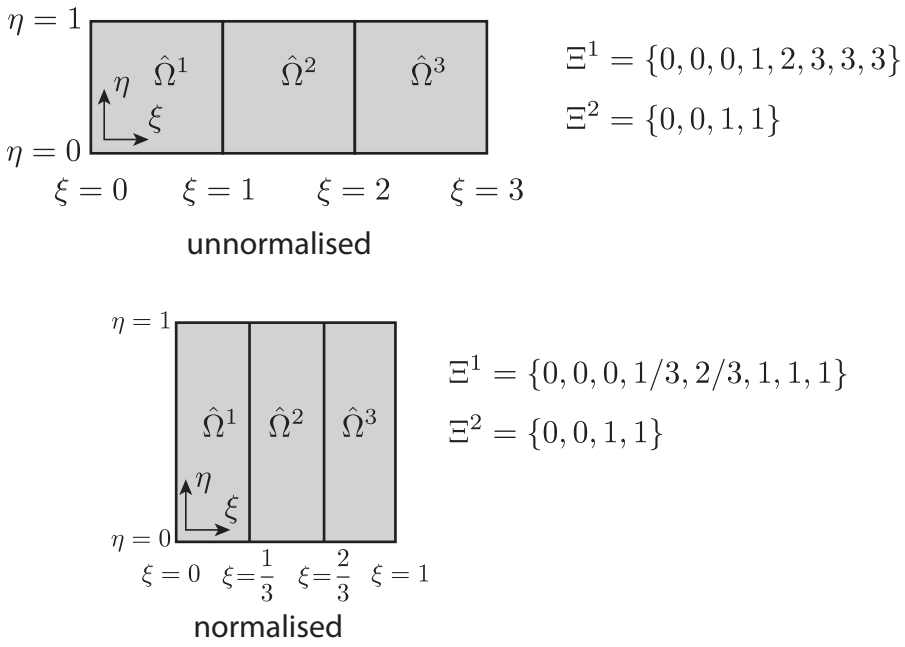


Figure 6: Parametric space defined by non-zero knot intervals. All parametric spaces defined for a B-spline or NURBS patch can be normalised to a unit interval, unit square, unit cube in 1D, 2D, 3D respectively. Knot lines provide a natural definition of element boundaries.

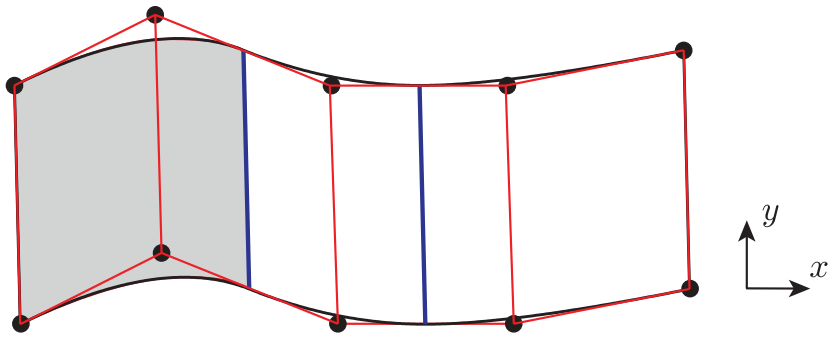


Figure 7: A 2D NURBS surface defined for knot vectors $\Xi^1 = \{0, 0, 0, 1, 2, 3, 3, 3\}$, $\Xi^2 = \{0, 0, 1, 1\}$. The control mesh is shown in red with control points denoted by black circles. Knot lines shown in blue indicate element boundaries.

3.2. Isogeometric formulation

Before outlining the details of isogeometric analysis in a FE context, it is instructive to consider the similarities and differences of IGA over conventional discretisation technology. Lagrangian basis functions are most commonly used to discretise both the geometry and unknown fields in an *isoparametric* fashion. In this way, exactly the same basis functions are used for both. For the majority of cases, the geometry is always approximated incurring a geometrical error which may lead to erroneous results, particularly for highly oscillatory problems. In addition, once a discretisation is generated from a given CAD model, the geometry information that has been lost can never be retrieved, forming a one-way process from discretisation to analysis. This has serious repercussions for efficiency, attenuated by the iterative nature of design.

Isogeometric analysis also makes use of an isoparametric formulation, but a key difference over its Lagrangian counterpart is the use of basis functions generated by CAD to discretise both the geometry and unknown fields. Not only is task of discretisation (meshing) greatly reduced or eliminated entirely, but a direct link is made with CAD, forming a bi-directional process. In addition, the use of CAD discretisations ensures that the *exact* geometry is used at all stages of analysis, incurring no geometrical error. But the key compelling feature of IGA is the unified nature of design and analysis.

We can summarise these points as:

- **Conventional finite element analysis**: the basis which is chosen to approximate the unknown field is also used to approximate the known geometry. This most commonly takes the form of (low order) Lagrangian basis functions. In most cases the geometry is only approximated. CAD and analysis are disparate.

- **Isogeometric analysis**: the basis is generated by CAD which captures the geometry exactly. This basis is also used to approximate the unknown field. Refinement may be required for the unknown fields, but the exact geometry is maintained at all stages of analysis. CAD and analysis are combined to form a unified process.

3.3. Isogeometric discretisation

The B-spline and NURBS discretisations outlined in Section 2 are written in terms of parametric coordinates, but to use such discretisations for analysis, we must provide a mapping that allows us to operate at the parent element level. We will outline the appropriate mappings in Section 3.4, but for now let us assume that B-spline and NURBS

basis functions can be written in terms of parent coordinates. This allows us to state the isoparametric discretisation used to approximate both the geometry and fields in IGA. For a given element e, the geometry is expressed as

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} \mathbf{P}_a^e R_a^e(\tilde{\boldsymbol{\xi}}) \tag{26}$$

where a is a local basis function index, $n_{en} = (p+1)^{d_p}$ is the number of non-zero basis functions over element e and $\mathbf{P}_a^e$,R_a^e are the control point and NURBS basis function associated with index a respectively. We employ the commonly used notation of an element connectivity mapping [116] which translates a local basis function index to a global index through

$$A = \mathrm{IEN}(a, e). \tag{27}$$

Global and local control points are therefore related through $\mathbf{P}_A \equiv \mathbf{P}_{\mathrm{IEN}(a,e)} \equiv \mathbf{P}_a^e$ with similar expressions for R_a^e. A field $\mathbf{u}(\mathbf{x})$ which governs our relevant PDE can also be discretised in a similar manner to (26) as

$$\mathbf{u}^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} \mathbf{d}_a^e R_a^e(\tilde{\boldsymbol{\xi}}) \tag{28}$$

where $\mathbf{d}_a^e$ represents a control (nodal) variable. In contrast to conventional discretisations, these coefficients are not in general interpolatory at nodes. This is similar to the case of meshless methods built on non-interpolatory shape functions such as the moving least squares (MLS) [117, 118, 119].

3.4. Mappings (change of variables)

The use of NURBS basis functions for discretisation introduces the concept of parametric space which is absent in conventional FE implementations. The consequence of this additional space is that an additional mapping must be performed to operate in parent element coordinates. As shown in Fig. 8, two mappings are considered for IGA with NURBS: a mapping $\tilde{\phi}^e : \tilde{\Omega} \to \hat{\Omega}^e$ and $\mathbf{S} : \hat{\Omega} \to \Omega$. The mapping $\mathbf{x}^e : \tilde{\Omega} \to \Omega^e$ is given by the composition $\mathbf{S} \circ \tilde{\phi}^e$.

Taking the case $d_p = d_s = 2$, an element defined by $\hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_i, \eta_{i+1}]$ is mapped from parent space to

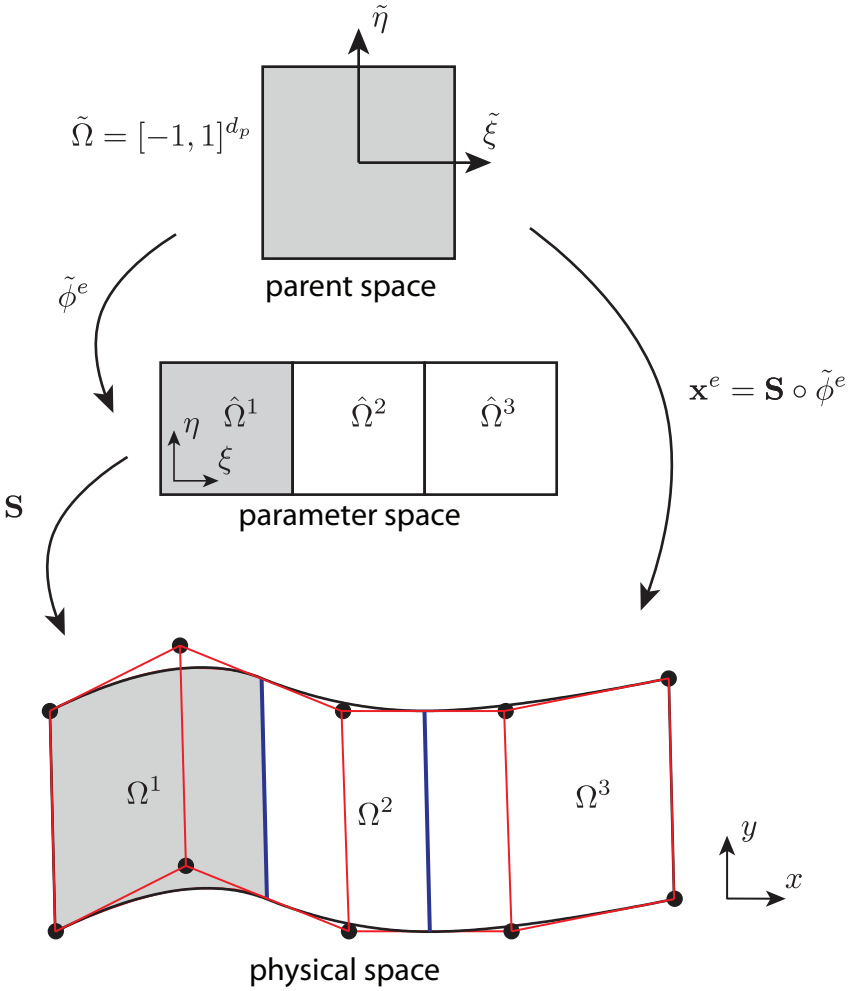


Figure 8: Diagrammatic interpretation of mappings from parent space through parametric space to physical space.

parametric space through

$$\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}) = \left\{ \begin{array}{c} \frac{1}{2}[(\xi_{i+1} - \xi_i)\tilde{\xi} + (\xi_{i+1} + \xi_i)] \\ \frac{1}{2}[(\eta_{j+1} - \eta_j)\tilde{\eta} + (\eta_{j+1} + \eta_j)] \end{array} \right\} \tag{29}$$

with an associated Jacobian determinant given by

$$|J_{\tilde{\boldsymbol{\xi}}}| = \frac{1}{4}(\xi_{i+1} - \xi_i)(\eta_{j+1} - \eta_j). \tag{30}$$

Similarly, the mapping from parametric space to physical space is given by the NURBS expressions of Eqs. (16), (19) and (22). In the case $d_p = d_s = 2$, the Jacobian of transformation for this mapping is represented by the matrix

$$\mathbf{J}_{\boldsymbol{\xi}} = \left[\begin{array}{cc} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{array} \right] \tag{31}$$

in which the components are calculated as

$$\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = \sum_{a=1}^{n_{en}} \mathbf{P}_a^e \frac{\partial R_a^e(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}. \tag{32}$$

The associated Jacobian determinant is denoted by $|J_{\boldsymbol{\xi}}|$.

The mapping $x^e : \tilde{\Omega} \to \Omega^e$, formed through the composition of the previous mappings, can be written as

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \left(\sum_{A=1}^{N} \mathbf{P}_A R_A \left(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}) \right) \right) \Bigg|_e \tag{33}$$

$$= \left(\sum_{a=1}^{n_{en}} \mathbf{P}_a R_a(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}})) \right) \Bigg|_e \tag{34}$$

$$= \sum_{a=1}^{n_{en}} \mathbf{P}_a^e R_a^e(\tilde{\boldsymbol{\xi}}) \tag{35}$$

$$\tag{36}$$

where N is the number of global basis functions and $(\cdot)|_e$ denotes that the expression $(\cdot)$ is restricted to element e. The

Jacobian determinant for this mapping is given by

$$|J| = |J_{\boldsymbol{\xi}}||J_{\tilde{\boldsymbol{\xi}}}|. \tag{37}$$

With this final mapping and Jacobian determinant, it is possible to integrate a function $f : \Omega \to \mathbb{R}$ over the physical domain as

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega^e} f(\mathbf{x})\,\mathrm{d}\Omega \tag{38}$$

$$= \sum_{e=1}^{n_{el}} \int_{\hat{\Omega}^e} f(\mathbf{x}(\boldsymbol{\xi}))\,|J_{\boldsymbol{\xi}}|\,\mathrm{d}\hat{\Omega} \tag{39}$$

$$= \sum_{e=1}^{n_{el}} \int_{\tilde{\Omega}} f(\mathbf{x}(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}})))\,|J_{\boldsymbol{\xi}}|\,|J_{\tilde{\boldsymbol{\xi}}}|\,\mathrm{d}\tilde{\Omega} \tag{40}$$

$$= \sum_{e=1}^{n_{el}} \int_{\tilde{\Omega}} f(\tilde{\boldsymbol{\xi}})\,|J|\mathrm{d}\tilde{\Omega} \tag{41}$$

with the final integral in a suitable form for application of standard Gauss-Legendre quadrature (hereafter named Gaussian quadrature). As detailed in [116] for Lagrangian basis functions, a rule of $(p+1)\times(q+1)$ Gaussian quadrature can be applied for two-dimensional elements in which p and q denote the orders of the chosen basis functions in the ξ and η direction. The same procedure is also used for NURBS basis functions in the present work, although it should be emphasised that Gaussian quadrature is not optimal for IGA. Research is currently focussed on optimal integration techniques such as that in [105, 106] in which an optimal quadrature rule, known as the half-point rule, has been applied.

Spatial derivatives of basis functions are also required for element assembly algorithms, and are calculated as

$$\left[\begin{array}{c} \dfrac{\partial R_a^e}{\partial \mathbf{x}} \end{array}\right] = \left[\begin{array}{c} \dfrac{\partial R_a^e}{\partial \boldsymbol{\xi}} \end{array}\right] \left[\begin{array}{cc} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \xi}{\partial y} \\ \dfrac{\partial \eta}{\partial x} & \dfrac{\partial \eta}{\partial y} \end{array}\right] = \left[\begin{array}{c} \dfrac{\partial R_a^e}{\partial \boldsymbol{\xi}} \end{array}\right] \mathbf{J}_{\boldsymbol{\xi}}^{-1}, \tag{42}$$

with the derivatives $\partial R_a^e/\partial \boldsymbol{\xi}$ obtained through Eq. (14).

3.5. One-dimensional IGA formulation

To illustrate the use of the isogeometric concept in a finite element context, a one-dimensional IGA formulation is developed through an example. Efforts are made to maintain the notation adopted in [116] and [4]. In the following, we define the domain $\Omega \subset \mathbb{R}$ with boundary $\Gamma \equiv \partial\Omega$. The boundary is partitioned as $\Gamma = \overline{\Gamma_D \cup \Gamma_N}$, $\Gamma_N \cap \Gamma_D = \emptyset$ where Γ_D and Γ_N denote the Dirichlet and Neumann boundaries respectively with an overline representing a closed set. Robin boundary conditions are not considered in the present work.

Letting $\Omega = (0, 1)$ and $\Gamma_D = \Gamma = \{0, 1\}$ with $\Gamma_N = \emptyset$, we seek the solution $u : \overline{\Omega} \to \mathbb{R}$ such that

$$\frac{d^2u(x)}{dx^2} + b(x) = 0, \quad x \in \overline{\Omega}, \tag{43}$$

with the Dirichlet boundary conditions specified through $g : \Gamma_D \to \mathbb{R}$ as

$$g(0) = 0, \quad g(1) = 0. \tag{44}$$

Choosing $b(x) = x$, it can be shown that the exact solution to Eq. (43) subject to Eq. (44) is given by

$$u(x) = -\frac{1}{6}x^3 + \frac{1}{6}x. \tag{45}$$

To construct the Galerkin formulation of the preceding problem, the infinite dimensional spaces $\mathscr{U}$ and $\mathscr{V}$ which represent the usual trial and test spaces respectively[7] are required. By multiplying Eq. (43) by a test function $w \in \mathscr{V}$ and applying integration by parts, the weak form of the problem reads: find $u \in \mathscr{U}$, such that

$$\int_\Omega \frac{dw}{dx}\frac{du}{dx}\,\mathrm{d}\Omega = \int_\Omega wb\,\mathrm{d}\Omega \quad \text{for all } w \in \mathscr{V}. \tag{46}$$

This is reduced to a finite-dimensional problem by creating finite-dimensional subspaces $\mathscr{U}^h \subset \mathscr{U}$ and $\mathscr{V}^h \subset \mathscr{V}$ both formed through a NURBS basis. Letting $g^h \in \mathscr{U}$ denote the finite-dimensional representation of g, the solution we

[7] Specifically, $\mathscr{U} = \{u : u \in H^1(\Omega), u = g \text{ on } \Gamma_D\}$, $\mathscr{V} = \{w : w \in H^1(\Omega), w = 0 \text{ on } \Gamma_D\}$ where $H^1(\Omega)$ represents a Sobolev space of order 1

wish to seek can be written as

$$u^h = v^h + g^h \tag{47}$$

which is valid for all $v^h \in \mathscr{V}^h$. Expression (47) underpins a key concept of the Galerkin formulation in that the solution u^h and test (weighting) function w^h are constructed from the same space of functions. We also see that the boundary condition data is 'built in' to the solution by the inclusion of the term g^h.

The approximation to our solution can be written in terms of NURBS basis functions as

$$u^h = \sum_{B=1}^{n_{eq}} d_B R_B(x) + \sum_{B=n_{eq}+1}^{n_{np}} g_B R_B(x) \tag{48}$$

where n_{eq} is the number of equations (number of unknowns) and n_{np} is the total number of nodal points (total degrees of freedom) in the system. $\{d_B\}_{B=1}^{n_{eq}}$ represents the set of all unknown control variables and $\{g_B\}_{B=n_{eq}+1}^{n_{np}}$ is the set of known Dirichlet control variables. Likewise, the test function is discretised as

$$w^h = \sum_{A=1}^{n_{eq}} c_A R_A(x) \tag{49}$$

Substitution of Eqs. (48) and (49) into Eq. (46) and noting that the values of the set $\{c_A\}$ are arbitrary, the discrete form is now written as

$$\left(\sum_{A=1}^{n_{eq}} \sum_{B=1}^{n_{eq}} \int_\Omega \frac{dR_A}{dx} \frac{dR_B}{dx} \, \mathrm{d}\Omega \right) d_B = \sum_{A=1}^{n_{eq}} \int_\Omega R_A b \, \mathrm{d}\Omega - \left(\sum_{A=1}^{n_{eq}} \sum_{B=n_{eq}+1}^{n_{np}} \int_\Omega \frac{dR_A}{dx} \frac{dR_B}{dx} \, \mathrm{d}\Omega \right) g_B \tag{50}$$

or, by defining the element stiffness matrix and element force vector as

$$\mathbf{K}_{AB} = \int_\Omega \frac{dR_A}{dx} \frac{dR_B}{dx} \, \mathrm{d}\Omega \tag{51}$$

$$\mathbf{F}_A = \int_\Omega R_A b \, \mathrm{d}\Omega - \sum_{B=n_{eq}+1}^{n_{np}} \int_\Omega \frac{dR_A}{dx} \frac{dR_B}{dx} \, \mathrm{d}\Omega, \tag{52}$$

(50) can expressed in matrix notation as

$$\mathbf{K}\mathbf{d} = \mathbf{F} \tag{53}$$

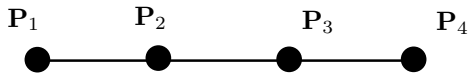
where

$$\mathbf{K} = [\mathbf{K}_{AB}] \tag{54}$$

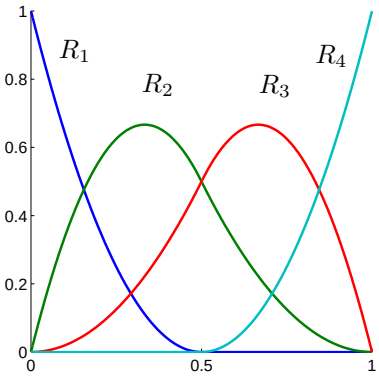
$$\mathbf{d} = \{d_B\} \tag{55}$$

$$\mathbf{F} = \{\mathbf{F}_A\} \quad A, B = 1, 2, \ldots n_{eq}. \tag{56}$$

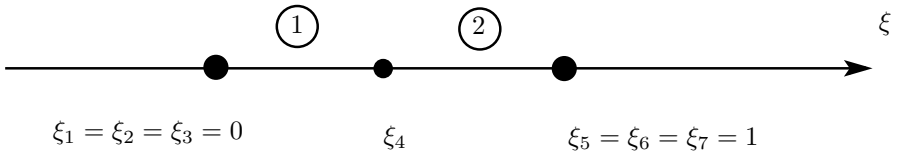
Eq. (53) is in a form amenable for computation.



A quadratic B-spline curve with 4 control points



4 quadratic basis functions



Parametric space

Figure 9: One dimensional isogeometric analysis example: exact geometry, quadratic basis functions and mesh in the parametric space. IEN(:,1)=[1 2 3] and IEN(:,2)=[2 3 4].

3.5.1. Assembly of system of equations

The global stiffness matrix $\mathbf{K}$ and global force vector $\mathbf{F}$ are assembled by looping over each element (non-zero knot interval) and inserting the appropriate element entries into their relevant rows and columns in the global system of equations. To allow integration over each element, a parent coordinate system is adopted making use of the mappings and Jacobian matrices outlined in Section 3.4. For example, for a particular element e with local basis function indices $a, b = 1, 2, \ldots, p+1$, we can compute a local element stiffness matrix $\mathbf{k}^e_{ab}$ using parent coordinates as

$$\mathbf{k}^e_{ab} = \int_{\tilde{\Omega}} \frac{dR^e_a}{dx} \frac{dR^e_b}{dx} |J| \, \mathrm{d}\tilde{\Omega} \tag{57}$$

where use has been made of expressions (42) and (37). This can be inserted into the global stiffness matrix through the element mapping given by Eq. (27).

To make this discussion more concrete, the element assembly process for a simple quadratic ($p = 2$) NURBS discretisation with $\Xi = \{0, 0, 0, 0.5, 1, 1, 1\}$ is outlined. We apply the discretisation to the problem defined by Eqs. (43) and (44). Fig. 9 illustrates the relevant NURBS discretisation including control points, basis functions and element definitions. Note that there are $p+1$ non-zero basis functions over an element e given by $\{R^e_a\}^{p+1}_{a=1}$. We can therefore summarise the element data for this discretisation as

element	knot interval	non-zero basis functions	control variables	control points
1	$[\xi_3, \xi_4]$	R_1, R_2, R_3	d_1, d_2, d_3	$\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$
2	$[\xi_4, \xi_5]$	R_2, R_3, R_4	d_2, d_3, d_4	$\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$

(58)

The implementation required to compute the element stiffness matrix is shown in Box 1. The code follows the same standard FEM routines presented in [120] where the notation $||\cdot||$ is used to signify the L^2 norm[8].

Fig. 10 illustrates the solution obtained for a NURBS discretisation for both two and four elements. Convergence to the exact solution is observed, verifying the present IGA implementation.

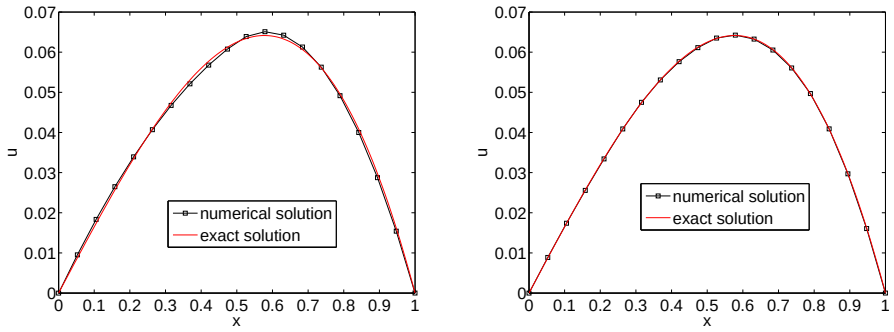
In order to assess how high order B-spline elements behave for problems with localized gradients, let us consider

[8]That is, for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $||\mathbf{x}|| = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2}$

Box 1 Element stiffness matrix evaluation for element $\hat{\Omega}^e = [\xi_i, \xi_{i+1}]$

1. $\mathbf{P} = [\mathbf{P}_1; \mathbf{P}_2; \mathbf{P}_3]$
2. $sctr = [\text{IEN}(1, e)\ \text{IEN}(2, e)\ \text{IEN}(3, e)]$
3. Set $\mathbf{k}^e = 0$
4. Loop over Gauss points (GPs) $\{\tilde{\xi}_j, \tilde{w}_j\} \quad j = 1, 2, \ldots, n_{gp}$*
   (a) Compute parametric coordinate $\xi = \tilde{\phi}^e(\tilde{\xi}_j)$ (see Eq. (29))
   (b) Compute derivatives $R^e_{a,\xi}$ $(a = 1, 2, 3)$ at point ξ
   (c) Define vector $\mathbf{R}_\xi = [R^e_{1,\xi}\ R^e_{2,\xi}\ R^e_{3,\xi}]$
   (d) Compute $|J_\xi| = ||\mathbf{R}_\xi \mathbf{P}||$
   (e) Compute $|J_{\tilde{\xi}}| = 0.5(\xi_{i+1} - \xi_i)$
   (f) Compute shape function derivatives $\mathbf{R}_x = J_\xi^{-1} \mathbf{R}_\xi^{\mathrm{T}}$
   (g) $\mathbf{k}^e = \mathbf{k}^e + \mathbf{R}_x \mathbf{R}_x^{\mathrm{T}} \, |J_\xi| \, |J_{\tilde{\xi}}| \, \tilde{w}_j$
5. End loop over GPs
6. Assemble $\mathbf{k}^e$ into the global matrix as $\mathbf{K}(sctr, sctr) = \mathbf{K}(sctr, sctr) + \mathbf{k}^e$

* $\tilde{w}_j$ denotes a Gauss point weight and n_{gp} is the total number of Gauss points.



(a) Two element NURBS mesh

(b) Four element NURBS mesh.

Figure 10: Quadratic NURBS solution for one-dimensional Poisson equation.

the following problem [119]

$$u_{,xx}(x) + b(x) = 0 \quad x \in [0,1]; \quad u(0) = 0, \quad u(1) = 1, \tag{59}$$

with

$$b(x) = \begin{cases} \left\{2\alpha^2 - 4[\alpha^2(x-0.5)]^2\right\} \exp\left\{-[\alpha(x-0.5)]^2\right\} & x \in [0.42, 0.58] \\ 0 & \text{otherwise} \end{cases} \tag{60}$$

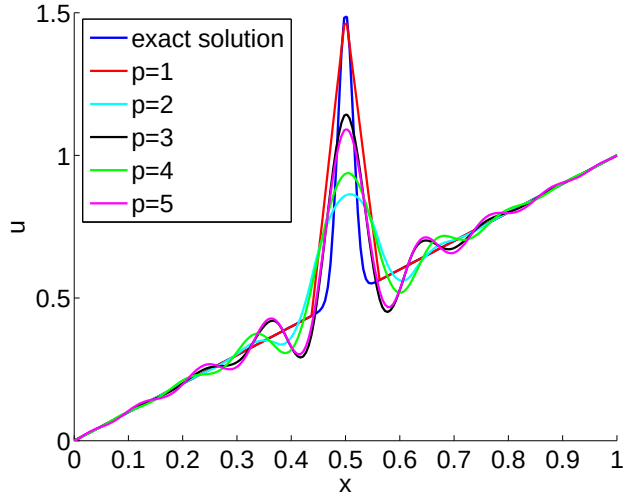
The exact solution of this problem is

$$u(x) = x + \exp\left\{-[\alpha(x-0.5)]^2\right\} x \in [0,1] \tag{61}$$

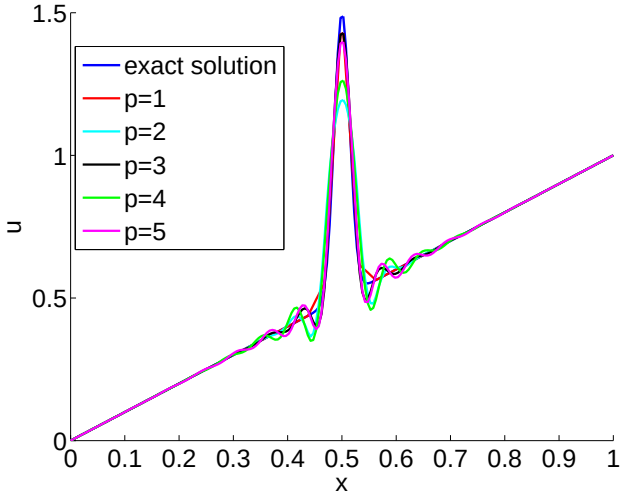
We use a value of 50 for α and the exact solution exhibits a sharp peek at location $x = 0.5$. We are going to solve this problem using elements of order ranging from one (linear elements) to five (quintic elements). To remove error in the numerical integration of the body force term, Eq. (**??**), 10 GPs were used for each element. We use the k-refinement (to be discussed in detail in Section 6.5) in building meshes of different basis orders. The initial mesh consists of one single linear element with knot vector $\Xi = \{0, 0, 1, 1\}$. The parametrization is thus linear and after performing k-refinement, the parametrization is still linear. Therefore, a point with $x = 0.3$ corresponds to $\xi = 0.3$ in the parameter space. The Matlab file for this problem is **iga1DStrongGradient.m** in the **iga** folder of our source code.

Figs. 11a,b shows the results obtained with meshes consisting of 16 and 32 elements of C^{p-1} continuity where p denotes the B-spline basis order. It is obvious that smooth basis is not suitable to problems with sharp gradients. Linear elements which have a C^0 continuity at location $x = 0.5$ (precisely at every knots) give better results than high order B-spline elements.

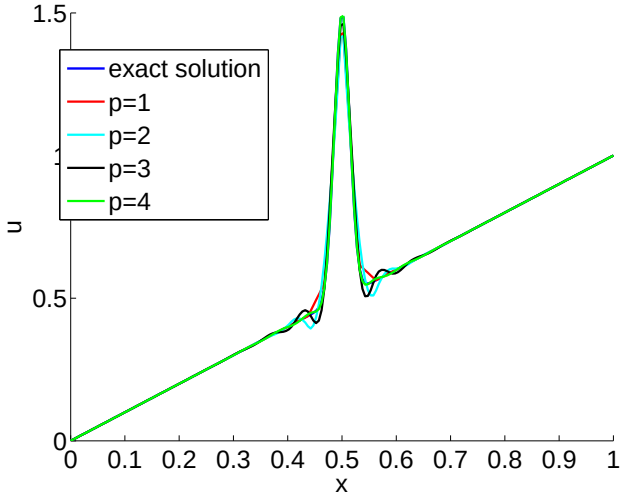
Knot insertion was made to insert the value 0.5 p times to the initial knot vector so that the basis is C^0 continuous at $x = 0.5$. For example for $p = 2$, a vector $\{0.5, 0.5\}$ was inserted to the knot vector. The results are given in Fig. 11c,d where the peak was captured better. Finally $0.42, 0.5, 0.58$ were added to the knots p times so that the basis are C^0 continuous at locations $x = 0.42, 0.5, 0.58$. The corresponding results are given in Fig. 12. With only 16 cubic elements (25 CPs) the exact solution was well captured. This example showed the flexibility of B-splines/NURBS–high



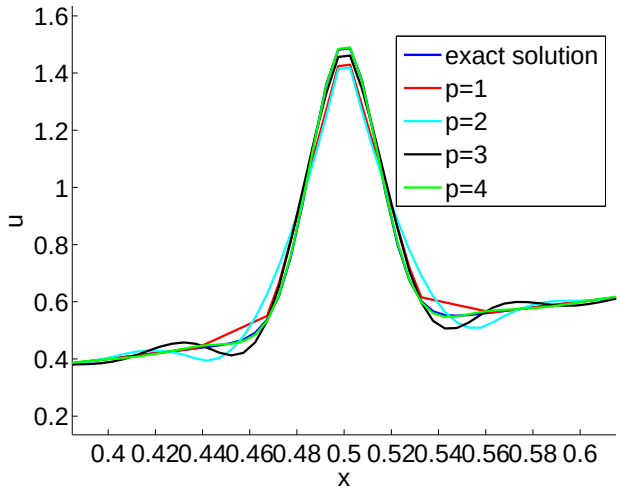
(a) 16 elements



(b) 32 elements

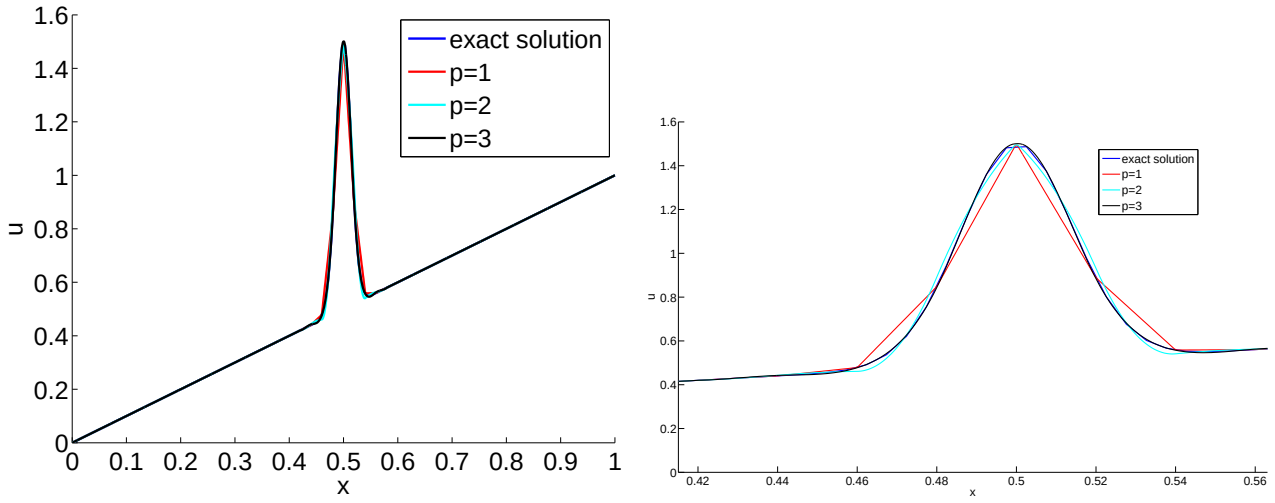


(c) 32 elements with C^0 continuity at $x = 0.5$



(d) 32 elements with C^0 continuity at $x = 0.5$ (close up)

Figure 11: Comparison of the IGA result against the exact solution for the one dimensional PDE given in Eq. (59).



(a) 16 elements with C^0 continuity at $x = 0.42, 0.5, 0.58$

(b) 16 elements with C^0 continuity at $x = 0.42, 0.5, 0.58$ (close up)

Figure 12: Comparison of the IGA result against the exact solution for the one dimensional PDE given in Eq. (59).

order functions and any level of continuity can be easily achieved.

4. Elasticity: two-dimensional implementation

The application of the FEM to elasticity is common and represents a familiar language to many researchers. We therefore outline a two-dimensional implementation of IGA for linear elasticity, highlighting the differences over conventional discretisations.

4.1. Assembly process for two dimensional elastostatic analysis

Consider a domain Ω, bounded by Γ. The boundary is partitioned into two sets: Γ_u and Γ_t with displacements prescribed on Γ_u and tractions $\overline{\mathbf{t}}$ prescribed on Γ_t: $\Gamma = \overline{\Gamma_t \cup \Gamma_u}$, $\Gamma_t \cap \Gamma_u = \emptyset$. The weak form of a linear elastostatics problem is to find $\mathbf{u}$ in the trial space [9], such that for all test functions $\delta\mathbf{u}$ in the test space [10],

$$\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \mathbf{D} : \boldsymbol{\varepsilon}(\delta\mathbf{u}) \mathrm{d}\Omega = \int_{\Gamma_t} \overline{\mathbf{t}} \cdot \delta\mathbf{u} \mathrm{d}\Gamma + \int_\Omega \mathbf{b} \cdot \delta\mathbf{u} \mathrm{d}\Omega, \tag{62}$$

[9] contains C^0 functions
[10] contains C^0 functions vanishing on Γ_u

where the elasticity matrix is denoted by $\mathbf{D}$ and $\mathbf{b}$ refers to a body force. Using the Galerkin method where the same shape functions $R_a(\tilde{\boldsymbol{\xi}})$ are used for both $\mathbf{u}$ and $\delta\mathbf{u}$, we can write

$$\mathbf{u}(\mathbf{x}) = \sum_{A=1}^{n_{np}} R_A(\tilde{\boldsymbol{\xi}})\mathbf{u}_A, \quad \delta\mathbf{u}(\mathbf{x}) = \sum_{A=1}^{n_{np}} R_A(\tilde{\boldsymbol{\xi}})\delta\mathbf{u}_A, \tag{63}$$

where $\mathbf{u}_A, \delta\mathbf{u}_A$ denote the nodal displacement and its variations, respectively and n_{np} is the total number of control points. In 2D, each control point has two unknowns–the x and y displacements, hence one writes $\mathbf{u}_A = \{u_{xA}, u_{yA}\}$. Proper modification can be made for 3D problems.

Substitution of these approximations into Eq. (62) and using the arbitrariness of the nodal variations gives the standard discrete set of equations $\mathbf{K}\ \mathbf{u} = \mathbf{f}$ with

$$\mathbf{K}_{AB} = \int_\Omega \mathbf{B}_A^{\mathrm{T}}\mathbf{D}\mathbf{B}_B \mathrm{d}\Omega, \quad \mathbf{f}_A = \int_{\Gamma_t} R_A \bar{\mathbf{t}} \mathrm{d}\Gamma + \int_\Omega R_A \mathbf{b} \mathrm{d}\Omega, \quad A, B = 1, 2, \ldots, n_{np}. \tag{64}$$

In two dimensions, the strain-displacement matrix $\mathbf{B}_A$ is given by

$$\mathbf{B}_A = \begin{bmatrix} R_{A,x} & 0 \\ 0 & R_{A,y} \\ R_{A,y} & R_{A,x} \end{bmatrix}, \tag{65}$$

where the shape function derivatives are computed according to Eq. (42) and $R_{A,x} \equiv dR_A/dx$.

We now consider a concrete two dimensional problem shown in Fig. 13. In this case, the knot vectors are $\Xi^1 = [0, 0, 1, 1]$ and $\Xi^2 = [0, 0, 0, 0.5, 1, 1, 1]$, respectively. The orders of the basis functions are $p = 1$ and $q = 2$. There two control points ($n = 2$) along the ξ direction and four control points ($m = 4$) along the η direction that results in 8 control points ($n_{np} = n \times m = 8$).

The domain and non-zero basis functions for Element 1 are given by

direction	knot interval	non zero basis
ξ	$[\xi_2, \xi_3]$	N_1, N_2
η	$[\eta_3, \eta_4]$	M_1, M_2, M_3

(66)

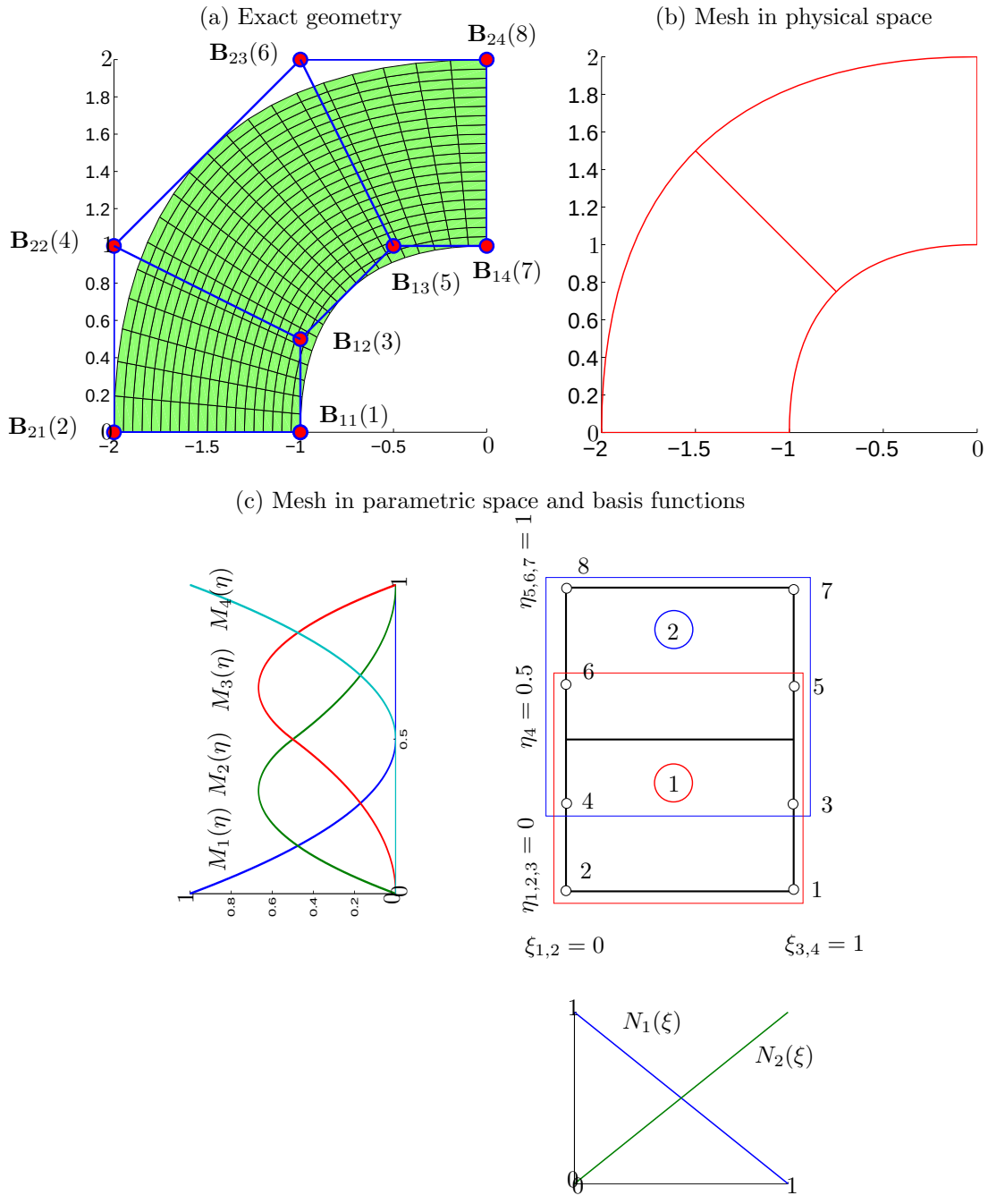


Figure 13: Two dimensional isogeometric analysis example: (a) exact geometry, (b) mesh in physical space and (c) mesh in the parametric space. The two rectangles are used to illustrate the control points belonging to each element.

Hence there are six non-zero basis functions on element $e = 1$ which can be assembled into a vector $\mathbf{R}$ as follows

$$\mathbf{R}_1 = [N_1M_1, N_2M_1, N_1M_2, N_2M_2, N_1M_3, N_2M_3]. \tag{67}$$

These six basis functions are associated with six global basis indices given by (in Matlab® notation)

$$\text{IEN}(:, 1) = [1, 2, 3, 4, 5, 6]. \tag{68}$$

Similarly, for element 2, the shape function vector is given by

$$\mathbf{R}_2 = [N_1M_2, N_2M_2, N_1M_3, N_2M_3, N_1M_4, N_2M_4], \tag{69}$$

with the associated global indices

$$\text{IEN}(:, 2) = [3, 4, 5, 6, 7, 8]. \tag{70}$$

Control points are stored in a two dimensional matrix of dimensions $n_{np} \times 2$. The connectivity data is stored in a two dimensional matrix of dimensions $(p + 1) * (q + 1) \times n_{el}$ where n_{el} denotes the number of elements. For the example under consideration, these two matrices are given by

$$\text{controlPts} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{21} & \mathbf{B}_{12} & \mathbf{B}_{22} & \mathbf{B}_{13} & \mathbf{B}_{23} & \mathbf{B}_{14} & \mathbf{B}_{24} \end{bmatrix}^{\mathrm{T}}, \quad \text{IEN} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}^{\mathrm{T}}. \tag{71}$$

The knot intervals along the ξ and η directions are stored in the following matrices

$$\text{elRangeU} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad \text{elRangeV} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \tag{72}$$

where the number of rows is equal to the number of elements in each direction ($n_s^1 - 1, n_s^2 - 1$ respectively).

With this vector of basis functions $\mathbf{R}$, the control points associated to this element, we can compute the derivatives

of the basis functions. The derivatives of the basis functions with respect to x are stored in the following vector

$$\mathbf{R}_{,x} = \begin{bmatrix} R_{1,x} & R_{2,x} & R_{3,x} & R_{4,x} & R_{5,x} & R_{6,x} \end{bmatrix}^{\mathrm{T}}, \tag{73}$$

Similarly, we have $\mathbf{R}_{,y}$ for the derivatives of the basis functions with respect to y. Having these basis function derivatives, we are now ready to define the $\mathbf{B}$ matrix for any element e

$$\mathbf{B}_e = \begin{bmatrix} \mathbf{R}[1]_{,x} & 0 & \mathbf{R}[2]_{,x} & 0 & \cdots \\ 0 & \mathbf{R}[1]_{,y} & 0 & \mathbf{R}[2]_{,y} & \cdots \\ \mathbf{R}[1]_{,y} & \mathbf{R}[1]_{,x} & \mathbf{R}[2]_{,y} & \mathbf{R}[2]_{,x} & \cdots \end{bmatrix}, \tag{74}$$

where the control point displacement vector is stored in the following order $\mathbf{u} = [u_{x1}, u_{y1}, u_{x2}, u_{y2}, \ldots, u_{xn_{np}}, u_{yn_{np}}]^{\mathrm{T}}$. The element stiffness matrix is then given by

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}_e^{\mathrm{T}} \mathbf{D} \mathbf{B}_e \mathrm{d}\Omega_e, \tag{75}$$

which is then assembled to the global stiffness matrix using the element connectivity matrix and the fact that a control point I corresponds to positions $2 * I - 1$ and $2 * I$ in the global displacement vector $\mathbf{u}$.

For implementation convenience, Box 2 gives a procedure of an isogeometric analysis of 2D elasticity problems. Note that this aims for a Matlab implementation and the whole elementary stiffness matrix is computed in one go. In order to compute the external force vector, it is convenient to define a boundary mesh as shown in Fig. 14. The computation of the external force vector $\int_\Gamma \mathbf{R}^{\mathrm{T}} \bar{\mathbf{t}} \mathrm{d}\Gamma$ then follows the one dimensional assembly procedure given in Section 3.5.

4.2. Boundary condition enforcement

Fig. 15 illustrates two kinds of Dirichlet boundary conditions: on edge AD, $u_x = 0$ and on edge BC, $u_y = \bar{u}$. The former is a homogeneous Dirichlet boundary condition (BC) while the latter is referred to as uniform inhomogeneous Dirichlet BCs. Homogeneous Dirichlet BCs can be enforced by setting the corresponding control variables as zeros (in this example, setting $u_{xI} = 0, I = 2, 4, 6, 8$). For edge BC, the inhomogeneous Dirichlet BCs can also be satisfied by setting $u_{yI} = \bar{u}, I = 1, 3, 5, 7$. This is due to the partition of unity property of the NURBS basis i.e., $u_y^{BC} = M_1(\eta)u_{y1} + M_2(\eta)u_{y3} + M_3(\eta)u_{y5} + M_4(\eta)u_{y7} = (M_1(\eta) + M_2(\eta) + M_3(\eta) + M_4(\eta))\bar{u} = \bar{u}$. Note that a boundary of a

Box 2 Procedure for isogeometric analysis of 2D elasticity problems.

1. Loop over elements, $e = 1, \ldots, nel$
   (a) Determine NURBS coordinates $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$ using elRangeU and elRangeV
   (b) Get connectivity array, $sctr = \text{IEN}(:, e)$
   (c) Define $sctrB(1, 1:2:2*nn) = 2*sctr - 1; sctrB(1, 2:2:2*nn) = 2*sctr$ for assembly*
   (d) $\mathbf{K}_e = \mathbf{0}$
   (e) Loop over Gauss points, $\{\tilde{\boldsymbol{\xi}}_j, \tilde{w}_j\} \quad j = 1, 2, \ldots, n_{gp}$**
      i. Compute $\boldsymbol{\xi}$ corresponding to $\tilde{\boldsymbol{\xi}}_j$ (Eq. 29)
      ii. Compute $|J_{\tilde{\xi}}|$ (Eq. 30)
      iii. Compute derivatives of shape functions $\mathbf{R}_{,\xi}$ and $\mathbf{R}_{,\eta}$ at $\boldsymbol{\xi}$
      iv. Compute $\mathbf{J}_\xi$ using controlPts$(sctr, :)$, $\mathbf{R}_{,\xi}$ and $\mathbf{R}_{,\eta}$ (Eq. 31)
      v. Compute Jacobian inverse $\mathbf{J}_\xi^{-1}$ and determinant $|J_\xi|$
      vi. Compute derivatives of shape functions $\mathbf{R}_{,\mathbf{x}} = [\mathbf{R}_{,\xi}\ \mathbf{R}_{,\eta}]\mathbf{J}_\xi^{-1}$ (Eq. 42)
      vii. Use $\mathbf{R}_{,\mathbf{x}}$ to build the strain-displacement matrix $\mathbf{B}$ (Eq. 74)
      viii. Compute $\mathbf{K}_e = \mathbf{K}_e + \tilde{w}_j |J_{\tilde{\xi}}||J_\xi| \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}$
   (f) End loop over Gauss points
   (g) Assemble $\mathbf{K}_e$: $\mathbf{K}(sctrB, sctrB) = \mathbf{K}(sctrB, sctrB) + \mathbf{K}_e$
2. End loop over elements

* nn denotes the number of control points per element i.e., nn=length(sctr).
** $\tilde{w}_j$ denotes a Gauss point weight and n_{gp} is the total number of Gauss points.

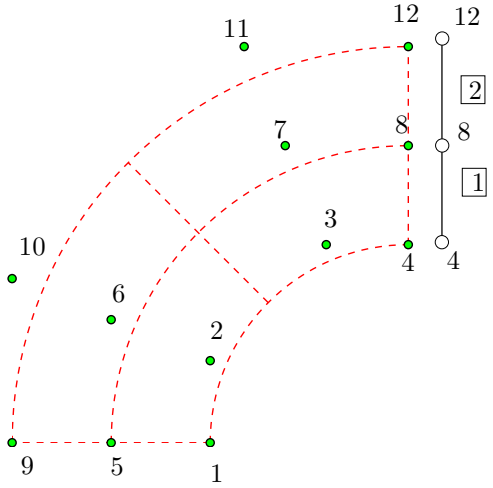


Figure 14: Boundary mesh for external force computation. A linear basis is used in the η direction. Assume a traction is applied on the edge containing nodes 4, 8 and 12. The boundary mesh is composed of two linear isogeometric elements 1 and 2.

NURBS surface is a NURBS curve due to the use of open knots. Inhomogeneous Dirichlet BCs applied on corner control points (black points in Fig. 15) are enforced by simply setting the corner control variables equal to the prescribed values since the NURBS shape functions at these points satisfy the Kronecker delta property (assuming the use of open knot vectors). This is called direct imposition of Dirichlet BCs.

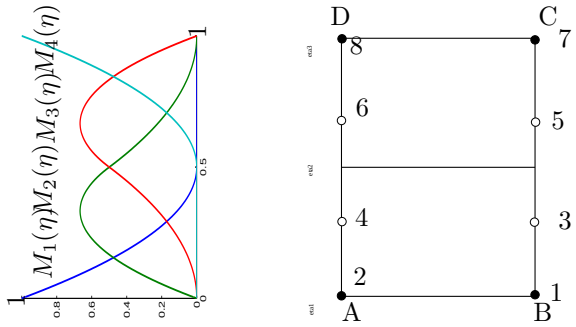


Figure 15: Imposing Dirichlet BCs: black points denote corner control points where the NURBS basis satisfies the Kronecker delta property.

For cases other than the ones previously discussed such as a prescribed displacement imposed at interior control point 3 or a non-uniform Dirichlet BC–Dirichlet BCs that vary from point to point applied on edge BC, special treatment of Dirichlet BCs have to be employed as is the case for meshless methods. Techniques available include

the Lagrange multiplier method, the penalty method, the augmented Lagrangian method and we refer to [119] for an overview of these techniques in the context of meshless methods. In [121] a transformation method was proposed to impose inhomogeneous Dirichlet BCs in IGAFEM. However this method requires modifications to the stiffness matrix which breaks the usual structure of a FE code. The authors in [71] presented a weak enforcement of general inhomogeneous Dirichlet BCs using a least squares minimization and the implementation is described in what follows. The same procedure was used in imposing BCs in meshless methods when coupling a fluid to a solid domain through a master-slave concept [122]. Imposing Dirichlet boundary conditions with Nitsche's method was presented in [123] for spline-based finite elements.

The basic idea of the least squares method is to find the parameters of the boundary control points that minimize the following quantity

$$\begin{aligned} J &= \frac{1}{2}\sum_{C} ||\mathbf{u}(\mathbf{x}_C) - \bar{\mathbf{u}}(\mathbf{x}_C)||^2 \\ &= \frac{1}{2}\sum_{C} \left|\left|\sum_{A} R_A(\mathbf{x}_C)\mathbf{q}_A - \bar{\mathbf{u}}(\mathbf{x}_C)\right|\right|^2, \end{aligned} \tag{76}$$

where $\mathbf{x}_C$ denotes a set of collocation points distributed on the essential boundary Γ_u, $\mathbf{q}_A$ are the parameters of the control points defining Γ_u, $\bar{\mathbf{u}}(\mathbf{x})$ represents the prescribed displacements and R_A represents the NURBS basis functions that are non-zero at $\mathbf{x}_C$ which are univariate NURBS functions $R_A(\xi)$ for the boundary of a NURBS surface is a NURBS curve. For the sake of clarity, let us consider the case where there is only one collocation point and a quadratic basis (thus there are 3 non-zero R_A at $\mathbf{x}_C$). So, we have

$$J = \frac{1}{2}\left|\left|R_1(\mathbf{x}_C)\mathbf{q}_1 + R_2(\mathbf{x}_C)\mathbf{q}_2 + R_3(\mathbf{x}_C)\mathbf{q}_3 - \bar{\mathbf{u}}(\mathbf{x}_C)\right|\right|^2. \tag{77}$$

The partial derivatives of J with respect to $\mathbf{q}_i$ are given by

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{q}_1} &= [R_1(\mathbf{x}_C)\mathbf{q}_1 + R_2(\mathbf{x}_C)\mathbf{q}_2 + R_3(\mathbf{x}_C)\mathbf{q}_3 - \bar{\mathbf{u}}(\mathbf{x}_C)]R_1(\mathbf{x}_C) \\
\frac{\partial J}{\partial \mathbf{q}_2} &= [R_1(\mathbf{x}_C)\mathbf{q}_1 + R_2(\mathbf{x}_C)\mathbf{q}_2 + R_3(\mathbf{x}_C)\mathbf{q}_3 - \bar{\mathbf{u}}(\mathbf{x}_C)]R_2(\mathbf{x}_C) \\
\frac{\partial J}{\partial \mathbf{q}_3} &= [R_1(\mathbf{x}_C)\mathbf{q}_1 + R_2(\mathbf{x}_C)\mathbf{q}_2 + R_3(\mathbf{x}_C)\mathbf{q}_3 - \bar{\mathbf{u}}(\mathbf{x}_C)]R_3(\mathbf{x}_C).
\end{aligned} \tag{78}$$

The condition $\frac{\partial J}{\partial \mathbf{q}} = 0$ thus gives the following linear system

$$\begin{bmatrix} R_1R_1 & R_2R_1 & R_3R_1 \\ R_1R_2 & R_2R_2 & R_3R_2 \\ R_1R_3 & R_2R_3 & R_3R_3 \end{bmatrix}_{\mathbf{x}_C} \begin{bmatrix} q_1^x & q_1^y \\ q_2^x & q_2^y \\ q_3^x & q_3^y \end{bmatrix} = \begin{bmatrix} \bar{u}_x(\mathbf{x}_C)R_1(\mathbf{x}_C) & \bar{u}_y(\mathbf{x}_C)R_1(\mathbf{x}_C) \\ \bar{u}_x(\mathbf{x}_C)R_2(\mathbf{x}_C) & \bar{u}_y(\mathbf{x}_C)R_2(\mathbf{x}_C) \\ \bar{u}_x(\mathbf{x}_C)R_3(\mathbf{x}_C) & \bar{u}_y(\mathbf{x}_C)R_3(\mathbf{x}_C) \end{bmatrix}. \tag{79}$$

By collecting all the NURBS basis at point $\mathbf{x}_C$ in a column vector $\mathbf{N}(\mathbf{x}_C)$, the control points displacements in the x and y directions in $\mathbf{q}_x$ and $\mathbf{q}_y$, respectively, Eq. (79) can be written in a compact form as

$$\begin{aligned}
[\mathbf{N}(\mathbf{x}_C)\mathbf{N}^{\mathrm{T}}(\mathbf{x}_C)]\mathbf{q}_x &= \bar{u}_x(\mathbf{x}_C)\mathbf{N}(\mathbf{x}_C) \\
[\mathbf{N}(\mathbf{x}_C)\mathbf{N}^{\mathrm{T}}(\mathbf{x}_C)]\mathbf{q}_y &= \bar{u}_y(\mathbf{x}_C)\mathbf{N}(\mathbf{x}_C).
\end{aligned} \tag{80}$$

Repeating the same analysis for other collocation points $\mathbf{x}_C$ on the Dirichlet boundary, one obtains the linear system $\mathbf{Aq} = \mathbf{b}$ with two different $\mathbf{b}$ (one for the x component and the other for the y component). The dimension of $\mathbf{A}$ is $n_D \times n_D$ where n_D denotes the number of control points defining the Dirichlet boundary. Having these boundary control point displacements, the enforcement of Dirichlet BCs (when solving $\mathbf{Ku} = \mathbf{f}$) are then treated as in standard FEM.

We note that this procedure involves only control points that define the essential boundary. This is in sharp contrast to meshless shape functions such as the MLS used in the Element Free Galerkin method [124] where the displacements at a point on the essential boundary depend not only on the nodes on that boundary but also the neighbouring interior nodes. It is said that NURBS therefore satisfy the so-called weak Kronecker delta property as local Maxent interpolants

approximations [125].

Listing 1 gives the Matlab® implementation of the least-squares method. Note that in our implementation, the collocation points are uniformly distributed in the parameter space. We refer to [71] for a discussion on the influence of the collocation points on the accuracy. We refer to Fig. 16 for an illustration of this method.

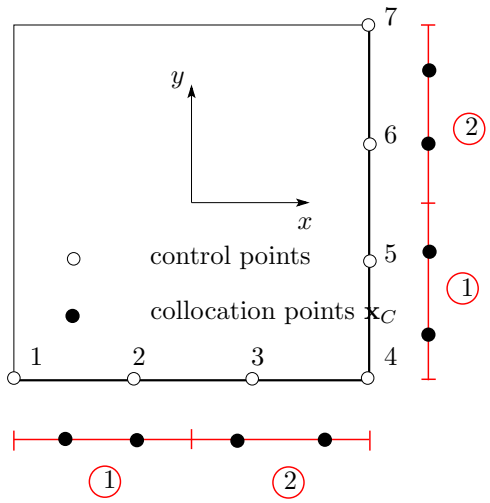


Figure 16: Illustration of the implementation of the least squares method: a quadratic NURBS surface with knot vectors $\Xi^1 = \{0, 0, 0, 0.5, 1, 1, 1\}$ and $\Xi^2 = \{0, 0, 0, 0.5, 1, 1, 1\}$. Essential BCs are imposed on the bottom and right edges. A one dimensional mesh for this boundary is created. The control points defining the essential boundary are numbered from one to the total number of boundary control points (seven in this example). Note that this numbering is required only for assembling the matrix $\mathbf{A}$.

Listing 1: Matlab® implementation of the least-squares method

```
A  = zeros(noDispNodes,noDispNodes);
bx = zeros(noDispNodes,1);
by = zeros(noDispNodes,1);
noxC   = 4; % number of collocation pts/element
% loop over bottom edge
for ie=1:noElemsU
    sctr   = bottomEdgeMeshIGA(ie,:); % standard connectivity array (global numbering)
    pts    = controlPts(sctr,:);      % control pts of element ie
    sctrA  = bndElement(ie,:);        % connectivity array for A matrix
    xiE    = elRangeU(ie,:);          % parametric coords of ie
    xiArr  = linspace(xiE(1),xiE(2),noxC); % collocation pts x_C
```

```
    for ic=1:noxC
        xi              = xiArr(ic);
        [N dNdxi]       = NURBS1DBasisDers(xi,p,uKnot,weights);
        A(sctrA,sctrA) = A(sctrA,sctrA) + N'*N;
        x               = N*pts;
        % exact displacements
        [ux,uy]    = exact_Griffith(x,E0,nu0,sigmato,xTip,seg,cracklength);
        bx(sctrA) = bx(sctrA) + ux*N';
        by(sctrA) = by(sctrA) + uy*N';
    end
end
% loop over other edges if neccessary
% solve the system Aq_x=bx and Aq_y=by
[LL UU] = lu(A);
qxTemp  = LL\bx;
qyTemp  = LL\by;
qx      = UU\qxTemp; qy      = UU\qyTemp;
% later, before solving Ku=f, qx and qy will be used  to enforce BCs as in conventional FEM.
```

In our Matlab® code implementation are provided for the penalty method, the Lagrange multiplier method and the least squares method. The implementation of the two former methods are considered standard and the reader is referred to [119] for details.

5. Extended isogeometric finite element method

There are basically two ways in which discontinuities can be modeled in the context of IGA: PUM based enrichment and knot insertion. For the former, there are works of [110, 71, 72, 73]. For the latter, we refer to [69, 76]. PUM based methods are general for they can be applied to any kinds of discontinuity such as weak and strong discontinuities whereas knot insertion used to produce the C^{-1} continuity is only suitable for cracks. Knot insertion found great applications for delamination analyses [76] where the crack path is known *a priori*. Note that in [69], knot insertion was used in combination with T-splines to model cracks of which trajectory is not known in advance. However the implementation is tedious.

In the category of PUM based IGA methods which are sometimes called XIGA (eXtended IGA), the method in [73] is different from existing XIGA formulations [110, 71, 72] in a way that discontinuities (holes/inclusions/cracks) are exactly isogeometrically represented by NURBS. Note that in most of XIGA techniques, discontinuities are defined implicitly using level set method. This section briefly presents the XIGA formulation of [71, 72] and implementation aspects are deferred to Section 6.7.

The extended finite element method (XFEM) (see e.g., [70], [126] for a recent review and open XFEM library [127]) is a local PUM enrichment method in which internal boundaries such as holes, inclusions, cracks are modeled independently of the FE discretisation which allows for crack growth modeling without remeshing. Two dimensional extended isogeometric finite element formulation (XIGA) was presented in [71, 72] in which the displacement field is enriched for traction-free crack modelling using the following approximation

$$\mathbf{u}^h(\mathbf{x}) = \sum_{I\in\mathcal{S}} R_I(\mathbf{x})\mathbf{u}_I + \sum_{J\in\mathcal{S}^c} R_J(\mathbf{x})H(\mathbf{x})\mathbf{a}_J + \sum_{K\in\mathcal{S}^f} R_K(\mathbf{x})\left(\sum_{\alpha=1}^{4} B_\alpha \mathbf{b}_K^\alpha\right), \tag{81}$$

where $R_{I,J,K}$ are the NURBS basis functions. In addition to the standard degrees of freedom (dofs) $\mathbf{u}_I$, additional dofs $\mathbf{a}_J$ and $\mathbf{b}_K^\alpha$ are introduced. The set $\mathcal{S}^c$ includes the control points/nodes whose support is cut by the crack and the set $\mathcal{S}^f$ are control points whose support contains the crack tip $\mathbf{x}_{tip}$, see Fig. 17. Note that we use a topological tip enrichment and in the literature another tip enrichment scheme called geometrical enrichment with a fixed area (to ensure that the role of enrichment in the approximation space does not vanish as the mesh is refined) is present see e.g., [128]. The Heaviside function H is given by

$$H(\mathbf{x}) = \begin{cases} +1 & \text{if } (\mathbf{x}-\mathbf{x}^*)\cdot\mathbf{n} \geq 0 \\ -1 & \text{otherwise} \end{cases}, \tag{82}$$

where $\mathbf{x}^*$ is the projection of point $\mathbf{x}$ on the crack and $\mathbf{n}$ denotes the outward normal vector to the crack. And the branch functions, which span the crack tip displacement field, are given by

$$[B_1, B_2, B_3, B_4](r,\theta) = \left[\sqrt{r}\sin\frac{\theta}{2}, \sqrt{r}\cos\frac{\theta}{2}, \sqrt{r}\sin\frac{\theta}{2}\cos\theta, \sqrt{r}\cos\frac{\theta}{2}\cos\theta\right], \tag{83}$$

where r and θ are polar coordinates in the local crack front coordinate system (see e.g., [70] for details). It is noted

that B_1 is discontinuous along the crack face.

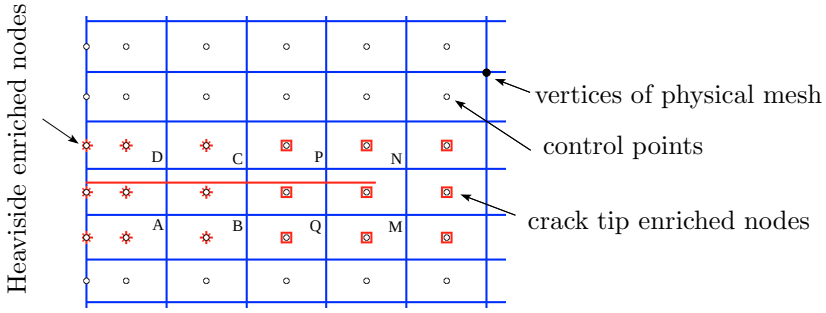


Figure 17: Illustration of enriched node sets $\mathcal{S}^f$ and $\mathcal{S}^c$ for a quadratic NURBS mesh. The thick red line denotes the crack.

Using the standard Galerkin procedure as outlined in Section 4.1 the discrete system of equations $\mathbf{Ku} = \mathbf{f}$ are formed by an enlarged $\mathbf{B}$ matrix given by

$$\mathbf{B} = \left[\begin{array}{c|c} \mathbf{B}^{\text{std}} & \mathbf{B}^{\text{enr}} \end{array} \right], \tag{84}$$

where $\mathbf{B}^{\text{std}}$ is the standard strain-displacement matrix $\mathbf{B}$ (cf. Eq. (65)) and $\mathbf{B}^{\text{enr}}$ is the enriched $\mathbf{B}$ matrix of which components are given by

$$\mathbf{B}_I^{\text{enr}} = \begin{bmatrix} (R_I)_{,x}\Psi_I + R_I(\Psi_I)_{,x} & 0 \\ 0 & (R_I)_{,y}\Psi_I + R_I(\Psi_I)_{,y} \\ (R_I)_{,y}\Psi_I + R_I(\Psi_I)_{,y} & (R_I)_{,x}\Psi_I + R_I(\Psi_I)_{,x} \end{bmatrix}, \tag{85}$$

Ψ_I may represent either the Heaviside function H or the branch functions B_α depending whether control point I is Heaviside or near tip enriched. The unknown vector $\mathbf{u}$ contains both displacements and enriched dofs. This extended IGAFEM can be implemented within an available IGAFEM code with little modification following the ideas given in the meshless review and computer implementation aspects paper [119]. Some implementation aspects will be given in Section 6.7. In our code enrichment for holes and material interfaces are also given. They are, however, not covered here because the implementation follows the ideas given here.

6. MIGFEM- A Matlab IGA (X)FEM code

In this section we describe shortly the open source IGA Matlab (X)FEM program which can be downloaded from `https://sourceforge.net/projects/cmcodes/`. The code supports one, two and three dimensional linear elasticity problems. Extended IGA for crack and material interface modelling is also implemented. Geometrically nonlinear solid and structural mechanics models are available. The features of the code include:

- Global h,p and k-refinement is provided for one, two and three dimensional meshes.
- Extended IGA for 2D/3D stationary traction-free cracks and material interfaces.
- Visualization of displacements and stresses in Paraview.
- Inhomogeneous Dirichlet boundary conditions are treated with the penalty method, the Lagrange multiplier method and the least squares method.
- Compatible multi-patch isogeometric formulation for two dimensional problems.
- Support for T-splines via the Bézier extraction operators.
- Structural elements including beams, plates and thin shells.
- Implicit Newmark scheme and explicit central difference scheme for time discretisation.
- Python scripts to extract Rhino3d NURBS surfaces to be used for IGA.

6.1. Data structure

MIGFEM follows the Matlab FEM code described in [120]. The main data structures include (1) *element* (store the element connectivity), (2) *controlPts* (store control point coordinates), (3) *weights* (store the weights) (4) K (stiffness matrix) and (5) f (external force vector). Contrary to FEM in which the element connectivity and nodal coordinates are inputs which have been created by a meshing program, in IGA, the input consists of CAD data including knots, control points, order of basis functions. Therefore, one has to construct the *element* matrix based on the knots and the basis orders.

Construction of the *element* matrix is illustrated by a 2D example shown in Fig. 18. Given the knot vectors $uKnot$ and $vKnot$ together with the orders of the basis p and q, one can compute the number of control points along the ξ and ζ directions, denoted by n and m. Then we define a two dimensional matrix of dimension $m \times n$ called *node_pattern* given by for the illustrated example shown in Fig. 18 which has 4 control points along ξ and 4 control points along η directions

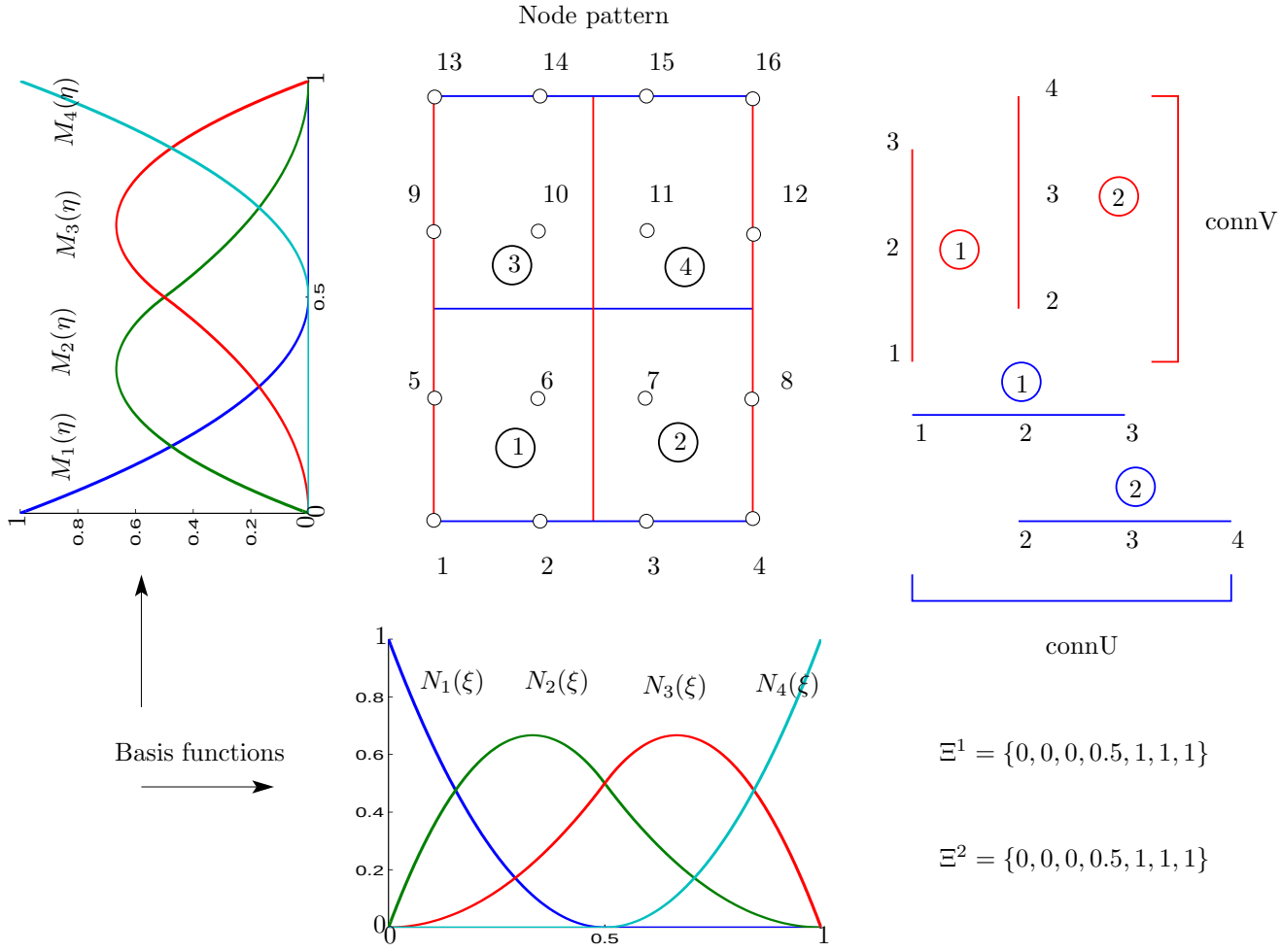


Figure 18: Two dimensional isogeometric analysis: mesh generation for a bi-quadratic NURBS surface (2×2 elements). The circles denote the control points.

$$node\_pattern = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}, \tag{86}$$

which is simply an application of Eq. (8) for defining global indexes.

The number of elements along the two directions $n_s^1 - 1, n_s^2 - 1$ are $noElemsU = \text{length}(\text{unique}(uKnot)) - 1$ and $noElemsV = \text{length}(\text{unique}(uKnot)) - 1$ (in the Matlab language). The connectivity matrix for the ξ direction, denoted by $connU$ which is a $noElemsU \times (p+1)$ matrix and the connectivity matrix for the η direction, denoted by $connV$ which is a $noElemsV \times (q+1)$ matrix are given by

$$connU = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}, \quad connV = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}, \tag{87}$$

for the example under consideration. Matrix $connU$ stores the indices of the non-zero basis functions at each knot span for the ξ direction. Having this information and the global indexes of the NURBS basis given in Eq. (86), we are able to define the connectivity matrix for the whole mesh, called $element$ which is a $noElemsU * noElemsV \times (p+1)(q+1)$ matrix. For the example being considered, this matrix reads

$$element = \begin{bmatrix} 1 & 2 & 3 & 5 & 6 & 7 & 9 & 10 & 11 \\ 2 & 3 & 4 & 6 & 7 & 8 & 10 & 11 & 12 \\ 5 & 6 & 7 & 9 & 10 & 11 & 13 & 14 & 15 \\ 6 & 7 & 8 & 10 & 11 & 12 & 14 & 15 & 16 \end{bmatrix}. \tag{88}$$

Note that the matrix $element$ is the transpose of the IEN matrix. We decided to use $element$ instead of IEN to be compatible with the FEM code [120] on which MIGEM is built.

Finally in order to compute the mapping from the parent domain $\tilde{\Omega}$ to the parametric space, we define the knot

intervals in two directions as

$$rangeU = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad rangeV = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 1 \end{bmatrix}. \tag{89}$$

In order to retrieve the parametric coordinates of a specific element, the matrix *index*, that is a $noElemsU * noElemsV \times 2$ matrix, is used. For a given element e, its parametric coordinates are determined by *rangeU(index(e,1),:); rangeV(index(e,2),:).*

The above discussion, together with the illustration given in Fig. 18 is implemented in Matlab in the file **generateIGA2DMesh.m**, located in folder **meshing**, and shown in Listing 2. In that folder, one can find similar M files for generating 1D and 3D meshes.

Listing 2: Mesh generation for two dimensional problems. Input are uKnot, vKnot, and p, q.

```
uniqueUKnots  = unique(uKnot); uniqueVKnots  = unique(vKnot);
noElemsU      = length(uniqueUKnots)-1;%#of elements xi dir.
noElemsV      = length(uniqueVKnots)-1;%#of elements eta dir.
noPtsX        = length(uKnot)-p-1; noPtsY        = length(vKnot)-q-1;
nodePattern   = zeros(noPtsY,noPtsX);
count = 1;
for i=1:noPtsY
    for j=1:noPtsX
        nodePattern(i,j) = count; count           = count + 1;
    end
end
% determine our element ranges and the corresponding knot indices along each direction
[elRangeU,elConnU] = buildConnectivity(p,uKnot,noElemsU);
[elRangeV,elConnV] = buildConnectivity(q,vKnot,noElemsV);
noElems = noElemsU * noElemsV;
element = zeros(noElems,(p+1)*(q+1));
e = 1;
for v=1:noElemsV
    vConn = elConnV(v,:);
    for u=1:noElemsU
```

```
        c = 1; uConn = elConnU(u,:);
        for i=1:length(vConn)
            for j=1:length(uConn)
              element(e,c) = nodePattern(vConn(i),uConn(j)); c = c + 1;
            end
        end
        e = e + 1;
    end
end
index = zeros(noElems,2);
count = 1;
for j=1:size(elRangeV,1)
    for i=1:size(elRangeU,1)
        index(count,1) = i; index(count,2) = j; count = count + 1;
    end
end
```

6.2. Shape function routines

The shape function routines (evaluate NURBS basis functions and derivatives with respect to parametric coordinates) are implemented using MEX files to improve the performance. They are located in folder **C_files**. Listing 3 gives some commonly used shape function routines to compute the NURBS basis function and their first derivatives in 1D, 2D and 3D at a given point. In the last line, second derivatives are also computed.

Listing 3: Shape function routines: uKnot,vKnot,wKnot store Ξ^1, Ξ^2, Ξ^3.

```
% R, dRdxi: row vectors i.e. dRdxi dimension is (1,3) for p=2
[R dRdxi]                 = NURBS1DBasisDers(Xi,p,uKnot,weights);
[R dRdxi dRdeta]          = NURBS2DBasisDers([Xi; Eta],p,q,uKnot,vKnot,weights');
[R dRdxi dRdeta dRdzeta] = NURBS3DBasisDers([Xi;Eta;Zeta],p,q,r,uKnot,vKnot,wKnot,weights');
[R dRdxi dRdeta dR2dxi dR2det dR2dxe] = NURBS2DBasis2ndDers([Xi; Eta],p,q,uKnot,vKnot,weights');
```

Listing 4: Matlab code for 2D spatial derivatives of shape functions.

```
1  % input: element e and parameter coordinates of a Gauss point Xi and Eta
```

```
2   sctr    = element(e,:);          %  element scatter vector
3   pts     = controlPts(sctr,:);    %  of dimension nn x 2 where nn: number of nodes
4   % derivate of R=Nxi*Neta w.r.t xi and eta
5   [dRdxi dRdeta] = NURBS2Dders([Xi;Eta],p,q,uKnot,vKnot,weights');
6   % Jacobian matrix
7   jacob       = pts'*[dRdxi' dRdeta'];
8   invJacob    = inv(jacob);
9   % nn x 2 matrix: first column derivatives wrt x ...
10  dRdx        = [dRdxi' dRdeta'] * invJacob;
```

6.3. Assembly process

The assembly of an IGA-FEM code is given in Listing 5 where it can be seen that the procedure is almost identical to that used in the conventional FEM. The minor differences lie in (1) the need of the elements in the parameter space (lines 4 to 7) and (2) the second map (from the parent domain to the parametric domain) in the numerical integration of the stiffness matrix (line 35).

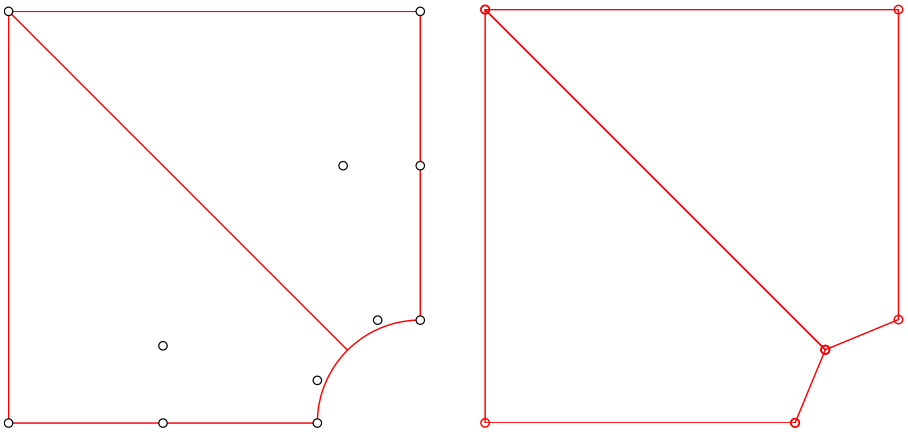
Listing 5: Matlab code for IGA (2D elasticity problems).

```
1   [W,Q]=quadrature(4, 'GAUSS', 2); % 4x4 point quadrature
2   for e=1:noElems % Loop over elements (knot spans)
3     idu     = index(e,1);
4     idv     = index(e,2);
5     xiE     = elRangeU(idu,:); % [xi_i,xi_i+1]
6     etaE    = elRangeV(idv,:); % [eta_j,eta_j+1]
7     sctr    = element(e,:);    %  element scatter vector
8     nn      = length(sctr); nn2 = 2*nn;
9     sctrB(1,1:2:2*nn) = 2*sctr-1;
10    sctrB(1,2:2:2*nn) = 2*sctr  ;
11    B       = zeros(3,2*nn);
12    pts     = controlPts(sctr,:);
13    for gp=1:size(W,1)                        % loop over Gauss points
14      pt       = Q(gp,:);
15      wt       = W(gp);
16      % compute coords in parameter space
```

```
    Xi      = parent2ParametricSpace(xiE,pt(1));
    Eta     = parent2ParametricSpace(etaE,pt(2));
    J2      = jacobianPaPaMapping(xiE,etaE);
    % derivate of R=Nxi*Neta w.r.t xi and eta
    [dRdxi dRdeta] = NURBS2Dders([Xi;Eta],p,q,uKnot,vKnot,weights');
    % Jacobian matrix
    jacob       = pts'*[dRdxi' dRdeta'];
    J1          = det(jacob); invJacob    = inv(jacob);
    dRdx        = [dRdxi' dRdeta'] * invJacob;
    % B matrix
    B(1,1:2:nn2)  = dRdx(1,:); B(2,2:2:nn2)  = dRdx(2,:);
    B(3,1:2:nn2)  = dRdx(2,:); B(3,2:2:nn2)  = dRdx(1,:);
    % compute elementary stiffness matrix and  assemble it to the global matrix
    K(sctrB,sctrB) = K(sctrB,sctrB) + B'*C*B*J1*J2*wt;
  end
end
```

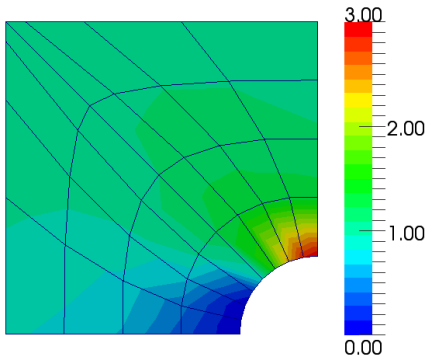
6.4. Post-processing

We present here a simple technique to visualize the IGA results that reuse available visualization techniques for finite elements. To simplify the exposé, only 2D cases are considered here. In the first step, a mesh consisting of four-noded quadrilateral (Q4) elements is generated, see Fig. 19. We call this mesh the visualization mesh (whose connectivity matrix is stored in *elementV* and nodal coordinates are stored in *node*) whose nodes are images of the knots ξ_i, η_j in the physical space. In the second step, quantities of interest e.g., stresses are computed at the nodes of the Q4 mesh. This mesh together with the nodal values can then be exported to a visualization program such as Paraview, see [129], for visualization. It should be emphasized that due to the high order continuity of the NURBS basis, there is no need to perform nodal averaging as required in standard C^0 finite element analysis to obtain smooth fields. Listing 6 gives the Matlab code (we have removed some code due to the similarity with the assembly code) for building the Q4 visualization mesh, computing the stresses at the nodes of this mesh and exporting the result to Paraview. The source code can be found in the file **plotStress1.m** located in folder **post-processing**. The results can be then visualized directly in Matlab or exported to Paraview, see Listing 7.



(a) NURBS mesh

(b) approximate Q4 mesh



(c) stress visualization on a refined mesh

Figure 19: Exact NURBS mesh (top left) and approximate Q4 mesh (top right) for visualization purpose. The nodes in the Q4 mesh are the intersections of the ξ and η knot lines. The bottom figure shows a contour plot of a stress field in Paraview. It should be emphasized that the mesh in (b) does not provide a sufficiently smooth contour plot for it to be directly usable. The result given in (c) was obtained with a refined NURBS mesh (hence a refined Q4 mesh).

Listing 6: Matlab code for computing stresses and displacements at nodes of the visualization mesh.

```
buildVisualizationMesh; % build visualization Q4 mesh
stress = zeros(noElems,4,3);
disp   = zeros(noElems,4,2);
for e=1:noElems
    idu    = index(e,1); idv  = index(e,2);
    xiE    = elRangeU(idu,:); etaE = elRangeV(idv,:);
    sctr   = element(e,:);        % element scatter vector
    sctrB  = [sctr sctr+noCtrPts];%vector scatters B matrix
    uspan = FindSpan(noPtsX-1,p,xiE(1),uKnot);
    vspan = FindSpan(noPtsY-1,q,etaE(1),vKnot);
    % loop over 4 nodes (instead of loop over GPs)
    gp = 1;
    for iv=1:2
      % Q4 elements, the nodes are numbered counter-clockwise
      if (iv==2) xiE = sort(xiE,'descend'); end
      for iu=1:2
        Xi  = xiE(iu); Eta = etaE(iv);
        [N dRdxi dRdeta]= NURBS2DBasisDersSpecial([Xi;Eta],p,q,uKnot,vKnot,weights',[uspan;vspan]);
        % B matrix as usual
        strain          = B*U(sctrB);
        stress(e,gp,:)  = C*strain;
        gp = gp +1;
     end
  end
end
```

Listing 7: Matlab code for post-processing.

```
% plot sigma_xx contour directly in Matlab
figure
plot_field(node,elementV,'Q4',stress(:,:,1));
% export to VTK format to plot in Mayavi or Paraview
sigmaXX = zeros(size(node,1),1);
sigmaYY = zeros(size(node,1),1);
```

```
sigmaXY = zeros(size(node,1),1);
dispX = zeros(size(node,1),1);
dispY = zeros(size(node,1),1);
for e=1:size(elementV,1)
    connect = elementV(e,:);
    for in=1:4
        nid = connect(in);
        sigmaXX(nid) = stress(e,in,1);
        sigmaYY(nid) = stress(e,in,2);
        sigmaXY(nid) = stress(e,in,3);
    end
end
VTKPostProcess(node,elementV,2,'Quad4','result.vtu',...
            [sigmaXX sigmaYY sigmaXY],[dispX dispY]);
```

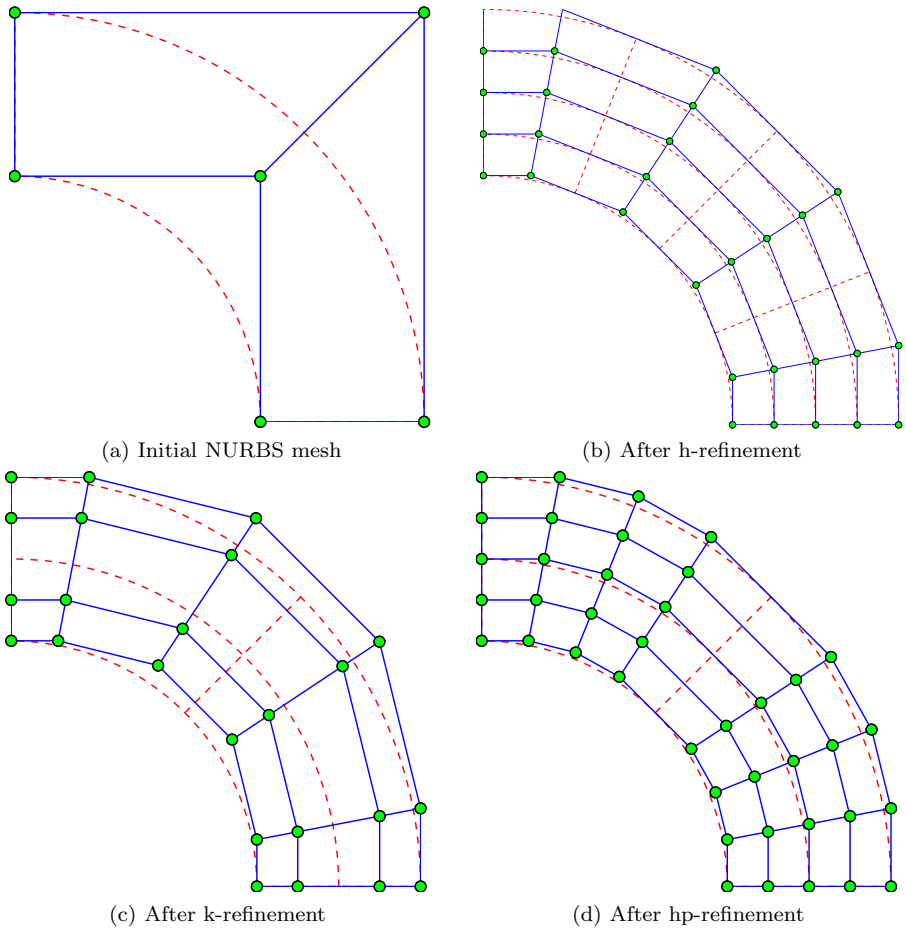
For three-dimensional problems, the same procedure is used where a mesh of tri-linear brick elements is created and the values of interest are computed at the nodes of this mesh (see file **plotStress3d.m**). The results are then exported to Paraview under a structured grid format (*.vts files), see the file **mshToVTK.m**).

6.5. h, p, k-refinement

For the refinement of NURBS, we reuse the NURBS Toolbox described in [108]. We construct a NURBS surface as shown in Fig.20a. The corresponding Matlab code is given in Listing 8. Using a uniform h-refinement (Listing 9) that divides a knot span into two, we obtain the mesh given in Fig.20b. If one needs to use $k-$refinement ($p-$refinement followed by $h-$refinement), then the code in Listing 10 can be used (see Fig.20c). Finally, Fig.20d gives the mesh which is obtained by the process in which $h-$refinement is employed first and then $p-$refinement is performed (see Listing 11). After using the NURBS toolbox, the NURBS object is then converted to MIGFEM data structures using the function **convert2DNurbs** located in folder **nurbs-util**.

Listing 8: Construct a NURBS surface using the NURBS toolbox.

```
a = 0.3; b = 0.5; % inner/outer radius
```



(a) Initial NURBS mesh (b) After h-refinement

(c) After k-refinement (d) After hp-refinement

Figure 20: Illustration of the utilization of the NURBS toolbox in building NURBS object (a). From the initial mesh, different meshes can be obtained using either $h-$refinement, $p-$refinement or combination thereof. As can be seen from (c) and (d), $k-$refinement (c) is more efficient than $hp-$refinement (d). The function **plotMesh.m** in folder **meshing** is used to plot NURBS mesh and control polygon.

```
%% knots, control points
uKnot = [0 0 0 1 1 1]; %quadratic basis
vKnot = [0 0 1 1];     %linear basis
% homogeneous coords (x*w,y*w,z*w,w)
% 3 pts in u-dir, 2 pts in v-dir
controlPts          = zeros(4,3,2);

controlPts(1:2,1,1) = [a;0];
controlPts(1:2,2,1) = [a;a;];
controlPts(1:2,3,1) = [0;a];

controlPts(1:2,1,2) = [b;0];
controlPts(1:2,2,2) = [b;b];
controlPts(1:2,3,2) = [0;b];

controlPts(4,:,:)   = 1;
controlPts(4,2,1) = 1/sqrt(2);
controlPts(4,2,2) = 1/sqrt(2);
% homogenous coordinates (x*w,y*w,z*w)
controlPts(1:2,2,1) = controlPts(1:2,2,1)*fac;
controlPts(1:2,2,2) = controlPts(1:2,2,2)*fac;
%% build NURBS object
solid = nrbmak(controlPts,{uKnot vKnot});
```

Listing 9: h-refinement using the NURBS toolbox.

```
refineLevel = 2;
for i=1:refineLevel
  uKnotVectorU = unique(uKnot);
  uKnotVectorV = unique(vKnot);
  % new knots along two directions
  newKnotsX =uKnotVectorU(1:end-1)+0.5*diff(uKnotVectorU);
  newKnotsY =uKnotVectorV(1:end-1)+0.5*diff(uKnotVectorV);

  newKnots  = {newKnotsX newKnotsY};
```

```
  solid      = nrbkntins(solid,newKnots);
  uKnot      = cell2mat(solid.knots(1));
  vKnot      = cell2mat(solid.knots(2));
end
convert2DNurbs % convert to the MIGFEM format
plotMesh(controlPts,weights,uKnot,vKnot,p,q,res,'r-','try.eps');
```

Listing 10: k-refinement using the NURBS toolbox.

```
% code from Listing 5
% order elevation, p=p+2, q=q+1
solid = nrbdegelev(solid,[2 1]);
% then, knot insertion using the code from Listing 6
convert2DNurbs % convert to the MIGFEM format
```

Listing 11: h-refinement followed by p-refinement using the NURBS toolbox.

```
% code from Listing 5
% then, knot insertion using the code from Listing 6
% order elevation, p=p+2, q=q+1
solid = nrbdegelev(solid,[2 1]);
convert2DNurbs % convert to the MIGFEM format
```

Listing 12: A typical input file for MIGFEM.

```
% code from Listing 5
% code from Listing 6 (if only h-refinement)
convert2DNurbs%convert to the MIGFEM format (if not done yet)
generateIGA2DMesh % build the mesh (element connectivity)
```

6.6. Input file for MIGFEM

A typical input file is given in Listing 12. This file replaces the standard FE mesh file. Note that, for backward compatibility with older versions of the MIGFEM code, some input files do not use the NURBS toolbox to create the NURBS object. For those input files, it is however impossible to perform order elevation and hence $k-$refinement. It is,

therefore, recommended to create the NURBS objects using the NURBS toolbox [108] which, besides the aforementioned refinement functionalities, also supports many useful operations such as extrusion, rotation etc.

Incorporating NURBS into an existing FE code cannot be considered a trivial task due to the presence of various spaces (index, parameter) that are not present in conventional FE codes, Bézier extraction [130, 52] provides a FE data structure that allows for a straightforward implementation of NURBS/T-splines into any FE codes. Appendix D briefly presents this concept and its implementation in MIGFEM.

6.7. XFEM implementation

There are certainly different ways to implement XIGA. We present a way that reuse most of the tools present in an existing XFEM code. We use the approximate Q4 mesh used for visualization as discussed in Section 6.4 for selection of enriched control points. The level set values defining the crack at the vertices of this mesh are then computed. Based on these level sets, elements cut by the crack and elements containing the crack tip can be determined [131]. For example, element ABCD in Fig. 17 is cut by the crack. In a FEM context, its four nodes are then enriched using the Heaviside function. In an isogeometric framework, however, the control points associated to this element are enriched. Listing 13 details the Matlab® implementation of this process. Note that this listing is taken directly from our XFEM code [119] with only one small modification and thus the proposed technique is considered simpler than that adopted in [72]. We emphasize that the crack geometry is defined in the physical space to keep the usual XFEM notation.

Listing 13: Selection of enriched control points/nodes.

```
enrich_node = zeros(noCtrPts,1);
count1 = 0;
count2 = 0;
for iel = 1 : numelem
    sctr    = elementV(iel,:);
    sctrIGA = element(iel,:);
    phi  = ls(sctr,1); % normal level set
    psi  = ls(sctr,2); % tangent level set
    if ( max(phi)*min(phi) < 0 )
        if max(psi) < 0
            count1 = count1 + 1 ; % one split element
            split_elem(count1) = iel;
```

```
            enrich_node(sctrIGA)   = 1;
        elseif max(psi)*min(psi) < 0
            count2 = count2 + 1 ; % one tip element
            tip_elem(count2) = iel;
            enrich_node(sctrIGA)   = 2;
        end
    end
end
split_nodes = find(enrich_node == 1);
tip_nodes   = find(enrich_node == 2);
```

Remark 6.1. Note that for simple geometries as those tackled in this paper, the use of level sets is certainly not necessary. More generally, describing open surfaces (cracks) with level sets requires two level sets functions that, for crack growth simulations, must be reinitialised for stability (this decreases accuracy) and reorthogonalised every few time steps. This is particularly cumbersome and probably explains along with the difficulties in dealing intersecting and branching cracks the recent trend of research efforts in the area of phase field models of fracture, see e.g., [132, 74], and the thick level set method [133].

Crack visualization We have $H(\mathbf{x}^+) - H(\mathbf{x}^-) = 2$ and $B_1(\mathbf{x}^+) - B_1(\mathbf{x}^-) = 2\sqrt{r}$, therefore the displacement jump at a point $\mathbf{x}$ on the crack face is given by

$$[\![\mathbf{u}]\!](\mathbf{x}) = 2\sum_{J\in\mathcal{S}^c} R_J(\mathbf{x})\mathbf{a}_J + 2\sqrt{r}\sum_{K\in\mathcal{S}^f} R_K(\mathbf{x})\mathbf{b}_K^1. \tag{90}$$

Note that the other branch functions B_α, $\alpha = 2, 3, 4$ are continuous functions and thus do not contribute to the displacement jump.

Fig. 21 illustrates the idea for crack visualization with the script **post-processing/crackedMeshNURBS.m** providing implementation details. The contour plots of the displacement and stress field of a mode I cracked sample are given in Fig. 22. Note that the stresses at points on the crack are simply set to zero (traction-free cracks) and the stresses of the new nodes of the tip element are interpolated from the values of the four nodes of this Q4 element.

Remark 6.2. As is the case for XFEM, integration over elements cut by the cracks usually requires subdivision of the elements into integration subcells. We refer to [134] for a recent discussion on this issue. In addition to this

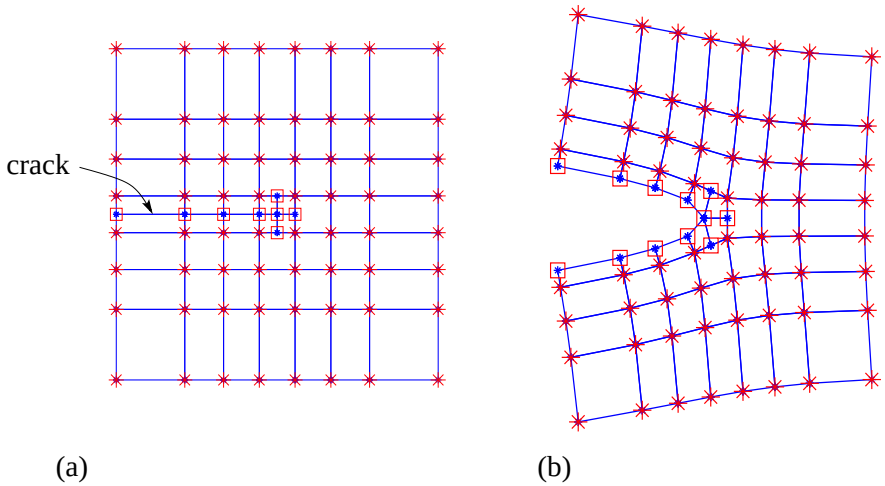


Figure 21: Crack visualization in XIGA: (a) build a mesh that is compatible to the crack by introducing double nodes along the crack (square nodes) and (b) assign the displacement jumps to these new nodes. Note that the square nodes in the tip element are used only for compatibility purposes. Exact mode I displacements are imposed on the bottom, right and top edge using the Lagrange multiplier method while Neumann BCs from the exact stress field are enforced on the left edge. We refer to [119] for a detailed description of this standard problem.

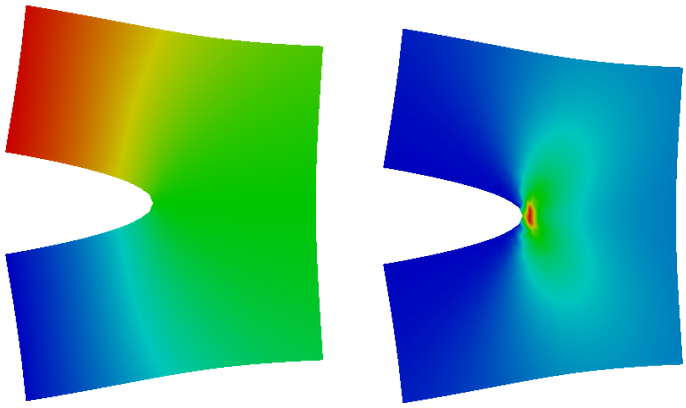


Figure 22: Contour plots on a cracked mesh: (a) vertical displacement and (b) normal stress in the vertical direction.

popular technique, a simple integration rule is also provided–elements crossed by the crack and tip-enriched elements are numerically integrated using a regular Gauss-Legendre quadrature with a large number of Gauss points as done in [135].

7. Structural mechanics

Thanks to the high order continuity provided by NURBS/T-splines, the implementation of rotation-free thin beam/-plate/shell elements becomes direct and simple. In this section, we are going to present the implementation of a rotation-free IGA Kirchhoff plate formulation (rotation free shell elements can be found in folder **structural-mechanics**). The plate geometry and the deflection are both approximated by NURBS. At control points there is only one unknown-the deflection or transverse displacement. For simplicity only isotropic elastic plates are considered. We refer to [136] for a treatment of plate theories.

The element stiffness matrix is defined as

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}_e^{\mathrm{T}} \mathbf{D} \mathbf{B}_e \mathrm{d}\Omega, \tag{91}$$

where the constitutive matrix $\mathbf{D}$ reads

$$\mathbf{D} = \frac{Eh^3}{12(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 0.5(1-\nu) \end{bmatrix}, \tag{92}$$

where E, ν are the Young's modulus and Poisson's ratio, respectively; h denotes the plate thickness and the element displacement-curvature matrix $\mathbf{B}_e$ that contains the second derivatives of the shape functions is given by

$$\mathbf{B}_e = \begin{bmatrix} R_{1,xx} & R_{2,xx} & \cdots R_{n,xx} \\ R_{1,yy} & R_{2,yy} & \cdots R_{n,yy} \\ 2R_{1,xy} & 2R_{2,xy} & \cdots 2R_{n,xy} \end{bmatrix}, \tag{93}$$

where n denotes the number of basis functions of element e and $R_{A,xx} \equiv d^2 R_A / dx^2$.

7.1. Boundary conditions

For clamped BCs one needs to fix the rotations. The nodal unknowns are, however, only the transverse displacements **w**. To fix the rotation of a boundary, we simple fix two row of control points at the boundary [20] because these control points define the tangent of the surface at the boundary (cf. Fig. 3), see Fig. 23.

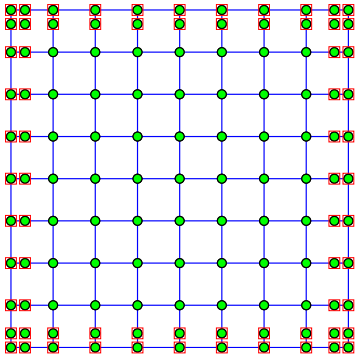


Figure 23: Enforcing BCs for a fully clamped plate: simply fixing the deflections of two rows of control points around the clamped boundary. Note that the set of CPs next to the boundary CPs are not artificially added to impose the rotations. They are simply the CPs defining the geometry of the plate.

7.2. Symmetry boundary conditions

Fig. 24 illustrates the use of symmetry boundary conditions when only 1/4 of the plate is modelled. Along the symmetry lines, the rotation should be zero which can be enforced by constraining the deflection (w) of two rows of control points along these lines together. These contraints can be implemented using a simple penalty technique as shown in Listing 14.

Listing 14: Enforcing symmetry BCs.

```
w      = 1e7;
penaltyStiffness = w*[1 -1;-1 1];
for i=1:length(topNodes)
    sctr  = [topNodes(i) nextToTopNodes(i)];
    K(sctr,sctr) = K(sctr,sctr) + penaltyStiffness;
end
% the same for the left two rows
```

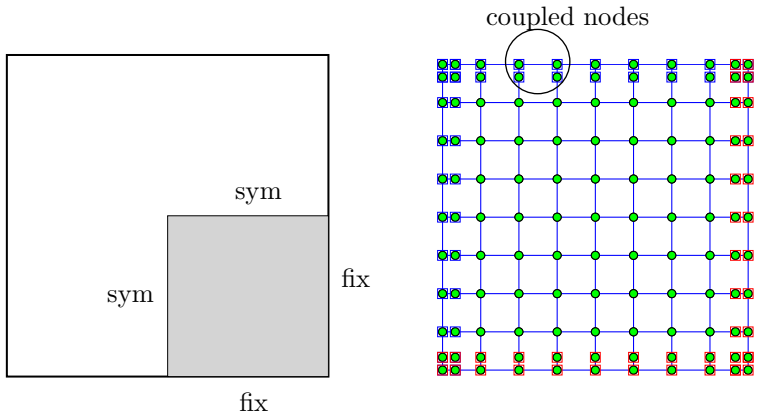


Figure 24: A fully clamped square plate: 1/4 model is analyzied using appropriate symmetry BCs. Along the symmetry lines, the rotation is fixed which can be achieved by enforcing the deflection of two rows of control points that define the tangent of the plate to have the same value.

For completeness, we give the implementation of the rotation free Kirchhoff plate elements in Listing 15. Note that we have skipped codes that are common with IGA code for 2D continua.

Listing 15: Computation of stiffness matrix for the rotation free Kirchhoff plate.

```
for gp=1:size(W,1)
    pt      = Q(gp,:); wt      = W(gp);
    % shape functions, first and second derivatives w.r.t natural coords
    [R dRdxi dRdeta dR2dxi dR2det dR2dxe]=NURBS2DBasis2ndDers([Xi; Eta],p,q,uKnot,vKnot,weights');
    jacob  = [dRdxi; dRdeta]           * pts; % 2x2 matrix
    jacob2 = [dR2dxi; dR2det; dR2dxe] * pts; % 3x2 matrix
    J1     = det(jacob);
    dxdxi = jacob(1,1); dydxi = jacob(1,2);
    dxdet = jacob(2,1); dydet = jacob(2,2);
    j33    = [dxdxi^2      dydxi^2      2*dxdxi*dydxi;
              dxdet^2      dydet^2      2*dxdet*dydet;
              dxdxi*dxdet dydxi*dydet dxdxi*dydet+dxdet*dydxi];
    % Jacobian inverse and spatial 1st and 2nd derivatives
    invJacob    = inv(jacob);
    dRdx        = invJacob*[dRdxi;dRdeta];
    dR2dx       = inv(j33)*([dR2dxi; dR2det; dR2dxe]-jacob2*dRdx);
    % B matrix
```

```
    B           = dR2dx;
    B(3,:)      = B(3,:)*2;
    % compute elementary stiffness matrix, sctr=element(e,:)
    K(sctr,sctr) = K(sctr,sctr) + B' * C * B * J1 * J2 * wt;
end
```

8. Verification examples

In this section, numerical examples in linear elasticity and linear elastic fracture mechanics in 2D and 3D are presented with the purpose to serve as a set of verification examples for MIGFEM. They include an infinite plate with a circular hole under constant in-plane tension, the pinched cylinder, an edge cracked plate in tension, a three-dimensional mode I fracture problem and a large deformation thin shell problem. Unless otherwise stated, standard direct imposition of Dirichlet BCs is used. Units are standard International System (SI) units.

8.1. Two and three dimensional solid mechanics

8.1.1. Infinite plate with a circular hole

The problem considered is that of an infinite plate with a circular hole in the centre under constant in-plane tension at infinity as shown in Fig. 25 where, due to symmetry, only a quarter of the plate is modeled. The plate dimension is taken to be $L \times L$ and the circular hole has a radius R. The exact stress field in the plate is given by

$$\sigma_{xx}(r,\theta) = 1 - \frac{R^2}{r^2}\left(\frac{3}{2}\cos 2\theta + \cos 4\theta\right) + \frac{3}{2}\frac{R^4}{r^4}\cos 4\theta \tag{94a}$$

$$\sigma_{yy}(r,\theta) = -\frac{R^2}{r^2}\left(\frac{1}{2}\cos 2\theta - \cos 4\theta\right) - \frac{3}{2}\frac{R^4}{r^4}\cos 4\theta \tag{94b}$$

$$\sigma_{xy}(r,\theta) = -\frac{R^2}{r^2}\left(\frac{1}{2}\sin 2\theta + \sin 4\theta\right) + \frac{3}{2}\frac{R^4}{r^4}\sin 4\theta, \tag{94c}$$

where r, θ are the usual polar coordinates centered at the center of the hole.

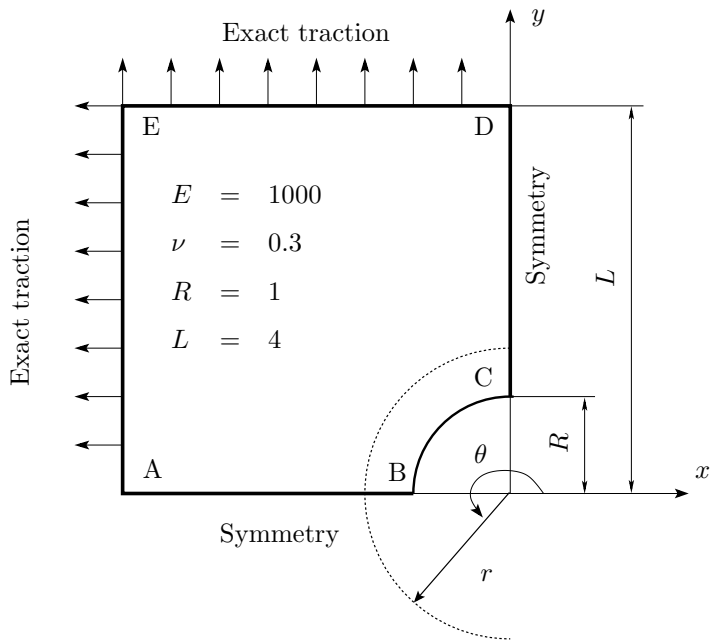


Figure 25: Infinite plate with a circular hole under constant in-plane tension: quarter model. Boundary conditions include: $u_y = 0$ (AB), $u_x = 0$ (CD), $\overline{\mathbf{t}}^{\mathrm{T}} = (-\sigma_{xx}, -\sigma_{xy})$ (AE), $\overline{\mathbf{t}}^{\mathrm{T}} = (\sigma_{xy}, \sigma_{yy})$ (ED).

The material properties are specified as $E = 10^3$, Poisson's ratio $\nu = 0.3$ and the geometry is such that $L = 4$, $R = 1$. A plane stress condition is assumed. The problem is solved with quadratic NURBS meshes such as those shown in Fig. 26. The control points and weights for the coarsest mesh can be found in [3] or file **plateHoleCkData.m**. Fig. 27, generated in Paraview, illustrates the contour plot of numerical σ_{xx}. Note that the stress concentration at point $(R, 3\pi/2)$ is well captured and a smooth stress field is obtained throughout.

Remark 8.1. Using the visualization technique described in Section 6.4 for this problem, a note should be made on the evaluation of the stress field at the top left corner where there are two coincident control points. This causes a singular Jacobian matrix. Therefore at this corner, the stresses at a point slightly shifted from the original position are used.

8.1.2. Pinched cylinder

In order to demonstrate the performance of the 3D IGA implementation, we consider the pinched cylinder problem as shown in Fig. 28. Note that we discretize the shell with solid NURBS elements. Due to symmetry, only 1/8 of the model is analyzed. A tri-quadratic NURBS mesh ($p = q = r = 2$) was used for the computation. Details can be

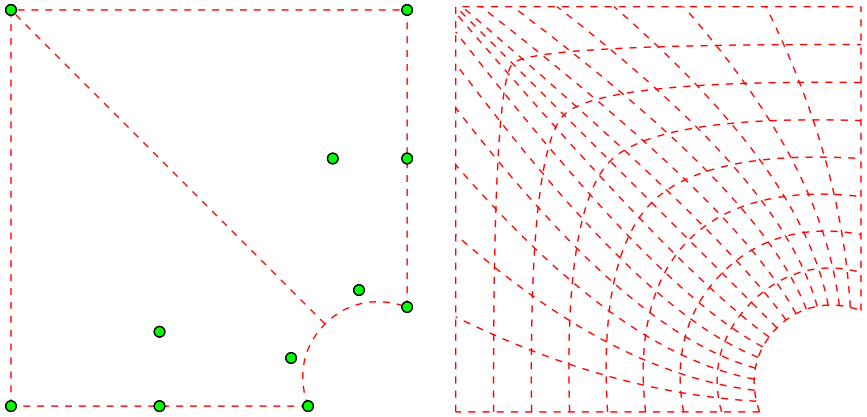


Figure 26: Plate with a hole: coarse mesh of 2 bi-quadratic elements (left) and refined mesh (right).

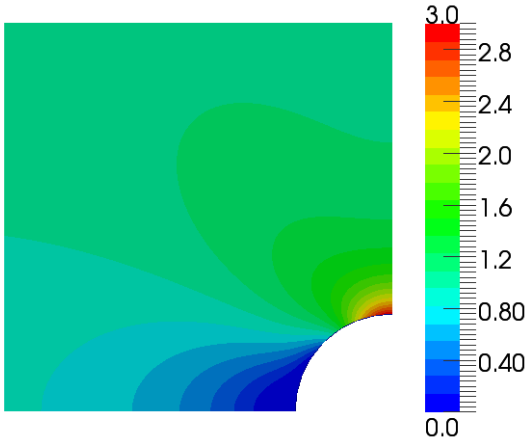


Figure 27: Plate with a hole: distribution of numerical σ_{xx} obtained with a 32×16 quadratic mesh having 4488 dofs.

found in the file **igaPinchedCylinder.m**. Fig. 29 shows the mesh and the contour plot of the displacement in the point load direction. Post-processing is done in Paraview and we refer to Section 6.4 for details. We recognise that the problem under consideration is a shell like structure that would be more accurately modelled using appropriate shell elements, but the example is merely intended to illustrate the ability of the method to analyse 3D geometries.

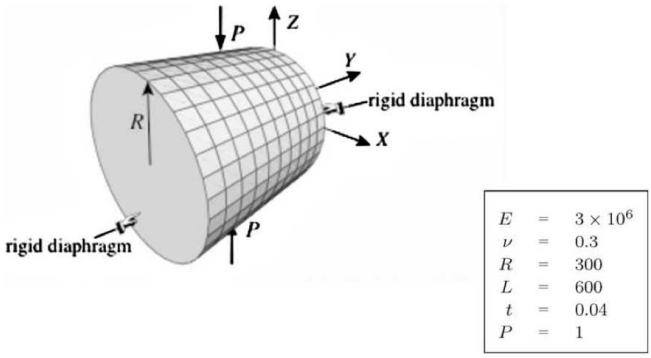


Figure 28: Pinched cylinder. Problem description and data [137].

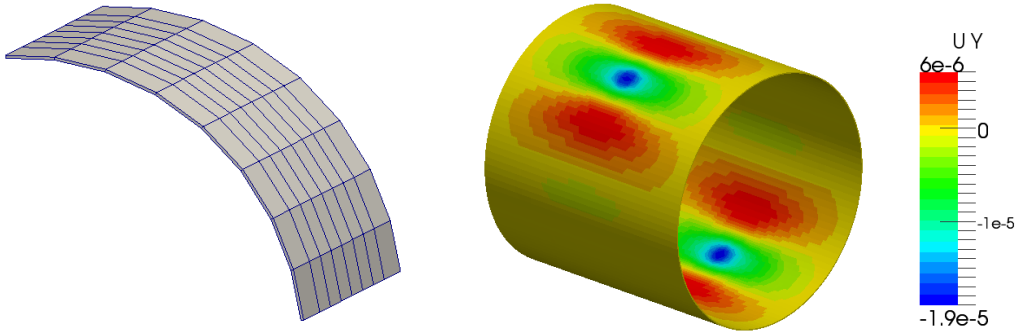


Figure 29: Pinched cylinder: (a) mesh of one octant of the cylinder and (b) contour plot of the displacement in the direction of the point load.

8.2. Two and three dimensional fracture mechanics

8.2.1. Edge cracked plate in tension

A plate of dimension $b \times 2h$ is loaded by a tensile stress $\sigma = 1.0$ along the top edge and bottom edge as shown in Fig. 30. In the computation, the displacement along the y-axis is fixed at the bottom edge and the bottom left corner

is fixed in both x and y directions. The material parameters are $E = 10^3$ and $\nu = 0.3$. A plane strain condition is assumed. The reference mode I stress intensity factor (SIF) for this problem is given in [138] and is calculated as

$$K_I = F\left(\frac{a}{b}\right)\sigma\sqrt{\pi a}, \tag{95}$$

where a is the crack length, b is the plate width, and $F(a/b)$ is an empirical function. For $a/b \leq 0.6$, function F is given by

$$F\left(\frac{a}{b}\right) = 1.12 - 0.23\left(\frac{a}{b}\right) + 10.55\left(\frac{a}{b}\right)^2 - 21.72\left(\frac{a}{b}\right)^3 + 30.39\left(\frac{a}{b}\right)^4. \tag{96}$$

In the present implementation, this problem is solved using both XFEM and an extended isogeometric formulation. The SIF is computed using an interaction integral. We refer to [70] for details.

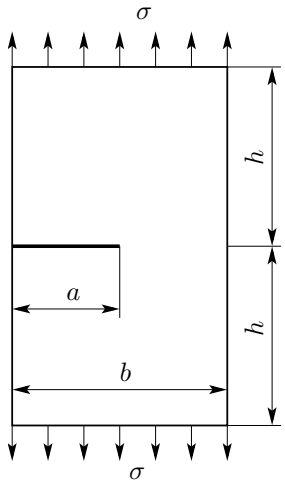
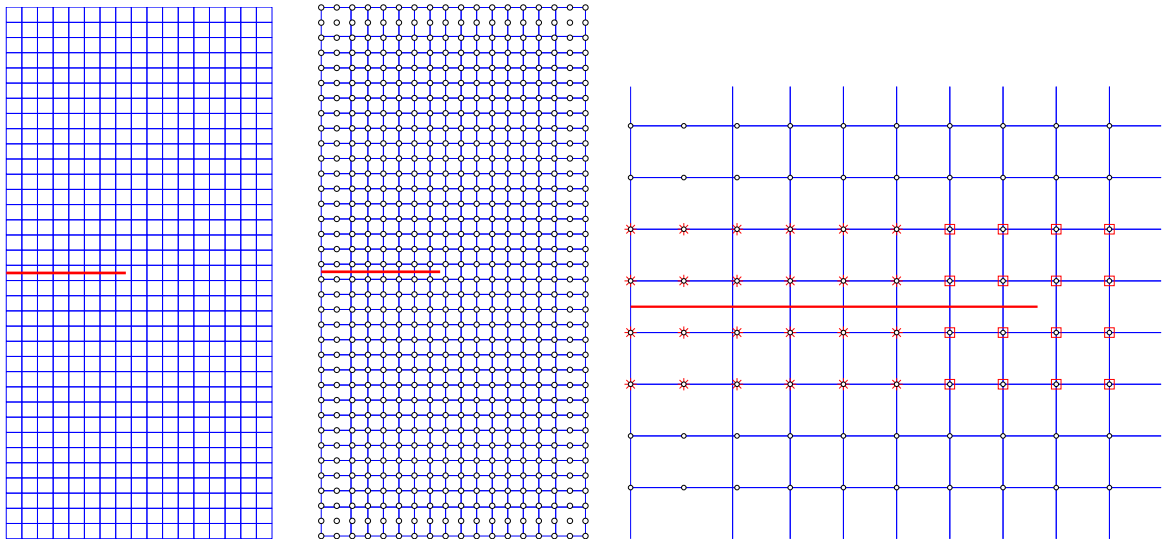


Figure 30: Edge cracked plate in tension: geometry and loading.

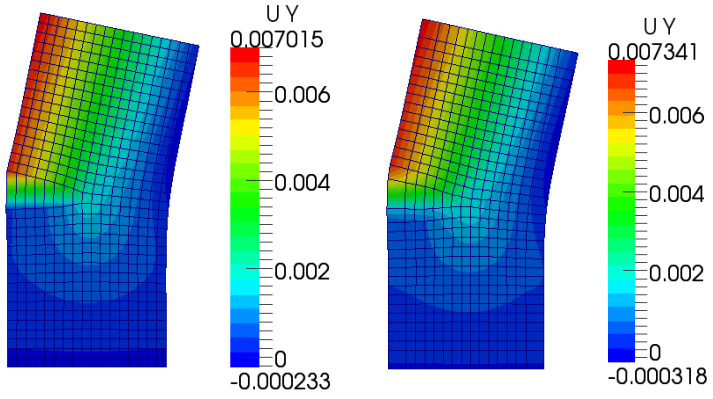
We first verify the implementation of the XIGA code by comparing the XIGA result with the XFEM result for the case of $a = 0.45$, $b = 1$ and $h = 1$. The XFEM and XIGA meshes are given in Fig. 31. Both meshes have the same uniform distribution of nodes/control points. For the XFEM mesh, bilinear Q4 elements are used. For the XIGA mesh, cubic ($p = q = 3$) B-spline basis functions are adopted. Fig. 32 shows the contour plots of the vertical displacement obtained with XFEM and XIGA.

We now consider the computation of the mode I SIF for a crack of length $a = 0.3$. The reference SIF for this



(a) XFEM mesh (Q4) (b) XIGA mesh (cubic Bsplines) (c) XIGA mesh, enriched nodes

Figure 31: Edge cracked plate: XFEM and XIGA meshes. Both have the same number of displacement dofs of 1296. The thick line denotes the crack. Square nodes denote tip enriched nodes whereas star nodes represent Heaviside enriched nodes.



(a) XFEM (b) XIGA

Figure 32: Edge cracked plate: u_y contour plots on deformed configuration (enlargement factor of 30 used).

problem is $K_I^{\text{ref}} = 1.6118$. Both a linear and a cubic B-spline basis are used for three different meshes. The results are given in Table 1. It should be emphasized that in the computation of the interaction integral, we use bilinear Lagrange shape functions i.e., shape functions of Q4 elements to compute the derivatives of the weight function. This guarantees that the weight function takes a value of unity on an open set containing the crack tip and vanishes on an outer contour as shown in Fig. 33. B-splines functions are not interpolatory and therefore cannot be used to approximate the weight functions.

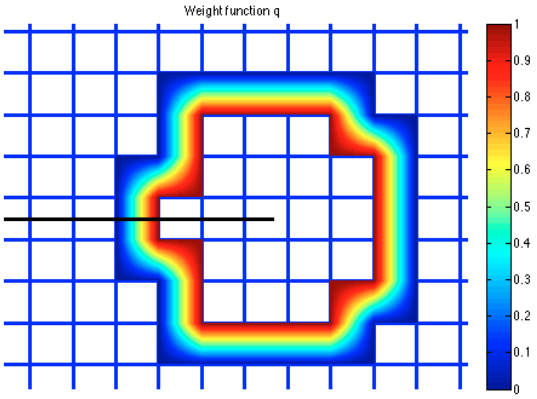


Figure 33: Distribution of weight function used in the computation of the interaction integral. Four-noded quadrilateral elements with bilinear Lagrange shape functions are used to interpolate the weight function.

mesh	disp. dofs	K_I (linear)	Error (%)	K_I (cubic)	Error (%)
9×18	324	1.4997	6.96	1.5560	3.46
18×36	1296	1.5823	1.83	1.6150	0.20
36×72	5184	1.5968	0.93	1.6117	0.01

Table 1: Edge cracked plate: SIFs results. The reference SIF is $K_I^{\text{ref}} = 1.6118$. Note that linear NURBS are equivalent to the conventional bilinear finite elements.

8.2.2. Three-dimensional mode I fracture problem

This example aims to show the capability of MIGFEM for solving three-dimensional (3D) fracture problems. For 3D cracks, the polar coordinates in the branch functions given in Eq. (83) are defined in terms of the level sets as [139]

$$r = \sqrt{\varphi(\xi,\eta,\zeta)^2 + \psi(\xi,\eta,\zeta)^2}, \quad \theta = \operatorname{atan}\left(\frac{\varphi(\xi,\eta,\zeta)}{\psi(\xi,\eta,\zeta)}\right), \tag{97}$$

where the level set field $\Phi = (\varphi, \psi)$ is interpolated as

$$\Phi(\xi,\eta,\zeta) = \sum_I R_I(\xi,\eta,\zeta)\Phi_I. \tag{98}$$

We refer to [139] for details concerning the derivatives of the branch functions with respect to the parametric coordinates (ξ,η,ζ).

The mode I 3D fracture problem we are solving is given in Fig. 34. The exact displacement field is given by

$$\begin{aligned} u_x(r,\theta) &= \frac{2(1+\upsilon)}{\sqrt{2\pi}}\frac{K_I}{E}\sqrt{r}\cos\frac{\theta}{2}\left(2-2\upsilon-\cos^2\frac{\theta}{2}\right) \\ u_y(r,\theta) &= 0 \\ u_z(r,\theta) &= \frac{2(1+\upsilon)}{\sqrt{2\pi}}\frac{K_I}{E}\sqrt{r}\sin\frac{\theta}{2}\left(2-2\upsilon-\cos^2\frac{\theta}{2}\right), \end{aligned} \tag{99}$$

where $K_I = \sigma\sqrt{\pi a}$ is the stress intensity factor, υ is Poisson's ratio and E is Young's modulus. In our example, $a = 100$ mm; $E = 10^7$ N/mm^2, $\upsilon = 0.3$, $\sigma = 10^4$ N/mm^2. On the bottom, right and top surfaces, essential BCs taken from Eq. (99) are imposed using the penalty method. A penalty parameter of $1e10$ was used. We note that this problem can be more effectively solved with two-dimensional elements. This example however aims at presenting how 3D extended IGA can be implemented. Furthermore, it also illustrates how Dirichlet BCs are enforced on surfaces rather than the usual case of line boundaries. To this end, a two-dimensional NURBS mesh for a given surface is generated from the set of control points that define this surface (see the file **surfaceMesh.m**).

The problem is first solved using a linear B-spline basis. A mesh of $9 \times 9 \times 1$ elements is used. The mesh, enriched nodes and comparison of the numerical deformed configuration against the exact profile are given in Fig. 35. Next, a

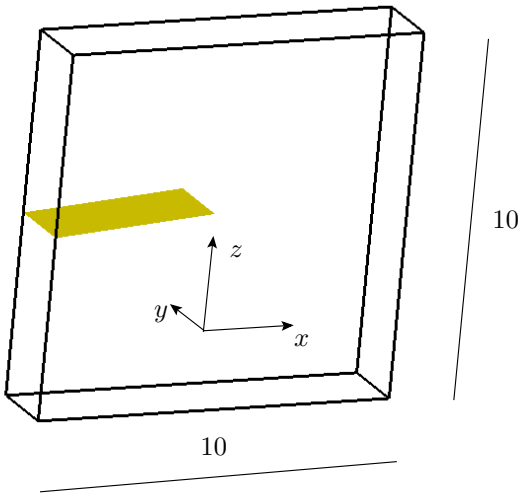


Figure 34: Three-dimensional mode I fracture problem: infinite plate with a center planar crack. The plate thickness is 2, the crack length is 5 and the crack width is 2. The crack is located in the mid-plane of the plate.

mesh of $7 \times 7 \times 2$ elements is used where, in the through-thickness direction, there are two linear elements ($q = 1$) and for the two other directions, a quadratic basis ($p = r = 2$) is used. The result is given in Fig. 36 and we note a good qualitative agreement between Figs. 35 and 36. We did not perform a SIF computation for this problem as 3D SIF computation is not yet implemented at present.

8.3. Structural mechanics

We consider the pull out of an open-ended cylindrical shell, see Fig. 37, as one of the most common benchmarks for geometrically nonlinear thin shell problems [140, 141]. Due to symmetry only 1/8 of the model is studied. The 1/8 model can be exactly described using one single quadratic-linear NURBS surface as shown in Fig.38. Refined meshes are then obtained from this initial mesh by using the k-refinement. The analysis was performed using a mesh of 8×8 bi-quartic NURBS elements (144 control points). The enforcement of symmetry BCs is achieved by constraining two row of control points as shown in the same figure. These constraints–the so-called multipoint linear homogeneous constraints are handled using the master-slave method [142] for the penalty method as described in Section 7.2 could endanger the convergence of the Newton-Raphson solver. Note that for this particular nonlinear problem, of which the M file is **igaGNLFreeEndCylinderShell.m**, the analysis was performed with a C++ implementation [143].

We adopt the Kirchhoff-Love thin shell model, see e.g., [144, 141] for details that involves only displacement dofs.

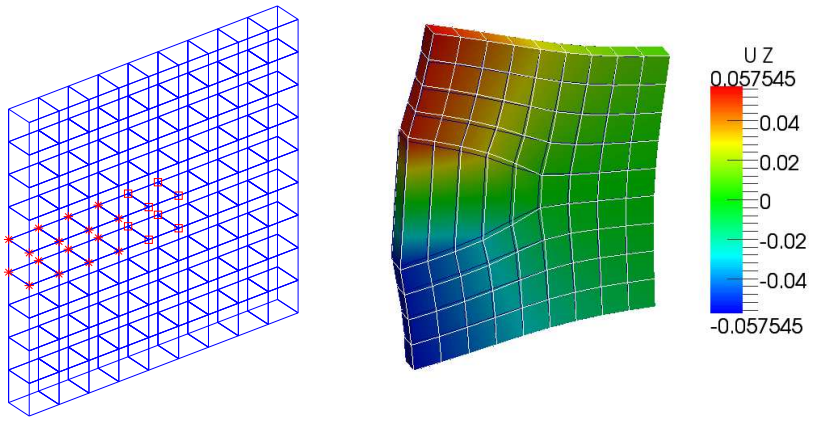


Figure 35: Three-dimensional mode I fracture problem: mesh of linear B-spline elements and enriched control points (left); numerical deformed configuration (magnification factor of 40) superimposed on the exact deformed configuration (right).

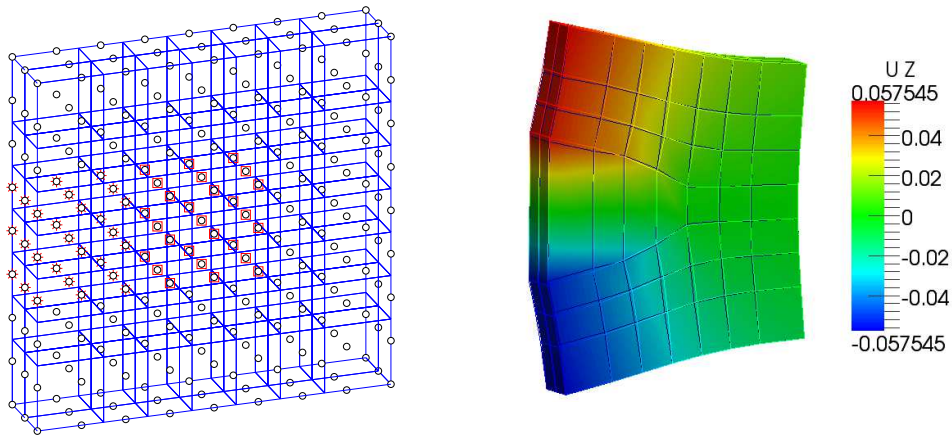


Figure 36: Three-dimensional mode I fracture problem: mesh of quadratic B-spline elements and enriched control points (left); numerical deformed configuration (magnification factor of 40) superimposed on the exact deformed configuration (right).

The maximum applied load $0.25P_{\max}$ is applied in 80 equal increments and for each increment the full Newton-Raphson method is used to solve for the displacements. Fig. 39 shows the deformed configuration of the shell and in Fig.40 the load versus the z-displacement at point A is plotted together with the result reported in [140].

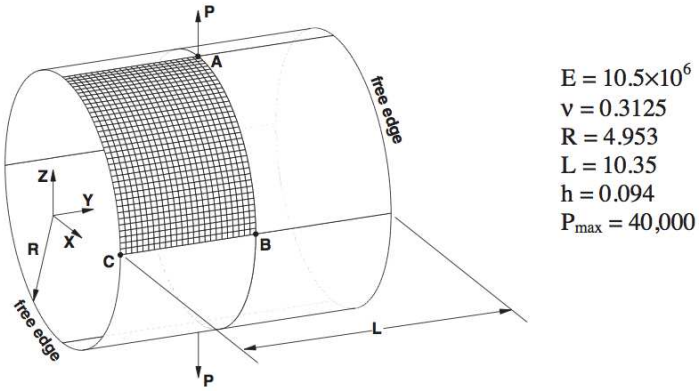


Figure 37: The open-end cylindrical shell subjected to radial pulling forces: problem setup [140].

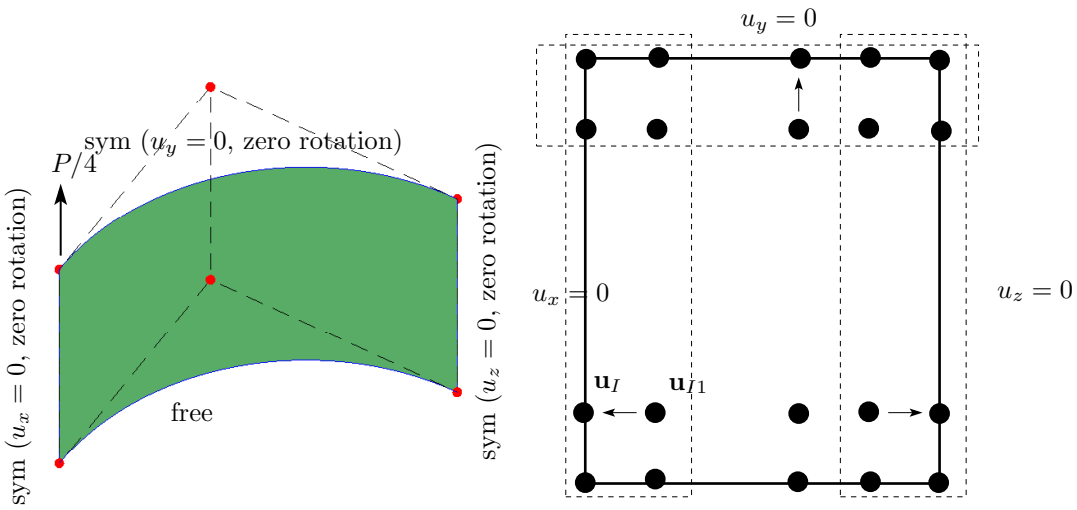


Figure 38: Open ended cylindrical shell: 1/8 of the model with one bi-quadratic NURBS surface. The control points and weights are given in file **freeEndsGNLCylinderShellData.m**. For symmetry edges, in order to satisfy the symmetry condition i.e., zero rotation, we constraint two rows of control points in the sense that the displacements of the control points right next to the control points locating on the symmetry edges ($\mathbf{u}_{I1}$) matches $\mathbf{u}_I$.

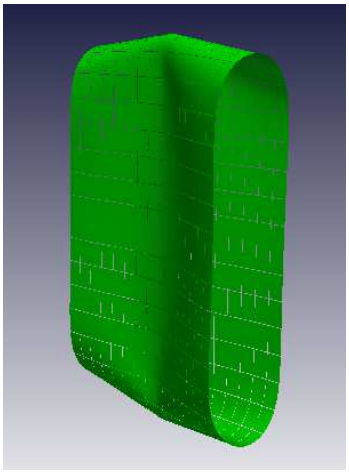


Figure 39: The open-ended cylindrical shell subjected to radial pulling forces: deformed shape without scaling.

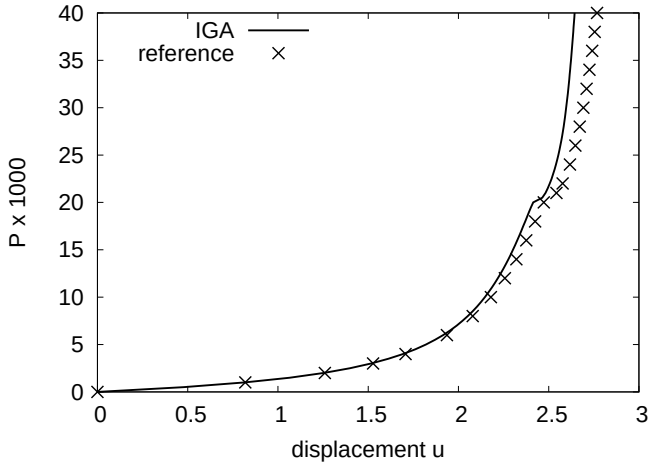


Figure 40: The open-ended cylindrical shell subjected to radial pulling forces: load-displacement responses.

9. Conclusion

We presented a Matlab® implementation for one, two and three-dimensional isogeometric finite element analysis for solid and structural mechanics. This paper is addressed to students or researchers who wish to learn the concepts of IGA in a clear and concise manner and is especially suited to those with solid mechanics applications in mind. NURBS are used throughout, where the underlying construction of the basis functions is detailed along with associated refinement algorithms essential for numerical analysis. Differences with conventional FE implementations are made clear with

the use of Matlab® source code to illustrate isogeometric FE concepts explicitly. In addition, the implementation of an isogeometric XFEM formulation for both two-dimensional and three-dimensional problems is described allowing for simple linear elastic fracture analysis to be performed directly from CAD data. The benefits of isogeometric analysis are made evident. Although not presented, the code supports multi-patch analysis in which compatibility between connecting patches is not required [145, 146]. Geometrical nonlinearities for solid elements under the Total Lagrange framework are provided as well. In addition, PUM enrichment for holes and inclusions is provided along with implementation of the least squares method for imposing essential boundary conditions. Mass matrices are implemented for almost every elements and popular time integration schemes such as Newmark and central difference explicit are available so that transient analysis on CAD objects can be performed. The code is available for download from `https://sourceforge.net/projects/cmcodes/` for Linux and Mac OS machines. A Windows version can be found at `http://www.softpedia.com/get/Science-CAD/igafem.shtml`.

The preliminary concepts and implementation details of isogeometric analysis have been described, but many challenges remain in the field. These include the creation of suitable volume discretisations from given CAD boundary representations, efficient integration schemes and suitable error estimators.

Acknowledgements

The authors would like to acknowledge the partial financial support of the Framework Programme 7 Initial Training Network Funding under grant number 289361 "Integrating Numerical Simulation and Geometric Design Technology". Stéphane Bordas also thanks partial funding for his time provided by 1) the EPSRC under grant EP/G042705/1 Increased Reliability for Industrially Relevant Automatic Crack Growth Simulation with the eXtended Finite Element Method and 2) the European Research Council Starting Independent Research Grant (ERC Stg grant agreement No. 279578) entitled "Towards real time multiscale simulation of cutting in non-linear materials with applications to surgical simulation and computer guided surgery". The first author would like to express his gratitude towards Professor L.J. Sluys at Delft University of Technology, The Netherlands for his support to VPN during the PhD period and the Framework Programme 7 Initial Training Network Funding. The authors acknowledge the comments of Dr. Robert Simpson at Cardiff University that has improved the paper.

Appendix A. Knot vector conventions

The definition of an open knot vector as $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ where the first and last knot values are repeated $p+1$ times is sometimes simplified to remove redundant information. It is found that if a knot vector is open, the first and last knot values play no role in the definition of the curve and can be removed entirely. Knot vectors are then defined as $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p-1}\}$ where the first and last knot vectors are repeated p times. This notation is used frequently through the CAGD community.

Furthermore, B-spline algorithms can be written entirely in terms of knot interval vectors [147], defined as

$$\Delta\Xi = \{\Delta\xi_1, \Delta\xi_2, \ldots, \Delta\xi_{n+p-2}\} \quad \text{where} \quad \Delta\xi_i = \xi_{i+1} - \xi_i \tag{A.1}$$

using the previous simplified knot vector notation. T-splines are based entirely on knot interval vector notation.

Appendix B. C^k and G^k continuity

Two types of continuity are commonplace in the CAGD community, referred to as C and G continuity. From a numerical analysis perspective, C continuity refers to the traditional definition of continuity in which, given a parametric function $f : \mathbb{R} \to \mathbb{R}^2$ which defines a 2D curve through a parameter t, the parameterisation is said to be C^k continuous at point a if $d^n f/dt^n$ is continuous (single-valued) for $n = 0, 1, \ldots, k$ at $t = a$. Intuitively, for a curve to be C^0, it must contain no discontinuities with t single-valued throughout its domain. Letting df/dt represent the 'speed' of the curve, C^1 continuity implies that the tangent vector is equal in magnitude and direction while also assuming C^0 continuity. C^2 continuity assumes both C^0 and C^1 continuity and requires that the 'acceleration' of the curve is equal in magnitude and direction.

G continuity (sometimes referred to as 'visual' continuity) is independent of the parameter t. For a curve to be G^0 continuous at a point it must contain no discontinuities, but the parameter value may have multiple values. For G^1 continuity, the tangent of the curve must be equal in its direction, but its magnitude may differ. And G^2 continuity implies that the direction of acceleration is continuous, but its magnitude may differ.

If desired, a G^k curve can be made C^k-continuous through an appropriate 'reparameterisation'.

Appendix C. Homogenous and non-homogenous coordinates

It is found that if rational basis functions are used, then it is often more appropriate to work in terms of homogeneous coordinates, since they can be applied straightforwardly to non-rational basis function algorithms. For example, in the case of NURBS, the use of homogenous coordinates allows B-spline algorithms to be used directly.

Given a physical coordinate $\mathbf{P}_A = (x_A, y_A, z_A)$ with weighting w_A, the corresponding four-dimensional homogeneous coordinate $\tilde{\mathbf{P}}_A$ is written as

$$\tilde{\mathbf{P}}_A = (x_A w_A, y_A w_A, z_A w_A, w_A) \tag{C.1}$$

$$= (\tilde{x}_A, \tilde{y}_A, \tilde{z}_A, w_A). \tag{C.2}$$

To convert back to non-homogenous coordinates, it is a simple case of dividing $\tilde{x}_A, \tilde{y}_A, \tilde{z}_A$ through by w_A.

Appendix D. Bézier extraction

Bézier extraction [130, 52] provides a tool to facilitate the incorporation of NURBS and even T-splines in to any FE codes. The idea is based on the fact that any B-splines basis can be written as a linear combination of Bernstein polynomials. Written for element e, the shape functions read

$$\mathbf{N}^e(\boldsymbol{\xi}) = \mathbf{C}^e \mathbf{B}(\tilde{\boldsymbol{\xi}}) \tag{D.1}$$

where $\mathbf{C}^e$ denotes the elemental Bézier extraction operator and $\mathbf{B}$ are the Bernstein polynomials which are defined on the parent element $\tilde{\Omega}$. Index space, knot vectors are all embedded in the Bézier extrators which are computed in a pre-processing step. Therefore, existing FE solvers can use NURBS/T-splines straightforwardly. We refer to folder **bezier-extraction** for examples.

References

[1] L. A. Piegl and W. Tiller. *The NURBS Book*. Springer, 1996.

[2] D. F. Rogers. *An Introduction to NURBS with Historical Perspective*. Academic Press, 2001.

[3] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005.

[4] J. A. Cottrell, T. J.R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009.

[5] P. Kagan, A. Fischer, and P. Z. Bar-Yoseph. New B-Spline Finite Element approach for geometrical design and mechanical analysis. *International Journal for Numerical Methods in Engineering*, 41(3):435–458, 1998.

[6] P. Kagan and A. Fischer. Integrated mechanically based CAE system using B-Spline finite elements. *Computer-Aided Design*, 32(8–9):539 – 552, 2000.

[7] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[8] İ. Temizer, P. Wriggers, and T.J.R. Hughes. Contact treatment in isogeometric analysis with NURBS. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1100–1112, 2011.

[9] L. Jia. Isogeometric contact analysis: Geometric basis and formulation for frictionless contact. *Computer Methods in Applied Mechanics and Engineering*, 200(5-8):726–741, 2011.

[10] İ. Temizer, P. Wriggers, and T.J.R. Hughes. Three-Dimensional Mortar-Based frictional contact treatment in isogeometric analysis with NURBS. *Computer Methods in Applied Mechanics and Engineering*, 209–212:115–128, 2012.

[11] L. De Lorenzis, İ. Temizer, P. Wriggers, and G. Zavarise. A large deformation frictional contact formulation using NURBS-bases isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 87(13):1278–1300, 2011.

[12] M.E. Matzen, T. Cichosz, and M. Bischoff. A point to segment contact formulation for isogeometric, NURBS based finite elements. *Computer Methods in Applied Mechanics and Engineering*, 255:27 – 39, 2013.

[13] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2976–2988, 2008.

[14] N. D. Manh, A. Evgrafov, A. R. Gersborg, and J. Gravesen. Isogeometric shape optimization of vibrating membranes. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1343–1353, 2011.

[15] X. Qian and O. Sigmund. Isogeometric shape optimization of photonic crystals via Coons patches. *Computer Methods in Applied Mechanics and Engineering*, 200(25-28):2237–2255, 2011.

[16] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199(29-32):2059–2071, 2010.

[17] R.N. Simpson, S.P.A. Bordas, J. Trevelyan, and T. Rabczuk. A two-dimensional isogeometric boundary element method for elastostatic analysis. *Computer Methods in Applied Mechanics and Engineering*, 209–212:87–100, 2012.

[18] M.A. Scott, R.N. Simpson, J.A. Evans, S. Lipton, S.P.A. Bordas, T.J.R. Hughes, and T.W. Sederberg. Isogeometric boundary element analysis using unstructured T-splines. *Computer Methods in Applied Mechanics and Engineering*, 254:197 – 221, 2013.

[19] D.J. Benson, Y. Bazilevs, M.C. Hsu, and T.J.R. Hughes. Isogeometric shell analysis: The Reissner–Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):276–289, 2010.

[20] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff-Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49-52):3902–3914, 2009.

[21] D.J. Benson, Y. Bazilevs, M.-C. Hsu, and T.J.R. Hughes. A large deformation, rotation-free, isogeometric shell. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1367–1378, 2011.

[22] L. Beirão da Veiga, A. Buffa, C. Lovadina, M. Martinelli, and G. Sangalli. An isogeometric method for the Reissner-Mindlin plate bending problem. *Computer Methods in Applied Mechanics and Engineering*, 209–212:45–53, 2012.

[23] T. K. Uhm and S. K. Youn. T-spline finite element method for the analysis of shell structures. *International Journal for Numerical Methods in Engineering*, 80(4):507–536, 2009.

[24] R. Echter, B. Oesterle, and M. Bischoff. A hierarchic family of isogeometric shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 254:170 – 180, 2013.

[25] D.J. Benson, S. Hartmann, Y. Bazilevs, M.-C. Hsu, and T.J.R. Hughes. Blended isogeometric shells. *Computer Methods in Applied Mechanics and Engineering*, 255:133 – 146, 2013.

[26] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff-Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering*, 199(37-40):2403–2416, 2010.

[27] Lu J. Circular element: Isogeometric elements of smooth boundary. *Computer Methods in Applied Mechanics and Engineering*, 198(30-32):2391–2402, 2009.

[28] J. Lu and X. Zhou. Cylindrical element: Isogeometric model of continuum rod. *Computer Methods in Applied Mechanics and Engineering*, 200(1-4):233–241, 2011.

[29] H. Gomez, T.J.R. Hughes, X. Nogueira, and V. M. Calo. Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. *Computer Methods in Applied Mechanics and Engineering*, 199(25-28):1828–1840, 2010.

[30] P. N. Nielsen, A. R. Gersborg, J. Gravesen, and N. L. Pedersen. Discretizations in isogeometric analysis of Navier-Stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 200(45-46):3242–3253, 2011.

[31] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method. *Journal of Computational Physics*, 229(9):3402–3414, 2010.

[32] Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and Y. Zhang. Isogeometric fluid-structure interaction: theory, algorithms, and computations. *Computational Mechanics*, 43:3–37, 2008.

[33] Y. Bazilevs, J.R. Gohean, T.J.R. Hughes, R.D. Moser, and Y. Zhang. Patient-specific isogeometric fluid-structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device. *Computer Methods in Applied Mechanics and Engineering*, 198(45-46):3534–3550, 2009.

[34] H. Gómez, V. M. Calo, Y. Bazilevs, and T.J.R. Hughes. Isogeometric analysis of the Cahn-Hilliard phase-field model. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4333–4352, 2008.

[35] C. V. Verhoosel, M. A. Scott, T. J. R. Hughes, and R. de Borst. An isogeometric analysis approach to gradient damage models. *International Journal for Numerical Methods in Engineering*, 86(1):115–134, 2011.

[36] P. Fischer, M. Klassen, J. Mergheim, P. Steinmann, and R. Müller. Isogeometric analysis of 2D gradient elasticity. *Computational Mechanics*, 47:325–334, 2010.

[37] A. Masud and R. Kannan. B-splines and NURBS based finite element methods for Kohn-Sham equations. *Computer Methods in Applied Mechanics and Engineering*, 241-244:112 – 127, 2012.

[38] J.A. Cottrell, A. Reali, Y. Bazilevs, and T.J.R. Hughes. Isogeometric analysis of structural vibrations. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5257–5296, 2006.

[39] T.J.R. Hughes, A. Reali, and G. Sangalli. Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS. *Computer Methods in Applied Mechanics and Engineering*, 197(49–50):4104 – 4124, 2008.

[40] C. H. Thai, H. Nguyen-Xuan, N. Nguyen-Thanh, T-H. Le, T. Nguyen-Thoi, and T. Rabczuk. Static, free vibration, and buckling analysis of laminated composite Reissner-Mindlin plates using NURBS-based isogeometric approach. *International Journal for Numerical Methods in Engineering*, 91(6), 2012.

[41] D. Wang, W. Liu, and H. Zhang. Novel higher order mass matrices for isogeometric structural vibration analysis. *Computer Methods in Applied Mechanics and Engineering*, pages –, 2013.

[42] J. A. Evans, Y. Bazilevs, I. Babuška, and T.J.R. Hughes. n-Widths, sup-infs, and optimality ratios for the k-version of the isogeometric finite element method. *Computer Methods in Applied Mechanics and Engineering*, 198(21-26):1726–1741, 2009.

[43] A. Shaw and D. Roy. Nurbs-based parametric mesh-free methods. *Computer Methods in Applied Mechanics and Engineering*, 197(17–18):1541 – 1567, 2008.

[44] Hyun-Jung Kim and Sung-Kie Youn. Spline-based meshfree method. *International Journal for Numerical Methods in Engineering*, 92(9):802–834, 2012.

[45] D. Großmann, B. Jüttler, H. Schlusnus, J. Barner, and Anh-Vu Vuong. Isogeometric simulation of turbine blades for aircraft engines. *Computer Aided Geometric Design*, 29(7):519 – 531, 2012.

[46] Thomas Elguedj, Julien Réthoré, and Aurélien Buteri. Isogeometric analysis for strain field measurements. *Computer Methods in Applied Mechanics and Engineering*, 200(1-4):40–56, 2011.

[47] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics*, 22:477–484, 2003.

[48] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Computer Methods in Applied Mechanics and Engineering*, 1437-1445:199, 2010.

[49] M.A. Scott, X. Li, T.W. Sederberg, and T.J.R. Hughes. Local refinement of analysis suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213-216:206–222, 2012.

[50] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[51] M. R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):264–275, 2010.

[52] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, 88(2):126–156, 2011.

[53] H. Wang, Y. He, X. Li, X. Gu, and H. Qin. Polycube splines. *Computer-Aided Design*, 40:721–733, 2008.

[54] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng. Polynomial splines over hierarchical T-meshes. *Graphical Models*, 70:76–86, 2008.

[55] T. Dokken and V. Skytt. Locally refined splines. In *Proceedings of IV European Conference On Computational Mechanics. Solids, Structures and Coupled Problems in Engineering*, Paris, France, 16-21 May 2010.

[56] N. Nguyen-Thanh, H. Nguyen-Xuan, S.P.A. Bordas, and T. Rabczuk. Isogeometric analysis using polynomial splines over hierarchical T-meshes for two-dimensional elastic solids. *Computer Methods in Applied Mechanics and Engineering*, 200(21-22):1892–1908, 2011.

[57] N. Nguyen-Thanh, J. Kiendl, H. Nguyen-Xuan, R. Wüchner, K.U. Bletzinger, Y. Bazilevs, and T. Rabczuk. Rotation free isogeometric thin shell analysis using PHT-splines. *Computer Methods in Applied Mechanics and Engineering*, 200(47-48):3410–3424, 2011.

[58] A.-V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49-52):3554–3567, 2011.

[59] D. Schillinger, L. Dedé, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. R. Hughes. An Isogeometric Design-through-analysis Methodology based on Adaptive Hierarchical Refinement of NURBS, Immersed Boundary Methods, and T-spline CAD Surfaces. *Computer Methods in Applied Mechanics and Engineering*, 242-252:116–150, 2012.

[60] P.B. Bornemann and F. Cirak. A subdivision-based implementation of the hierarchical b-spline finite element method. *Computer Methods in Applied Mechanics and Engineering*, 253:584 – 598, 2013.

[61] H. Speleers, C. Manni, F. Pelosi, and M. L. Sampoli. Isogeometric analysis with Powell-Sabin splines for advection-diffusion-reaction problems. *Computer Methods in Applied Mechanics and Engineering*, 221–222:132 – 148, 2012.

[62] S. K. Kleiss, B. Jüttler, and W. Zulehner. Enhancing isogeometric analysis by a finite element-based local refinement strategy. *Computer Methods in Applied Mechanics and Engineering*, 213–216:168 – 182, 2012.

[63] D. Burkhart, B. Hamann, and G. Umlauf. Iso-geometric Finite Element Analysis Based on Catmull-Clark : subdivision Solids. *Computer Graphics Forum*, 29(5):1575–1584, 2010.

[64] A. Wawrzinek, K. Hildebrandt, and K. Polthier. Koiter's thin shells on catmull-clark limit surfaces. In *Proceedings of the Vision, Modeling, and Visualization Workshop*, Berlin, Germany, 4-6 October 2011.

[65] Hyun-Jung Kim, Yu-Deok Seo, and Sung-Kie Youn. Isogeometric analysis with trimming technique for problems of arbitrary complex topology. *Computer Methods in Applied Mechanics and Engineering*, 199(45-48):2796–2812, 2010.

[66] R. Sevilla, S. Fernández-Méndez, and A. Huerta. 3D NURBS-enhanced finite element method (NEFEM). *International Journal for Numerical Methods in Engineering*, 88(2):103–125, 2011.

[67] R. Sevilla, S. Fernández-Méndez, and A. Huerta. NURBS-enhanced finite element method (NEFEM). *International Journal for Numerical Methods in Engineering*, 76(1):56–83, 2008.

[68] R. Schmidt, R. Wuchner, and K.U. Bletzinger. Isogeometric analysis of trimmed NURBS geometries. *Computer Methods in Applied Mechanics and Engineering*, 241–244:93 – 111, 2012.

[69] C. V. Verhoosel, M. A. Scott, R. de Borst, and T. J. R. Hughes. An isogeometric approach to cohesive zone modeling. *International Journal for Numerical Methods in Engineering*, 87(15):336–360, 2011.

[70] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999.

[71] E. De Luycker, D. J. Benson, T. Belytschko, Y. Bazilevs, and M. C. Hsu. X-FEM in isogeometric analysis for linear fracture mechanics. *International Journal for Numerical Methods in Engineering*, 87(6):541–565, 2011.

[72] S. S. Ghorashi, N. Valizadeh, and S. Mohammadi. Extended isogeometric analysis for simulation of stationary and propagating cracks. *International Journal for Numerical Methods in Engineering*, 2012. In Press.

[73] A. Tambat and G. Subbarayan. Isogeometric enriched field approximations. *Computer Methods in Applied Mechanics and Engineering*, 245–246:1 – 21, 2012.

[74] M. J. Borden, C. V. Verhoosel, M. A. Scott, T. J.R. Hughes, and C. M. Landis. A phase-field description of dynamic brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 217–220:77 – 95, 2012.

[75] V. P. Nguyen and H. Nguyen-Xuan. High-order B-splines based finite elements for delamination analysis of laminated composites. *Composite Structures*, 102:261–275, 2013.

[76] V. P. Nguyen, P. Kerfriden, and S. Bordas. Isogeometric cohesive elements for two and three dimensional composite delamination analysis. *Composites Science and Technology*, 2013. http://arxiv.org/abs/1305.2738.

[77] E. Cervera and J. Trevelyan. Evolutionary structural optimisation based on boundary representation of NURBS. Part I: 2D algorithms. *Computers and Structures*, 83:1902–1916, 2005.

[78] R.A.K. Sanches, P.B. Bornemann, and F. Cirak. Immersed B-spline (i-spline) finite element method for geometrically complex domains. *Computer Methods in Applied Mechanics and Engineering*, 200(13–16):1432 – 1445, 2011.

[79] E. Rank, M. Ruess, S. Kollmannsberger, D. Schillinger, and A. Düster. Geometric modeling, isogeometric analysis and the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 249–252(0):104 – 115, 2012.

[80] M. Moumnassi, S. Belouettar, E. Béchet, S.P.A. Bordas, D. Quoirin, and M. Potier-Ferry. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering*, 200(5–8):774 – 796, 2011.

[81] G. Legrain. A nurbs enhanced extended finite element approach for unfitted cad analysis. *Computational Mechanics*, pages 1–17, 2013.

[82] F. Auricchio, L. da Veiga Beirão, A. Buffa, C. Lovadina, A. Reali, and G. Sangalli. A fully locking-free isogeometric approach for plane linear elasticity problems: A stream function formulation. *Computer Methods in Applied Mechanics and Engineering*, 197(1-4):160–172, 2007.

[83] T. Elguedj, Y. Bazilevs, V.M. Calo, and T.J.R. Hughes. B-bar and f-bar projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2732–2762, 2008.

[84] R. Echter and M. Bischoff. Numerical efficiency, locking and unlocking of NURBS finite elements. *Computer Methods in Applied Mechanics and Engineering*, 199(5–8):374 – 382, 2010.

[85] T. Elguedj, Y. Bazilevs, V.M. Calo, and T.J.R. Hughes. F-bar projection method for finite deformation elasticity and plasticity using nurbs based isogeometric analysis. *International Journal of Material Forming*, 1(1):1091–1094, 2008.

[86] Rui P. R. Cardoso and J. M. A. Cesar de Sa. The enhanced assumed strain method for the isogeometric analysis of nearly incompressible deformation of solids. *International Journal for Numerical Methods in Engineering*, 92(1):56–78, 2012.

[87] R. L. Taylor. Isogeometric analysis of nearly incompressible solids. *International Journal for Numerical Methods in Engineering*, 87(1-5):273–288, 2011.

[88] S. Lipton, J.A. Evans, Y. Bazilevs, T. Elguedj, and T.J.R. Hughes. Robustness of isogeometric structural discretizations under severe mesh distortion. *Computer Methods in Applied Mechanics and Engineering*, 199(5–8):357 – 373, 2010.

[89] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V.M. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213–216:353 – 361, 2012.

[90] F. Auricchio, L. Da Veiga, T.J.R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010.

[91] F. Auricchio, L. Beiro da Veiga, T.J.R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation for elastostatics and explicit dynamics. *Computer Methods in Applied Mechanics and Engineering*, 249–252:2 – 14, 2012.

[92] T. Takacs and B. Jüttler. Existence of stiffness matrix integrals for singularly parameterized domains in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49-52):3568–3582, 2011.

[93] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Parameterization of computational domain in isogeometric analysis: Methods and comparison. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2021–2031, 2011.

[94] E. Cohen, T. Martin, R.M. Kirby, T. Lyche, and R.F. Riesenfeld. Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):334–356, 2010.

[95] R. Schmidt, J. Kiendl, K.-U. Bletzinger, and R. Wüchner. Realization of an integrated structural design process: analysis-suitable geometric modelling and isogeometric analysis. *Computing and Visualization in Science*, 13(7):315–330, 2010.

[96] Xianlian Zhou and Jia Lu. Nurbs-based galerkin method and application to skeletal muscle modeling. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, SPM '05, pages 71–78, New York, NY, USA, 2005. ACM.

[97] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and T.J.R. Hughes. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering*, 196(29-30):2943–2959, 2007.

[98] Y. Zhang, W. Wang, and T.J.R. Hughes. Solid T-spline construction from boundary representations for genus-zero geometry. *Computer Methods in Applied Mechanics and Engineering*, 249–252:185 – 197, 2012.

[99] Y. Zhang, W. Wang, and T.J.R. Hughes. Conformal solid T-spline construction from boundary T-spline representations. *Computational Mechanics*, pages 1–9, 2012.

[100] T. Martin, E. Cohen, and R. M. Kirby. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Computer Aided Geometric Design*, 26(6):648–664, 2009.

[101] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A.-V. Vuong. Swept volume parameterization for isogeometric analysis. In E.R. Hancock, R.R. Martin, and M.A. Sabin, editors, *Mathematics of Surfaces XIII*, volume 5654 of *Lecture Notes in Computer Science*, pages 19–44. Springer Berlin Heidelberg, 2009.

[102] J.M. Escobar, J.M. Cascn, E. Rodrguez, and R. Montenegro. A new approach to solid modeling with trivariate T-splines based on mesh optimization. *Computer Methods in Applied Mechanics and Engineering*, 200(45-46):3210–3222, 2011.

[103] M.J. Peake, J. Trevelyan, and G. Coates. Extended isogeometric boundary element method (xibem) for two-dimensional helmholtz problems. *Computer Methods in Applied Mechanics and Engineering*, 259:93 – 102, 2013.

[104] R.N. Simpson, S.P.A. Bordas, H. Lian, and J. Trevelyan. An isogeometric boundary element method for elastostatic analysis: 2D implementation aspects. *Computers & Structures*, 118:2 – 12, 2013.

[105] T.J.R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):301–313, 2010.

[106] F. Auricchio, F. Calabro, T.J.R. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 249–252:15 – 27, 2012.

[107] A.-V. Vuong, Ch. Heinrich, and B. Simeon. ISOGAT: a 2D tutorial MATLAB code for Isogeometric Analysis. *Computer Aided Geometric Design*, 27(8):644–655, 2010.

[108] C. de Falco, A. Reali, and R. Vázquez. GeoPDEs: a research tool for Isogeometric Analysis of PDEs. *Advances in Engineering Software*, 42(12):1020–1034, 2011.

[109] D. Rypl and B. Patzák. From the finite element analysis to the isogeometric analysis in an object oriented computing environment. *Advances in Engineering Software*, 44:116–125, 2012.

[110] D. J. Benson, Y. Bazilevs, E. De Luycker, M.-C. Hsu, M. Scott, T. J. R. Hughes, and T. Belytschko. A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to XFEM. *International Journal for Numerical Methods in Engineering*, 83(6):765–785, 2010.

[111] R.L. Taylor. A finite element analysis program, programmer manual. `http://www.ce.berkeley.edu/rlt/feap/`, 2001.

[112] L. Dalcin. PetIGA: A framework for high performance Isogeometric Analysis (using Petsc). `https://bitbucket.org/dalcinl/petiga`, 2011.

[113] A. Ratnani and E. Sonnendrucker. Isogeometric analysis in reduced magnetohydrodynamics. *Computational Science & Discovery*, 5(014007), 2012.

[114] A. Ratnani. Pigasus, Isogeometric Analysis simulations in Python. *INRIA report*, 2012.

[115] Rhino. CAD modeling and design toolkit. `www.rhino3d.com`.

[116] T.J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Mineola, NY, 2000.

[117] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10(5):307–318, 1992.

[118] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

[119] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813, 2008.

[120] J. Chessa. Programming the finite element method with Matlab. Northwestern University, http://www.tam.northwestern.edu/jfc795/Matlab/, 2002.

[121] D. Wang and J. Xuan. An improved NURBS-based isogeometric analysis with enhanced treatment of essential boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 199(37-40):2425–2436, 2010.

[122] T. Rabczuk, R. Gracie, J. H. Song, and T. Belytschko. Immersed particle method for fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 81(1):48–71, 2010.

[123] A. Embar, J. Dolbow, and I. Harari. Imposing dirichlet boundary conditions with nitsche's method and splinebased finite elements. *International Journal for Numerical Methods in Engineering*, 83(7):877–898, 2010.

[124] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[125] M. Arroyo and M. Ortiz. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *International Journal for Numerical Methods in Engineering*, 65(13):2167–2202, 2006.

[126] T. P. Fries and T. Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3):253–304, 2010.

[127] S. Bordas, P. V. Nguyen, C. Dunant, A. Guidoum, and H. Nguyen-Dang. An extended finite element library. *International Journal for Numerical Methods in Engineering*, 71(6):703–732, 2007.

[128] P. Laborde, J. Pommier, Y. Renard, and M. Salaun. High order extended finite element method for cracked domains. *International Journal for Numerical Methods in Engineering*, 190(47):6183–6200, 2004.

[129] A. Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware Inc., 2007.

[130] M. J. Borden, M. A. Scott, J. A. Evans, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering*, 87(15):15–47, 2011.

[131] N. Sukumar, D. L. Chopp, N. Moës, and T. Belytschko. Modelling holes and inclusions by level sets in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190:6183–6200, 2000.

[132] C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. *International Journal for Numerical Methods in Engineering*, 83(10):1273–1311, 2010.

[133] N. Moës, C. Stolz, P.-E. Bernard, and N. Chevaugeon. A level set based model for damage growth: The thick level set approach. *International Journal for Numerical Methods in Engineering*, 86(3):358–380, 2011.

[134] S. Natarajan, D. R. Mahapatra, and S. P. A. Bordas. Integrating strong and weak discontinuities without integration subcells and example applications in an XFEM/GFEM framework. *International Journal for Numerical Methods in Engineering*, 83(3):269–294, 2010.

[135] T. Elguedj, A. Gravouil, and A. Combescure. Appropriate extended functions for x-fem simulation of plastic fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 195(7–8):501 – 515, 2006.

[136] J. N. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill Education, 3rd edition, 2005.

[137] C. A. Felippa. Advanced Finite Element Methods (lecture notes). `http://www.colorado.edu/engineering/CAS/courses.d/AFEM.d/`, 2001.

[138] H. P. Tada, P. C. Paris, and G. R. Irwin. *The Stress Analysis of Cracks Handbook*. Del Research Corporation, St. Louis, MO, 1985.

[139] J. Shi, D. Chopp, J. Lua, N. Sukumar, and T. Belytschko. Abaqus implementation of extended finite element method using a level set representation of three-dimensional fatigue crack growth and life predictions. *Engineering Fracture Mechanics*, 77:2840–2863, 2010.

[140] K.Y. Sze, X.H. Liu, and S.H. Lo. Popular benchmark problems for geometric nonlinear analysis of shells. *Finite Elements in Analysis and Design*, 40(11):1551 – 1569, 2004.

[141] D. Millán, A. Rosolen, and M. Arroyo. Nonlinear manifold learning for meshfree finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 93(7):685–713, 2013.

[142] C. A. Felippa. Introduction to Finite Element Methods. Multifreedom constraints II (lecture notes). `http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/`, 2001.

[143] E. J. Lingen and M. Stroeven. Jem/Jive-a C++ numerical toolkit for solving partial differential equations. `http://www.habanera.nl/`.

[144] T.D. Nguyen and G. N. Wells. Geometrically nonlinear formulation for thin shells without rotation degrees of freedom. *Computer Methods in Applied Mechanics and Engineering*, 197(33–40):2778–2788, 2008.

[145] V. P. Nguyen, P. Kerfriden, M. Brino, S.P.A. Bordas, and E. Bonisoli. Nitsche's method for two and three dimensional nurbs patch coupling. *Computational Mechanics*, 2013. submitted.

[146] V. P. Nguyen, P. Kerfriden, S. Claus, and S.P.A. Bordas. A Nitsche's method for mixed dimensional analysis: conforming and non-conforming solid-structure coupling. 2013. In preparation.

[147] T. W. Sederberg. Computer Aided Geometric Design: Course Notes. `http://www.tsplines.com/educationportal.html`, 2010.