Structural and Multidisciplinary Optimization (2023) 66:146
https://doi.org/10.1007/s00158-023-03590-0

EDUCATIONAL PAPER

MATLAB implementations for 3D geometrically nonlinear topology optimization: 230-line code for SIMP method and 280-line code for MMB method

Yanfang Zhao[1,2] · Guikai Guo[2] · Wenjie Zuo[1,2]

Received: 30 September 2022 / Revised: 6 May 2023 / Accepted: 13 May 2023

Abstract
Topology optimization can generate the innovative material layout to meet the performance demands. Meanwhile, the computational programs with educational purposes were also released for topology optimization. However, few educational papers concerning topology optimization focused on three-dimensional structures considering the geometrical nonlinearity. Therefore, this paper presents complete MATLAB codes for three-dimensional geometrically nonlinear topology optimization for the educational purposes. Two sets of MATLAB codes can be downloaded from the attachments: the 230-line code using the SIMP method and the 280-line code using the moving morphable bars. The 280-line code using the moving morphable bars in the Appendix is composed of 59 lines for the parameter initialization, 59 lines for the main loop of the optimization loop, 64 lines for the relative intermediate subfunctions, and 96 lines for the finite element analysis considering the geometrical nonlinearity. The validity of the codes is confirmed by the numerical examples during the MATLAB implementation.

Keywords Topology optimization · Geometrical nonlinearity · 3D structures · MATLAB · SIMP · Moving morphable bars (MMB)

1 Introduction

Since the seminal paper for topology optimization (TO) was published by using the homogenization method (Bendsøe and Kikuchi 1988), TO has been extensively studied, which can generate the innovative material layout for the prescribed structures by satisfying the design demands. Besides, several critical TO reviews can be recommended for the readers (Deaton and Grandhi 2013; Liu et al. 2018; Zhu et al. 2020; Wang et al. 2021).

To highly understand the basic principle of TO, the relative programs were published for educational purposes. These programs not only were helpful for the newcomers to study TO, but also facilitated the flourished development of TO. For reading convenience, Table 1 collects and lists several program implementations of the existed TO methods. In this table, the solid isotropic material with penalization (SIMP) method (Sigmund 2001; Zuo and Saitou 2017), as one of the density-based TO approaches was extensively used, especially since the publication of the 99-line code in 2001 (Sigmund 2001). Then, other improved and developed versions of the SIMP method were published for two-dimensional (2D) structures (Andreassen et al. 2010; Talischi et al. 2012; Tavakoli and Mohseni 2014; Kim et al. 2020; Zhu et al. 2021). The 169-line code (Liu and Tovar 2014) and the C# code (Lagaros et al. 2018) only focused on the three-dimensional (3D) structural TO. To compact and improve the efficiency of the TO, the new 99-line code and its extension version by 125-line code were constructed for 2D and 3D structures, respectively (Ferrari and Sigmund 2020). By combining the SIMP and homogenization methods, the materials with extreme properties were designed by the 119-line code (Xia and Breitkopf 2015), and the concurrent TO for multiscale composite structures was illustrated in (Gao et al. 2019).

Responsible Editor: Xiaojia Shelly Zhang

Wenjie Zuo
zuowenjie@jlu.edu.cn

[1] State Key Laboratory of Automotive Simulation and Control, Jilin University, Changchun 130025, People's Republic of China

[2] School of Mechanical and Aerospace Engineering, Jilin University, Changchun 130025, People's Republic of China

Published online: 13 June 2023

Table 1 Educational programs for TO

Approach	Names	Authors and years	Environment	Structure	FEA
SIMP	99-line	Sigmund (2001)	MATLAB	2D	Linear
	88-line	Andreassen et al. (2010)	MATLAB	2D	Linear
	PolyTop	Talischi et al. (2012)	MATLAB	2D	Linear
	169-line	Liu and Tovar (2014)	MATLAB	**3D**	Linear
	115-line	Tavakoli and Mohseni (2014)	MATLAB	2D	Linear
	213-line	Chen et al. (2018)	MATLAB, ANSYS	2D	Geometrically nonlinear
	C# 3D	Lagaros et al. (2018)	SAP2000	**3D**	Linear
	108-line	Kim et al. (2020)	FreeFEM	2D	Linear
	99-line	Ferrari and Sigmund (2020)	MATLAB	2D	Linear
	125-line	Ferrari and Sigmund (2020)	MATLAB	**3D**	Linear
	89-line	Zhu et al. (2021)	FreeFEM	2D	Geometrically nonlinear
Homogenization method + SIMP	119-line	Xia and Breitkopf (2015)	MATLAB	2D	Linear
	86-line	Gao et al. (2019)	MATLAB	2D	Linear
	86-line	Gao et al. (2019)	MATLAB	**3D**	Linear
LSM	129-line	Challis (2010)	MATLAB	2D	Linear
	185-line	Laurain (2018)	FEniCS	2D	Linear
	88-line	Wei et al. (2018)	MATLAB	2D	Linear
	62-line	Yaghmaei et al. (2020)	MATLAB	2D	Linear
	108-line	Kim et al. (2020)	FreeFEM	2D	Linear
	80-line	Wang and Kang (2021)	MATLAB	2D	Linear
	100-line	Wang and Kang (2021)	MATLAB	**3D**	Linear
ESO/BESO	Soft-kill	Huang and Xie (2010)	MATLAB	2D	Linear
	100-line	Zuo and Xie (2015)	Python, Abaqus	**3D**	Linear
	SERA	Ansola Loyola et al. (2018)	MATLAB	2D	Linear
	78-line/99-line	Xia et al. (2018)	MATLAB	2D	Linear
	137-line	Han et al. (2021)	MATLAB	2D	Geometrically nonlinear
MMC	188-line	Zhang et al. (2016)	MATLAB	2D	Linear
	218-line	Du et al. (2022)	MATLAB	2D	Linear
	256-line	Du et al. (2022)	MATLAB	**3D**	Linear
TOBS	101-line	Picelli et al. (2020)	MATLAB	2D	Linear
DVTO	128-line	Liang and Cheng (2019)	MATLAB	2D	Linear
GPM	54-line	Smith and Norato (2020)	MATLAB	2D	Linear
MMB	209-Line	Zhao et al. (2021)	MATLAB	2D	Linear

The presentation of numerical procedures for the level-set method (LSM) is also beneficial to the basic understanding of its principles (Wang et al. 2003; Allaire et al. 2004). The codes can be referred from the 129-line code for the discrete LSM (Challis 2010), the 88-line code for the parameterized LSM (Wei et al. 2018), and the 62-line code for the filter-based LSM (Yaghmaei et al. 2020). The 185-line code and the 108-line code were written by using FEniCS (Laurain 2018) and FreeFEM (Kim et al. 2020), respectively. The 80-line code and 100-line code were respectively provided for 2D and 3D structures (Wang and Kang 2021). Besides, several available codes by the evolutionary structural optimization (ESO) method and its improved version BESO method were presented to optimize the 2D structures, such as Soft-kill (Huang and Xie 2010), SERA (Ansola Loyola et al. 2018), 78-line/99-line (Xia et al. 2018), and 137-line (Han et al. 2021). The 3D TO using the BESO method was implemented using the 100-line code on the Python and Abaqus commercial software (Zuo and Xie 2015). Subsequently, the 101-line code (Picelli et al. 2020) was published for the TO of binary structures (TOBS). The MATLAB code concerning the discrete variable TO (DVTO) contained the 128-line code (Liang and Cheng 2019). Recently, the MATLAB codes for the explicit TO methods have successively received widespread attention. For example, the 188-line code for 2D structures (Guo et al. 2014; Zhang et al. 2016) and its easy-to-extend version (Du et al. 2022) were based on the moving morphable component (MMC) method.

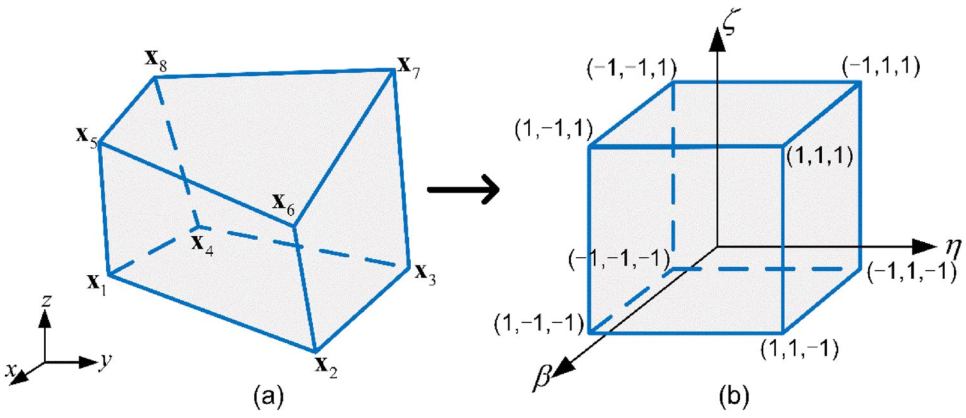


Fig. 1 Description of hexahedral element: **a** Deformed element; **b** Reference element

Subsequently, the 54-line code in the reference (Smith and Norato 2020) just calculated the distance and the implementation details can be found for the geometry projection method (GPM). The moving morphable bar (MMB) method proposed in the reference (Hoang and Jang 2017) was further developed to the 209-line code for the 3D structure (Zhao et al. 2021).

From these aforementioned references and Table 1, the majority of these published programs were developed on the linear finite element analysis (FEA). In particular, it was worth emphasizing that structures in engineering are more efficient under the large deformation assumption, so the geometrical nonlinearity should be recognized as an important consideration in TO design. For geometrically nonlinear TO, the excessive distortion at the low-density elements causes the converged issue of the Newton–Raphson iterations and affects the final optimized result (Kemmler et al. 2005). Therefore, many methods have been proposed to address the excessive distortion, such as the convergence criterion relaxation (Buhl et al. 2000; Pedersen et al. 2001), the removal and reintroduction method for low-density elements (Bruns and Tortorelli 2003), the extreme soft hyperelastic material added method (Luo et al. 2015), the element connectivity parameterization method (Moon and Yoon 2013), the element deformation scaling method (van Dijk et al. 2014), the polyconvex constitutive model (Lahuerta et al. 2013), the interpolation schemes about the elastic energy density (Wang et al. 2014), and the super element condensation method (Hou et al. 2020).

However, there were currently only three references for the educational codes of geometrically nonlinear TO. The 213-line code was constructed on MATLAB and Abaqus, which transferred the data between these two platforms and then caused numerical issues with the sensitivity information (Chen et al. 2018). The 89-line code was established on the open-source program platform FreeFEM (Zhu et al. 2021). The 137-line code on the ESO method was just built based on MATLAB and the calculation of the solid-deformed elements was not reflected in the code (Han et al. 2021). These three codes only focused on 2D structures and did not consider 3D structures. Moreover, MATLAB is more suitable for the code presentation for educational purposes. If the readers want to know more researches related to geometrically nonlinear TO, these literatures can be referred (Luo et al. 2015; Chen et al. 2021; Zhao et al. 2022; Zhu et al. 2022).

Therefore, this paper presents complete MATLAB codes concerning the geometrically nonlinear TO for 3D structures. This proposed method can obtain slightly nonsymmetric optimized results. It is noted that some preliminary notions about the symmetry in topology optimization were outlined by Rozvany GIN Rozvany (2010), who also illustrated that the optimal topologies were non-symmetric in some cases involved the buckling effects. In this paper, the slightly non-symmetric results considering the geometrical nonlinearity by the proposed method can also be regarded as the further development for the non-symmetry of the references (Buhl et al. 2000; Rozvany 2010). The readers can download two sets of MATLAB codes from the attachments: 230-line code using the SIMP method and 280-line code using the MMB method. The 280-line code in the Appendix is mainly used to illustrate the MATLAB implementation. The outline of the paper is summarized as follows. The derivation of the 3D geometrically nonlinear FEA is presented in Sect. 2. Additionally, Sect. 3 constructs the geometrically nonlinear TO. The illustration of the MATLAB implementation by the MMB method is presented in Sect. 4. Then Sect. 5 summarizes the final conclusive remarks.

2 3D geometrically nonlinear FEA

2.1 Basic conversion matrix

The structural large deformation can be addressed by the FEA considering the geometrical nonlinearity. Then, the hexahedral elements are used to discretize the 3D structural domain. The actual elements in Fig. 1a are deformed, which can be solved by transforming them into the reference elements shown in Fig. 1b. The total Lagrangian formulations are adopted to construct the structural deformation, and the interpolations for the displacements and coordinates are expressed by the isoparametric shape function N_I

$$\mathbf{u} = \sum_{I=1}^{8} N_I \mathbf{u}_I,\ \mathbf{x} = \sum_{I=1}^{8} N_I \mathbf{x}_I, \tag{1}$$

where $\mathbf{x}_I$ is the nodal coordinate, $\mathbf{u}_I$ is the displacement. $\mathbf{F}$ is the deformation gradient which is equal to $\mathbf{1} + \nabla_0 \mathbf{u}$. Then $\mathbf{E}$ is the Lagrangian strain written as follows:

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{1}\right). \tag{2}$$

For clarity, E_{kl} and S_{ij} are the second-order tensors that represent the Lagrangian strain $\mathbf{E}$ and the second Piola–Kirchhoff stress $\mathbf{S}$. Besides, D_{ijkl} is the fourth-order tensor that denotes the elastic constitutive matrix $\mathbf{D}$, then the constitutive relationship is written as follows:

$$S_{ij} = D_{ijkl} E_{kl}. \tag{3}$$

The elastic modulus $\mathbf{D}$ is calculated by the power law approach (Sigmund 2007):

$$\mathbf{D} = \left[\rho_{\min} + \rho_e^{\eta}\left(1 - \rho_{\min}\right)\right]\mathbf{D}_0, \tag{4}$$

where η is the penalty factor; $\rho_{\min}$ is set as the minimum value to avoid the numerical issue; ρ_e is the element density. tiaozhanfang2022

2.2 The residual of equilibrium equation

The equilibrium equation for the process of geometrically nonlinear TO is stated as follows:

$$\iiint_{\Omega_0} \bar{\mathbf{u}}^{\mathrm{T}} \mathbf{f}^{\mathrm{b}} d\Omega + \iint_{\Gamma_o^s} \bar{\mathbf{u}}^{\mathrm{T}} \mathbf{t}\, d\Gamma = \iiint_{\Omega_0} \mathbf{S} : \bar{\mathbf{E}}\, d\Omega \tag{5}$$

where $\bar{\mathbf{u}}$ is the virtual displacement, $\bar{\mathbf{E}}$ is the variational form of the strain, $\mathbf{f}^{\mathrm{s}}$ and $\mathbf{f}^{\mathrm{b}}$ are respectively the surface and the body forces. Then the residual for this geometrically nonlinear process is defined as follows:

$$\begin{aligned}&\iiint_{\Omega_0} \mathbf{S} : \bar{\mathbf{E}}\, d\Omega - \left(\iiint_{\Omega_0} \bar{\mathbf{u}}^{\mathrm{T}} \mathbf{f}^{\mathrm{b}} d\Omega + \iint_{\Gamma_o^s} \bar{\mathbf{u}}^{\mathrm{T}} \mathbf{t}\, d\Gamma\right) \\ &= \bar{\mathbf{u}}^{\mathrm{T}}\left[\mathbf{F}^{\mathrm{int}} - \mathbf{F}^{ext}\right] = \bar{\mathbf{u}}^{\mathrm{T}} \mathbf{R}(\mathbf{u})\end{aligned} \tag{6}$$

and the residual force vector $\mathbf{R}(\mathbf{u})$ is defined as follows:

$$\mathbf{R}(\mathbf{u}) = \mathbf{F}^{\mathrm{int}} - \mathbf{F}^{ext} = \mathbf{0}, \tag{7}$$

where $\mathbf{F}^{\mathrm{ext}}$ is the external force vector, and $\mathbf{F}^{\mathrm{int}}$ is the internal force vector which is calculated as follows:

$$\mathbf{F}^{\mathrm{int}} = \iiint_{\Omega_0} \left(\mathbf{B}_{\mathrm{N}}\right)^{\mathrm{T}} \mathbf{S}\, d\Omega. \tag{8}$$

2.3 Linearized and discretized equation

According to the Newton–Raphson (NR) method (Bathe 2006; Kim 2015), the linearized form of Eq. (6) is

$$\bar{\mathbf{u}}^{\mathrm{T}}\left\{\iiint_{\Omega_0} \left[\left(\mathbf{B}_{\mathrm{N}}\right)^{\mathrm{T}} \mathbf{D}\left(\mathbf{B}_{\mathrm{N}}\right) + \left(\mathbf{B}_{\mathrm{G}}\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{B}_{\mathrm{G}}\right)\right] d\Omega\right\}\Delta\mathbf{u} = -\bar{\mathbf{u}}^{\mathrm{T}}\mathbf{R}(\mathbf{u}), \tag{9}$$

where the displacement increment is $\Delta\mathbf{u}$, and the global stiffness matrix $\mathbf{K}_{\mathrm{T}}$ is by derived as follows:

$$\begin{aligned}\mathbf{K}_{\mathrm{T}} &= \iiint_{\Omega_0} \left[\left(\mathbf{B}_{\mathrm{G}}\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{B}_{\mathrm{G}}\right) + \left(\mathbf{B}_{\mathrm{N}}\right)^{\mathrm{T}} \mathbf{D}\left(\mathbf{B}_{\mathrm{N}}\right)\right] d\Omega \\ &= \frac{\partial \mathbf{R}}{\partial \mathbf{u}}\end{aligned}, \tag{10}$$

where the related matrices $\mathbf{H}$, $\mathbf{B}_{\mathrm{N}}$, $\mathbf{B}_{\mathrm{G}}$ are expressed as follows:

$$\mathbf{H} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{12} & S_{22} & S_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{13} & S_{23} & S_{33} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{11} & S_{12} & S_{13} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{12} & S_{22} & S_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{13} & S_{23} & S_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{11} & S_{12} & S_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{12} & S_{22} & S_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{13} & S_{23} & S_{33} \end{bmatrix}_{9\times 9}, \tag{11}$$

$$\mathbf{B}_{\mathrm{G}} = \begin{bmatrix} N_{1,1} & 0 & 0 & N_{2,1} & 0 & 0 & \cdots & N_{8,1} & 0 & 0 \\ N_{1,2} & 0 & 0 & N_{2,2} & 0 & 0 & \cdots & N_{8,2} & 0 & 0 \\ N_{1,3} & 0 & 0 & N_{2,3} & 0 & 0 & \cdots & N_{8,3} & 0 & 0 \\ 0 & N_{1,1} & 0 & 0 & N_{2,1} & 0 & \cdots & 0 & N_{8,1} & 0 \\ 0 & N_{1,2} & 0 & 0 & N_{2,2} & 0 & \cdots & 0 & N_{8,2} & 0 \\ 0 & N_{1,3} & 0 & 0 & N_{2,3} & 0 & \cdots & 0 & N_{8,3} & 0 \\ 0 & 0 & N_{1,1} & 0 & 0 & N_{2,1} & \cdots & 0 & 0 & N_{8,1} \\ 0 & 0 & N_{1,2} & 0 & 0 & N_{2,2} & \cdots & 0 & 0 & N_{8,2} \\ 0 & 0 & N_{1,3} & 0 & 0 & N_{2,3} & \cdots & 0 & 0 & N_{8,3} \end{bmatrix}_{9\times 24}, \tag{12}$$

$$
\mathbf{B}_\mathrm{N} = \begin{bmatrix}
F_{11}N_{1,1} & F_{21}N_{1,1} & F_{31}N_{1,1} & \cdots & F_{11}N_{8,1} & F_{21}N_{8,1} & F_{31}N_{8,1} \\
F_{12}N_{1,2} & F_{22}N_{1,2} & F_{32}N_{1,2} & \cdots & F_{12}N_{8,2} & F_{22}N_{8,2} & F_{32}N_{8,2} \\
F_{13}N_{1,3} & F_{23}N_{1,3} & F_{33}N_{1,3} & \cdots & F_{13}N_{8,3} & F_{23}N_{8,3} & F_{33}N_{8,3} \\
\begin{pmatrix} F_{11}N_{1,2} \\ +F_{12}N_{1,1} \end{pmatrix} & \begin{pmatrix} F_{21}N_{1,2} \\ +F_{22}N_{1,1} \end{pmatrix} & \begin{pmatrix} F_{31}N_{1,2} \\ +F_{32}N_{1,1} \end{pmatrix} & \cdots & \begin{pmatrix} F_{11}N_{8,2} \\ +F_{12}N_{8,1} \end{pmatrix} & \begin{pmatrix} F_{21}N_{8,2} \\ +F_{22}N_{8,1} \end{pmatrix} & \begin{pmatrix} F_{31}N_{8,2} \\ +F_{32}N_{8,1} \end{pmatrix} \\
\begin{pmatrix} F_{13}N_{1,2} \\ +F_{12}N_{1,3} \end{pmatrix} & \begin{pmatrix} F_{23}N_{1,2} \\ +F_{22}N_{1,3} \end{pmatrix} & \begin{pmatrix} F_{33}N_{1,2} \\ +F_{32}N_{1,3} \end{pmatrix} & \cdots & \begin{pmatrix} F_{13}N_{8,2} \\ +F_{12}N_{8,3} \end{pmatrix} & \begin{pmatrix} F_{23}N_{8,2} \\ +F_{22}N_{8,3} \end{pmatrix} & \begin{pmatrix} F_{33}N_{8,2} \\ +F_{32}N_{8,3} \end{pmatrix} \\
\begin{pmatrix} F_{13}N_{1,1} \\ +F_{11}N_{1,3} \end{pmatrix} & \begin{pmatrix} F_{23}N_{1,1} \\ +F_{21}N_{1,3} \end{pmatrix} & \begin{pmatrix} F_{33}N_{1,1} \\ +F_{31}N_{1,3} \end{pmatrix} & \cdots & \begin{pmatrix} F_{13}N_{8,1} \\ +F_{11}N_{8,3} \end{pmatrix} & \begin{pmatrix} F_{33}N_{8,1} \\ +F_{31}N_{8,3} \end{pmatrix} & \begin{pmatrix} F_{33}N_{8,1} \\ +F_{31}N_{8,3} \end{pmatrix}
\end{bmatrix}_{6\times24}, \tag{13}
$$

where $N_{i,j}$ is denoted as the derivative of N_i with X_j. Afterwards, the updated displacement is written as $\mathbf{u}+\Delta\mathbf{u}$. The derivations for $\mathbf{F}^{\mathrm{int}}$ and $\mathbf{K}_\mathrm{T}$ can be referred from the FEA books in details (Bathe 2006; Kim 2015).

3 TO considering geometrical nonlinearity

3.1 Model

For the geometrically nonlinear TO, the objective is generally minimizing the end-compliance by subjecting to the volume constraint, which can be written as follows (Buhl et al. 2000):

$$
\begin{aligned}
&\mathrm{Min}\ C = \left(\mathbf{F}^{ext}\right)^\mathrm{T}\mathbf{u} \\
&\mathrm{s.t.} \begin{cases} \mathbf{R} = \mathbf{F}^{\mathrm{int}} - \mathbf{F}^{ext} = \mathbf{0} \\ \sum_{e=1}^{n_e} \rho_e V_e - fV_0 \le 0 \\ \mathbf{x}_{\min} \le \mathbf{x} \le \mathbf{x}_{\max} \end{cases},
\end{aligned} \tag{14}
$$

where C is the objective function, V_e is the element volume, f represents the volume fraction, and V_0 is the structural initial volume. $\mathbf{x}$ is the design variables, $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ are respectively the lower and upper design variables values. n_e is the element number. ρ_e is the element density.

3.2 Sensitivity analysis

The objective sensitivity in Eq. (14) is determined as follows since the design variable have no effect on the external force:

$$
\frac{\partial C}{\partial \rho_e} = \left(\mathbf{F}^{ext}\right)^\mathrm{T}\frac{\partial \mathbf{u}}{\partial \rho_e}. \tag{15}
$$

And $\partial\mathbf{u}/\partial\rho_e$ is difficult to solve, so the objective is modified by using the adjoint method (Buhl et al. 2000). The multiplier $\boldsymbol{\lambda}$ is introduced into the objective function due to $\mathbf{R}=\mathbf{0}$, which is modified as follows:

$$
\tilde{C} = \left(\mathbf{F}^{ext}\right)^\mathrm{T}\mathbf{u} + \boldsymbol{\lambda}\mathbf{R}. \tag{16}
$$

Then the corresponding sensitivity in Eq. (15) is also modified as follows:

$$
\begin{aligned}
\frac{\partial \tilde{C}}{\partial \rho_e} &= \left(\mathbf{F}^{ext}\right)^\mathrm{T}\frac{\partial \mathbf{u}}{\partial \rho_e} + \boldsymbol{\lambda}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \rho_e} + \frac{\partial \mathbf{R}}{\partial \rho_e}\right) \\
&= \left(\left(\mathbf{F}^{ext}\right)^\mathrm{T} + \boldsymbol{\lambda}\mathbf{K}_T\right)\frac{\partial \mathbf{u}}{\partial \rho_e} + \boldsymbol{\lambda}\frac{\partial \mathbf{R}}{\partial \rho_e}
\end{aligned}. \tag{17}
$$

To fully neglect the influence of $\partial\mathbf{u}/\partial\rho_e$, $\boldsymbol{\lambda}$ is calculated as follows:

$$
\left(\mathbf{F}^{ext}\right)^\mathrm{T} = -\boldsymbol{\lambda}\mathbf{K}_\mathrm{T}. \tag{18}
$$

Since $\mathbf{R}$ is the zero vector, $\boldsymbol{\lambda}$ is chosen at random. The above equation for solving the Lagrange multiplier $\boldsymbol{\lambda}$ is correct only with the absence of non-zero Dirichlet boundary conditions. Eventually, the modified objective sensitivity is obtained by

$$
\frac{\partial \tilde{C}}{\partial \rho_e} = -\mathbf{K}_\mathrm{T}^{-1}\left(\mathbf{F}^{ext}\right)^\mathrm{T}\sum_{e=1}^{n_e}\frac{\partial \mathbf{F}_e^{\mathrm{int}}}{\partial \rho_e}. \tag{19}
$$

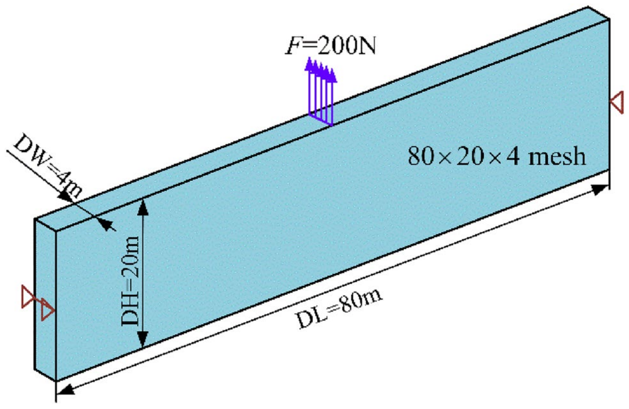


Fig. 2 Design domain of slender beam

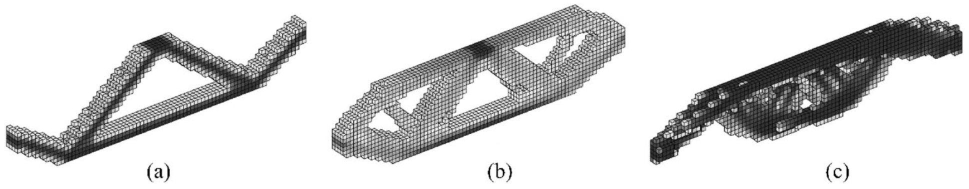


Fig. 3 Optimized results using SIMP method with $F=200$ N: **a** Linear FEA and the OC solution; **b** Geometrically nonlinear FEA and MMA solution (400 steps); **c** Geometrically nonlinear FEA and MMA different parameters solution (200 steps)

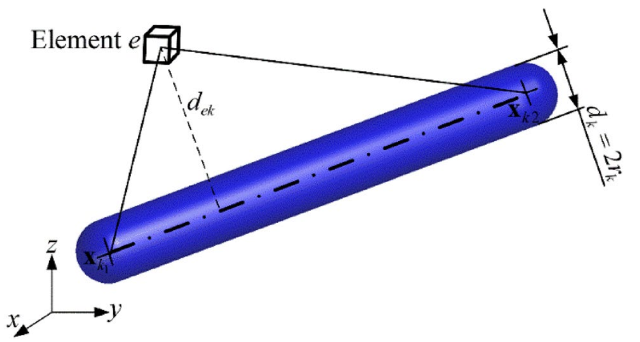


Fig. 4 Element density description using the MMB method

3.3 Illustration for SIMP-based geometrically nonlinear TO

For the geometrically nonlinear TO, the sensitivity filter in the 230-line code for SIMP method is adopted to modify the sensitivities as follows:

$$\overline{\frac{\partial \tilde{C}}{\partial \rho_e}} = \frac{1}{\max(\gamma, \rho_e) \sum_{i \in N_e} H_{ei}} \sum_{i \in N_e} H_{ei} \rho_i \frac{\partial \tilde{C}}{\partial \rho_i}, \tag{20}$$

where N_e is the element number in a circle centered on the element i with the radius $r_{\min}$. H_{ei} is a weight factor and γ is a small constant to avoid the numerical issue.

The slender beam in Fig. 2 is used to illustrate the 230-line code for SIMP geometrically nonlinear TO. The Poisson's ratio μ is 0.4. The volume fraction is 0.2. The Young modulus E_0 is 3 GPa. The linear-optimized result is obtained by the optimality criterion (OC) method, as depicted in Fig. 3a. Then the geometrically nonlinear results are obtained by using the Method of Moving Asymptotes (MMA) with different parameter settings (Svanberg 1987), as depicted in Fig. 3b, c. The significant difference between the linear and geometric nonlinear results was also observed. The more detailed analysis can be found in the reference (Guo et al. 2022). More importantly, the current contours of the optimized results for SIMP-based geometrically nonlinear TO are not clear, but readers can add the Heaviside projection filter to obtain the clear contours of the optimized result, which cause a slight increase in the computational cost. Furthermore, to obtain the clear and explicit contours of optimized results, the element density ρ_e can also be further established and illustrated by the MMB method in the subsequent sections.

3.4 MMB-based geometrically nonlinear TO

The element density ρ_e is constructed by the geometrical parameters of the MMBs, which are chosen as the design variables, namely, $\mathbf{x} = \left\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_b}\right\}$ are the design variables. N_b is the MMBs number. The design variables of the k-th MMB are $\mathbf{x}_k = \left\{x_{k1}, y_{k1}, z_{k1}, x_{k2}, y_{k2}, z_{k2}, r_k\right\}$. Then the element density ρ_e is constructed from the position relationship in Fig. 4:

$$\rho_e = 1 - \prod_{k=1}^{N_b} \frac{1}{1 + \exp\left[-\beta\left(d_{ek} - r_k\right)\right]} \tag{21}$$

where the distance function d_{ek} is constructed from the reference (Guo et al. 2022).

When the MMB method is adopted to establish the element density, then the final objective sensitivity in Eq. (19) is further derived as follows:

$$\frac{\partial \tilde{C}}{\partial \psi_k} = \frac{\partial \tilde{C}}{\partial \rho_e} \frac{\partial \rho_e}{\partial \psi_k}, \tag{22}$$

where ψ_k denotes one of $\mathbf{x}_k = \left\{x_{k1}, y_{k1}, z_{k1}, x_{k2}, y_{k2}, z_{k2}, r_k\right\}$. The detailed derivation for the sensitivity $\partial \rho_e / \partial \psi_k$ can refer to the references (Zhao et al. 2021; Guo et al. 2022).

4 MATLAB implementation by MMB method

This paper is mainly about complete MATLAB codes for 3D geometrically nonlinear TO. If the SIMP method is directly adopted in this paper, the element densities are chosen as the design variables. If the MMB method is adopted to construct

Table 2 MATLAB code parameter meanings

Parameter	Meaning	Parameter	Meaning
DL	Domain length	df0dx	Compliance sensitivity
DW	Domain weight	dfdx	Volume sensitivity
DH	Domain height	Loop	Iteration number for main loop
nelx	Element number in x direction	maxloop	Maximum iteration for main loop
nely	Element number in y direction	loop	NR iteration number
nelz	Element number in z direction	IterMax	Maximum number of NR iterations
xn	Interval MMBs number in x direction	xthreshold	Low density threshold
yn	Interval MMBs number in y direction	ResidualForce	Residual force vector
zn	Interval MMBs number in z direction	ForceVector	External force vector
volfrac	Volume fraction	Fint	Internal force vector
EC	Center coordinates of all element	F	Deformation gradient
rk	Radius	StrainMatrix	Lagrangian strain
xmax	Minimum variables values	uselessNode	Node surrounded by low densities
xmin	Maximum variables values	Nodes2Ele	Adjacent element for each node
E	Young’s modulus	Nodes	All nodal coordinates
nu	Poisson’s ratio	du	Displacement increment
level	Density isosurface value	U	Displacement
Eledensity	All element density	Eles	All elements DOFs
variable	All design variable	Lamada	Multiplier $\boldsymbol{\lambda}$
dek	Distance between element and MMBs	TOL	NR iteration tolerance
dekdx	Sensitivity for the distance	BN	$\mathbf{B}_N$ in Eq. (13)
Var_num	Variable number of each bar	BG	$\mathbf{B}_G$ in Eq. (12)
nn	The tnumber of design variables	SHEAD	$\mathbf{H}$ in Eq. (11)
bata	Parameter β	EKF	Element stiffness matrix
Comp	End-compliance	GKF	Global stiffness matrix

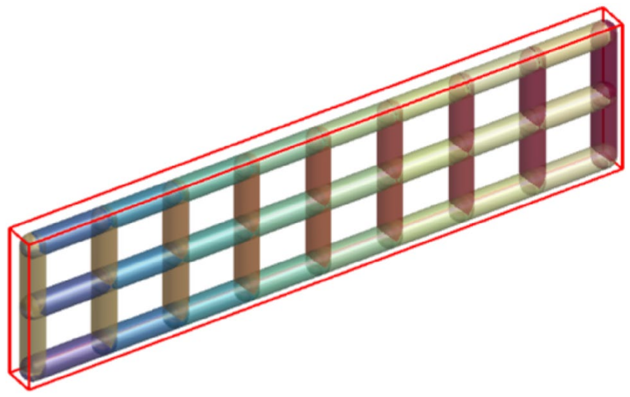


Fig. 5 Initial layout of slender beam

the element density, the design variables are the MMBs geometrical parameters. Furthermore, the MMB method is also essentially the explicit density-based TO method. Therefore, the frameworks of the MATLAB codes for these two methods are basically the same. The main function in MATLAB for the SIMP-based geometrically nonlinear TO contains 230 lines, which are not shown here and can be downloaded from the attachment. Then the 280-line codes by the MMB method in the Appendix are mainly used to illustrate the 3D geometrically nonlinear TO. The codes adopt the MMA to solve the established model. The classical slender beam using the same material properties as in Fig. 2 is again used to illustrate the MATLAB codes.

4.1 Main function

The following command line invokes the main code in MATLAB:

```
function MMB_3D_GNTO(DL,DW,DH,nelx,nely,nelz,xn,yn,zn,volfrac,rk)
```

where the parameter meanings for the MATLAB code are described in Table 2.

4.2 Parameters initialization: lines 2–60

(1) Lines 2–20: Preparing initial parameters, such as material properties, the magnitude of the external force, the center and nodal coordinates of the element, and the file name for saving the iterative result. The iterative number of the optimization process is chosen as 100.
(2) Lines 20–28: Setting up the initial variables, the maximum and minimum values for all design variables. As depicted in Fig. 5, the layout with forty-two solid MMBs with the radius 1.5 m is initially determined by function `[Layout3Face_HM]`. The function `[Layout3Face_HM]` can be easily modified for the initial variables of other examples, so it is not shown here. The upper and lower values for the design variables can be set from the reference (Guo et al. 2022). The MMBs are deleted to accelerate the optimization process if the corresponding radius is less than 0.1 m.
(3) Lines 29–36: Setting up the MMA parameters.
(4) Lines 37–44: Determining the degrees of freedom (DOFs) for the loading and boundary conditions for a slender beam, as shown in Fig. 2.
(5) Lines 45–60: Preparing the parameters for FEA. Line 60 is used to find adjacent elements for each node by function `[NodeSurroundingElement]` in lines 218–224 to subsequently eliminate the influence of the low-density elements.

4.3 Main loop of the optimization process: lines 61–119

(1) Lines 61–62: Preparing the main loop parameters for the optimization.

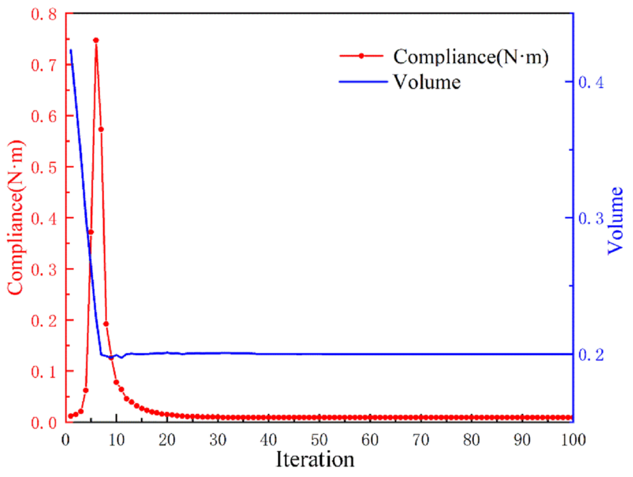
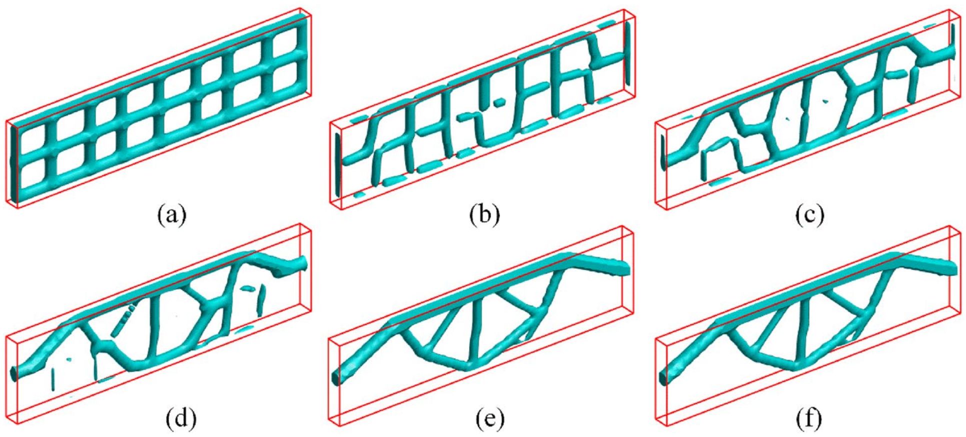


Fig. 7 Iterative histories for the optimized result considering the geometrical nonlinearity

(2) Lines 63–75: Obtaining the element densities by the MMB method. And the distance function `[DEK]` is modified and compacted by comparing with the reference (Zhao et al. 2021).
(3) Lines 76–79: Displaying and saving the figures of the iterative result by the element density isosurface in function `[visualizeLevelSet_1]`, as shown in Fig. 6, which shows some intermediate iterative results considering the geometrical linearity.
(4) Lines 80–81: Finding nodal numbers surrounded by low-density elements by function `[NodeSurround-WithVoidElement]`.
(5) Lines 82–84: Geometrically nonlinear FEA and solving the residual force sensitivity with the element density by function `[GNFEA]`, which are illustrated in details later.
(6) Lines 85–98: Calculating the corresponding sensitivities with the design variables, respectively.

Fig. 6 Intermediate iterative results with $F=200$ N considering the geometrical linearity. **a** 2nd; **b** 10th; **c** 15th; **d** 20th; **e** 50th; **f** 100th



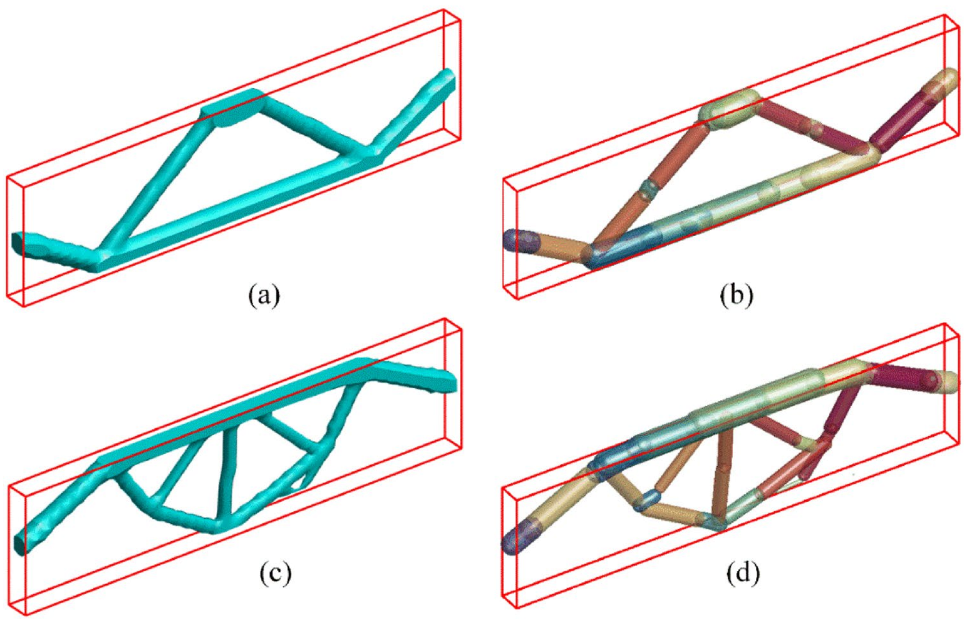


Fig. 8 Optimized results using the force $F = 200$N. **a** Linear result. **b** Final layout of linear result by the MMBs geometrical shape. **c** Geometrically nonlinear result. **d** Final layout of geometrically nonlinear result by the MMBs geometrical shape

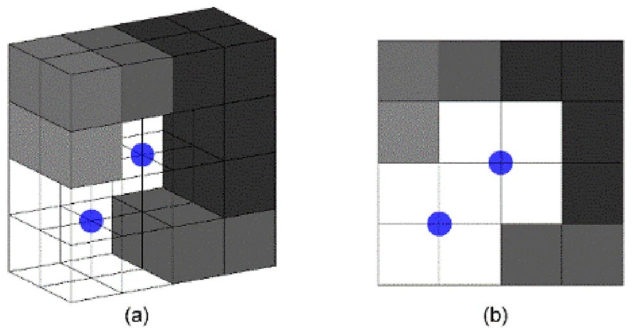


Fig. 9 Nodes surrounded by minimum density elements. **a** Oblique view; **b** Front view

(7) Lines 99–119: Solving the optimization model by the MMA. To accelerate the optimization process, the minimal bars are directly neglected in lines 110–111, which can also be deleted in the codes.

Then the slender beam considering the geometrical nonlinearity is optimized and thereby shown in Fig. 8c, d, which is slightly nonsymmetric and explained in the reference (Guo et al. 2022). Meanwhile, the iterative histories are presented for the end-compliance and volume fraction of this nonlinear result in Fig. 7. It is easy to observe that the optimization process converges so quickly to achieve the geometrically nonlinear result. Moreover, for a clear comparison, the linear-optimized result in Fig. 8a, b is obtained by the linear MATLAB codes from the main codes in the reference (Zhao et al. 2021) and the modified function `[DEK]` in this paper. By comparing these two optimized results with different FEA, this example shows that the geometrical nonlinearity has the substantial impact on the eventual optimized result.

4.4 Function [GNFEA]: lines 120–216

(1) Lines 121–126: Initialize the parameters for geometrically nonlinear FEA. And the symbol `D0` is the predefined constitutive $\mathbf{D}_0$ in Eq. (4) which is calculated by the Poisson ratio μ and Young's modulus E_0.

$$\mathbf{D}_0 = \frac{E_0}{(1+\mu)(1-2\mu)} \times \begin{bmatrix} 1-\mu & \mu & \mu & 0 & 0 & 0 \\ & 1-\mu & \mu & 0 & 0 & 0 \\ & & 1-\mu & 0 & 0 & 0 \\ & & & (1-2\mu)/2 & 0 & 0 \\ & & & & (1-2\mu)/2 & 0 \\ & & & & & (1-2\mu)/2 \end{bmatrix}_{6\times 6} . \quad (23)$$

(2) Lines 127–188: NR iterative solution: Calculating the displacement, the internal force vector, the residual force vector, and the global stiffness matrix by the Gaussian point integral. Especially for lines181–186, the method in the references (Buhl et al. 2000; Guo et al. 2022) is used to alleviate the non-convergence issues by neglecting the excessive distortion in the nodal DOFs of low-density elements, as shown in Fig. 9.

(3) Lines 189–216: Calculating the residual force sensitivity with the element densities.

4.5 The relative intermediate subfunctions: lines 217–280

(1) Lines 218–224: Function `[NodeSurrounding-Element]` is used to find adjacent elements for each node.
(2) Lines 225–230: Function `[NodeSurroundWith-VoidElement]` is used to determine the nodal numbers surrounded by low-density elements.
(3) Lines 231–250: Function `[Shape function]` is used to calculate the derivation of the shape function and Jacobi matrix.
(4) Lines 251–280: Function `[DEK]` is used to calculate the distance function and its sensitivities from the modified formulations in the reference (Guo et al. 2022), which denotes function `[DEK]` is compressed by compared with the reference (Zhao et al. 2021).

4.6 Analysis of the optimized results

By considering the geometrically nonlinear FEA, the clear comparison of the optimized results between the SIMP and MMB methods has been listed in Table 3. From this table, the end-compliances for the three optimized results are almost identical. Besides, it clearly shows that the solution time for a typical iterative step by the MMB method is slightly less than the SIMP method, because the number of design variables for the MMB method is 294, which is less than the 6400 design variables SIMP method. The geometrically nonlinear FEA consumes the most computational time, followed by the sensitivity analysis using less time, and the MMA solution using the least time. However, the optimized result by the MMB method can be obtained by quickly converged than the SIMP method.

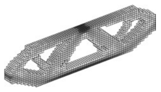
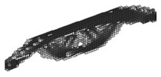
To further show the resistance to deformation, the optimized results in Fig. 8 are imposed on the large force $F = 2 \times 10^8$ N. The structural deformations are depicted in Fig. 10, which also shows that the linear-optimized result is more prone to the local buckling. Moreover, here the end-compliance for the linear-optimized result in Fig. 10a is 9.96×10^9 N m, whereas the end-compliance for the geometrically linear-optimized result in Fig. 10b is 7.87×10^9 N m. Namely, it clearly shows that the geometrically linear-optimized result is more resistant to the structural deformation.

4.7 Extensions for other loading and boundary conditions

The MATLAB codes can be conveniently implemented by modifying the loading and boundary conditions for other examples. To optimize the double-clamped beam in Fig. 11, the volume fraction f is 0.1. The external force F is 10^6 N. Since F is quite large, and then the objective function amplification factor for the MMA in line 3 is changed to `scale=1.0/10000000`. The initial parameters are the same as the slender beam, such as material parameters. The dimensions of the design domain are easily modified in the main function line. The boundary and loading conditions in lines 37–44 are changed to the following codes:

```
% User-defined load DOFs for double clamped beam
 loadnid=(nelx+1)*(nely+1)*nelz+(nelx+1)*(nely/2)+nelx/2+1;
loaddof = 3*loadnid;
% User-defined support fixed DOFs
[jf,kf] = meshgrid(1:nely+1,1:nelz+1);
fixednidL = (kf-1)*(nely+1)*(nelx+1)+(jf-1)*(nelx+1)+1;
fixednidR=fixednidL+nelx;
fixednid =[fixednidL fixednidR];
Fixeddofs = [3*fixednid(:); 3*fixednid(:)-1; 3*fixednid(:)-2];
```

Table 3 Comparison between the MMB and the SIMP methods for a typical iteration step

Approaches	SIMP (MMA)[a]	SIMP (MMA)	MMB (MMA)
Optimized results			
FEA	52.5634 s	52.5634 s	52.0771 s
Sensitivity	29.5173 s	29.4859 s	28.0879 s
MMA	0.9818 s	0.9707 s	0.0050 s
A typical iterative step	83.06 s	83.02 s	80.17 s
End-compliance	0.009 N m	0.008 N m	0.009 N m
Total iterative steps	400 steps	200 steps	100 steps

[a]Using different parameters in MMA to get better convergence

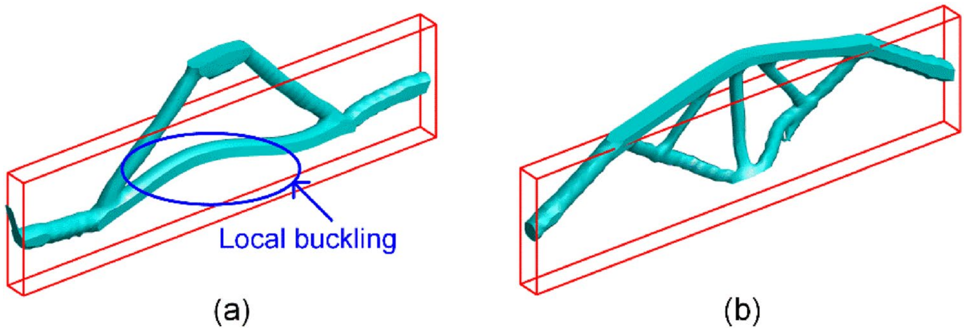


Fig. 10 Structural deformation by the force $F = 2 \times 10^8$ N. **a** Linear result with the end-compliance 9.96×10^9 N m. **b** Geometrically linear result with the end-compliance 7.87×10^9 N m

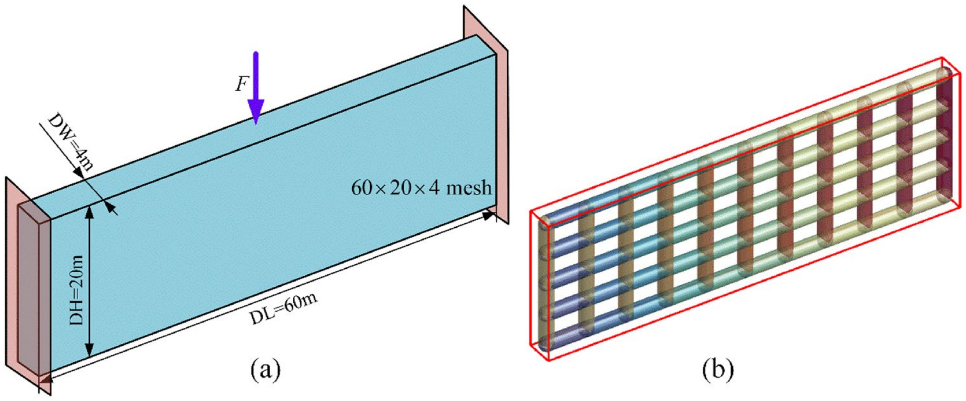


Fig. 11 Double-clamped beam. **a** Structural domain. **b** Initial MMBs layout

Subsequently, the geometrically nonlinear result is clearly displayed in Fig. 12c, and its final layout using the MMBs is depicted in Fig. 12d. Meanwhile, the linear-optimized result and its final layout are presented in Fig. 12a, b, respectively. This example also demonstrates the effectiveness of the codes. Furthermore, this double-clamped beam has been optimized by another initial layout for different load magnitudes in the reference (Guo et al. 2022). Therefore, the influence of the load magnitude on the final geometrically nonlinear-optimized results is not discussed here.

To further verify the effect of the geometrical nonlinearity on the optimized results, more different external forces are set for the double-clamped beam in Fig. 11. The corresponding geometrically linear-optimized results are listed in Table 4. It can be seen that with the increase of the external force, the shapes of the optimized results show the distinct difference. Namely, it also shows that the geometrical nonlinearity has an important influence on the optimized results under the larger forces.

The optimized results in Table 4 can be used to further illustrate the symmetry and non-symmetry for optimized results. When the magnitude of the external force is small, the displacement deformation is also very small. So these optimized results considering geometrical nonlinearity can

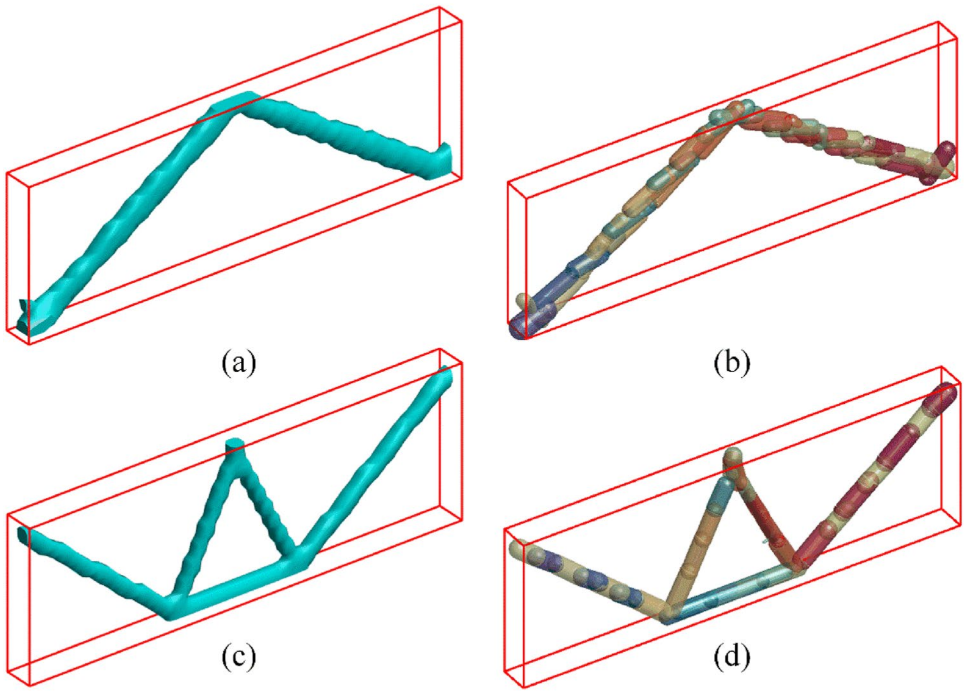


Fig. 12 Optimized results. **a** Linear result. **b** Final layout of linear result by the MMBs geometrical shape. **c** Geometrically nonlinear result. **d** Final layout of geometrically nonlinear result by the MMBs geometrical shape

Table 4 Geometrically nonlinear-optimized results using different force magnitudes

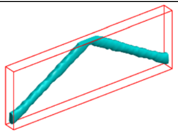
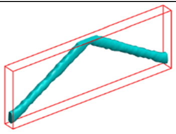
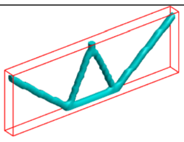
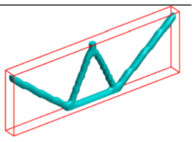
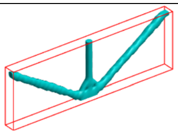
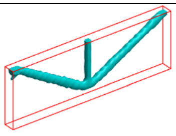
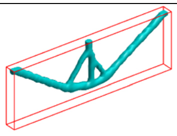
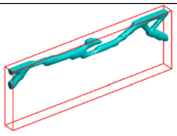
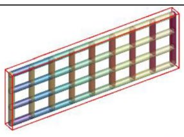
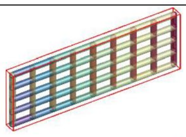
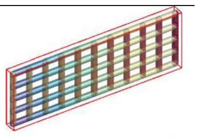
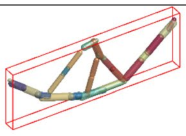
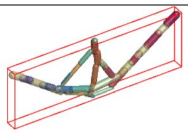
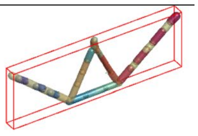
Force F (N)	10^3	10^5	1×10^6	5×10^6
Optimized results				
Force F (N)	2×10^7	3×10^7	4×10^7	2×10^8
Optimized results				

Table 5 Geometrically optimized results by using three different initial layouts

Bar number	59	76	84
Initial layout			
Optimized results			
End-compliance	30771.438	29056.164	26257.726

be considered as linear-optimized results, which do not involve the nonlinear buckling. Afterwards, with the increase of the force, the geometrical nonlinearity begins to affect the shape of the optimized results. The optimized results are slightly non-symmetric due to the buckling effects, which are identical to the explanation in (Buhl et al. 2000; Rozvany 2010). Moreover, it is found that when the loading force F is further increased, the shapes of the optimized results change again, which are also not studied and observed for 2D structures in the literature (Buhl et al. 2000; Zhu et al. 2018, 2022). Besides, the final bars are distributed toward the upper part of the optimized result, as shown in the last data in Table 4. It also reveals that geometrically nonlinearity can affect the shape of the optimized result and its symmetry.

To illustrate the effect of the initial layout on the final optimized result, three different initial layouts are used for double-clamped beam in Fig. 11 by imposing the same external force 2×10^6 N. The numbers of the bars for these three initial layouts are 59, 76, and 84, respectively. The corresponding geometrically optimized results are listed in Table 5, which clearly show that the initial layout has an impact on the optimized result. As the number of the bars increases, the end-compliance for the optimized results decreases. Therefore, to obtain excellent optimized results, more number of bars should be appropriately selected as the initial layout.

5 Conclusion

This paper constructs two sets of MATLAB codes for 3D geometrically nonlinear TO by the SIMP and MMB methods. The frameworks of these two MATLAB codes are basically the same by adopting the end-compliance to establish the optimization model. The only difference is the element density function determined by the MMB method, which can explicitly describe the contours of the optimized results. To avoid redundant elaboration, the 280-line code by the MMB method is selected for educational purposes, which contains 261 main lines by excluding the 19 lines for the explanation. The 280 lines of this code refer to the main code, which illustrates details of the implementation considered for the parameter initialization, geometrically nonlinear finite element analysis, the main loop, etc. Then it is enough for the readers to fully understand the educational codes. These codes are completely in MATLAB without the consideration of other software. In future works, these codes will provide the convenience for the further extension of the 3D geometrically nonlinear structures.

Appendix

```
function MMB_3D_GNTO(DL,DW,DH,nelx,nely,nelz,xn,yn,zn,volfrac,rk)
penal=3;
scale=1.0;    % Objective function amplification factor for MMA
E=3e9;        % Young's modulus
nu=0.4;       % Poisson ratio
xthreshold=0.01; % Low density threshold
force=200 ;   % Magnitude of external force
IterMax=16;   % Maximum number of Newton-Raphson iterations
pmin=1e-4;
level = 0.6; % Density isosurface values
parent_dir_name ='GNTO results'; % File name for saving the figures
mkdir(parent_dir_name);
% Calculate the coordinates of the center points for all elements
EL=DL/nelx; EW=DW/nely; EH=DH/nelz;
[ey,ex,ez]=meshgrid(EW * [0.5 : nely],EL * [0.5 : nelx],EH * [0.5 : nelz]);
EC.dim = 3;EC.xs = cell(EC.dim, 1);
EC.xs{1}=ex; EC.xs{2}=ey; EC.xs{3}=ez;
% Calculate all nodal coordinates
[ey1,ex1,ez1] = meshgrid(0:nely, 0:nelx, 0:nelz);
Nodes=[ex1(:) ey1(:) ez1(:)];
% Initial layout of the design domain by MMBs
[variable,N]=Layout3Face_HM(DL,DW,DH,xn,yn,zn,rk);
% Maximum and minimum values for the design variables
xmin=[ 0.0 ; 0.0 ;  0.0 ; 0.0 ;0.0 ; 0.0 ; -5];
[Var_num,~]=size(xmin);
xmin=repmat(xmin,N,1);
xmax=[DL ;DW;DH ;DL ;DW; DH;  DH/4];
xmax=repmat(xmax,N,1);
% MMA parameters
m=1;   nn=Var_num*N;
[low,upp]=deal(ones(nn,1));
c=5000*ones(m,1);
a0=1;
[d,b]=deal(zeros(m,1));
xy00=variable(:);
[xold1,xold2]=deal(xy00);
% Nodal DOFs for the loading force
loadnid=(nelx+1)*(nely+1)*nelz+(nelx/2+1):(nelx+1):(nelx+1)*(nely+1)*(nelz+1);
loaddof = 3*loadnid;
% Nodal DOFs at the fixed end
fixednidL =(nelx+1)*(nely+1)*nelz/2+1:(nelx+1):(nelx+1)*(nely+1)*(nelz/2+1);
fixednidR=fixednidL+nelx;
fixednid =[fixednidL fixednidR];
Fixeddofs = [3*fixednid(:); 3*fixednid(:)-1; 3*fixednid(:)-2];
% Prepare for FEA
nele = nelx*nely*nelz;
ndof = 3*(nelx+1)*(nely+1)*(nelz+1);
Alldofs=1:ndof;
Freedofs=setdiff(Alldofs,Fixeddofs);
ForceVector = sparse(loaddof,1,force,ndof,1);
nodegrd = reshape(1:(nelx+1)*(nely+1),nelx+1,nely+1);
nodeids = reshape(nodegrd(1:end-1,1:end-1),nely*nelx,1);
nodeidz = 0:(nely+1)*(nelx+1):(nelz-1)*(nely+1)*(nelx+1);
nodeids = repmat(nodeids,size(nodeidz))+repmat(nodeidz,size(nodeids));
edofVec = 3*nodeids(:);
edofMat=repmat(edofVec,1,24)+repmat([-2 -1 0 1 2 3 3*nelx+[4 5 6 1 2 3] ...
    3*(nely+1)*(nelx+1)+[-2 -1 0 1 2 3 [3*nelx+[4 5 6 1 2 3]]]] ,nele,1);
Eles= repmat(nodeids(:),1,8)+repmat([0 1 nelx+[2 1] ...
    (nely+1)*(nelx+1)+[0 1 nelx+[2 1]]],nele,1);
```

```
[Nodes2Ele]=NodeSurroundingElement(Nodes,Eles); %Find the adjacent cell for each node
Loop=1; change=1; maxloop=101; Lamada=zeros(ndof,1);
while change>0.00001 && Loop<maxloop
    % Calculate the element density by MMBs
    bata=min(4+Loop*(4-2)/30,6);
    alctr=min(0.993+Loop*0.005/50,0.9999);  %MMA parameter
    pa=1.;
    dek=cell(N,1);
    dekdx=cell(N,1);
    for k=1:N
        [dek{k},dekdx{k}]=DEK(xy00(Var_num*k-Var_num+1:Var_num*k),
ex(:),ey(:),ez(:));
        mm=-bata*(dek{k}-xy00(Var_num*k));
        pa=pa*1./(1.+exp(mm));
    end
    PE=1.-pa;
    Eledensity=reshape(PE,nelx,nely,nelz);
    % Show the figure for the optimized result
    clf;   h = visualizeLevelSet_1(EC, Eledensity, 'surface', level);
    FileName=[parent_dir_name,'\Fig1_',int2str(Loop),'.png'];
    saveas(h,FileName);
    % Find nodes surrounded by low-density elements
    [uselessNode]=NodeSurroundWithVoidElement(Nodes2Ele, PE,xthreshold);
    % Geometrically nonlinear FEA + Residual force sensitivity with the element density
    [U,GKF,dRdpe]=GNFEA(ndof,nele,Freedofs,IterMax,ForceVector,Nodes,Eles,edofMat,
E,nu,PE,penal,uselessNode,pmin);
    Lamada(Freedofs,:)=GKF(Freedofs,Freedofs)\ForceVector(Freedofs,:);
    dfdpe=full(-Lamada'*dRdpe)';
    % Objective function and sensitivity analysis
    Comp=ForceVector'*U; % End-compliance
    df0dx=zeros(Var_num*N,1); % The sensitivity of the objective function
    dfdx=zeros(Var_num*N,1);   % The sensitivity of the constraint function
    for k=1:N
        penal=3;
        mid=exp(-bata*(dek{k}-xy00(Var_num*k)));
        dpedek=-(1.-PE).*bata.* mid./(1.+ mid);
        dpedrk=-1*dpedek;
        drkdx=[0,0,0,0,0,0,1];
        df0dx(Var_num*k-Var_num+1:Var_num*k,1)=(dfdpe.*dpedek)'*dekdx{k}+dfdpe'*
dpedrk*drkdx;
        dfdx(Var_num*k-Var_num+1:Var_num*k,1)=(dpedek'*dekdx{k})+sum(dpedrk*drkdx);
    end
    %MMA solve
    xval=xy00;
    xold=xy00;
    f0val =Comp*scale;
    df0dx=df0dx*scale;
    df0dx2=0*df0dx;
    fval=sum(PE)/(nelx*nely*nelz*volfrac)-1;
    dfdx=dfdx/(nelx*nely*nelz*volfrac);
    dfdx2=0*dfdx;
    [xmma,ymma,zmma,lam,xsi,eta,mu,zet,ss,low,upp] = mmasub(m,nn,Loop,xval,xmin,
xmax,xold1,xold2,f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,b,c,d,alctr);
    xy00=xmma;
    % Delete the minimal bars to accelerate the process
    [xy00,xold,xold1,xmin,xmax,low,upp,N,nn]=DeleteBar(xy00,xold,xold1,xmin,xmax,
low,upp,N,Var_num);
    xold2 = xold1;
    xold1 = xy00;
    change=max(max(abs( xy00-xold)));
    disp([' It.:' sprintf('%4i\t',Loop)'Obj.:' sprintf('%6.3f\t',Comp)'Vol.: '...
        sprintf('%6.4f\t',mean(PE)) 'ch.:' sprintf('%6.4f\t',change)]);
    Loop = Loop + 1;
end
end
% === Geometrically nonlinear FEA + Residual force sensitivity with the element density ===
function [U,GKF,dRdpe]=GNFEA(ndof,nele,Freedofs,IterMax,ForceVector,Nodes,
Elements,edofMat,E,nu,x,p,uselessNode,pmin)
U=zeros(ndof,1); loop = 0; % Iteration number
du=zeros(ndof,1);           % Displacement increment
ResidualForceMax=max(abs(ForceVector(Freedofs)));
D0=LinearElasticD(E,nu);    % Given constitutive tensor
TOL=1;% Newtonian iteration tolerance
while loop<IterMax  && ResidualForceMax>TOL
    Dimension=3;
    Fint=zeros(ndof,1);      % Initialize the internal force vector
    GKF=sparse(ndof,ndof);  %Initialize the tangent stiffness array
    GaussCoordinate=[-0.57735026918963D0, 0.57735026918963D0];
    GaussWeight=[1.00000000000000D0, 1.00000000000000D0];
    for i=1:nele
        D=(x(i)^(p)*(1-pmin)+pmin)*D0;
        ElementNodeCoordinate=Nodes(Elements(i,:),:); % Element node coordinate matrix
        edof=edofMat(i,:);
        ElementDisplacement0=U(edof);
        ElementDisplacement=reshape(ElementDisplacement0,Dimension,8);
        % Calculate all Gaussian points in each element
        for LX=1:2, for LY=1:2, for LZ=1:2
          E1=GaussCoordinate(LX); E2=GaussCoordinate(LY); E3=GaussCoordinate(LZ);
          [dNdx, JacobiDET] = ShapeFunction([E1 E2 E3], ElementNodeCoordinate);
          FAC=GaussWeight(LX)*GaussWeight(LY)*GaussWeight(LZ)*JacobiDET;
          F=ElementDisplacement*dNdx' + eye(3); %Calculate the deformation gradient
          StrainMatrix=0.5*(F'*F-eye(3));   % Calculate Lagrangian strain
          Strain=[StrainMatrix(1,1) StrainMatrix(2,2) StrainMatrix(3,3)
2*StrainMatrix(1,2) 2*StrainMatrix(2,3) 2*StrainMatrix(1,3)]';
          GaussPointStress=D*Strain;
          BN=zeros(6,24);
          BG=zeros(9,24);
          for I=1:8
            COL=(I-1)*3+1:(I-1)*3+3;
BN(:,COL)=[dNdx(1,I)*F(1,1) dNdx(1,I)*F(2,1) dNdx(1,I)*F(3,1);
             dNdx(2,I)*F(1,2) dNdx(2,I)*F(2,2) dNdx(2,I)*F(3,2);
             dNdx(3,I)*F(1,3) dNdx(3,I)*F(2,3) dNdx(3,I)*F(3,3);
             dNdx(1,I)*F(1,2)+dNdx(2,I)*F(1,1) dNdx(1,I)*F(2,2)+dNdx(2,I)*F(2,1)
dNdx(1,I)*F(3,2)+dNdx(2,I)*F(3,1);
             dNdx(2,I)*F(1,3)+dNdx(3,I)*F(1,2) dNdx(2,I)*F(2,3)+dNdx(3,I)*F(2,2)
dNdx(2,I)*F(3,3)+dNdx(3,I)*F(3,2);
             dNdx(1,I)*F(1,3)+dNdx(3,I)*F(1,1) dNdx(1,I)*F(2,3)+dNdx(3,I)*F(2,1)
dNdx(1,I)*F(3,3)+dNdx(3,I)*F(3,1)];
BG(:,COL)=[dNdx(1,I)  0        0;
             dNdx(2,I)  0        0;
             dNdx(3,I)  0        0;
             0        dNdx(1,I)  0;
             0        dNdx(2,I)  0;
             0        dNdx(3,I)  0;
             0        0        dNdx(1,I);
             0        0        dNdx(2,I);
             0        0        dNdx(3,I)];
            end
            Fint(edof) = Fint(edof) + FAC*BN'*GaussPointStress; % Internal force
            SIG=[GaussPointStress(1) GaussPointStress(4) GaussPointStress(6);
                  GaussPointStress(4) GaussPointStress(2) GaussPointStress(5);
                  GaussPointStress(6) GaussPointStress(5) GaussPointStress(3)];
            SHEAD=zeros(9);
            SHEAD(1:3,1:3)=SIG;  SHEAD(4:6,4:6)=SIG;  SHEAD(7:9,7:9)=SIG;
            EKF = BN'*D*BN + BG'*SHEAD*BG; % Calculate element stiffness matrix
            GKF(edof,edof)=GKF(edof,edof)+FAC*EKF;% Assemble global stiffness matrix
         end; end; end
    end
    ResidualForce = ForceVector-Fint;
    du(Freedofs,:) = GKF(Freedofs,Freedofs)\ResidualForce(Freedofs,:);
    U = U + du;
    if  loop>0 && ~isempty(uselessNode)
        uselessDOF=[3*uselessNode-1;3*uselessNode];% Low density elements nodal DOFs
        Freedofs=setdiff(Freedofs,uselessDOF);   % Update freedoms DOFs for convergence
        ResidualForceMax=max(abs(ResidualForce(Freedofs)));
        disp([' N-R process.' sprintf('%4i\t:',loop) 'Residual: ' sprint
('%6.3f\t',full(ResidualForceMax)) ]);
    end
    loop = loop + 1;
end
dRdpe=sparse(ndof,size(Elements,1));%Residual force sensitivity with element densities
for i=1:nele
    dDdx=(p*x(i)^(p-1)*(1-pmin))*D0;
    ElementNodeCoordinate=Nodes(Elements(i,:),:);
    edof=edofMat(i,:);
    ElementDisplacement0=U(edof);
    ElementDisplacement=reshape(ElementDisplacement0,Dimension,8);
    for LX=1:2, for LY=1:2, for LZ=1:2
          E1=GaussCoordinate(LX); E2=GaussCoordinate(LY); E3=GaussCoordinate(LZ);
          [dNdx, JacobiDET] = ShapeFunction([E1 E2 E3], ElementNodeCoordinate);
          FAC=GaussWeight(LX)*GaussWeight(LY)*GaussWeight(LZ)*JacobiDET;
          F=ElementDisplacement*dNdx' + eye(3); %Calculate the deformation gradient
          StrainMatrix=0.5*(F'*F-eye(3));   % Calculate Lagrangian strain
          Strain=[StrainMatrix(1,1) StrainMatrix(2,2) StrainMatrix(3,3)
2*StrainMatrix(1,2) 2*StrainMatrix(2,3) 2*StrainMatrix(1,3)]';
          BN=zeros(6,24);
          for I=1:8
            COL=(I-1)*3+1:(I-1)*3+3;
BN(:,COL)=[dNdx(1,I)*F(1,1) dNdx(1,I)*F(2,1) dNdx(1,I)*F(3,1);
             dNdx(2,I)*F(1,2) dNdx(2,I)*F(2,2) dNdx(2,I)*F(3,2);
             dNdx(3,I)*F(1,3) dNdx(3,I)*F(2,3) dNdx(3,I)*F(3,3);
             dNdx(1,I)*F(1,2)+dNdx(2,I)*F(1,1) dNdx(1,I)*F(2,2)+dNdx(2,I)*F(2,1)
dNdx(1,I)*F(3,2)+dNdx(2,I)*F(3,1);
             dNdx(2,I)*F(1,3)+dNdx(3,I)*F(1,2) dNdx(2,I)*F(2,3)+dNdx(3,I)*F(2,2)
dNdx(2,I)*F(3,3)+dNdx(3,I)*F(3,2);
             dNdx(1,I)*F(1,3)+dNdx(3,I)*F(1,1) dNdx(1,I)*F(2,3)+dNdx(3,I)*F(2,1)
dNdx(1,I)*F(3,3)+dNdx(3,I)*F(3,1)];
          end
           dRdpe(edof,i) = dRdpe(edof,i) + FAC*BN'*dDdx*Strain;
     end; end; end
end
end
% === Obtain elements and nodes during deformation large displacement ===
function [Nodes3Ele]=NodeSurroundingElement(Nodes,Eles)
Nodes3Ele=sparse(size(Nodes,1),size(Eles,1));
for i=1:size(Nodes,1)
    [hang,~]=find(Eles==i);
    Nodes3Ele(i,hang)=ones(1,size(hang,1));
end
end
function [uselessNode]=NodeSurroundWithVoidElement(Nodes2Ele, xPhy,xmin)
xPhyMatrix=repmat(xPhy',size(Nodes2Ele,1),1);
a=Nodes2Ele.*xPhyMatrix;
```

```
[max_a,~]=max(a,[],2);
[uselessNode]=find(max_a<xmin);% Find node numbers surrounded by low densities
end
% === Calculate the derivation of shape function and Jacobi matrix ===
function [dNdx, JacobiDET] = ShapeFunction(GaussPoint, ElementNode)
ParentNodes=[-1  1  1 -1 -1  1  1 -1;
   -1 -1  1  1 -1 -1  1  1;
   -1 -1 -1 -1  1  1  1  1]; %The coordinates for the reference element
ParentNDerivative=zeros(3,8);
for I=1:8
   XPoint = ParentNodes(1,I);
   YPoint = ParentNodes(2,I);
   ZPoint = ParentNodes(3,I);
   ShapePart = [1+GaussPoint(1)*XPoint 1+GaussPoint(2)*YPoint 1+GaussPoint(3)
*ZPoint];
   ParentNDerivative(1,I) = 0.125*XPoint*ShapePart(2)*ShapePart(3);
   ParentNDerivative(2,I) = 0.125*YPoint*ShapePart(1)*ShapePart(3);
   ParentNDerivative(3,I) = 0.125*ZPoint*ShapePart(1)*ShapePart(2);
end
Jacobi = ParentNDerivative*ElementNode;% Calculate Jacobi matrix
JacobiDET = det(Jacobi);
JacobiINV=inv(Jacobi);
dNdx=JacobiINV*ParentNDerivative;% Calculate the derivation of shape function
end
% === Forming modified dek and dekdx for each MMB ===
function [dek,dekdx]=DEK(xk,xe,ye,ze)
x1=xk(1); y1=xk(2);z1=xk(3);x2=xk(4); y2=xk(5);  z2=xk(6);
Smin=1E-9;
lk=sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
a1=xe-x1;  a2=ye-y1;  a3=ze-z1;
b1=xe-x2;  b2=ye-y2;  b3=ze-z2;
n=length(xe); %n is the number of the elements
dek=zeros(n,1); dekdx=zeros(n,7);   num_all=[1:n]';
num1=find((a1.*(x2-x1)+a2.*(y2-y1)+a3.*(z2-z1))<=0);
dek(num1)=max(Smin,sqrt(a1(num1).^2+ a2(num1).^2+ a3(num1).^2));
dekdx(num1,1)=-a1(num1)./dek(num1);
dekdx(num1,2)=-a2(num1)./dek(num1);
dekdx(num1,3)=-a3(num1)./dek(num1);
num2=find((b1.*(x1-x2)+b2.*(y1-y2)+b3.*(z1-z2))<=0);
dek(num2)=max(Smin,sqrt(b1(num2).^2+ b2(num2).^2+ b3(num2).^2));
dekdx(num2,4)=-b1(num2)./dek(num2);
dekdx(num2,5)=-b2(num2)./dek(num2);
dekdx(num2,6)=-b3(num2)./dek(num2);
num3=setdiff(num_all,[num1;num2]);
A=b2.*a3-a2.*b3; B=a1.*b3-b1.*a3; C=b1.*a2-a1.*b2;
sqrtABC=sqrt(A.^2+B.^2+C.^2);
dek(num3)=sqrtABC(num3)./lk; fm=max(Smin,(lk^3).*sqrtABC);
dekdx(num3,1)=((-b3(num3).*B(num3)+b2(num3).*C(num3))*lk^2+(sqrtABC(num3).^2).
*(x2-x1))./fm(num3);
dekdx(num3,2)=(( b3(num3).*A(num3)-b1(num3).*C(num3))*lk^2+(sqrtABC(num3).^2).
*(y2-y1))./fm(num3);
dekdx(num3,3)=((-b2(num3).*A(num3)+b1(num3).*B(num3))*lk^2+(sqrtABC(num3).^2).
*(z2-z1))./fm(num3);
dekdx(num3,4)=(( a3(num3).*B(num3)-a2(num3).*C(num3))*lk^2-(sqrtABC(num3).^2).
*(x2-x1))./fm(num3);
dekdx(num3,5)=((-a3(num3).*A(num3)+a1(num3).*C(num3))*lk^2-(sqrtABC(num3).^2).
*(y2-y1))./fm(num3);
dekdx(num3,6)=(( a2(num3).*A(num3)-a1(num3).*B(num3))*lk^2-(sqrtABC(num3).^2).
*(z2-z1))./fm(num3);
end
```

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 12172148), and the Plan for Scientific and Technological Development of Jilin Province (Grant No. 20210101058JC).

Declarations

Competing interest The authors declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Replication of results Matlab code is listed in the Appendix, which can be download from the journal website or https://www.researchgate.net/profile/Yanfang-Zhao-2.

References

Allaire G, Jouve F, Toader AM (2004) Structural optimization using sensitivity analysis and a level-set method. J Copmut Phys 194:363–393. https://doi.org/10.1016/j.jcp.2003.09.032

Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2010) Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidisc Optim 43:1–16. https://doi.org/10.1007/s00158-010-0594-7

Ansola Loyola R, Querin OM, Garaigordobil Jiménez A, Alonso Gordoa C (2018) A sequential element rejection and admission (SERA) topology optimization code written in Matlab. Struct Multidisc Optim 58:1297–1310. https://doi.org/10.1007/s00158-018-1939-x

Bathe KJ (2006) Finite element procedures. Prentice Hall, New Jersey

Bendsøe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224. https://doi.org/10.1016/0045-7825(88)90086-2

Bruns TE, Tortorelli DA (2003) An element removal and reintroduction strategy for the topology optimization of structures and compliant mechanisms. Int J Numer Methods Eng 57:1413–1430. https://doi.org/10.1002/nme.783

Buhl T, Pedersen CBW, Sigmund O (2000) Stiffness design of geometrically nonlinear structures using topology optimization. Struct Multidisc Optim 19:93–104. https://doi.org/10.1007/s001580050089

Challis VJ (2010) A discrete level-set topology optimization code written in Matlab. Struct Multidisc Optim 41:453–464. https://doi.org/10.1007/s00158-009-0430-0

Chen Q, Zhang X, Zhu B (2018) A 213-line topology optimization code for geometrically nonlinear structures. Struct Multidisc Optim 59:1863–1879. https://doi.org/10.1007/s00158-018-2138-5

Chen Z, Long K, Wang X, Liu J, Saeed N (2021) A new geometrically nonlinear topology optimization formulation for controlling maximum displacement. Eng Optim 53:1283–1297. https://doi.org/10.1080/0305215x.2020.1781106

Deaton JD, Grandhi RV (2013) A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct Multidisc Optim 49:1–38. https://doi.org/10.1007/s00158-013-0956-z

Du Z, Cui T, Liu C, Zhang W, Guo Y, Guo X (2022) An efficient and easy-to-extend Matlab code of the moving morphable component (MMC) method for three-dimensional topology optimization. Struct Multidisc Optim 65:158. https://doi.org/10.1007/s00158-022-03239-4

Ferrari F, Sigmund O (2020) A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D. Struct Multidisc Optim 62:2211–2228. https://doi.org/10.1007/s00158-020-02629-w

Gao J, Luo Z, Xia L, Gao L (2019) Concurrent topology optimization of multiscale composite structures in Matlab. Struct Multidisc Optim 60:2621–2651. https://doi.org/10.1007/s00158-019-02323-6

Guo X, Zhang W, Zhong W (2014) Doing topology optimization explicitly and geometrically: a new moving orphable components based framework. J Appl Mech 81:081009. https://doi.org/10.1115/1.4027609

Guo G, Zhao Y, Zuo W (2022) Explicit and efficient topology optimization for three-dimensional structures considering geometrical nonlinearity. Adv Eng Softw 173:103238. https://doi.org/10.1016/j.advengsoft.2022.103238

Han Y, Xu B, Liu Y (2021) An efficient 137-line MATLAB code for geometrically nonlinear topology optimization using bi-directional evolutionary structural optimization method. Struct Multidisc Optim 63:2571–2588. https://doi.org/10.1007/s00158-020-02816-9

Hoang VN, Jang GW (2017) Topology optimization using moving morphable bars for versatile thickness control. Comput Methods Appl Mech Eng 317:153–173. https://doi.org/10.1016/j.cma.2016.12.004

Hou J, Gu X, Zhu J, Wang J, Zhang W (2020) Topology optimization of joint load control with geometrical nonlinearity. Chinese J Aeronaut 33:372–382. https://doi.org/10.1016/j.cja.2019.01.024

Huang XD, Xie YM (2010) A further review of ESO type methods for topology optimization. Struct Multidisc Optim 41:671–683. https://doi.org/10.1007/s00158-010-0487-9

Kemmler R, Lipka A, Ramm E (2005) Large deformations and stability in topology optimization. Struct Multidisc Optim 30:459–476. https://doi.org/10.1007/s00158-005-0534-0

Kim N (2015) Introduction to nonlinear finite element analysis. Springer, US. https://doi.org/10.1007/978-1-4419-1745-4

Kim C, Jung M, Yamada T, Nishiwaki S, Yoo J (2020) FreeFEM++ code for reaction-diffusion equation–based topology optimization: for high-resolution boundary representation using adaptive mesh refinement. Struct Multidisc Optim 62:439–455. https://doi.org/10.1007/s00158-020-02498-3

Lagaros ND, Vasileiou N, Kazakis G (2018) A C# code for solving 3D topology optimization problems using SAP2000. Optim Eng 20:1–35. https://doi.org/10.1007/s11081-018-9384-7

Lahuerta RD, Simoes ET, Campello EMB, Pimenta PM, Silva ECN (2013) Towards the stabilization of the low density elements in topology optimization with large deformation. Comput Mech 52:779–797. https://doi.org/10.1007/s00466-013-0843-x

Laurain A (2018) A level set-based structural optimization code using FEniCS. Struct Multidisc Optim 58:1311–1334. https://doi.org/10.1007/s00158-018-1950-2

Liang Y, Cheng G (2019) Further elaborations on topology optimization via sequential integer programming and Canonical relaxation algorithm and 128-line MATLAB code. Struct Multidisc Optim 61:411–431. https://doi.org/10.1007/s00158-019-02396-3

Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. Struct Multidisc Optim 50:1175–1196. https://doi.org/10.1007/s00158-014-1107-x

Liu J, Gaynor AT, Chen S, Kang Z, Suresh K, Takezawa A, Li L, Kato J, Tang J, Wang CCL, Cheng L, Liang X, To AC (2018) Current and future trends in topology optimization for additive manufacturing. Struct Multidisc Optim 57:2457–2483. https://doi.org/10.1007/s00158-018-1994-3

Luo Y, Wang MY, Kang Z (2015) Topology optimization of geometrically nonlinear structures based on an additive hyperelasticity technique. Comput Methods Appl Mech Eng 286:422–441. https://doi.org/10.1016/j.cma.2014.12.023

Moon SJ, Yoon GH (2013) A newly developed qp-relaxation method for element connectivity parameterization to achieve stress-based topology optimization for geometrically nonlinear structures. Comput Methods Appl Mech Eng 265:226–241. https://doi.org/10.1016/j.cma.2013.07.001

Pedersen CBW, Buhl T, Sigmund O (2001) Topology synthesis of large-displacement compliant mechanisms. Int J Numer Methods Eng 50:2683–2705. https://doi.org/10.1002/nme.148

Picelli R, Sivapuram R, Xie YM (2020) A 101-line MATLAB code for topology optimization using binary variables and integer programming. Struct Multidisc Optim 63:935–954. https://doi.org/10.1007/s00158-020-02719-9

Rozvany GIN (2010) On symmetry and non-uniqueness in exact topology optimization. Struct Multidisc Optim 43:297–317. https://doi.org/10.1007/s00158-010-0564-0

Sigmund O (2001) A 99 line topology optimization code written in Matlab. Struct Multidisc Optim 21:120–127. https://doi.org/10.1007/s001580050176

Sigmund O (2007) Morphology-based black and white filters for topology optimization. Struct Multidisc Optim 33:401–424. https://doi.org/10.1007/s00158-006-0087-x

Smith H, Norato JA (2020) A Matlab code for topology optimization using the geometry projection method. Struct Multidisc Optim 62:1579–1594. https://doi.org/10.1007/s00158-020-02552-0

Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. Int J Numer Methods Eng 24:359–373. https://doi.org/10.1002/nme.1620240207

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 Multidisc Optim 45:329–357. https://doi.org/10.1007/s00158-011-0696-x

Tavakoli R, Mohseni SM (2014) Alternating active-phase algorithm for multimaterial topology optimization problems: a 115-line MATLAB implementation. Struct Multidisc Optim 49:621–642. https://doi.org/10.1007/s00158-013-0999-1

van Dijk NP, Langelaar M, van Keulen F (2014) Element deformation scaling for robust geometrically nonlinear analyses in topology optimization. Struct Multidisc Optim 50:537–560. https://doi.org/10.1007/s00158-014-1145-4

Wang Y, Kang Z (2021) MATLAB implementations of velocity field level set method for topology optimization: an 80-line code for 2D and a 100-line code for 3D problems. Struct Multidisc Optim 64:4325–4342. https://doi.org/10.1007/s00158-021-02958-4

Wang MY, Wang X, Guo D (2003) A level set method for structural topology optimization. Comput Methods Appl Mech Eng 192:227–246. https://doi.org/10.1016/S0045-7825(02)00559-5

Wang F, Lazarov BS, Sigmund O, Jensen JS (2014) Interpolation scheme for fictitious domain techniques and topology optimization of finite strain elastic problems. Comput Methods Appl Mech Eng 276:453–472. https://doi.org/10.1016/j.cma.2014.03.021

Wang C, Zhao Z, Zhou M, Sigmund O, Zhang XS (2021) A comprehensive review of educational articles on structural and multidisciplinary optimization. Struct Multidisc Optim 64:2827–2880. https://doi.org/10.1007/s00158-021-03050-7

Wei P, Li ZY, Li XP, Wang MY (2018) An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions. Struct Multidisc Optim 58:831–849. https://doi.org/10.1007/s00158-018-1904-8

Xia L, Breitkopf P (2015) Design of materials using topology optimization and energy-based homogenization approach in Matlab. Struct Multidisc Optim 52:1229–1241. https://doi.org/10.1007/s00158-015-1294-0

Xia L, Xia Q, Huang XD, Xie YM (2018) Bi-directional evolutionary structural optimization on advanced structures and materials: a comprehensive review. Arch Comput Method Eng 25:437–478. https://doi.org/10.1007/s11831-016-9203-2

Yaghmaei M, Ghoddosian A, Khatibi MM (2020) A filter-based level set topology optimization method using a 62-line MATLAB code. Struct Multidisc Optim 62:1001–1018. https://doi.org/10.1007/s00158-020-02540-4

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 Multidisc Optim 53:1243–1260. https://doi.org/10.1007/s00158-015-1372-3

Zhao Y, Hoang VN, Jang GW, Zuo W (2021) Hollow structural topology optimization to improve manufacturability using three-dimensional moving morphable bars. Adv Eng Softw 152:102955. https://doi.org/10.1016/j.advengsoft.2020.102955

Zhao Y, Guo G, Bai J, Zuo W (2022) Hollow structural topology optimization considering geometrical nonlinearity using three-dimensional moving morphable bars. Eng Comput 38:5603–5616. https://doi.org/10.1007/s00366-022-01701-x

Zhu B, Chen Q, Wang R, Zhang X (2018) Structural topology optimization using a moving morphable component-based method considering geometrical nonlinearity. J Mech Des 140:081403. https://doi.org/10.1115/1.4040547

Zhu B, Zhang X, Zhang H, Liang J, Zang H, Li H, Wang R (2020) Design of compliant mechanisms using continuum topology optimization: a review. Mech Mach Theory 143:103622. https://doi.org/10.1016/j.mechmachtheory.2019.103622

Zhu B, Zhang X, Li H, Liang J, Wang R, Li H, Nishiwaki S (2021) An 89-line code for geometrically nonlinear topology optimization written in FreeFEM. Struct Multidisc Optim 63:1015–1027. https://doi.org/10.1007/s00158-020-02733-x

Zhu B, Wang R, Zhang H, Li H, Liang J, Zhang X, Li H, Nishiwaki S (2022) An approach for geometrically nonlinear topology optimization using moving wide-Bezier components with constrained ends. J Mech Des 144:011704. https://doi.org/10.1115/1.4051872

Zuo W, Saitou K (2017) Multi-material topology optimization using ordered SIMP interpolation. Struct Multidisc Optim 55:477–491. https://doi.org/10.1007/s00158-016-1513-3

Zuo ZH, Xie YM (2015) A simple and compact Python code for complex 3D topology optimization. Adv Eng Softw 85:1–11. https://doi.org/10.1016/j.advengsoft.2015.02.006