

Structural and Multidisciplinary Optimization

IgaTop: an implementation of topology optimization for structures using IGA in Matlab

--Manuscript Draft--

Manuscript Number:		
Full Title:	IgaTop: an implementation of topology optimization for structures using IGA in Matlab	
Article Type:	Research Paper	
Funding Information:	Fundamental Research Funds for the Central Universities of Huazhong University of Science and Technology (CN) (5003123021)	Dr Jie Gao
	Program for HUST Academic Frontier Youth Team (2017QYTD04)	Prof. Liang Gao
Abstract:	In this paper, the key intention is to present a compact and efficient Matlab code for the implementation of the Isogeometric Topology Optimization (ITO). A main function IgaTop2D with eight inputs in a 56-line Matlab code is developed, mainly including nine components, 1) Non-uniform rational B-splines (NURBS) to construct the geometrical model in a subfunction Geom_Mod; 2) A subfunction Pre_IGA to prepare the IsoGeometric Analysis (IGA); 3) Define Dirichlet and Neumann boundary conditions in a subfunction Boun_Cond; 4) Initialize control densities and the densities at Gauss quadrature points, implemented from line 11 to 20 of the main function; 5) A subfunction Shep_Fun to develop the smoothing mechanism; 6) IGA to solve the structural responses involving three steps: compute all IGA element stiffness matrices in Stiff_Ele2D subfunction, a subfunction Stiff_Ass2D to implement the assembly of all IGA element stiffness matrices, and a Solving subfunction; 7) Calculate the objective function and sensitivity analysis in lines 32-46 of the main function; 8) Update design variables using OC; 9) Present the optimized designs using Plot_Data and Plot_Topy subfunctions. Finally, several numerical examples are tested to show the effectiveness of the ITO Matlab implementation IgaTop2D, which are attached in the Appendix, also offering an entry point for newcomers who have interest in the field of ITO.	
Corresponding Author:	Jie Gao, Ph.D Huazhong University of Science and Technology WUHAN, Hubei CHINA	
Corresponding Author Secondary Information:		
Corresponding Author's Institution:	Huazhong University of Science and Technology	
Corresponding Author's Secondary Institution:		
First Author:	Jie Gao, Ph.D	
First Author Secondary Information:		
Order of Authors:	Jie Gao, Ph.D	
	Lin Wang, Ph.D	
	Zhen Luo, Ph.D	
	Liang Gao, Ph.D	
Order of Authors Secondary Information:		
Author Comments:	Thanks for your work to process this manuscript.	
Suggested Reviewers:	Timon Rabczuk, Ph.D Chair of Computational Mechanics, Bauhaus-Universität Weimar timon.rabczuk@tdtu.edu.vn The research focus of Prof. Rabczuk is Computational Mechanics with emphasis on method development for problems involving topology optimization for materials design	

	using isogeometric analysis, and he is not close to us
	Yingjun Wang, Ph.D Associate Professor, South China University of Technology wangyj84@scut.edu.cn The research focuses of A/Prof. Yingjun Wang mainly contains the developments of isogeometric topology optimization methods and applications, and he is not close to us.
	Qing-Hua Qin, Ph.D Professor, Australian National University qinghua.qin@anu.edu.au The research focuses of Prof. Qing-Hua Qin mainly contains the computational mechanics, but he is not close to us.
	Zhan Kang, Ph.D Professor, Dalian University of Technology zhankang@dlut.edu.cn Kang is an excellent researcher in topology optimization, but is not close to us. He has published many relevant papers in topology optimization field. In this paper, the density distribution function extends his basic works in constructing the material density field only by the Shepard function.
	Glaucio H. Paulino, Ph.D Professor, GeorgiaTech - Georgia Institute of Technology paulino@gatech.edu Glaucio H. Paulino is a famous profesor who focuses on the field of topology optimization for many years, and he is not close to us.
	Xiaodong Huang, Ph.D Professor, Swinburne University of Technology xhuang@swin.edu.au Prof. Xiaodong Huang is a famous professor who focuses on the field of topology optimization, and he is not close to us

IgaTop: an implementation of topology optimization for structures using IGA in Matlab

Jie Gao^{1,2†}, Lin Wang^{1,2}, Zhen Luo^{4*}, Liang Gao^{3**}

¹Department of Engineering Mechanics, School of Aerospace Engineering, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan, Hubei 430074, China

²Hubei Key Laboratory for Engineering Structural Analysis and Safety Assessment, Huazhong University of Science and Technology, Wuhan, 430074, China

³The State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan, Hubei 430074, China

⁴The School of Mechanical and Mechatronic Engineering, University of Technology Sydney, 15 Broadway, Ultimo, NSW 2007, Australia

[†]Corresponding author: Tel.: +86-27-8755 7742; E-mail: JieGao@hust.edu.cn

^{*}Corresponding author: Tel.: +61-2-95142994; E-mail: zhen.luo@uts.edu.au (A/Prof Z. Luo)

^{**}Corresponding author: Tel.: +86-27-8755 7742; E-mail: GaoLiang@mail.hust.edu.cn (Prof L. Gao)

Abstract

In this paper, the key intention is to present a compact and efficient Matlab code for the implementation of the Isogeometric Topology Optimization (ITO). A main function `IgaTop2D` with eight inputs in a 56-line Matlab code is developed, mainly including nine components, 1) Non-uniform rational B-splines (NURBS) to construct the geometrical model in a subfunction `Geom_Mod`; 2) A subfunction `Pre_IGA` to prepare the IsoGeometric Analysis (IGA); 3) Define Dirichlet and Neumann boundary conditions in a subfunction `Boun_Cond`; 4) Initialize control densities and the densities at Gauss quadrature points, implemented from line 11 to 20 of the main function; 5) A subfunction `Shep_Fun` to develop the smoothing mechanism; 6) IGA to solve the structural responses involving three steps: compute all IGA element stiffness matrices in `Stiff_Ele2D` subfunction, a subfunction `Stiff_Ass2D` to implement the assembly of all IGA element stiffness matrices, and a `Solving` subfunction; 7) Calculate the objective function and sensitivity analysis in lines 32-46 of the main function; 8) Update design variables using OC; 9) Present the optimized designs using `Plot_Data` and `Plot_Topy` subfunctions. Finally, several numerical examples are tested to show the effectiveness of the ITO Matlab implementation `IgaTop2D`, which are attached in the Appendix, also offering an entry point for newcomers who have interest in the field of ITO.

Keywords: Topology optimization; Isogeometric analysis; NURBS; Matlab.

1 Introduction

Since the seminal research that a homogenization method is applied to generate the optimal topologies in structural design (Bendsøe and Kikuchi 1988), the field of topology optimization has undergone extensive developments due to its superior capability to search for the optimal material distribution with the expected structural performance in a pre-defined design domain. Currently, several topology optimization methods with their specific functions have been developed in recent years, such as the Solid Isotropic Material with Penalization (SIMP) method (Zhou and Rozvany 1991; Bendsøe and Sigmund 1999), the Evolutionary Structural Optimization (ESO) method (Xie and Steven 1993), the Level-Set Method (LSM) (Wang et al. 2003; Allaire et al. 2004), the Moving Morphable Components/Voids (MMC/Vs) method (Guo et al. 2014; Yang et al. 2016; Zhang et al. 2017) and etc.. Meanwhile, these developed topology optimization methods have been also applied to discuss several numerical optimization problems, such as, the concurrent topology optimization (Xia and Breitkopf 2014; Li et al. 2016; Wang et al. 2017a; Gao et al. 2019b), materials design (Sigmund 1994; Xia and Breitkopf 2015), heat conduction (Kato et al. 2018; Zhao et al. 2020b) and etc..

As we know, the classic Finite Element Method (FEM) (Hughes 2012) has achieved a broad of applications in topology optimization to solve the unknown structural responses in the numerical implementation. In the FEM, spline basis functions are employed in the construction of the Computer-Aided Design (CAD) model, whereas Lagrangian and Hermitian polynomials are used in Computer-Aided Engineering (CAE) model.

The disunification might cause several limitations in the implementation of the FEM to solve the structural responses, 1) the finite element mesh can only approximate the initial structural geometry, which lower the numerical precision in analysis; 2) the low-order (C^0) continuity of structural responses exists between the neighboring finite elements, also in the higher-order finite elements; 3) a prohibitive time cost to achieve a finite element mesh with higher quality. As a promising and powerful alternative of the FEM, IsoGeometric Analysis (IGA) has been proposed by Hughes and his co-workers (Hughes et al. 2005; Cottrell et al. 2009) to perform finite element analysis. An important and superior feature that the unification of the CAD model and CAE model using the same spline basis functions in IGA can effectively remove the above numerical deficiencies of the FEM, which might offer more benefits for the latter optimization.

The first work (Seo et al. 2010a) addressed the shape optimization and discussed its extension to topological design based on IGA, and then realized isogeometric topology optimization using trimmed spline surfaces to represent the outer and inner boundaries of design (Seo et al. 2010b). In (Hassani et al. 2012), a control point based SIMP method was employed and IGA was applied to solve structural responses instead of the

FEM. (Shojaee et al. 2012) discussed the composition of IGA with LSM to realize the structural topology optimization. A phase field model for topology optimization and IGA for the spatial approximation in the analysis were discussed in (Dedè et al. 2012). (Qian 2013) developed a B-spline space for the density-based topology optimization. In (Wang and Benson 2016), an Isogeometric Topology Optimization (ITO) method that integrated the non-uniform rational b-splines (NURBS)-based IGA and a parametrized level set method (Wang and Wang 2006; Luo et al. 2007) was developed for the minimal compliance problems. Later, the combination of LSM and IGA has also been discussed in (Jahangiry and Tavakkoli 2017) for stress problem and (Ghasemi et al. 2017) for flexoelectric materials. The multiresolution topology optimization using IGA was also addressed (Lieu and Lee 2017a) and then applied to discuss the multi-material optimization (Lieu and Lee 2017b). The topology optimization for the multi-material and functionally graded structures using IGA was also studied in (Taheri and Suresh 2017). In (Hou et al. 2017), the combination of IGA and MMC to develop an explicit ITO method was studied. Later, the discussions about the developments of the explicit ITO method using MMC/Vs and IGA were extensively discussed (Xie et al. 2018, 2020; Gai et al. 2020; Du et al. 2020; Zhang et al. 2020). (Gao et al. 2019a) developed an effective and efficient ITO method using an enhanced density distribution function to display the structural topology and the IGA to solve the structural responses. The developed ITO method was then employed to study the rational design of auxetic metamaterials (Gao et al. 2019d), ultra-lightweight architected materials (Xu et al. 2020). Later, a Multi-material ITO (M-ITO) method (Gao et al. 2020a) was proposed on the basis of (Gao et al. 2019a) and then applied to discuss the computational design of auxetic composites with different deformation mechanisms (Gao et al. 2020b). The applications of IGA to discuss the design of auxetic metamaterials were also realized using shape optimization (Wang et al. 2017b; Wang and Poh 2018). An IGA-based parametric LSM with a model order reduction was developed for the design of auxetic metamaterials (Nguyen et al. 2020). The T-splines-based ITO method was developed in (Zhao et al. 2020a) to discuss the optimization of arbitrarily shaped design domains. The ITO for anisotropic metamaterials to control high-frequency electromagnetic wave was addressed in (Nishi et al. 2020).

Since the seminar work for the implementation of topology optimization based on the 99-line Matlab code (Sigmund 2001), a large number of educational papers with the compact Matlab codes have been published to considerably promote the developments of topology optimization. As a promising alternative of the 99-line Matlab code, a 88-line Matlab code for the implementation of topology optimization with the higher computational efficiency was developed in (Andreassen et al. 2011). The Matlab code for a discrete level-

set topology optimization was also given (Challis 2010). A 199-line Matlab code for Pareto-optimal tracing in topology optimization was discussed in (Suresh 2010). (Huang and Xie 2010) provided the details of the Matlab implementations of the ESO method. A Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes was addressed in (Talischi et al. 2012), and (Sanders et al. 2018) presented the implementation of the multi-material topology optimization in Matlab. A 3D topology optimization with the efficiency was also performed in Matlab by (Liu and Tovar 2014) and (Ferrari and Sigmund 2020) with new generation 99-line Matlab code. A Matlab code for materials design using topology optimization was presented in (Xia and Breitkopf 2015), and the Matlab implementation of concurrent topology optimization was also given in (Gao et al. 2019c). A Matlab code for a level set-based topology optimization method using a reaction diffusion equation is performed (Otomori et al. 2015). (Da et al. 2018) provided the Matlab code for the ESO method with smooth boundary representation. An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions was given in (Wei et al. 2018). The implementation of geometrically nonlinear structures using a 213-line Matlab code is shown in (Chen et al. 2019). The implementation of MMC method with the ersatz material model in Matlab is presented in (Zhang et al. 2016). (Liang and Cheng 2020) provided a 128-line Matlab code for the topology optimization via sequential integer programming and Canonical relaxation algorithm. As far as the implementation of IGA in Matlab, a suite of free software tools, namely GeoPDEs, for applications on IGA is provided in (de Falco et al. 2011), and a powerful version GeoPDEs 3.0 with a new design for the implementation of IGA in Octave and Matlab was presented (Vázquez 2016). A brief overview and systematically implementations of IGA was provided in (Nguyen et al. 2015). Meanwhile, a simplified introduction and implementation details for the incorporation of NURBS-based IGA technique within the existing FEA code was also given in (Agrawal and Gautam 2019).

The introducing of IGA into topology optimization instead of FEM to develop the ITO methods for several numerical optimization problems has received more and more attentions in recent years. The superior merits of IGA with more benefits in the topology optimization problems have been also shown in the above works. However, to the best knowledge of the authors, a systematic description about the implementation of the ITO method in Matlab is still in lack. Hence, in the current work, the main intention is to lower the barrier of the ITO to attract newcomers and serve as an entry-level tutoring for researchers who have an interest to familiarize with the ITO by providing a detailed Matlab implementation for the ITO method proposed in (Gao et al. 2019a). The rest of this paper is organized as follows: a brief description about the ITO method

for the compliance-minimization problem is given in Section 2, and Section 3 provides the implementations in detail for the ITO method. In Section 4, several numerical results are given to show the effectiveness of the current Matlab implementations, and the paper ends with the concluding remarks in Section 5.

2 The ITO formulation for the compliance-minimization

In (Gao et al. 2019a), an ITO method with more effectiveness and efficiency is proposed using an enhanced Density Distribution Function (DDF) and IGA. As we know, the basic intention of topology optimization is to seek for the optimal layout of materials in a design domain. In the current ITO method, the optimizer aims to find an optimal DDF with sufficient smoothness and continuity to represent the structural topology, and its iso-contour/surface represents the structural boundaries. In IGA, the same NURBS basis functions in the DDF are applied to develop the solution space for the unknown structural responses in the analysis. Overall, the densities at control points work as design variables in the optimization, and the mathematical formulation of the ITO method for the compliance-minimization can read as:

$$\left\{ \begin{array}{l} \text{Find: } \rho_{i,j} \quad (i = 1, 2, \dots, n; j = 1, 2, \dots, m) \\ \text{Min: } J(\mathbf{u}, \mathcal{X}) = \frac{1}{2} \int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{u})^T \mathbf{D}(\mathcal{X}(\xi, \eta)) \boldsymbol{\varepsilon}(\mathbf{u}) d\Omega \\ \text{S.t: } \begin{cases} G(\mathcal{X}) = \frac{1}{|\Omega|} \int_{\Omega} \mathcal{X}(\xi, \eta) v_0 d\Omega - V_{\max} \leq 0 \\ a(\mathbf{u}, \delta \mathbf{u}) = l(\delta \mathbf{u}), \quad \mathbf{u}|_{\Gamma_D} = \mathbf{g}, \quad \forall \delta \mathbf{u} \in H^1(\Omega) \\ 0 \leq \rho_{i,j} \leq 1 \end{cases} \end{array} \right. \quad (1)$$

where $\rho_{i,j}$ corresponds to the initial densities defined at control points, namely control densities. n and m denote the numbers of control points in two different parametric directions ξ and η , respectively. J is the objective function defined by the structural compliance. $\mathcal{X}$ is the DDF to represent the structural topology in the design domain Ω . G is the volume constraint, where v_0 is the volume fraction of solid material and $V_{\max}$ is the maximal material consumption. $\mathbf{u}$ denotes the displacement field in the design domain, which will be solved by IGA, rather than the FEM. $\mathbf{g}$ indicates the prescribed displacement vector on the Dirichlet boundary Γ_D . $\delta \mathbf{u}$ denotes the virtual displacement field belonging to the Sobolev space $H^1(\Omega)$. a is the bilinear energy function, and l is the linear load function, defined as:

$$\left\{ \begin{array}{l} a(\mathbf{u}, \delta \mathbf{u}) = \int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{u})^T \mathbf{D}(\mathcal{X}(\xi, \eta)) \boldsymbol{\varepsilon}(\delta \mathbf{u}) d\Omega \\ l(\delta \mathbf{u}) = \int_{\Omega} \mathbf{f} \delta \mathbf{u} d\Omega + \int_{\Gamma_N} \mathbf{h} \delta \mathbf{u} d\Gamma_N \end{array} \right. \quad (2)$$

where $\mathbf{f}$ is the body force, and $\mathbf{h}$ is the boundary traction on the Neumann boundary Γ_N .

3 IgaTop: A Matlab implementation of the ITO method

Before implementing the ITO in Matlab, we should be familiar with the basic conception of IGA with the spaces and their relationships. It should be noticed that IGA is only considered presently in the context of NURBS, namely the parametric space, physical space and parent space.

Parametric space: In the definition of NURBS for the structural geometry, the knot vectors with on ordered set of increasing parameters should be given. In IGA, the parametric space can be viewed as a pre-image of the NURBS mapping, and it is defined by the non-zero intervals of knot vectors. Because all knot vectors can be normalized, the corresponding parametric space can be reduced to a unit interval, square or cube. In the mathematical language, a symbol $\hat{\Omega} \in \mathbb{R}^d$ denotes the parametric space, and the related sets contain parametric coordinates $(\Xi, \mathcal{H}, \mathcal{Z})_{3D}$, in which $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$, $\mathcal{H} = \{\eta_1, \eta_2, \dots, \eta_{m+q+1}\}$ and $\mathcal{Z} = \{\zeta_1, \zeta_2, \dots, \zeta_{l+r+1}\}$. ξ , η and ζ denote three parametric directions, respectively. p , q and r are the corresponding orders of NURBS basis functions, respectively. n , m and l denote the numbers of NURBS basis functions, also control points, in three parametric directions, respectively. Hence, the parametric space can be also defined as: $\hat{\Omega} = [\xi_1, \xi_{n+p+1}] \otimes [\eta_1, \eta_{l+r+1}] \otimes [\zeta_1, \zeta_{m+q+1}]$.

Physical space: A series of control points in spatial should be chosen in the definition of structural geometry using NURBS, which constitute a control mesh in spatial. In the mathematical language, the symbol Ω is applied to denote the physical space with a coordinate system $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. A NURBS mapping is defined to transform the parametric space to the physical space. In the physical space, the non-interpolatory of control points at the structural geometry is a nature feature. It is a notable difference compared to the conventional Lagrangian meshes in the FEM.

Parent space: It is an additional space defined for the numerical integration in IGA to compute the stiffness matrices of all IGA elements, termed by $\tilde{\Omega} = [-1, 1]^{\mathbb{R}^d}$. The symbol $(\tilde{\xi}, \tilde{\eta}, \tilde{\zeta})$ is applied to denote the coordinates in the parent space.

Mappings: Compared to the conventional FEM, the use of NURBS basis functions to construct the solution space in analysis introduces the parametric space in IGA. The isoparametric formulation is also employed to evaluate the elementary stiffness matrices. Hence, two mappings should be defined in IGA, including a mapping $\mathbf{X}: \hat{\Omega} \rightarrow \Omega$ from the parametric space to the physical space, and an affine mapping $\mathbf{Y}: \tilde{\Omega} \rightarrow \hat{\Omega}$ from the parent space to the parametric space.

In this section, we provide a detailed description about the Matlab implementation of the ITO method for the compliance-minimization problem. A main function `IgaTop2D` with a 56-line Matlab code is defined for the implementation of the ITO, and it mainly includes eight components, namely construct geometrical model using NURBS (a sub function `Geom_Mod` with a 27-line Matlab code is implemented in line 5), preparation for IGA (a sub function `Pre_IGA` with a 39-line Matlab code is implemented in line 7), define Dirichlet and Neumann boundary conditions (a sub function `Boun_Con` with a 38-line Matlab code is implemented in line 9), initialize design variables and the DDF at Gauss quadrature points (lines 11-20), define the smoothing mechanism (a sub function `Shep_Fun` with a 22-line Matlab code is implemented in line 22), NURBS-based IGA to solve structural responses (a sub function `Stiff_Ele2D` with a 33-line Matlab code is called in line 28, a sub function `Stiff_Ass2D` with a 18-line Matlab code is called in line 29 and a sub function `Solving` with a 14-line Matlab code is called in line 30), compute objective function and sensitivity analysis (lines 32-46), the representation of the optimized solutions (a sub function `Plot_Data` with a 16-line Matlab code is called in line 25 and a sub function `Plot_Topy` with a 20-line Matlab code is called in line 47), and update design variables and the DDF (a sub function `OC` with a 14-line Matlab code is implemented in line 52). A simple illustration for the Matlab implementation of the ITO method is given in **Fig.1**.

As far as the implementation of the ITO for a simple case, the main function in 2D code is called from the Matlab prompt of the following line:

```
IgaTop2D(L, W, Order, Num, BoundCon, Vmax, penal, rmin)
```

where `L` and `W` denotes the structural sizes, namely the length and width, respectively. `Order` is an array contains two parameters which denotes the elevated orders of NURBS basis functions in two parametric directions. `Num` is also an array contains two parameters which denotes the total numbers of knots in the unit interval $[0, 1]$ with two different parametric directions. It should be noticed that the new knots are assumed to be uniformly inserted in the corresponding unit interval. `BoundCon` denotes the choice of the boundary and loads conditions. In the current Matlab code, five numerical cases will be discussed in the latter, namely the cantilever beam (`BoundCon = 1`), MBB beam (`BoundCon = 2`), Michell-type structure (`BoundCon = 3`), L beam (`BoundCon = 4`) and a quarter annulus (`BoundCon = 5`). `Vmax` is the maximal material consumption. `penal` is the penalty parameter in the optimization to push design variables towards 0 or 1. `rmin` is the parameter to control the influence area of the current control point in Shepard function, which is the radius length of the circle domain along the normal parametric directions.

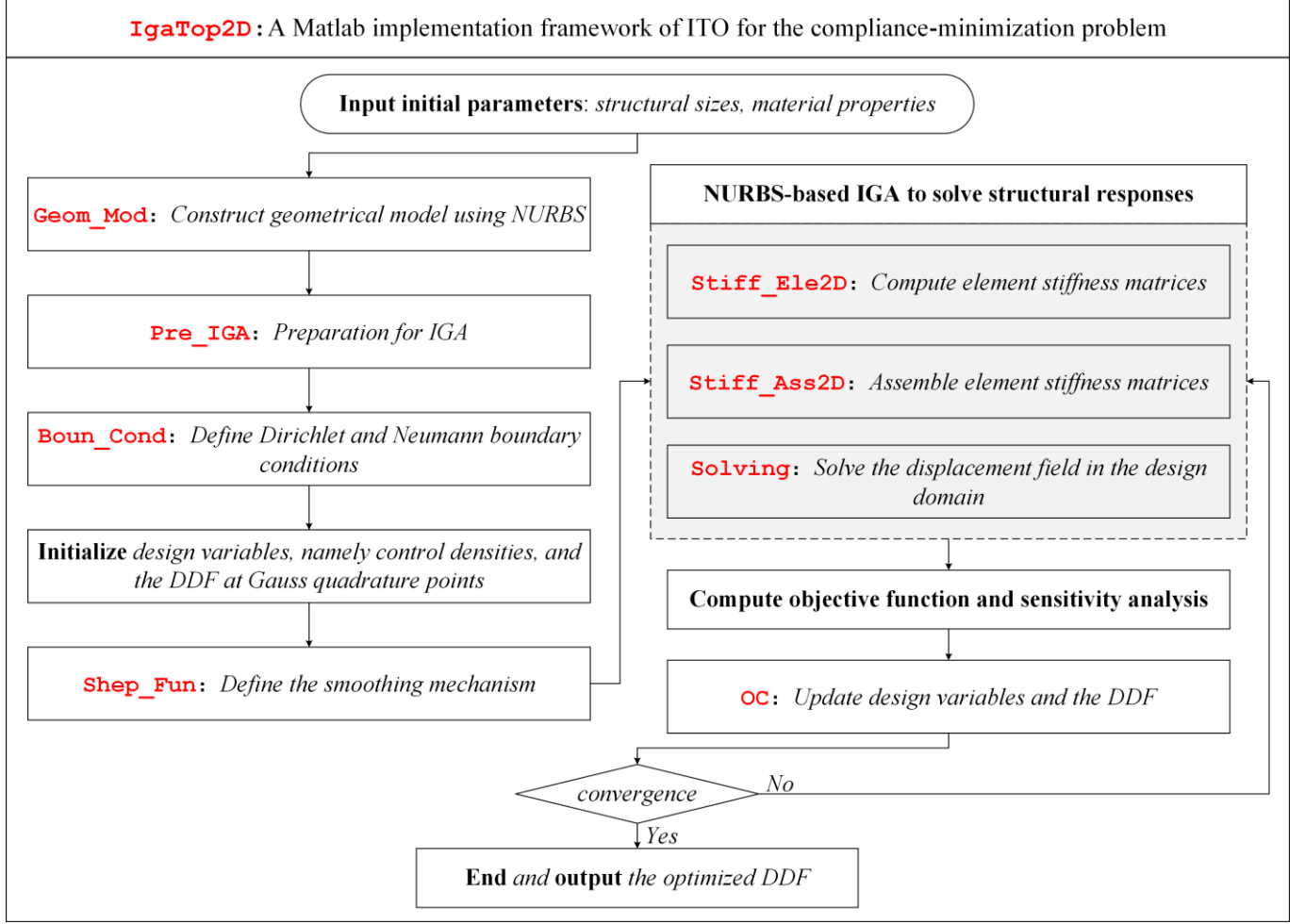


Fig.1 A Matlab implementation framework of ITO: IgaTop2D

3.1 Geom_Mod: Construct geometrical model using NURBS

As shown in **Fig.2**, a NURBS surface for a quarter annulus is given, where the structural geometry is shown in **Fig.2 (a)**, the corresponding NURBS surface is presented in **Fig.2 (b)** and the associated NURBS basis functions in two parametric directions are shown in **Fig.2 (c)** and **(d)**. We can easily find that the definition of the structural geometry using NURBS needs control points and the related NURBS basis functions. In **Fig.2 (b)**, control points plotted with the red color constitute the control grid in the spatial. The mathematical model of the NURBS surface is given as:

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m R_{i,j}^{p,q}(\xi, \eta) \mathbf{P}_{i,j} \quad (3)$$

where $\mathbf{S}$ denotes the NURBS surface for the structural geometry in 2D. $\mathbf{P}_{i,j}$ is the $(i,j)_{th}$ control point. $R_{i,j}^{p,q}$ is the NURBS basis function, which is defined by the B-spline basis functions, given as:

$$R_{i,j}^{p,q}(\xi, \eta) = \frac{N_{i,p}(\xi) M_{j,q}(\eta) \omega_{ij}}{\sum_{i=1}^n \sum_{j=1}^m N_{i,p}(\xi) M_{j,q}(\eta) \omega_{ij}} \quad (4)$$

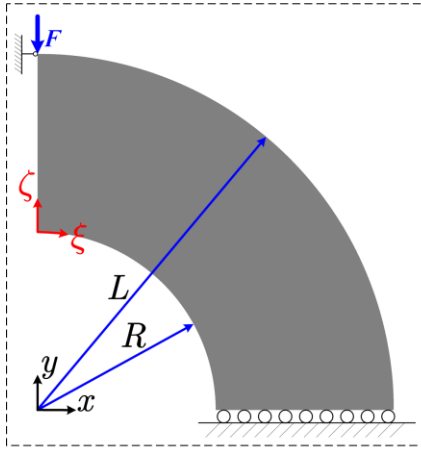
where $N_{i,p}$ and $M_{j,q}$ are the B-spline basis functions in two parametric directions, respectively. The B-spline basis functions are defined recursively using the Cox-de-Boor formula (De Boor 1978), starting with piecewise constants ($p = 0$),

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

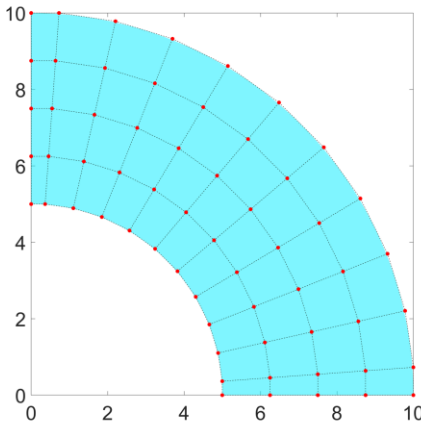
For $p \geq 1$, the B-spline basis functions are defined by:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \quad (6)$$

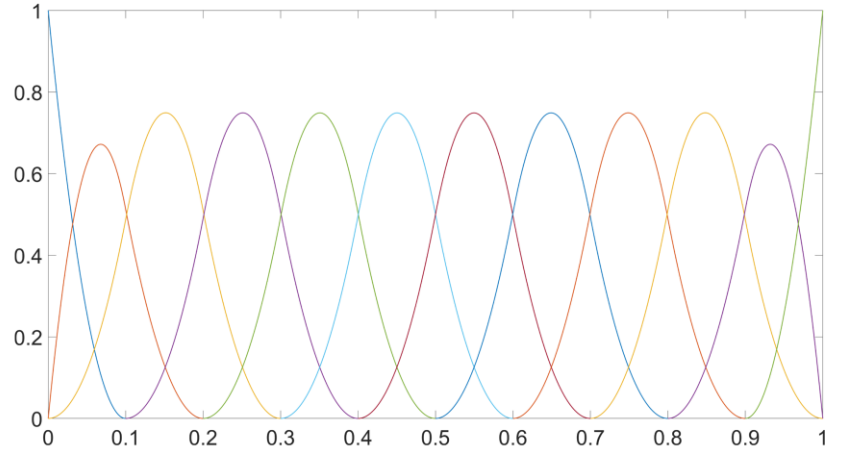
It should be noted that the fractions with the form 0/0 are equal to zero in Eq.(6).



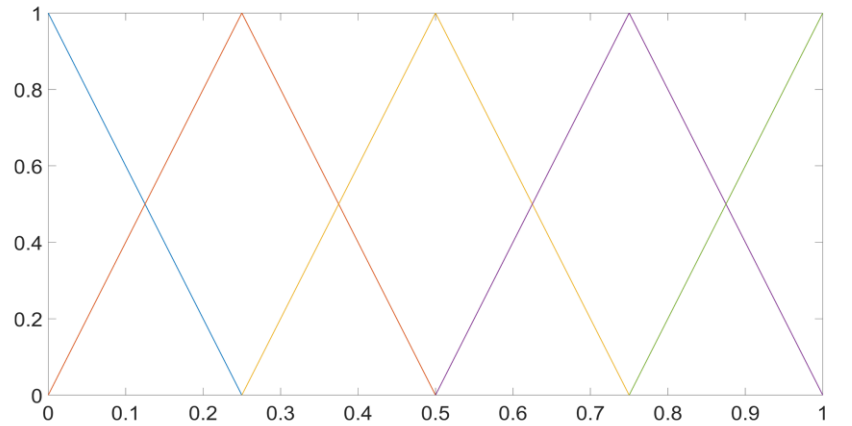
(a) Quarter annulus



(b) NURBS surface



(c) NURBS basis functions in the first parametric direction



(d) NURBS basis functions in the second parametric direction

Fig.2 A NURBS surface for a quarter annulus

As far as the implementation of NURBS to develop the structural geometry, the sub function `Geom_Mod` with five input parameters (`L`, `W`, `Order`, `Num` and `BoundCon`) is called from the Matlab prompt by the command in line 5 of the main function `IgaTop2D`:

```
NURBS = Geom_Mod(L, W, Order, Num, BoundCon)
```

The output parameter is NURBS. It is a structure array containing six fields, namely form, dim, number, coefs, knots and order. As far as an example of **Fig.2 (b)** with the input parameters (L=10, W=10, Order=[0 1], Num=[11 5] and BoundCon=5), the output parameter can read as:

NURBS	<pre> form: 'B-NURBS' dim: 4 number: [12 6] coefs: [4×12×6 double] knots: {[0 0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1] [0 0 0 0.25 0.50 0.75 1 1 1]} order: [3 3] </pre>
-------	--

Geom_Mod: In this sub function, lines 2-21 define the initial knot vectors in two parametric directions and the corresponding control points with the homogeneous coordinates $(\omega x, \omega y, \omega z, \omega)$ are also provided, in which ω denotes the weight parameter in the definition of NURBS basis functions. In the Geom_Mod, a NURBS toolbox developed by D.M. Spink (Spink et al. 2010), and the detailed numerical algorithms for NURBS can refer to (Piegl and Tiller 2012). The function nrbmak in the NURBS toolbox is applied to construct the NURBS surface 1 using the initial knot vectors and control points, presented in **Fig.3 (b)**. The function nrbdegelev is employed to elevate the order of NRUBS basis function in the second parametric direction, the corresponding NURBS surface 2 is displayed in **Fig.3 (c)**. Based on the NURBS surface 2, a series of new knots are uniformly inserted in the initial knot vectors, realized by the function nrbkntins. The corresponding NURBS surface 3 is shown in **Fig.3 (d)**. The detailed implementations to construct the NURBS surface, the elevation of the orders and the insertion of knots are called by the Matlab lines:

```

NURBS = nrbmak(coefs, knots);
NURBS = nrbdegelev(NURBS, Order);
NURBS = nrbkntins(NURBS, {setdiff(iknot_u, NURBS.knots{1}), ...
setdiff(iknot_v, NURBS.knots{2})});

```

The above process to develop the NURBS surface for the quarter annulus corresponds to the k -refinement, namely firstly elevate the orders of NURBS basis functions and secondly insert the knots in the initial knot vectors. Compared to the p -refinement, a much smaller number of NURBS basis functions are required in the k -refinement. The details about the k -refinement of NURBS can refer to (Cottrell et al. 2009). As far as

Geom_Mod, the corresponding Matlab code can read as:

```

1 function NURBS = Geom_Mod(L, W, Order, Num, BoundCon)
2 switch BoundCon
3     case {1, 2, 3}
4         knots{1} = [0 0 1 1]; knots{2} = [0 0 1 1];

```

```

5       ControlPts(:,:,1) = [0 L; 0 0; 0 0; 1 1];
6       ControlPts(:,:,2) = [0 L; W W; 0 0; 1 1];
7
1       case 4
2       8       knots{1} = [0 0 0.5 1 1]; knots{2} = [0 0 1 1];
3       9       ControlPts(:,:,1) = [0 0 L; L 0 0; 0 0 0; 1 1 1];
4       10      ControlPts(:,:,2) = [W W L; L W W; 0 0 0; 1 1 1];
5       11
6       11      case 5
7       12      W = W/2;
8       13      knots{1} = [0 0 0 1 1 1]; knots{2} = [0 0 1 1];
9       14      ControlPts(:,:,1) = [0 W W; W W 0; 0 0 0; 1 sqrt(2)/2 1];
10      15      ControlPts(:,:,2) = [0 L L; L L 0; 0 0 0; 1 sqrt(2)/2 1];
11
12     16 end
13
14     17 coefs = zeros(size(ControlPts));
15     18 coefs(1,::) = ControlPts(1,::).*ControlPts(4,::);
16     19 coefs(2,::) = ControlPts(2,::).*ControlPts(4,::);
17     20 coefs(3,::) = ControlPts(3,::).*ControlPts(4,::);
18     21 coefs(4,::) = ControlPts(4,::);
19
20     22 NURBS = nrbmak(coefs, knots);
21     23 NURBS = nrbdegelev(NURBS,Order);
22     24 nrbplot(NURBS,[100 100],'light','on')
23     25 iknot_u = linspace(0,1,Num(1)); iknot_v = linspace(0,1,Num(2));
24     26 NURBS =
25     nrbkntins(NURBS,{setdiff(iknot_u,NURBS.knots{1}),setdiff(iknot_v,NURBS.knots{2})});
26
27 end

```

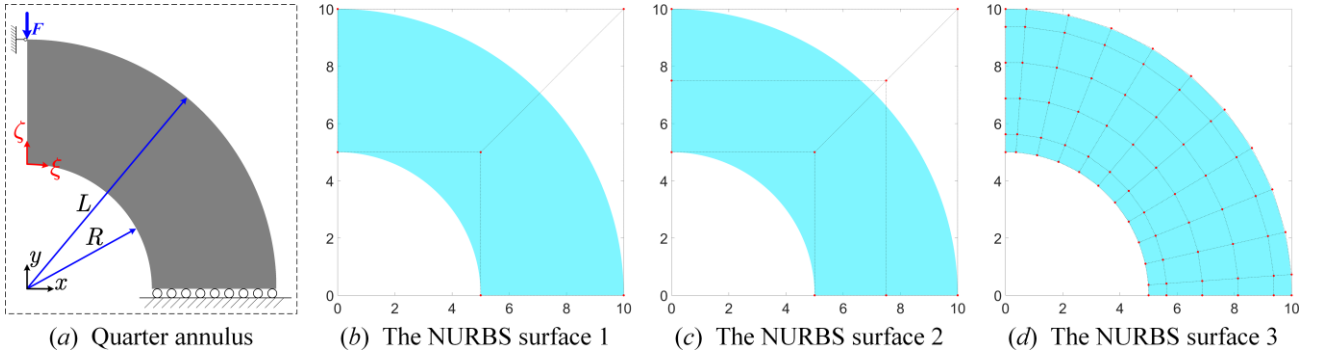


Fig.3 Three different NURBS surfaces for the quarter annulus

3.2 Pre_IGA: Preparation for IGA

In the preparation for IGA, the Matlab code focuses on the development of the numbers of control points, IGA elements, and Gauss quadrature points. The Matlab code for the preparation of IGA is called from the prompt of the following line with only one input parameter (NURBS):

```
[CtrPts, Ele, GauPts] = Pre_IGA(NURBS)
```

The output parameters are CtrPts, Ele, and GauPts. CtrPts is a structural array containing five fields, and the corresponding details are given as:

CtrPts.Cordis	The cartesian coordinates of control points in the physical space, namely (x, y, z, ω)
CtrPts.Num	The total number of control points
CtrPts.NumU	The total number of control points in the first parametric direction
CtrPts.NumV	The total number of control points in the second parametric direction
CtrPts.Seque	The numbers of all control points

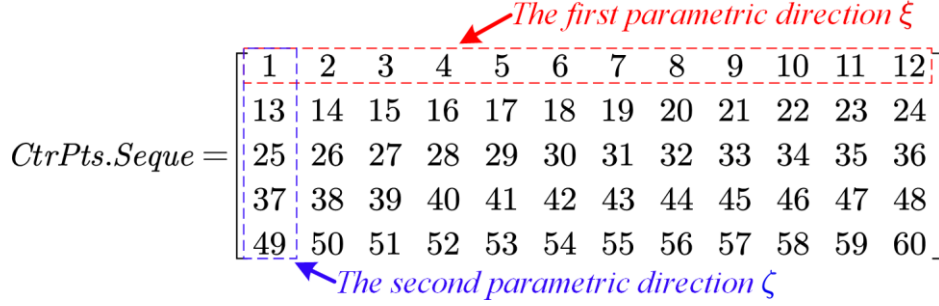


Fig.4 The numbers of all control points

The numbers of all control points with the corresponding parametric directions is shown in **Fig.4**. We can easily find that each control point are identified with the corresponding number ordered by the arc (the first parametric direction)-wise left-to-right and bottom-to-up, and the details of the numbers of control points are also presented in **Fig.7 (a)**.

Ele is also a structure array. Eleven fields are contained in this struct, namely 1) NumU: the total number of IGA elements in the first parametric direction; 2) NumV: the total number of IGA elements in the second parametric direction; 3) Num: the total number of IGA elements; 4) Seque: the numbers of IGA elements in the physical space, and the numbering manner in all IGA elements is same as control points, namely the arc-wise left-to-right and bottom-to-up, also shown in **Fig.7 (b)**; 5) KnotsU: the knot span related to each IGA element in the first parametric direction; 6) KnotsV: the knot span related to each IGA element in the second parametric direction; 7) CtrPtsNum: the total number of control points that have the influence on an IGA element, and also denotes the total number of nonzero NURBS basis functions in an IGA element; 8) CtrPtsNumU: the total number of control points that have the influence on each IGA element in the first parametric direction; 9) CtrPtsNumV: the total number of control points that have the influence on each IGA element in the second parametric direction; 10) CtrPtsCon: the numbers of control points that have the influence on each IGA element, the i -th row of this matrix contains the six (equal to CtrPtsNum) indices of control points that have the effect on the i -th IGA element, similar to the matrix edofMat in 88-line Matlab code (Andreassen et al. 2011), and the details of this matrix is given in **Fig.5**; 11) GauPtsNum: the number of Gauss quadrature points for the latter numerical integration in each IGA element.

$$Ele.CtrPtsCon = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 2 & 3 & 4 & 14 & 15 & 16 \\ 3 & 4 & 5 & 16 & 17 & 18 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 46 & 47 & 48 & 58 & 59 & 60 \end{bmatrix} \begin{matrix} \leftarrow \text{IGA element 1} \\ \leftarrow \text{IGA element 2} \\ \leftarrow \text{IGA element 3} \\ \\ \leftarrow \text{IGA element 40} \end{matrix}$$

Fig.5 The matrix of `Ele.CtrPtsCon`

`GauPts` is also a structure array, which includes six fields; namely 1) `QuaPts`: the coordinates of Gauss quadrature points in the bi-unit parent space; 2) `Weigh`: the corresponding weight parametric of each Gauss quadrature point; 3) `Num`: the total number of Gauss quadrature points; 4) `CorU`: the corresponding knots of Gauss quadrature points in the first parametric direction when mapping Gauss quadrature points from the parent space to parametric space; 5) `CorV`: the corresponding knots of Gauss quadrature points in the second parametric direction when mapping Gauss quadrature points from the parent space to the parametric space; 6) `Seque`: the numbers of Gauss quadrature points, the i -th row of this matrix has the nine (equal to `Ele.GauPtsNum`) indices of Gauss quadrature points which are located in the i -th IGA element. The details of the matrix `GauPts.Seque` are shown in **Fig.6**. The numbering mechanism of Gauss quadrature points in the quarter annulus is given in **Fig.7** (c), similar to the numbering way of control points.

$$GauPts.Seque = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 352 & 353 & 354 & 355 & 356 & 357 & 358 & 359 & 360 \end{bmatrix} \begin{matrix} \leftarrow \text{IGA element 1} \\ \leftarrow \text{IGA element 2} \\ \leftarrow \text{IGA element 3} \\ \\ \leftarrow \text{IGA element 40} \end{matrix}$$

Fig.6 The matrix of `GauPts.Seque`

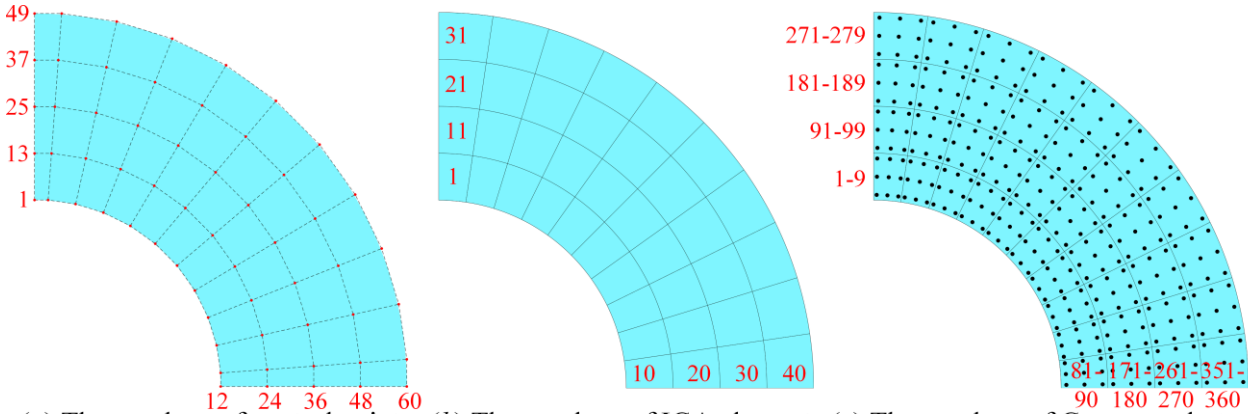


Fig.7 The numbers of control points, IGA elements and all Gauss quadrature points

`Pre_IGA`: This subfunction focuses on calculating the related data for the latter analysis. In lines 3-13, the Matlab implementation of the structural array `CtrPts` with five fields (`Cordis`, `Num`, `NumU`, `NumV` and `Seque`) is performed. The structural array `Ele` with eleven fields (`NumU`, `NumV`, `Num`, `Seque`, `KnotsU`, `KnotsV`, `CtrPtsNum`, `CtrPtsNumU`, `CtrPtsNumV`, `CtrPtsCon` and `GauPtsNum`) is implemented

from line 15 to 26 of the subfunction code. A subfunction Guadrature is called to calculate the Gauss quadrature points with their corresponding weights. The Matlab implementation of the GauPts array with six fields (QuaPts, Weigh, Num, CorU, CorV and Seque) is programed in lines 26-37. The details for the Matlab code of the subfunction Pre_IGA are given as:

```
1
2
3
4
5
6
7 1 function [CtrPts, Ele, GauPts] = Pre_IGA(NURBS)
8 2 %% the unique knots in two parametric directions
9
10 3 Knots.U = unique(NURBS.knots{1})';
11 4 Knots.V = unique(NURBS.knots{2})';
12
13 5 %% the information of control points including the physical
14   coordinates, numbers, sequence
15 6 CtrPts.Cordis = NURBS.coefs(:, :);
16 7 CtrPts.Cordis(1, :) = CtrPts.Cordis(1, :)./CtrPts.Cordis(4, :); % the
17   X Cartesian coordinates of control points;
18 8 CtrPts.Cordis(2, :) = CtrPts.Cordis(2, :)./CtrPts.Cordis(4, :); % the
19   Y Cartesian coordinates of control points;
20 9 CtrPts.Cordis(3, :) = CtrPts.Cordis(3, :)./CtrPts.Cordis(4, :); % the
21   Z Cartesian coordinates of control points;
22
23 10 CtrPts.Num = prod(NURBS.number); % the
24   total number of control points or basis functions;
25
26 11 CtrPts.NumU = NURBS.number(1); % the
27   total number of control points or basis functions in the first
28   parametric;
29
30 12 CtrPts.NumV = NURBS.number(2); % the
31   total number of control points or basis functions in the second
32   parametric;
33
34 13 CtrPts.Seque = reshape(1:CtrPts.Num, CtrPts.NumU, CtrPts.NumV)';
35
36 14 %% the information of the elements (knot spans) in the parametric
37   space, including the numbers, sequence
38
39 15 Ele.NumU = numel(unique(NURBS.knots{1}))-1; % the
40   number of elements in the first parametric direction
41
42 16 Ele.NumV = numel(unique(NURBS.knots{2}))-1; % the
43   number of elements in the second parametric direction
44
45 17 Ele.Num = Ele.NumU*Ele.NumV; % the
46   number of all elements in the structure
47
48 18 Ele.Seque = reshape(1:Ele.Num, Ele.NumU, Ele.NumV)';
49
50 19 Ele.KnotsU = [Knots.U(1:end-1) Knots.U(2:end)]; % the
51   unique knots of the elements in the first parametric direction
52
53 20 Ele.KnotsV = [Knots.V(1:end-1) Knots.V(2:end)]; % the
54   unique knots of the elements in the second parametric direction
55
56 21 Ele.CtrPtsNum = prod(NURBS.order);
57
58 22 Ele.CtrPtsNumU = NURBS.order(1); Ele.CtrPtsNumV = NURBS.order(2);
59
60 23 [~, Ele.CtrPtsCon] = nrbbasisfun((sum(Ele.KnotsU, 2)./2)',
61   (sum(Ele.KnotsV, 2)./2)'), NURBS);
62
63 24 %% the information of the Gauss quadrature points in the parent
64
65
```

```
space
25 [GauPts.Weigh, GauPts.QuaPts] = Quadrature(3, numel(NURBS.order));
26 Ele.GauPtsNum = numel(GauPts.Weigh);
1 27 GauPts.Num = Ele.Num*Ele.GauPtsNum;
2 28 GauPts.Seque = reshape(1:GauPts.Num, Ele.GauPtsNum, Ele.Num)';
3 29 GauPts.CorU = zeros(Ele.Num, Ele.GauPtsNum); GauPts.CorV =
4 zeros(Ele.Num, Ele.GauPtsNum);
5 30 for ide = 1:Ele.Num
6 31     [idv, idu] = find(Ele.Seque == ide); % The
7 two idices in two parametric directions for an element
8 32     Ele_Knot_U = Ele.KnotsU(idu, :); % The
9 knot span in the first parametric direction for an element
10 33     Ele_Knot_V = Ele.KnotsV(idv, :); % The
11 knot span in the second parametric direction for an element
12 34     for i = 1:Ele.GauPtsNum
13 35         GauPts.CorU(ide, i) = ((Ele_Knot_U(2) -
14 Ele_Knot_U(1)).*GauPts.QuaPts(i, 1) +
15 (Ele_Knot_U(2)+Ele_Knot_U(1)))/2;
16 36         GauPts.CorV(ide, i) = ((Ele_Knot_V(2) -
17 Ele_Knot_V(1)).*GauPts.QuaPts(i, 2) +
18 (Ele_Knot_V(2)+Ele_Knot_V(1)))/2;
19 37     end
20 38 end
21 39 end
22
23
24
25
26
27
28
29
30
31
32
```

33 The 20-line Matlab code for the subfunction Quadrature is also given as:

```
34
35 1 function [quadweight, quadpoint] = Quadrature(quadorder, dim)
36 2 quadpoint = zeros(quadorder^dim, dim);
37 3 quadweight = zeros(quadorder^dim, 1);
38 4 r1pt=zeros(quadorder, 1);
39 5 r1wt=zeros(quadorder, 1);
40 6 r1pt(1) = 0.774596669241483;
41 7 r1pt(2) = -0.774596669241483;
42 8 r1pt(3) = 0.000000000000000;
43 9 r1wt(1) = 0.5555555555555556;
44 10 r1wt(2) = 0.5555555555555556;
45 11 r1wt(3) = 0.8888888888888889;
46 12 n=1;
47 13 for i = 1:quadorder
48 14     for j = 1:quadorder
49 15         quadpoint(n, :) = [ r1pt(i), r1pt(j)];
50 16         quadweight(n) = r1wt(i)*r1wt(j);
51 17         n = n+1;
52 18     end
53 19 end
54 20 end
55
56
57
58
59
60
61
62
63
64
65
```

3.3 Boun_Conc: Define Dirichlet and Neumann boundary conditions

The Matlab implementation for the definition of Dirichlet and Neumann boundary conditions is realized in the sub function Boun_Conc with four input parameters, including CtrPts, BoundCon, NURBS and Dofs.Num. The corresponding Matlab line in the main function is given as:

```
[DBoudary, F] = Boun_Conc(CtrPts, BoundCon, NURBS, Dofs.Num)
```

There are two output parameters in the Matlab line, namely DBoudary and F. DBoudary is a structural array only containing one field CtrPtsOrd that denotes the imposed locations of the force in the physical space. However, if the total number of control points in one parametric direction is an even, and no control point will be located at the center of the physical space along one parametric direction, and it is possible to impose the corresponding force at the exact control point. Hence, a simple method defined in (Wang and Benson 2016) is employed here: 1) compute the parametric coordinates of the point that a force should be imposed; 2) evaluate all the nonzero NURBS basis functions at the current parametric coordinates R_s ; 3) impose the force $R_s F$ at all related control points. From line 3 to 29, the Dirichlet and Neumann boundary conditions for five numerical cases are provided, where the lines 5-8 of Matlab code are developed for the cantilever beam, lines 10-13 are defined for MBB beam, the Matlab code in lines 15-18 is programmed for Michell-type structure, the Matlab code for L beam is in lines 20-23 and lines 25-28 are developed for the quarter annulus. The numerical implementation of the imposed force in Matlab is programmed in lines 30-

38. The Matlab code of the subfunction Boun_Conc is given as:

```
1 function [DBoudary, F] = Boun_Conc(CtrPts, BoundCon, NURBS,
2   Dofs_Num)
3   %% boundary conditions
4   switch BoundCon
5       case 1 % Cantilever beam
6           DBoudary.CtrPtsOrd = CtrPts.Seque(:,1);
7           load.u = 1; load.v = 0.5;
8           [N, id] = nrbbasisfun([load.u; load.v], NURBS);
9           NBoudary.CtrPtsOrd = id'; NBoudary.N = N;
10      case 2 % MBB beam
11          DBoudary.CtrPtsOrd1 = CtrPts.Seque(1,1);
12          DBoudary.CtrPtsOrd2 = CtrPts.Seque(1,end);
13          load.u = 0.5; load.v = 1;
14          [N, id] = nrbbasisfun([load.u; load.v], NURBS);
15          NBoudary.CtrPtsOrd = id'; NBoudary.N = N;
16      case 3 % Michell-type structure
17          DBoudary.CtrPtsOrd1 = CtrPts.Seque(1,1);
```

```
DBoundary.CtrPtsOrd2 = CtrPts.Seque(1,end);
16     load.u = 0.5; load.v = 0;
17     [N, id] = nrbbasisfun([load.u; load.v], NURBS);
18     NBoundary.CtrPtsOrd = id'; NBoundary.N = N;
19 case 4 % L beam
20     DBoundary.CtrPtsOrd = CtrPts.Seque(:,1);
21     load.u = 1; load.v = 1;
22     [N, id] = nrbbasisfun([load.u; load.v], NURBS);
23     NBoundary.CtrPtsOrd = id'; NBoundary.N = N;
24 case 5 % A quarter annulus
25     DBoundary.CtrPtsOrd = CtrPts.Seque(:,end);
26     load.u = 0; load.v = 1;
27     [N, id] = nrbbasisfun([load.u; load.v], NURBS);
28     NBoundary.CtrPtsOrd = id'; NBoundary.N = N;
29 end
30 %% the force imposed on the structure
31 F = zeros(Dofs_Num,1);
32 switch BoundCon
33     case {1,2,3,4}
34         F(NBoundary.CtrPtsOrd+CtrPts.Num) = -1*NBoundary.N;
35     case 5
36         F(NBoundary.CtrPtsOrd) = -1*NBoundary.N;
37 end
38 end
```

3.4 Initialize control densities and the DDF at Gauss quadrature points

In the current ITO method, an enhanced DDF with the sufficient smoothness and continuity is developed to represent material distribution in the design domain. The construction of the DDF mainly involves three steps: 1) assign a series of discrete densities defined at control points, termed by control densities; 2) Define a smoothing mechanism to improve the smoothness of control densities; 3) A linear combination of NURBS basis functions (used in the construction of the NURBS surface for the initial structural geometry) with the smoothed control densities is applied to develop the corresponding DDF for the whole structural geometry. The mathematical equation of the DDF is given as:

$$\mathcal{X}(\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m R_{i,j}^{p,q}(\xi, \eta) \tilde{\rho}_{i,j} \quad (7)$$

In the optimization, the initial control densities work as design variables to derive the advancement of the DDF, until the optimal material distribution with the expected structural performance can be found. In the representation of designs, control densities, acting as design variables, should be provided. Meanwhile, the densities at Gauss quadrature points are calculated to evaluate the stiffness matrices of all IGA elements.

The representation of the DDF at Gauss quadrature points should be also given. Finally, it is noted that the DDF is a representation of the density in the whole design domain. The corresponding NURBS surface for the density should be also presented.

The initial control densities with the values equal to one are defined in line 11 of the main program. With an input parameter `GauPts.Cor` that denotes the coordinates of Gauss quadrature points in the parametric space, the Matlab command `nrbeval(NURBS, GauPts.Cor)` is called to compute the coordinates of Gauss quadrature points in the physical space, namely the outputs `GauPts.PCor` and `GauPts.Pw`. Then, the subfunction `nrbbasisfun` in NURBS toolbox is employed to evaluate the values of NURBS basis functions at the parametric coordinates of Gauss quadrature points, and the outputs contain `N` and `id`. In the matrix of `N`, the i -th row contains six values of NURBS basis functions at the corresponding parametric coordinate of the i -th Gauss quadrature point. In the matrix of `id`, the i -th row has six numbers of nonzero NURBS basis functions at the corresponding parametric coordinate of the i -th Gauss quadrature point. The first-order derivatives of NURBS basis functions with respect to parametric directions are calculated at the corresponding parametric coordinate of the i -th Gauss quadrature point using `nrbbasisfunder` a subfunction in the NURBS toolbox, denoted by `dRu` and `dRv`. Based on Eq.(7), the values of the DDF at Gauss quadrature points can be evaluated by calling the Matlab code in line 20. The corresponding Matlab code for the initialization of control densities and the DDF at Gauss quadrature points is given as:

```
X.CtrPts = ones(CtrPts.Num,1);
GauPts.Cor = [reshape(GauPts.CorU',1,GauPts.Num);
reshape(GauPts.CorV',1,GauPts.Num)];
[GauPts.PCor,GauPts.Pw] = nrbeval(NURBS, GauPts.Cor);
GauPts.PCor = GauPts.PCor./GauPts.Pw;
[N, id] = nrbbasisfun(GauPts.Cor, NURBS);
R = zeros(GauPts.Num, CtrPts.Num);
for i = 1:GauPts.Num, R(i,id(i,:)) = N(i,:); end
R = sparse(R);
[dRu, dRv] = nrbbasisfunder(GauPts.Cor, NURBS);
X.GauPts = R*X.CtrPts;
```

3.5 Shep_Fun: Define the smoothing mechanism

The main intention of the Shepard function is to improve the smoothness of the initial control densities in each iteration, and the corresponding mathematical model is given as:

$$\tilde{\rho}_{i,j} = \sum_{i=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{M}} \psi(\rho_{i,j}) \rho_{i,j} = \sum_{i=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{M}} \left(w(\rho_{i,j}) / \sum_{\hat{i}=1}^{\mathcal{N}} \sum_{\hat{j}=1}^{\mathcal{M}} w(\rho_{\hat{i},\hat{j}}) \right) \rho_{i,j} \quad (8)$$

where ψ is the Shepard function (Shepard 1968; Kang and Wang 2011, 2012). $\mathcal{N}$ and $\mathcal{M}$ are the total numbers of control densities located at the local support area of the current control density in two parametric directions respectively. Here, the compactly supported radial basis functions with C^4 continuity are adopted to define w , given as,

$$w(\vartheta) = (1 - \vartheta)_+^6 (35\vartheta^2 + 18\vartheta + 3) \quad (9)$$

where $\vartheta = r/r_m$, and r_m is the radius of the influence area. r is the Euclidean distance from the current control density and the other control density in the local support domain that corresponds to a blue circle area shown in **Fig.8**.

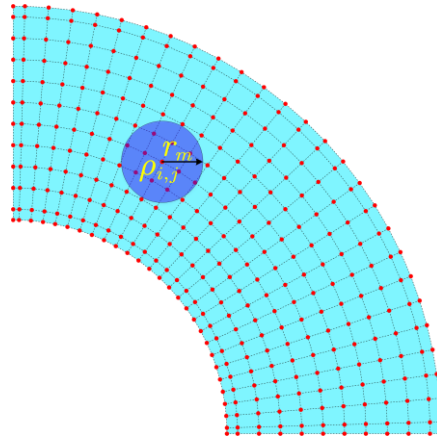


Fig.8 The Shepard function to smooth control densities

In the Matlab code, the Shepard function to define the smoothing mechanism is implemented by calling the line 22 with two input parameters (CtrPts and rmin) and two outputs (Sh and Hs). It is noted that rmin in the Matlab implementation denotes the radius length of the circle along the normal parametric directions, generally equal to 2. As we can see, the Matlab implementation of the smoothing mechanism is similar to the filtering mechanisms in the 88-line SIMP code (Andreassen et al. 2011). In actual, they have the intrinsic difference, and the detailed discussions can refer to (Gao et al. 2019a). The Matlab code of the subfunction Shep_Fun can read as:

```

1  function [Sh, Hs] = Shep_Fun(CtrPts, rmin)
2  Ctr_NumU = CtrPts.NumU; Ctr_NumV = CtrPts.NumV;
3  iH = ones(Ctr_NumU*Ctr_NumV*(2*(ceil(rmin)-1)+1)^2,1);
4  jH = ones(size(iH)); sH = zeros(size(iH));
5  k = 0;
6  for j1 = 1:Ctr_NumV
7      for i1 = 1:Ctr_NumU
8          e1 = (j1-1)*Ctr_NumU+i1;
9          for j2 = max(j1-(ceil(rmin)-1),1):min(j1+(ceil(rmin)-1), Ctr_NumV)

```

```
10         for i2 = max(i1-(ceil(rmin)-1),1):min(i1+(ceil(rmin)-1),Ctr_NumU)
11             e2 = (j2-1)*Ctr_NumU+i2;
12             k = k+1;
13             iH(k) = e1;
14             jH(k) = e2;
15             theta = sqrt((j1-j2)^2+(i1-i2)^2)./rmin/sqrt(2);
16             sH(k) = (max(0, (1-theta)).^6).*(35*theta.^2 +
17                 18*theta + 3);
18         end
19     end
20 end
21 Sh = sparse(iH,jH,sH); Hs = sum(Sh,2);
22 end
```

3.6 IGA to solve structural responses

In IGA, the same NURBS basis functions used in the above construction of the initial structural geometry and the DDF are kept unchanged to develop the solution space for the unknown structural responses, like the displacement, in analysis. Hence, the corresponding mathematical model is also given as:

$$\mathbf{U}(\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m R_{i,j}^{p,q}(\xi, \eta) \mathbf{U}_{i,j} \quad (10)$$

As we can see, Eq.(10) has the same mathematical form with Eq.(3) and (7), with only a minor change of physical meanings of control coefficients. In Eq.(3), the physical coordinates of control points are used. In Eq.(7), each control point is assigned by a density, and control densities work as the coefficients. In analysis, each control point will be also assigned by structural responses, namely control responses acting as control coefficients in the mathematical equation. In the Matlab implementation to solve structural responses, three key components are involved, namely 1) calculate stiffness matrices of all IGA elements, it is realized by calling a subfunction `Stiff_Ele2D` in line 28 of the main function; 2) assemble all IGA element stiffness matrices using a subfunction `Stiff_Ass2D` implemented in line 29 of the main program; and 3) solve the structural response by a subfunction `Solving` called in line 30.

3.6.1 `Stiff_Ele2D`

As far as the computation of IGA element stiffness matrices, the mathematical model can be given as:

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega_e \quad (11)$$

The isoparametric formulation is employed in the calculation of element stiffness matrix, and the detailed derivations in Matlab language can refer to (Gao et al. 2019c). In the evaluation of element stiffness matrix, the Gauss quadrature method is employed, and the corresponding integration is performed in a bi-unit area, namely the bi-unit parent element space. As already discussed, two mappings $\mathbf{X}$ from the parametric space to physical space and $\mathbf{Y}$ from the parent space to parametric space should be defined in IGA. An illustration in detail of these two mappings is provided in Fig.9.

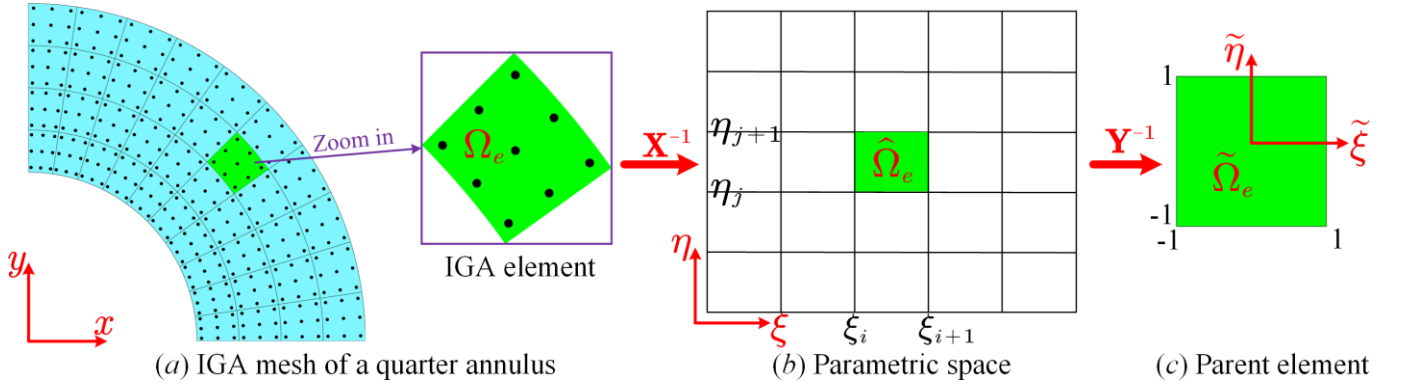


Fig.9 The numerical integration of IGA element

Hence, Eq.(11) can be transformed into an another form, given as:

$$\mathbf{K}_e = \int_{\tilde{\Omega}_e} \mathbf{B}^T \mathbf{D} \mathbf{B} |\mathbf{J}_1| |\mathbf{J}_2| d\tilde{\Omega}_e \quad (12)$$

where $\mathbf{J}_1$ and $\mathbf{J}_2$ are the Jacobi matrices of these two mappings $\mathbf{X}$ and $\mathbf{Y}$, respectively. Meanwhile, the Gauss quadrature method is employed to calculate Eq.(12). It is noticed that nine Gauss quadrature points are chosen in each IGA element to solve the IGA element stiffness matrices. Eq.(12) can be transformed into a new form, given as:

$$\mathbf{K}_e = \sum_{i=1}^3 \sum_{j=1}^3 \{ \mathbf{B}^T(\xi_i, \eta_j) (\mathcal{X}(\xi_i, \eta_j))^T \mathbf{D}_0 \mathbf{B}(\xi_i, \eta_j) |\mathbf{J}_1| |\mathbf{J}_2| \varpi_i \varpi_j \varpi_k \} \quad (13)$$

The Matlab implementation to solve all IGA element stiffness matrices is called in line 28, given as:

```
[KE, dKE, dv_dg] = Stiff_Ele2D(X, penal, Emin, DH, CtrPts, Ele,
GauPts, dRu, dRv);
```

The subfunction `Stiff_Ele2D` has twelve input parameters, namely 1) `X` is a structural array containing three fields to present the distribution of densities, including `CtrPts` field for control densities, `GauPts` field for the densities at Gauss quadrature points and `DDF` field for the densities in design domain; 2) `penal` is the penalty parameter, equal to 3; 3) `Emin` is the Young's modulus of void materials to avoid numerical

singularity; 4) DH is material constitutive elastic tensor matrix; 5-7) three structure arrays $CtrPts$, Ele and $GauPts$; 8-9) dRu and dRv corresponds to the first-order derivatives of NURBS basis functions with respect to two parametric directions, respectively. $Stiff_Ele2D$ outputs three parameters, in which KE is a cell matrix containing all IGA element stiffness matrices, dKE is a cell matrix containing the first-order derivatives of all IGA element stiffness matrices with respect to the densities of the Gauss quadrature points. dv_dg includes the first-order derivatives of volume constraint with respect to the densities of the Gauss quadrature points.

$Stiff_Ele2D$: the initial definitions of KE , dKE and dv_dg are implemented in lines 2-4. In lines 6-32, a Matlab loop is programmed to compute all IGA element stiffness matrices. The knots, numbers of control points and their corresponding physical coordinates of control points in each IGA element are firstly called in lines 7-11. Then the computation of the current IGA element stiffness matrix is completed in a sub loop from line 14 to 29. In this sub loop, Jacobi matrix J_1 which denotes the first-order derivatives of the physical space with respect to parametric space is firstly calculated in lines 15-18, including $dPhy_dPara$ and $J1$. The strain-displacement matrix in each IGA element is calculated from line 19 to 21, namely Be . In lines 22-24, the second Jacobi matrix J_2 which denotes the first-order derivatives of the parametric space with respect to the parent space is calculated, namely $J2$. Then, the current IGA element stiffness matrix Ke , its derivatives dKe with respect to the densities of Gauss quadrature points in the current IGA element and the first-order derivatives of volume constraint dv_dg with respect to the densities of Gauss quadrature points in the current IGA element are calculated in lines 26-28, respectively. In the Matlab code, the calculation of IGA element stiffness matrices needs the information containing physical coordinates of control points, the densities at the Gauss quadrature points and so on. The code of the subfunction $Stiff_Ele2D$ is given as follows:

```
1 function [KE, dKE, dv_dg] = Stiff_Ele2D(X, penal, Emin, DH, CtrPts,
2   Ele, GauPts, dRu, dRv)
3
4 KE = cell(Ele.Num,1);
5 dKE = cell(Ele.Num,1);
6 dv_dg = zeros(GauPts.Num,1);
7 Nen = Ele.CtrPtsNum;
8 for ide = 1:Ele.Num
9     [idv, idu] = find(Ele.Seque == ide); % The two
10    indices in two parametric directions for an element
11    Ele_Knot_U = Ele.KnotsU(idu,:); % The knot
12    span in the first parametric direction for an element
13    Ele_Knot_V = Ele.KnotsV(idv,:); % The knot
```

span in the second parametric direction for an element

```
10 Ele_NoCtPt = Ele.CtrPtsCon(ide,:); % The number
of control points in an element
11 Ele_CoCtPt = CtrPts.Cordis(1:2, Ele_NoCtPt); % The
coordinates of the control points in an element
12 Ke = zeros(2*Nen, 2*Nen);
13 dKe = cell(Ele.GauPtsNum, 1);
14 for i = 1:Ele.GauPtsNum
15     GptOrder = GauPts.Seque(ide, i);
16     dR_dPara = [dRu(GptOrder,:); dRv(GptOrder,:)];
17     dPhy_dPara = dR_dPara*Ele_CoCtPt';
18     J1 = dPhy_dPara;
19     dR_dPhy = inv(J1)*dR_dPara;
20     Be(1,1:Nen) = dR_dPhy(1,:); Be(2,Nen+1:2*Nen) =
dR_dPhy(2,:);
21     Be(3,1:Nen) = dR_dPhy(2,:); Be(3,Nen+1:2*Nen) =
dR_dPhy(1,:);
22     dPara_dPare(1,1) = (Ele_Knot_U(2)-Ele_Knot_U(1))/2; % the
mapping from the parametric space to the parent space
23     dPara_dPare(2,2) = (Ele_Knot_V(2)-Ele_Knot_V(1))/2;
24     J2 = dPara_dPare; J = J1*J2; % the
mapping from the physical space to the parent space;
25     weight = GauPts.Weigh(i)*det(J); % Weight
factor at this point
26     Ke = Ke + (Emin+X.GauPts(GptOrder,:).^penal*(1-
Emin))*weight*(Be'*DH*Be);
27     dKe{i} = (penal*X.GauPts(GptOrder,:).^(penal-1)*(1-
Emin))*weight*(Be'*DH*Be);
28     dv_dg(GptOrder) = weight;
29 end
30 KE{ide} = Ke;
31 dKE{ide} = dKe;
32 end
33 end
```

3.6.2 Stiff_Ass2D

All IGA element stiffness matrices are assembled into a global stiffness matrix K by calling a subfunction `stiff_Ass2D` in line 29 of the main function with five input parameters, namely 1) the cell matrix `KE` contains all IGA element stiffness matrices; 2-3) two structural arrays including the information of control points and all IGA elements, namely `CtrPts` and `Ele`. 4) `Dim` denotes the structural dimension; and 5) `Dofs.Num` is the total number of Degree of Freedoms (DOFs). The output parameter `K` denotes the global stiffness matrix. The corresponding Matlab command can read as:

```
[K] = Stiff_Ass2D(KE, CtrPts, Ele, Dim, Dofs.Num);
```

Stiff_Ass2D: the indices II and JJ for two directions of the matrix KX which is another form of global stiffness matrix and a count ntriplets for the loop program are defined from line 2 to 4. The Matlab code of the loop to realize the assembly of all IGA element stiffness matrices into KX is implemented in lines 5-16. A final assembly to obtain the global stiffness matrix K with a sparse form is performed in line 17 of the subfunction. The Matlab code of Stiff_Ass2D is listed as:

```
1  function [K] = Stiff_Ass2D(KE, CtrPts, Ele, Dim, Dofs_Num)
2  II = zeros(Ele.Num*Dim*Ele.CtrPtsNum*Dim*Ele.CtrPtsNum,1);
3  JJ = II; KX = II;
4  ntriplets = 0;
5  for ide = 1:Ele.Num
6      Ele_NoCtPt = Ele.CtrPtsCon(ide,:);
7      edof = [Ele_NoCtPt, Ele_NoCtPt+CtrPts.Num];
8      for krow = 1:numel(edof)
9          for kcol = 1:numel(edof)
10             ntriplets = ntriplets+1;
11             II(ntriplets) = edof(krow);
12             JJ(ntriplets) = edof(kcol);
13             KX(ntriplets) = KE{ide}(krow,kcol);
14         End
15     end
16 end
17 K = sparse(II,JJ,KX,Dofs_Num,Dofs_Num); K = (K+K')/2;
18 end
```

3.6.3 Solving

The solving of global displacement field is realized in a subfunction Solving with six inputs implemented in line 30 of the main program. In six input parameters, the detailed information of control points CtrPts, boundary conditions DBoundary and BoundCon, DOFs Dofs, the global stiffness matrix K and the load force F are included. The output corresponds to the global displacement field U in the design domain. The implementation of the subfunction Solving is called by a Matlab line:

```
U = Solving(CtrPts, DBoundary, Dofs, K, F, BoundCon);
```

Solving: In lines 2 to 9, five different numerical cases are defined with the corresponding displacements equal to 0 at Dirichlet boundary conditions, namely U_fixeddofs and V_fixeddofs. Then, the matrix U is solved in lines 10 to 13. The Matlab code of the subfunction Solving is given as:

```
1  function U = Solving(CtrPts, DBoundary, Dofs, K, F, BoundCon)
2  switch BoundCon
3      case {1, 4, 5}
```

```

4         U_fixeddofs = DBoundary.CtrPtsOrd;
5         V_fixeddofs = DBoundary.CtrPtsOrd + CtrPts.Num;
6     case {2,3}
1      7         U_fixeddofs = DBoundary.CtrPtsOrd1;
2      8         V_fixeddofs = [DBoundary.CtrPtsOrd1; DBoundary.CtrPtsOrd2] +
3      9         CtrPts.Num;
4      10 end
5      11 Dofs.Ufixed = U_fixeddofs; Dofs.Vfixed = V_fixeddofs;
6      12 Dofs.Free = setdiff(1:Dofs.Num, [Dofs.Ufixed; Dofs.Vfixed]);
7      13 U = zeros(Dofs.Num,1);
8      14 U(Dofs.Free) = K(Dofs.Free,Dofs.Free)\F(Dofs.Free);
9      15 end

```

3.7 Objective function and sensitivity analysis

As far as the derivations of sensitivity analysis of the objective and constraint functions, the reads can refer to (Gao et al. 2019a). firstly, we compute the first-order derivatives of the objective and constraint functions with respect to the DDF, given as:

$$\begin{cases} \frac{\partial J}{\partial \mathcal{X}} = -\frac{1}{2} \int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{u})^T (\gamma \mathcal{X}^{\gamma-1} \mathbf{D}_0) \boldsymbol{\varepsilon}(\mathbf{u}) d\Omega \\ \frac{\partial G}{\partial \mathcal{X}} = \frac{1}{|\Omega|} \int_{\Omega} v_0 d\Omega \end{cases} \quad (14)$$

In the numerical implementations, the DDF at Gauss quadrature points are considered in Eq.(14), and the Gauss quadrature method is still adopted here to compute the above equations. During the optimization, the initial control densities act as design variables, we should drive the first-order derivatives of the objective and constraint functions with respect to the initial control densities. A smoothing mechanism using Shepard function and a linear combination of the NURBS basis functions with the smoothed control densities are involved in the DDF. Using the chain rule, we derive the derivatives of the DDF with respect to the initial control densities, given as:

$$\frac{\partial \mathcal{X}}{\partial \rho_{i,j}} = \frac{\partial \mathcal{X}}{\partial \tilde{\rho}_{i,j}} \frac{\partial \tilde{\rho}_{i,j}}{\partial \rho_{i,j}} = R_{i,j}^{p,q}(\xi, \eta) \psi(\rho_{i,j}) \quad (15)$$

The final detailed form of sensitivity analysis of the objective and constraint functions with respect to design variables can be explicitly described by:

$$\begin{cases} \frac{\partial J}{\partial \rho_{i,j}} = -\frac{1}{2} \int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{u})^T \gamma (\mathcal{X}(\xi, \eta))^{\gamma-1} R_{i,j}^{p,q}(\xi, \eta) \psi(\rho_{i,j}) \mathbf{D}_0 \boldsymbol{\varepsilon}(\mathbf{u}) d\Omega \\ \frac{\partial G}{\partial \rho_{i,j}} = \frac{1}{|\Omega|} \int_{\Omega} R_{i,j}^{p,q}(\xi, \eta) \psi(\rho_{i,j}) v_0 d\Omega \end{cases} \quad (16)$$

The Matlab computation of sensitivity analysis is implemented in lines 32-46 of the main function, mainly containing two steps based on the derivations of sensitivity analysis. Firstly, the first-order derivatives of the objective and constraint functions with respect to the DDF at the Gauss quadrature points are computed, and the corresponding Matlab implementation is called by the following command:

```
1 J = 0;
2 dJ_dg = zeros(GauPts.Num,1);
3
4 for ide = 1:Ele.Num
5     Ele_NoCtPt = Ele.CtrPtsCon(ide,:);
6     edof = [Ele_NoCtPt, Ele_NoCtPt+CtrPts.Num];
7     Ue = U(edof,1);
8     J = J + Ue'*KE{ide}*Ue;
9     for i = 1:Ele.GauPtsNum
10         GptOrder = GauPts.Seque(ide, i);
11         dJ_dg(GptOrder) = -Ue'*dKE{ide}{i}*Ue;
12     end
13 end
14 Data(loop,1) = J; Data(loop,2) = mean(X.GauPts(:));
```

Secondly, the chain derivatives of the DDF with respect to the initial control densities are calculated, and the Matlab implementation for the derivatives of the objective and constraint functions with respect to the initial control densities is realized by calling the corresponding command:

```
33 dJ_dp = R'*dJ_dg; dJ_dp = Sh*(dJ_dp./Hs);
34 dv_dp = R'*dv_dg; dv_dp = Sh*(dv_dp./Hs);
```

3.8 OC: Update design variables and DDF

After computing the sensitivity analysis of the objective and constraint functions with respect to the initial control densities, the Optimality Criteria (OC) method is employed here to solve the numerical optimization problems. The Matlab calling of the OC method is implemented in line 52 of the main function, and a sub function OC is developed with seven input parameters, mainly including the DDF at control densities and Gauss quadrature points presented in a structural array X, the Shepard function and NURBS basis functions developed in the matrices Sh, Hs and R, respectively, the sensitivity analysis of the objective and constraint functions with respect to design variables given in the matrices dJ_dp and dv_dp, and Vmax denotes the maximal material consumption in the optimization. The output parameter only contains the updated DDF at control densities and Gauss quadrature points. The Matlab calling of the OC method is performed in the corresponding command, given as:

```
61 X = OC(X, R, Vmax, Sh, Hs, dJ_dp, dv_dp)
```

OC: In line 2, the updated parameters $l1$, $l2$ and $move$ are defined. The evolving of the design variables is implemented in a while loop from line 3 to 13, until the constraint for the maximal material consumption is satisfied. It should be noted the smoothing mechanism for control densities works in each iteration, and the densities at the Gauss quadrature points are applied to approximately calculate material volume fraction.

The Matlab code of the subfunction OC are given as:

```
1 function X = OC(X, R, Vmax, Sh, Hs, dJ_dp, dv_dp)
2 l1 = 0; l2 = 1e9; move = 0.2;
3 while (l2-l1)/(l1+l2) > 1e-3
4     lmid = 0.5*(l2+l1);
5     X.CtrPts_new = max(0,max(X.CtrPts-move, min(1,
6     min(X.CtrPts+move,X.CtrPts.*sqrt(-dJ_dp./dv_dp/lmid)))));
7     X.CtrPts_new = (Sh*X.CtrPts_new)./Hs;
8     X.GauPts = R*X.CtrPts_new;
9     if mean(X.GauPts(:)) > Vmax
10         l1 = lmid;
11     else
12         l2 = lmid;
13 end
14 end
```

3.9 Plot_Data and Plot_Topy: Representation of numerical results

In the final representation of numerical results, five numerical designs in the optimization will be presented, namely 1) the representation of design variables which corresponds to the densities at control points, namely control densities, in the design domain, presented in **Fig.10 (a)**. It is noted that the vertical direction denotes the values of control densities. 2) The representation of the densities at Gauss quadrature points, shown in **Fig.10 (b)**; in the optimization, Gauss quadrature method is employed to calculate all IGA element stiffness matrices, in each IGA element, nine Gauss quadrature points are chosen. In the final representation, there is no need to map control densities into densities at the center of all IGA elements working as IGA element densities. The mapping will also introduce a large number of intermediate densities. The direct display of the densities at Gauss quadrature points is more reasonable to present material distribution. 3) The DDF in the design domain to represent the overall material distribution is also shown in **Fig.10 (c)**, a function `surf` in Matlab is adopted here to approximately present the DDF using a family of samples. 4) The 2D view of the densities at Gauss quadrature points with the values larger than 0.5 is given to approximately describe the topology, shown in **Fig.10 (d)**. We can easily obtain that it can be viewed as a discrete distribution of a series of point-densities in design domain, similar to the layout of element-densities. 5) It is assumed that

the iso-contour (the value equal to 0.5) of the DDF represents the structural boundaries, and the discussions about the rationality of the value equal to 0.5 can refer to (Gao et al. 2019a, 2020a). A function `contourf` in Matlab is used here to approximately plot the structural topology, and the iso-contour of the DDF, namely the topology, is shown in **Fig.10 (e)**.

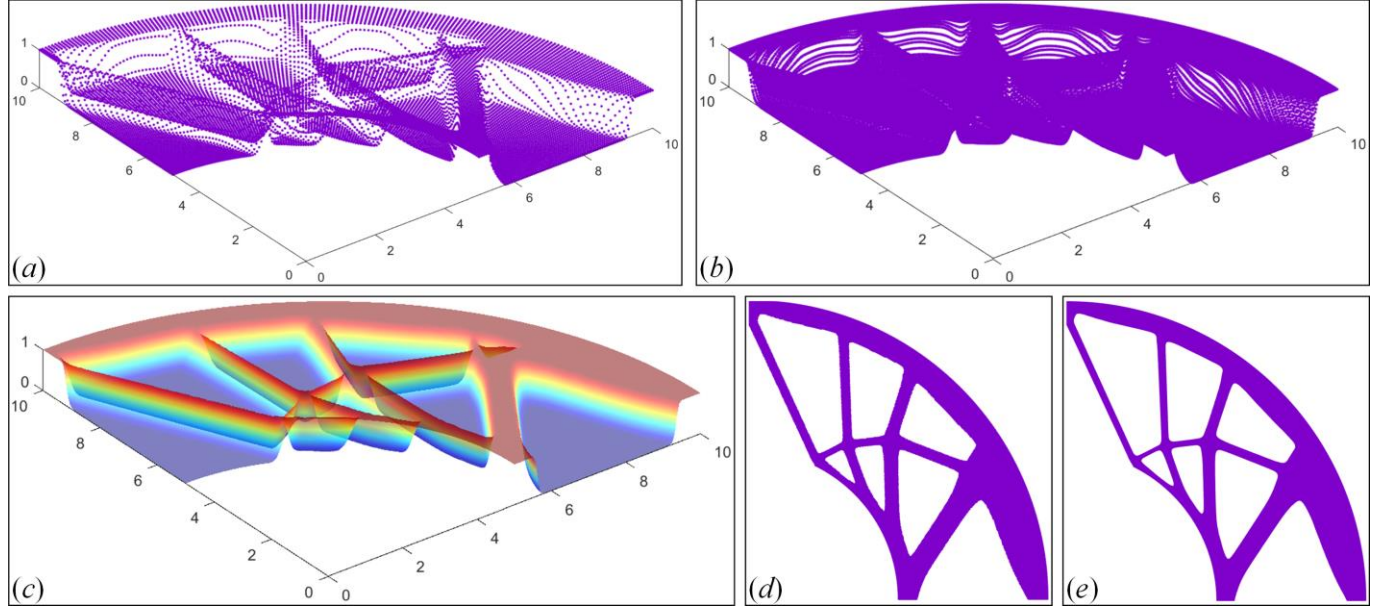


Fig.10 The representation of the optimized designs: a) the densities at control points, namely control densities; b) the DDF at Gauss quadrature points; c) the DDF; d) the 2D view for the densities at Gauss quadrature points with the values higher than 0.5; e) the iso-contour of the DDF, namely the topology

The representation of numerical results involves two subfunctions `Plot_Data` and `Plot_Topy`. Firstly, the Matlab implementation of the subfunction `Plot_Data` with two input parameters (`Num` and `NURBS`) is called in line 25 of the main function to construct some compulsory data for the latter representation. The corresponding Matlab command is given as:

```
[DenFied, Pos] = Plot_Data(Num, NURBS);
```

Two output parameters are contained, namely `DenFied` and `Pos`. The first output parameter denotes the detailed information for the latter samples to plot the DDF using the Matlab function `surf`, including the knot vectors, and the corresponding physical coordinates of all samples. The Matlab implementation of the `DenFied` is performed in lines 7-15. The second out parameter indicates the figure positions of numerical designs, implemented in lines 2-6 of the subfunction `Plot_Data` and its Matlab code is given as:

```
1 function [DenFied, Pos] = Plot_Data(Num, NURBS)
2 bdwidth = 5; topbdwidth = 30; scnsz = get(0, 'ScreenSize');
3 Pos.p1 = [bdwidth, 3/5*scnsz(4)+bdwidth-50, scnsz(3)/2-
4           2*bdwidth, 2*scnsz(4)/5-(topbdwidth+bdwidth)];
5 Pos.p2 = [Pos.p1(1)+scnsz(3)/2, Pos.p1(2), Pos.p1(3), Pos.p1(4)];
6 Pos.p3 = [bdwidth, 1/6*scnsz(4)+bdwidth-100, scnsz(3)/2-
```

```
2*bdwidth, 2*scnsize(4)/5-(topbdwidth + bdwidth)];
6 Pos.p4 = [Pos.p1(1)+scnsize(3)/2, Pos.p3(2), Pos.p3(3), Pos.p3(4)];
7 Uknots = linspace(0,1,10*Num(1)); Vknots = linspace(0,1,10*Num(2));
1 [N_f, id_f] = nrbbasisfun({Uknots, Vknots}, NURBS);
2
3 [PCor_U, PCor_W] = nrbeval(NURBS, {Uknots, Vknots});
4
5 PCor_U = PCor_U./PCor_W;
6
7 PCor_Ux = reshape(PCor_U(1,:), numel(Uknots), numel(Vknots))';
8
9 PCor_Uy = reshape(PCor_U(2,:), numel(Uknots), numel(Vknots))';
10
11 DenFied.N = N_f; DenFied.id = id_f;
12
13 DenFied.U = Uknots; DenFied.V = Vknots;
14
15 DenFied.Ux = PCor_Ux; DenFied.Uy = PCor_Uy;
16 end
```

The Matlab command for the subfunction `Plot_Topy` with seven input parameters, mainly including the DDF structural array `X`, control points and Gauss quadrature points (`GauPts` and `CtrPts`), the detailed information for the samples of the DDF (`DenFied`), the structural sizes `L` and `W`, and the figure position `Pos`, is given as:

```
[X] = Plot_Topy(X, GauPts, CtrPts, DenFied, L, W, Pos);
```

The Matlab code for the representation of control densities is implemented in lines 2-4 of the subfunction, and the representation of the densities at Gauss quadrature points is realized from line 5 to 7. The Matlab implementation of the representation of the DDF is called in lines 8-12. Lines 13-16 plot the representation of 2D-view of the densities at Gauss quadrature points with the values larger than 0.5. In lines 17-19, the representation of the structural topology is implemented.

```
1 function [X] = Plot_Topy(X, GauPts, CtrPts, DenFied, L, W, Pos)
2 h1 = figure(1); clf; set(h1, 'Position', Pos.p1, 'color', [1 1 1]);
3 plot3(CtrPts.Cordis(1,:), CtrPts.Cordis(2,:), X.CtrPts, '.', 'color', [0
4 .5 0 0.8]);
5 axis equal; axis([0 L 0 W 0 1]);
6 h2 = figure(2); clf; set(h2, 'Position', Pos.p1, 'color', [1 1 1]);
7 plot3(GauPts.PCor(1,:), GauPts.PCor(2,:), X.GauPts, '.', 'color', [0.5 0
8 0.8]);
9 axis equal; axis([0 L 0 W 0 1]);
10 h3 = figure(3); clf; set(h3, 'Position', Pos.p2, 'color', [1 1 1]);
11 X.DDF = sum(DenFied.N.*X.CtrPts(DenFied.id), 2);
12 X.DDF = reshape(X.DDF, numel(DenFied.U), numel(DenFied.V))';
13 surf(DenFied.Ux, DenFied.Uy, X.DDF); shading interp;
14 colormap(jet(256)); alpha(0.5);
15 axis equal; grid off; axis([0 L 0 W 0 1]);
16 h4 = figure(4); clf; set(h4, 'Position', Pos.p3, 'color', [1 1 1]);
17 GauPts_PCor = GauPts.PCor(1:2, X.GauPts>=0.5);
18 plot(GauPts_PCor(1,:), GauPts_PCor(2,:), '.', 'color', [0.5 0 0.8]);
```

```
16 axis equal; axis off;
17 h5 = figure(5); clf; set(h5, 'Position', Pos.p4, 'color', [1 1 1]);
18 contourf(DenFied.Ux, DenFied.Uy, X.DDF, [0.5 0.5], 'facecolor',
1 [0.5 0 0.8], 'edgecolor', [1 1 1]);
2
3 19 axis equal; axis off;
4
5 20 end
6
7
```

4 Numerical results

In this section, several numerical examples are tested to demonstrate the effectiveness and efficiency of the Matlab code IgaTop2D for the ITO method. In all numerical examples, the Poisson's ratio is defined as 0.3, and Young's moduli for solids and voids are equal to 1 and 1e-9, respectively. A personal laptop with the Matlab R2018b (9.5.0.944444) is used in the current work. In next examples, the quarter annulus with boundary and load conditions, shown in **Fig.3 (a)** is first considered here to present the basic and positive features of the current ITO method. Then, a L beam with boundary and load conditions is optimized by the Matlab code. Finally, several benchmarks, including the cantilever beam, Michell-type structure and MBB beam, are all tested using the current Matlab implementation framework. In the main function IgaTop2D, the terminal criterion is that the L^∞ norm of the difference of control densities between two consecutive iterations is less than 1% or the maximum 150 iteration steps are reached.

4.1 Quarter annulus

In this example, the quarter annulus with the boundary and load conditions shown in **Fig.3 (a)** is considered. The structural sizes L and R are defined as 10 and 5, respectively. The allowable material consumption in the optimization is equal to 40%. In the construction of the geometrical model for the quarter annulus, the orders of NURBS basis functions in two parametric directions are both set as 3. Hence, the input parameter `Order` should be equal to [0 1]. It is assumed that 101×51 knots with the unique values are used, and the input parameter `Num` is [101 51]. The corresponding total number of all IGA elements in the physical space is equal to 100×50 . The total number of control points to define the NURBS surface for the quarter annulus is equal to 102×52 . The optimization of the quarter annulus using the current ITO method is performed by calling the Matlab command:

```
IgaTop2D(10, 10, [0 1], [101 51], 5, 0.4, 3, 2)
```

The initial designs of the quarter annulus are presented in **Fig.11**, where the distribution of control densities is displayed in **Fig.11 (a)** with all values equal to 1 in the design domain, and the distribution of the densities at Gauss quadrature points is presented in **Fig.11 (b)** also with all values equal to one, and **Fig.11 (c)** shows

the layout of the DDF in the whole design domain. The initialization of the DDF keeps consistent with the 88-line SIMP code with equal values of all element in the design domain. As also already discussed in (Gao et al. 2019a), the equal values in all control densities can offer more benefits for the latter optimization of structures, namely avoiding a local optimum occurred in the design.

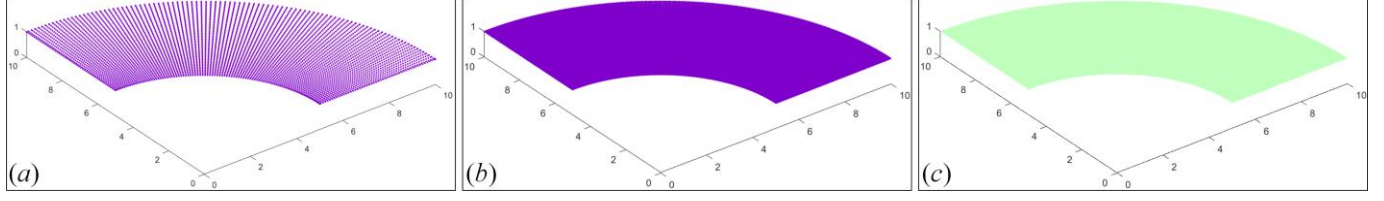


Fig.11 The initial designs of the quarter annulus: a) control densities; b) the densities at Gauss quadrature points; and c) the DDF in the design domain

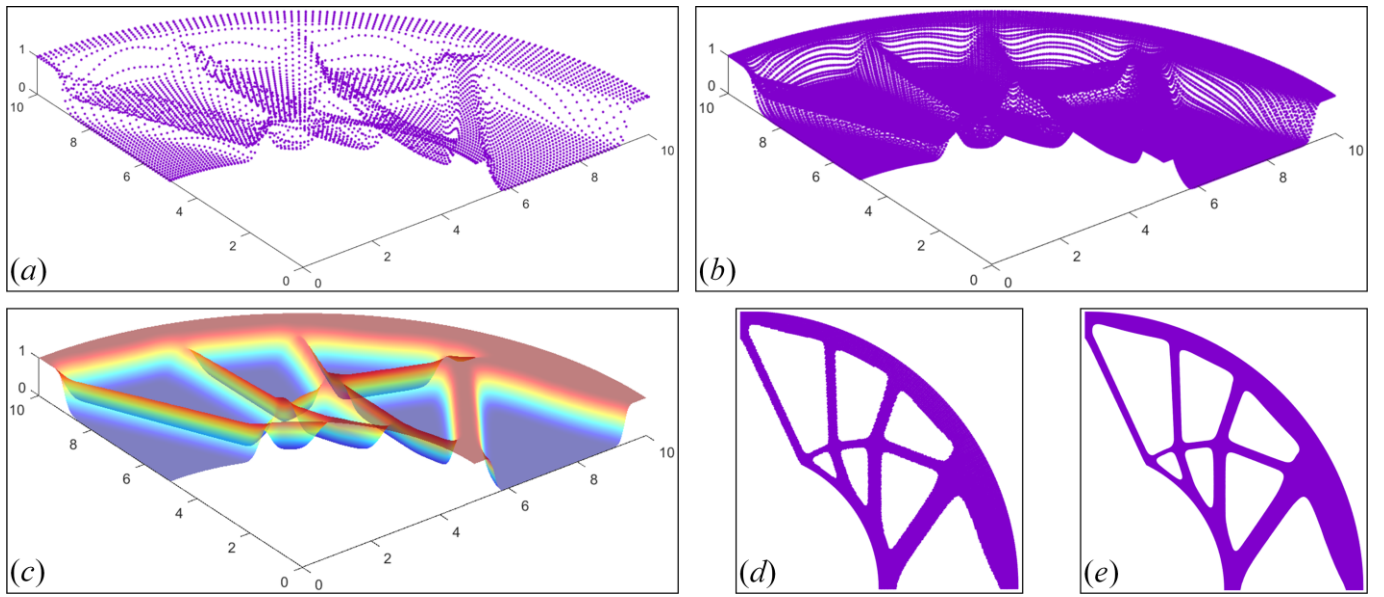


Fig.12 The optimized designs of the quarter annulus: a) control densities; b) the densities at Gauss quadrature points; c) the DDF in the design domain; d) the 2D view of the densities with values higher than 0.5 at Gauss quadrature points; e) the structural topology

In **Fig.12**, the optimized designs of the quarter annulus using the current Matlab code are provided, namely 1) the final distribution of control densities is presented in **Fig.12 (a)** with a series of discrete densities; 2) the optimized layout of the densities at Gauss quadrature points is shown in **Fig.12 (b)** also with a family of discrete densities in the design domain; 3) the optimized DDF is displayed in **Fig.12 (c)**; 4) The 2D view of the densities with values higher than 0.5 at Gauss quadrature points is presented in **Fig.12 (d)**; and 5) the final structural topology defined by the iso-contour (values equal to 0.5) of the DDF in **Fig.12 (c)** is shown in **Fig.12 (e)**. As we can easily see, the final optimized DDF is characterized with the sufficient smoothness and continuity, and the optimized structural topology is also featured with the smooth structural boundaries and distinct material interfaces between solids and voids. As already discussed in (Gao et al. 2019a), the definition of the structural topology using the optimized DDF in an implicit manner is simple but efficient.

As shown in **Fig.13**, the convergent iterations of the structural compliance and volume fraction during the optimization are provided, where the black curve represents the evolving of the objective function and the advancement of volume fraction is displayed by the red curve. The final optimized value of the structural compliance is equal to 106.75, and the material volume fraction can arrive at the prescribed maximal value of the consumption, namely 0.4. As we can easily see, the iterative curves of the objective and constraint function are both featured with the a clear, smooth and fast convergence within the maximum iterative step equal to 79, which shows the effectiveness and efficiency of the ITO method and also presents the validity of the current Matlab code `IgaTop2D` for the numerical implementation of the ITO method.

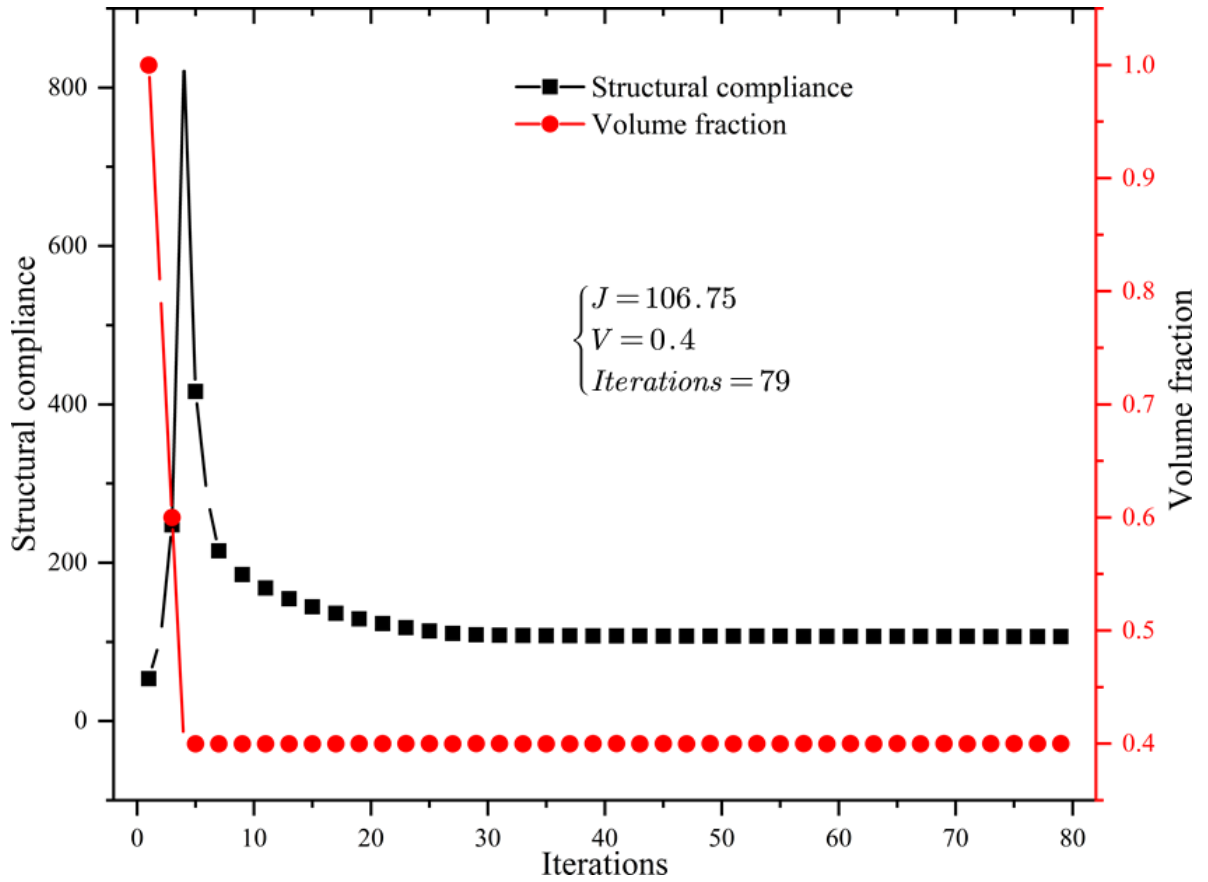


Fig.13 The convergent histories of the optimization for the quarter annulus

Meanwhile, it should be noted the iterative curve of the volume fraction in **Fig.13** shows the variation of the DDF in the optimization, rather than the final topology. The volume fraction of the final topology can be solved by slightly modifying the DDF, namely $\mathcal{X} \rightarrow 1$, if $\mathcal{X} \geq 0.5$ and $\mathcal{X} \rightarrow 0$, if $\mathcal{X} < 0.5$. The value of the volume fraction of the optimized topology is equal to 0.404, and we can obtain that it is nearly equal to the prescribed value of material maximal consumption. Meanwhile, nine intermediate designs of the DDF in the optimization are also presented in **Fig.14**, including iterations 1, 3, 5, 12, 20, 27, 37, 57 and 79. We can easily find that the DDF can keep the sufficient smoothness and continuity in the optimization, without the wavy or zigzag features in the optimization.

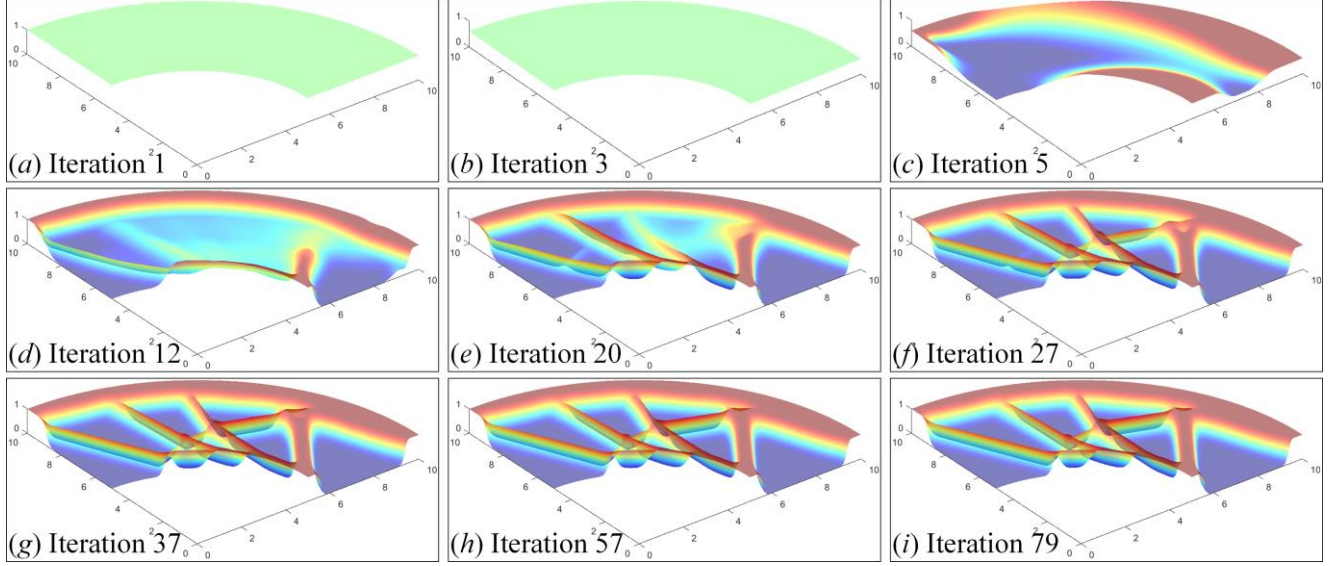


Fig.14 Iterations of the DDF in the optimization

4.2 L beam

In this example, a L beam with the load and boundary conditions defined in **Fig.15** will be optimized using the developed Matlab code of the ITO method. As far as the L beam, the origin of the physical coordinate plotted with the blue color is different from the origin of the parametric coordinate with the red color, shown in **Fig.15**. In the NURBS parametrization of the L beam, the first parametric direction is the L-shape in the beam, the second parametric direction is along the horizontal ordinate of the L beam, shown in **Fig.15**. The structural sizes (L and W) of the L beam are set a 10 and 5, respectively. In the optimization, the maximal value of material volume fraction is defined as 0.3. In the NURBS parametrization of the L beam, we still choose the NURBS basis functions with the third order in two parametric directions, and the corresponding input parameter `Order` should be equal to [1 1]. A same number of knots 101×51 with no repetitive values is still considered in the current example, and the input parameter `Num` should be [101 51]. The IGA mesh contain 100×50 IGA elements, and the total number of control points in the definition of NURBS surface is equal to 102×52 .

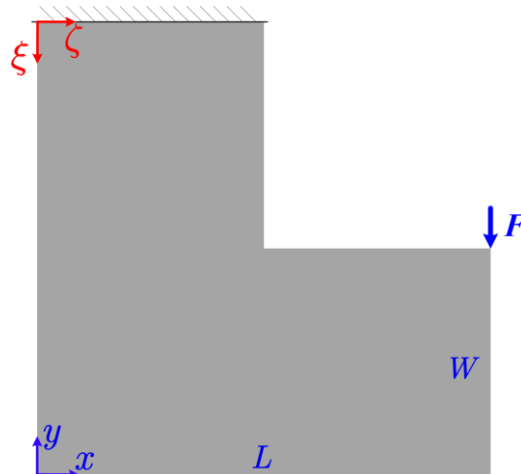


Fig.15 The structural geometry of L beam

As shown in **Fig.16**, the initial three designs containing the distribution of control densities in **Fig.16 (a)** with values equal to one, the layout of densities at Gauss quadrature points in **Fig.16 (b)** with values equal to one and the overall distribution of the DDF in the whole design domain in **Fig.16 (c)** are both provided. The ITO for the current L beam is implemented by calling a line of Matlab code:

```
IgaTop2D(10, 5, [1 1], [101 51], 4, 0.3, 3, 2)
```

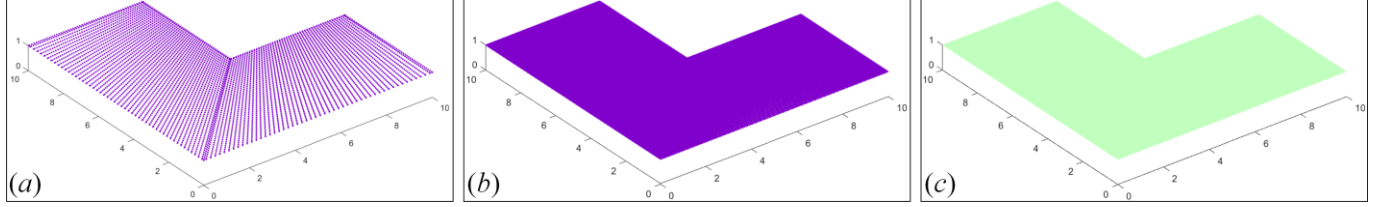


Fig.16 The initial designs: a) control densities; b) the densities at Gauss quadrature points; c) the DDF

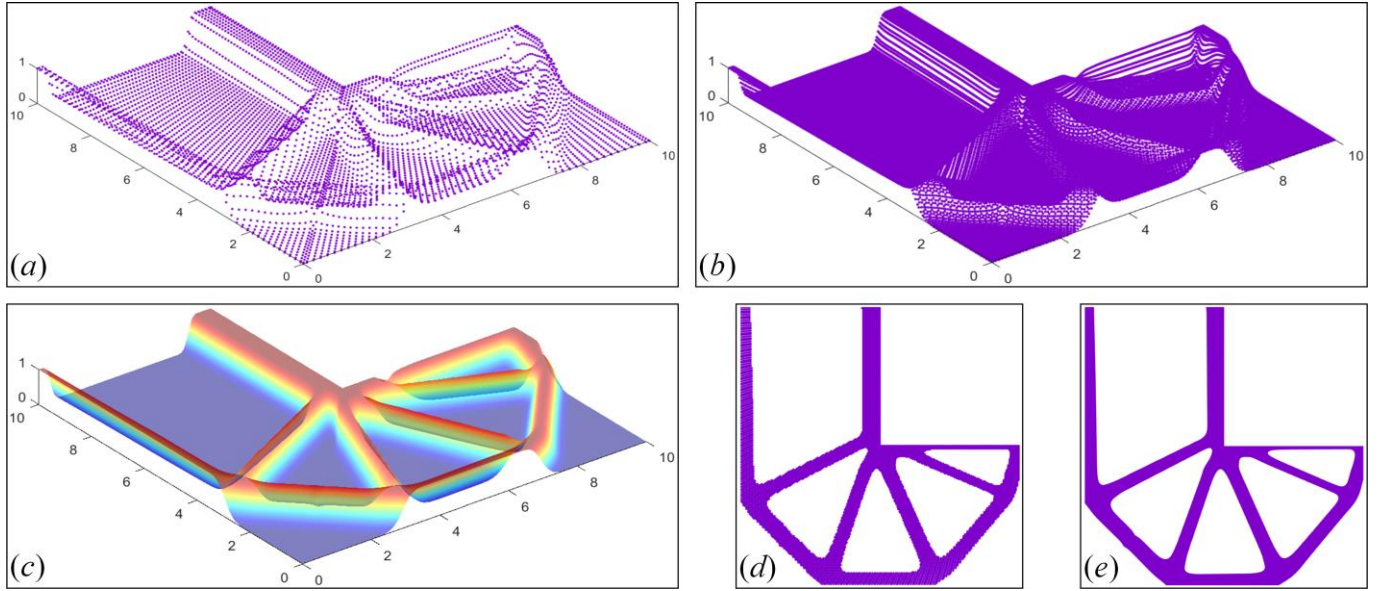


Fig.17 The optimized designs of the quarter annulus: a) control densities; b) the densities at Gauss quadrature points; c) the DDF in the design domain; d) the 2D view of the densities with values higher than 0.5 at Gauss quadrature points; e) the structural topology

The optimized numerical designs are presented in **Fig.17**, where the optimized layout of control densities is shown in **Fig.17 (a)**, the optimized distribution of the densities at Gauss quadrature points is presented in **Fig.17 (b)**, the optimized DDF is distributed in **Fig.17 (c)**, the view in 2D of the densities with the values higher than 0.5 at Gauss quadrature points is presented in **Fig.17 (d)**, and the final topology of the L beam is provided in **Fig.17 (e)**. Firstly, we can easily see that the optimized DDF is featured with the smoothness and continuity, which can offer more benefits for the latter representation of the structural topology. That is, the topology can have the smooth structural boundaries and distinct interfaces using the iso-contour (0.5) of the DDF. The optimization of the L beam is terminated at the 45th iteration step, where the final optimized objective function is equal to 177.01 and the material volume fraction of the DDF is equal to 0.3. the volume

fraction of the final topology shown in **Fig.17** (e) is equal to 0.303, and it is also almost equal to the pre-defined material volume fraction value 0.3, which can show the reasonability of the representation of the structural boundaries using the iso-contour of the DDF in an implicit manner.

4.3 Other numerical examples

In this subsection, three numerical cases will be discussed, including the classic cantilever beam, Michell-type structure and MBB beam. A same feature of these three structures is that a rectangular design domain is considered in the optimization. As far as the rectangular design domain, the physical coordinate system and the parametric coordinate system will be coincided.

In the optimization of the cantilever beam, the structural sizes (L and W) are also set a 10 and 5, respectively. The orders of NURBS basis functions in the parametrization are also equal to 3, and the input parameter `Order` is equal to [1 1]. A family of 161×81 unique knots are used and the corresponding input parameter `Num` should be [161 81]. In the IGA mesh, 160×80 IGA elements are contained to discrete the cantilever beam, and the total number of control points should be equal to 162×82 .

In the optimization of the Michell structure, the corresponding L and W in the structural sizes are defined as 10 and 4, respectively. The orders of NURBS basis functions in the parametrization of the geometry are also equal to 3. The corresponding input parameter `Order` is also equal to [1 1]. Meanwhile, a family of 101×41 unrepetitive knots are considered in the parametric space, and the related input parameter `Num` is [101 41]. In the IGA mesh, 100×40 IGA elements are employed in the discretization of the Michell-type structure, and the total number of control points should be equal to 102×42 .

In the optimization of the MBB beam, the related structure sizes L and W are set as 18 and 3, respectively. In the NURBS parametrization of the MBB beam, the third order NURBS basis functions are used, and the input parameter `Order` is also equal to [1 1]. Similarly, a series of 241×41 knots with unrepetitive values are considered in the parametric space, and the corresponding input parameter `Num` is [241 41]. 240×40 IGA elements are included in the IGA mesh to discretize the MBB beam. The total number of control points should be equal to 242×42 .

The corresponding Matlab implementations for the cantilever beam, MBB beam and Michell-type structure are realized by calling the following Matlab commands:

```
IgaTop2D(10, 5, [1 1], [161 81], 1, 0.2, 3, 2)
IgaTop2D(18, 3, [1 1], [241 41], 2, 0.2, 3, 2)
IgaTop2D(10, 4, [1 1], [101 41], 3, 0.2, 3, 2)
```

The first line realize the optimization of the cantilever beam, and the optimizations of MBB beam and the Michell-type structures are realized in the second and third lines, respectively.

The optimized results for three numerical examples are shown in **Fig.18-20**, where the optimized designs of the cantilever beam are presented in **Fig.18**, **Fig.19** presents the final optimized designs of Michell-type structure, and the optimized designs of MBB beam are shown in **Fig.20**. The optimized layouts of control densities for the cantilever beam, Michell-type structure and MBB beam are presented in **Fig.18 (a)**, **Fig.19 (a)** and **Fig.20 (a)**, respectively. As shown in **Fig.18 (c)**, **Fig.19 (c)** and **Fig.20 (c)**, the final optimized DDFs of the cantilever beam, Michell-type structure and MBB beam are all provided, and a similar feature that the sufficient smoothness and continuity can be seen in the DDFs. Meanwhile, the structural topologies of three structures are all featured with the smooth structural boundaries and distinct interfaces between solids and voids, shown in **Fig.18 (e)**, **Fig.19 (e)** and **Fig.20 (d)**. The optimized numerical results of the cantilever beam, Michell-type structure and MBB beam can obviously demonstrate the effectiveness and efficiency of the Matlab code IgaTop2D for the ITO method.

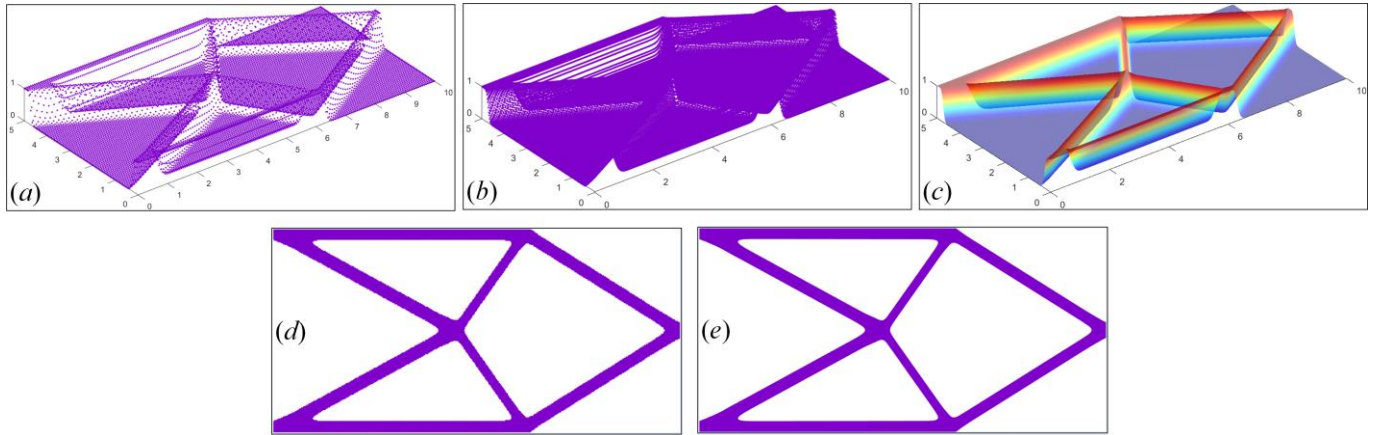


Fig.18 The optimized distributions of cantilever beam: *a)* control densities; *b)* the densities at Gauss quadrature points; *c)* the 2D view of the densities with values higher than 0.5 at Gauss quadrature points; *d)* the DDF in the design domain; *e)* the structural topology

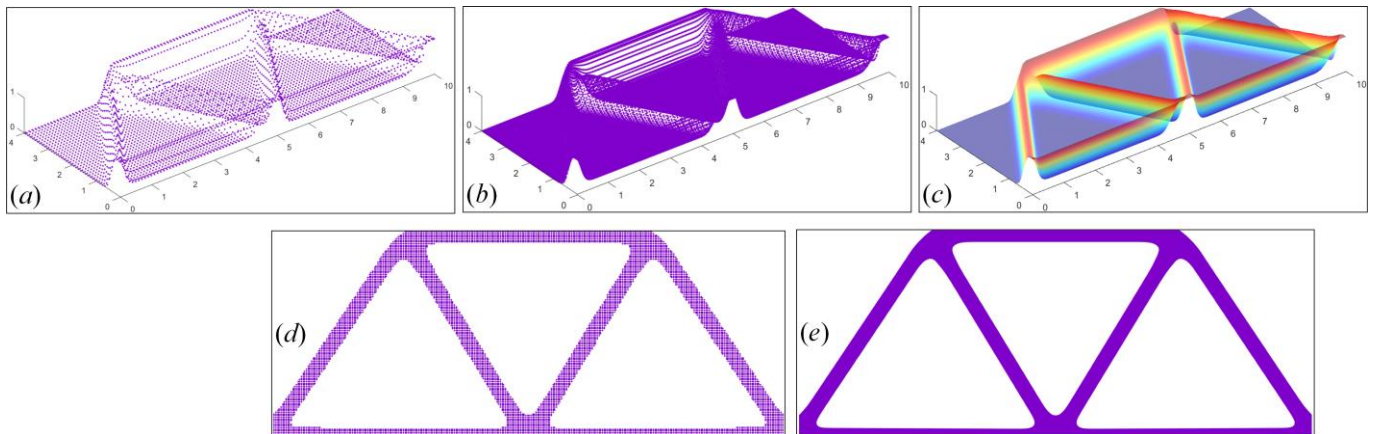


Fig.19 The optimized distributions of Michell-type structure: *a)* control densities; *b)* the densities at Gauss quadrature points; *c)* the 2D view of the densities with values higher than 0.5 at Gauss quadrature points; *d)* the DDF in the design domain; *e)* the structural topology

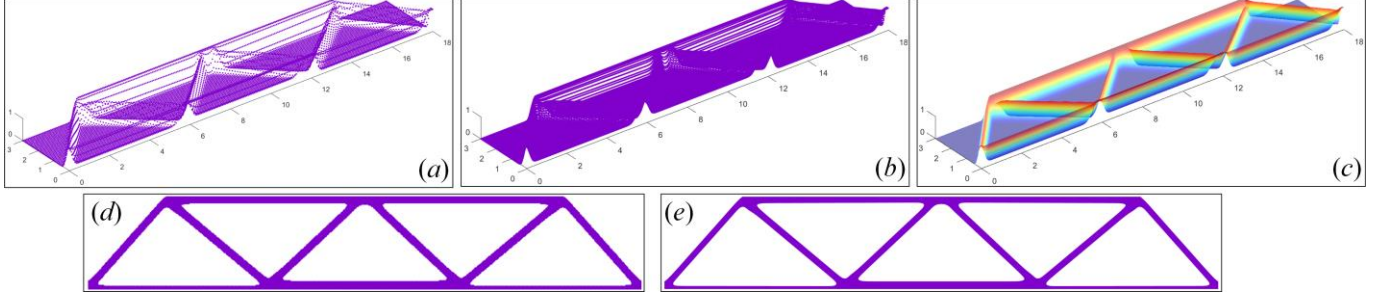


Fig.20 The optimized distributions of MBB beam: a) control densities; b) the densities at Gauss quadrature points; c) the 2D view of the densities with values higher than 0.5 at Gauss quadrature points; d) the DDF in the design domain; e) the structural topology

5 Conclusions

In this paper, a compact and efficient Matlab implementation framework is presented for the ITO, where a main function `IgaTop2D` with eleven subfunctions (`Geom_Mod`, `Pre_IGA`, `Quadrature`, `Shep_Fun`, `Boun_Cond`, `Stiff_Ele2D`, `Stiff_Ass2D`, `Solving`, `OC`, `Plot_Data` and `Plot_Topy`) for the optimization is developed. The Matlab implementation framework mainly involves the construction of the geometrical model using NURBS, the preparation for IGA, the definition of boundary conditions, initialize the DDF at control points and Gauss quadrature points, the definition of smoothing mechanism, the IGA to solve structural responses; calculate the objective function and sensitivity analysis, update design variables and the DDF, and finally present the optimized designs. Finally, five numerical examples are presented to show the effectiveness of the current Matlab implementation framework of the ITO method.

Acknowledgments

The authors wish to thank Dr. Phu Nguyen, a Lecturer from Department of Civil Engineering, Monash University. Dr. Phu Nguyen offers the complete Matlab code of IGA in (Nguyen et al. 2015) for us to extensively understand the concept and numerical implementation of IGA.

This work was partially supported by the Fundamental Research Funds for the Central Universities (5003123021), and the Program for HUST Academic Frontier Youth Team (2017QYTD04).

Declaration of Interest Statement:

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results

The main function of the Matlab code is listed in the Appendix, and all subfunctions with the Matlab code are presented in this paper. The ‘nurbs’ toolbox in Octave and Matlab can be obtained from <https://octave.sourceforge.io/nurbs/>. All the results presented in this paper can be reproduced with the Matlab code.

Appendix: a 56-line MATLAB code for the main function IgaTop2D

```
1 1 function IgaTop2D(L, W, Order, Num, BoundCon, Vmax, penal, rmin)
2 2 %% Material properties
3 3 path = genpath(pwd); addpath(path);
4 4 E0 = 1; Emin = 1e-9; nu = 0.3; DH=E0/(1-nu^2)*[1 nu 0; nu 1 0; 0 0
5 (1-nu)/2];
6 5 NURBS = Geom_Mod(L, W, Order, Num, BoundCon); close all
7 6 %% Preparation for IGA
8 7 [CtrPts, Ele, GauPts] = Pre_IGA(NURBS);
9 8 Dim = numel(NURBS.order); Dofs.Num = Dim*CtrPts.Num;
10 9 [DBoudary, F] = Boun_Cond(CtrPts, BoundCon, NURBS, Dofs.Num);
11 10 %% Initialization of control design variables
12 11 X.CtrPts = ones(CtrPts.Num,1);
13 12 GauPts.Cor = [reshape(GauPts.CorU',1,GauPts.Num);
14 reshape(GauPts.CorV',1,GauPts.Num)];
15 13 [GauPts.PCor,GauPts.Pw] = nrbeval(NURBS, GauPts.Cor);
16 14 GauPts.PCor = GauPts.PCor./GauPts.Pw;
17 15 [N, id] = nrbbasisfun(GauPts.Cor, NURBS);
18 16 R = zeros(GauPts.Num, CtrPts.Num);
19 17 for i = 1:GauPts.Num, R(i,id(i,:)) = N(i,:); end
20 18 R = sparse(R);
21 19 [dRu, dRv] = nrbbasisfunder(GauPts.Cor, NURBS);
22 20 X.GauPts = R*X.CtrPts;
23 21 %% Smoothing mechanism
24 22 [Sh, Hs] = Shep_Fun(CtrPts, rmin);
25 23 %% Start optimization in a loop
26 24 change = 1; nloop = 150; Data = zeros(nloop,2); Iter_Ch =
27 zeros(nloop,1);
28 25 [DenFied, Pos] = Plot_Data(Num, NURBS);
29 26 for loop = 1:nloop
30 27 %% IGA to evaluate the displacement responses
31 28 [KE, dKE, dv_dg] = Stiff_Ele2D(X, penal, Emin, DH, CtrPts, Ele,
32 GauPts, dRu, dRv);
33 29 [K] = Stiff_Ass2D(KE, CtrPts, Ele, Dim, Dofs.Num);
34 30 U = Solving(CtrPts, DBoudary, Dofs, K, F, BoundCon);
35 31 %% Objective function and sensitivity analysis
36 32 J = 0;
37 33 dJ_dg = zeros(GauPts.Num,1);
38 34 for ide = 1:Ele.Num
39 35 Ele_NoCtPt = Ele.CtrPtsCon(ide,:);
40 36 edof = [Ele_NoCtPt, Ele_NoCtPt+CtrPts.Num];
41 37 Ue = U(edof,1);
42 38 J = J + Ue'*KE{ide}*Ue;
43 39 for i = 1:Ele.GauPtsNum
44 40 GptOrder = GauPts.Seque(ide, i);
```

```
41         dJ_dg(GptOrder) = -Ue'*dKE{ide}{i}*Ue;
42     end
43 end
1 44 Data(loop,1) = J; Data(loop,2) = mean(X.GauPts(:));
2 45 dJ_dp = R'*dJ_dg; dJ_dp = Sh*(dJ_dp./Hs);
3 46 dv_dp = R'*dv_dg; dv_dp = Sh*(dv_dp./Hs);
4 47 %% Print and plot results
5 48 fprintf(' It.:%5i Obj.:%11.4f
6 Vol.:%7.3fch.:%7.3f\n',loop,J,mean(X.GauPts(:)),change);
7 49 [X] = Plot_Topy(X, GauPts, CtrPts, DenFied, L, W, Pos);
8 50 if change < 0.01, break; end
9 51 %% Optimality criteria to update design variables
10 52 X = OC(X, R, Vmax, Sh, Hs, dJ_dp, dv_dp);
11 53 change = max(abs(X.CtrPts_new(:)-X.CtrPts(:))); Iter_Ch(loop) =
12 change;
13 54 X.CtrPts = X.CtrPts_new;
14 55 end
15 56 end
```

References

- Agrawal V, Gautam SS (2019) IGA: a simplified introduction and implementation details for finite element users. J Inst Eng Ser C 100:561–585
- Allaire G, Jouve F, Toader AM (2004) Structural optimization using sensitivity analysis and a level-set method. J Comput Phys 194:363–393
- Andreassen E, Clausen A, Schevenels M, et al (2011) Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidiscip Optim 43:1–16
- Bendsøe M, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224
- Bendsøe MP, Sigmund O (1999) Material interpolation schemes in topology optimization. Arch Appl Mech 69:635–654
- Challis VJ (2010) A discrete level-set topology optimization code written in Matlab. Struct Multidiscip Optim 41:453–464
- Chen Q, Zhang X, Zhu B (2019) A 213-line topology optimization code for geometrically nonlinear structures. Struct Multidiscip Optim 59:1863–1879
- Cottrell JA, Hughes TJR, Bazilevs Y (2009) Isogeometric Analysis: Toward Integration of CAD and FEA
- Da D, Xia L, Li G, Huang X (2018) Evolutionary topology optimization of continuum structures with smooth boundary representation. Struct Multidiscip Optim 57:2143–2159
- De Boor C (1978) A practical guide to splines. Springer-Verlag New York
- de Falco C, Reali A, Vázquez R (2011) GeoPDEs: A research tool for Isogeometric Analysis of PDEs. Adv Eng Softw 42:1020–1034
- Dedè L, Borden MJ, Hughes TJR (2012) Isogeometric analysis for topology optimization with a phase field model. Arch Comput Methods Eng 19:427–465
- Du B, Zhao Y, Yao W, et al (2020) Multiresolution isogeometric topology optimisation using moving morphable voids. Comput Model Eng Sci 122:1119–1140
- Ferrari F, Sigmund O (2020) A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D. Struct Multidiscip Optim. doi: 10.1007/s00158-020-02629-w
- Gai Y, Zhu X, Zhang YJ, et al (2020) Explicit isogeometric topology optimization based on moving morphable voids with closed B-spline boundary curves. Struct Multidiscip Optim 61:963–982

- Gao J, Gao L, Luo Z, Li P (2019a) Isogeometric topology optimization for continuum structures using density distribution function. *Int J Numer Methods Eng* 119:991–1017
- Gao J, Luo Z, Li H, Gao L (2019b) Topology optimization for multiscale design of porous composites with multi-domain microstructures. *Comput Methods Appl Mech Eng* 344:451–476
- Gao J, Luo Z, Xia L, Gao L (2019c) Concurrent topology optimization of multiscale composite structures in Matlab. *Struct Multidiscip Optim* 60:2621–2651
- Gao J, Luo Z, Xiao M, et al (2020a) A NURBS-based Multi-Material Interpolation (N-MMI) for isogeometric topology optimization of structures. *Appl Math Model* 81:818–843
- Gao J, Xiao M, Gao L, et al (2020b) Isogeometric topology optimization for computational design of re-entrant and chiral auxetic composites. *Comput Methods Appl Mech Eng* 362:112876
- Gao J, Xue H, Gao L, Luo Z (2019d) Topology optimization for auxetic metamaterials based on isogeometric analysis. *Comput Methods Appl Mech Eng* 352:211–236
- Ghasemi H, Park HS, Rabczuk T (2017) A level-set based IGA formulation for topology optimization of flexoelectric materials. *Comput Methods Appl Mech Eng* 313:239–258
- Guo X, Zhang W, Zhong W (2014) Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *J Appl Mech* 81:081009
- Hassani B, Khanzadi M, Tavakkoli SM (2012) An isogeometrical approach to structural topology optimization by optimality criteria. *Struct Multidiscip Optim* 45:223–233
- Hou W, Gai Y, Zhu X, et al (2017) Explicit isogeometric topology optimization using moving morphable components. *Comput Methods Appl Mech Eng* 326:694–712
- Huang X, Xie Y-MM (2010) A further review of ESO type methods for topology optimization. *Struct Multidiscip Optim* 41:671–683
- Hughes TJR (2012) *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation
- Hughes TJR, Cottrell JAA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput Methods Appl Mech Eng* 194:4135–4195
- Jahangiry HA, Tavakkoli SM (2017) An isogeometrical approach to structural level set topology optimization. *Comput Methods Appl Mech Eng* 319:240–257
- Kang Z, Wang Y (2011) Structural topology optimization based on non-local Shepard interpolation of density field. *Comput Methods Appl Mech Eng* 200:3515–3525
- Kang Z, Wang Y (2012) A nodal variable method of structural topology optimization based on Shepard interpolant. *Int J Numer Methods Eng* 90:329–342
- Kato J, Ogawa S, Ichibangase T, Takaki T (2018) Multi-phase field topology optimization of polycrystalline microstructure for maximizing heat conductivity. *Struct Multidiscip Optim* 57:1937–1954
- Li H, Luo Z, Zhang N, et al (2016) Integrated design of cellular composites using a level-set topology optimization method. *Comput Methods Appl Mech Eng* 309:453–475
- Liang Y, Cheng G (2020) Further elaborations on topology optimization via sequential integer programming and Canonical relaxation algorithm and 128-line MATLAB code. *Struct Multidiscip Optim* 61:411–431
- Lieu QX, Lee J (2017a) Multiresolution topology optimization using isogeometric analysis. *Int J Numer Methods Eng* 112:2025–2047
- Lieu QX, Lee J (2017b) A multi-resolution approach for multi-material topology optimization based on isogeometric analysis. *Comput Methods Appl Mech Eng* 323:272–302
- Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. *Struct Multidiscip Optim* 50:1175–1196
- Luo Z, Wang MY, Tong L, Wang S (2007) Shape and topology optimization of compliant mechanisms using a parameterization level set method. *J Comput Phys* 227:680–705
- Nguyen C, Zhuang X, Chamoin L, et al (2020) Three-dimensional topology optimization of auxetic metamaterial using isogeometric analysis and model order reduction. *Comput Methods Appl Mech Eng* 371:113306

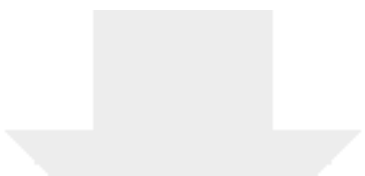
- Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T (2015) Isogeometric analysis: An overview and computer implementation aspects. *Math Comput Simul* 117:89–116
- Nishi S, Yamada T, Izui K, et al (2020) Isogeometric topology optimization of anisotropic metamaterials for controlling high-frequency electromagnetic wave. *Int J Numer Methods Eng* 121:1218–1247
- Otomori M, Yamada T, Izui K, Nishiwaki S (2015) Matlab code for a level set-based topology optimization method using a reaction diffusion equation. *Struct Multidiscip Optim* 51:1159–1172
- Piegl L, Les T, Tiller W (2012) *The NURBS Book*. Springer Science & Business Media
- Qian X (2013) Topology optimization in B-spline space. *Comput Methods Appl Mech Eng* 265:15–35
- Sanders ED, Pereira A, Aguiló MA, Paulino GH (2018) PolyMat: an efficient Matlab code for multi-material topology optimization. *Struct Multidiscip Optim* 58:2727–2759
- Seo Y-D, Kim H-J, Youn S-K (2010a) Shape optimization and its extension to topological design based on isogeometric analysis. *Int J Solids Struct* 47:1618–1640
- Seo Y-D, Kim H-J, Youn S-K (2010b) Isogeometric topology optimization using trimmed spline surfaces. *Comput Methods Appl Mech Eng* 199:3270–3296
- Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM national conference*. ACM, pp 517–524
- Shojaee S, Mohamadianb M, Valizadeh N (2012) Composition of isogeometric analysis with level set method for structural topology optimization. *Int J Optim Civ Eng* 2:47–70
- Sigmund O (1994) Materials with prescribed constitutive parameters: An inverse homogenization problem. *Int J Solids Struct* 31:2313–2329
- Sigmund O (2001) A 99 line topology optimization code written in Matlab. *Struct Multidiscip Optim* 21:120–127
- Spink M, Claxton D, Falco C de, Vazquez R (2010) *NURBS toolbox*. Octave Forge
- Suresh K (2010) A 199-line Matlab code for Pareto-optimal tracing in topology optimization. *Struct Multidiscip Optim* 42:665–679
- Taheri AH, Suresh K (2017) An isogeometric approach to topology optimization of multi-material and functionally graded structures. *Int J Numer Methods Eng* 109:668–696
- Talischi C, Paulino GH, Pereira A, Menezes IFM (2012) PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. *Struct Multidiscip Optim* 45:329–357
- Vázquez R (2016) A new design for the implementation of isogeometric analysis in Octave and Matlab: *GeoPDEs 3.0*. *Comput Math with Appl* 72:523–554
- Wang MY, Wang X, Guo D (2003) A level set method for structural topology optimization. *Comput Methods Appl Mech Eng* 192:227–246
- Wang S, Wang MY (2006) Radial basis functions and level set method for structural topology optimization. *Int J Numer Methods Eng* 65:2060–2090
- Wang Y, Benson DJ (2016) Isogeometric analysis for parameterized LSM-based structural topology optimization. *Comput Mech* 57:19–35
- Wang Y, Chen F, Wang MY (2017a) Concurrent design with connectable graded microstructures. *Comput Methods Appl Mech Eng* 317:84–101
- Wang Z-P, Poh LH (2018) Optimal form and size characterization of planar isotropic petal-shaped auxetics with tunable effective properties using IGA. *Compos Struct* 201:486–502
- Wang Z-P, Poh LH, Dirrenberger J, et al (2017b) Isogeometric shape optimization of smoothed petal auxetic structures via computational periodic homogenization. *Comput Methods Appl Mech Eng* 323:250–271
- Wei P, Li Z, Li X, Wang MY (2018) An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions. *Struct Multidiscip Optim* 58:831–849
- Xia L, Breitkopf P (2014) Concurrent topology optimization design of material and structure within FE2 nonlinear multiscale analysis framework. *Comput Methods Appl Mech Eng* 278:524–542

- Xia L, Breitkopf P (2015) Design of materials using topology optimization and energy-based homogenization approach in Matlab. *Struct Multidiscip Optim* 52:1229–1241
- Xie X, Wang S, Xu M, et al (2020) A hierarchical spline based isogeometric topology optimization using moving morphable components. *Comput Methods Appl Mech Eng* 360:112696
- Xie X, Wang S, Xu M, Wang Y (2018) A new isogeometric topology optimization using moving morphable components based on R-functions and collocation schemes. *Comput Methods Appl Mech Eng* 339:61–90
- Xie YM, Steven GP (1993) A simple evolutionary procedure for structural optimization. *Comput Struct* 49:885–969
- Xu J, Gao L, Xiao M, et al (2020) Isogeometric topology optimization for rational design of ultra-lightweight architected materials. *Int J Mech Sci* 166:105103
- Yang WY, Zhang WS, Guo X (2016) Explicit structural topology optimization via Moving Morphable Voids (MMV) approach. In: 2016 Asian Congress of Structural and Multidisciplinary Optimization, Nagasaki, Japan. p 98
- Zhang W, Li D, Kang P, et al (2020) Explicit topology optimization using IGA-based moving morphable void (MMV) approach. *Comput Methods Appl Mech Eng* 360:112685
- Zhang W, Yang W, Zhou J, et al (2017) Structural topology optimization through explicit boundary evolution. *J Appl Mech* 84:011011
- Zhang W, Yuan J, Zhang J, Guo X (2016) A new topology optimization approach based on Moving Morphable Components (MMC) and the ersatz material model. *Struct Multidiscip Optim* 53:1243–1260
- Zhao G, Yang J, Wang W, et al (2020a) T-Splines based isogeometric topology optimization with arbitrarily shaped design domains. *Comput Model Eng Sci* 123:1033–1059
- Zhao Q, Fan C-M, Wang F, Qu W (2020b) Topology optimization of steady-state heat conduction structures using meshless generalized finite difference method. *Eng Anal Bound Elem* 119:13–24
- Zhou M, Rozvany GIN (1991) The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. *Comput Methods Appl Mech Eng* 89:309–336

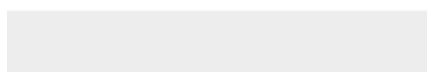
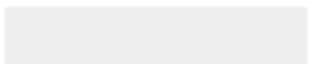




Click here to access/download
Supplementary Material
Boun_Cond.m

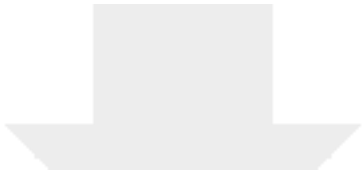


Click here to access/download
Supplementary Material
CASE.m

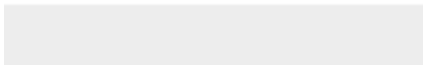
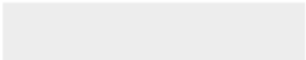


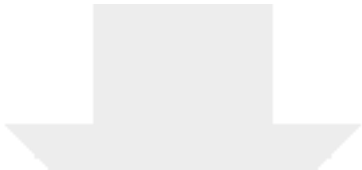


Click here to access/download
Supplementary Material
Geom_Mod.m

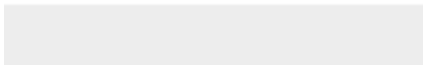
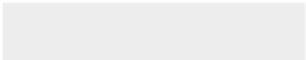


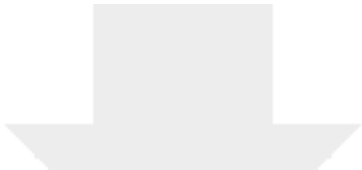
Click here to access/download
Supplementary Material
Guadrature.m



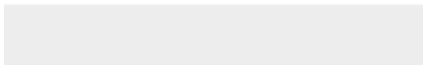
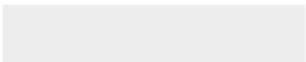


Click here to access/download
Supplementary Material
OC.m



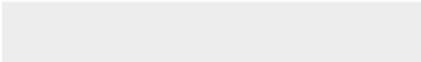


Click here to access/download
Supplementary Material
Plot_Data.m



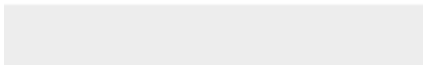


Click here to access/download
Supplementary Material
Plot_Topy.m





Click here to access/download
Supplementary Material
Pre_IGA.m

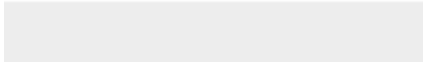


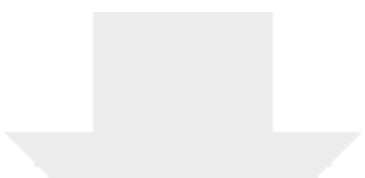


Click here to access/download
Supplementary Material
Shep_Fun.m



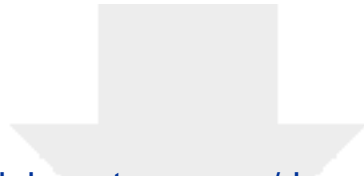
Click here to access/download
Supplementary Material
Solving.m





Click here to access/download
Supplementary Material
Stiff_Ass2D.m





Click here to access/download
Supplementary Material
Stiff_Ele2D.m

