Structural and Multidisciplinary Optimization
https://doi.org/10.1007/s00158-020-02816-9

EDUCATIONAL PAPER

An efficient 137-line MATLAB code for geometrically nonlinear topology optimization using bi-directional evolutionary structural optimization method

Yongsheng Han[1] · Bin Xu[1] · Yuanhao Liu[1]

Received: 21 October 2020 / Revised: 23 November 2020 / Accepted: 4 December 2020

Abstract
Topology optimization, as a powerful conceptual design method, has been widely adopted in both academic research and industrial applications. To further promote the development of topology optimization, many computer programs have been published for educational purposes over the past decades. However, most of the computer programs are constructed based on a linear assumption. On the basis of bi-directional evolutionary structural optimization (BESO) method, the paper presents a MATLAB implementation of the geometrically nonlinear topology optimization code for compliance minimization of statically loaded structures. Excluding 19 lines which are used for explanation, only 118 lines are needed for the initialization of the design parameters, nonlinear finite element analysis, sensitivity calculation, sensitivity filtration, and topological design variables update. Different design problems can be solved by modifying several lines in the proposed program. The complete 137-line code is included as an Appendix and is intended for educational purposes only.

Keywords Topology optimization · MATLAB code · Geometrically nonlinear · BESO method · Education

1 Introduction

Topology optimization, as a powerful structural design method, aims to find an optimized structural topology or material layout within a given design domain for specified objectives, constraints, and boundary conditions (Han et al. 2019). Since the seminal paper (Bendsøe and Kikuchi 1988), topology optimization has attracted wide academic (Bendsøe and Sigmund 2003; Deaton and Grandhi 2014; Huang and Xie 2010a, 2010b) and industrial (Zhu et al. 2016) interests due to its huge potential in engineering applications and its intrinsic mathematical challenges. Various approaches have been presented in the literature, such as homogenization method (Bendsøe and Kikuchi 1988; Suzuki and Kikuchi 1991), solid isotropic material with penalization (SIMP) method (Bendsøe 1989; Zhou and Rozvany 1991; Rozvany 2001), level set method (Sethian and Wiegmann 2000; Wang et al. 2003), moving morphable components (MMC) method (Guo et al. 2016; Wang et al. 2019), evolutionary structural optimization (ESO) method (Xie and Steven 1993, 1994; Rong et al. 2001) and its improved version bi-directional ESO method (Huang and Xie 2007, 2009; Huang and Xie 2010a, 2010b) which not only allows inefficient materials to be removed but also adds efficient materials simultaneously, and the more recent iso-geometric analysis (IGA) method (Qian 2013). For a critical and comprehensive review of established topology optimization methods and their applications, the readers may refer to some papers (Sigmund and Maute 2013a, 2013b; Deaton and Grandhi 2014; Zhu et al. 2020a).

With the development of topology optimization methods, computational programs concerning structural topology optimization have been gradually published in the literature. These programs, which are constructed for

Responsible Editor: Emilio Carlos Nelli Silva

Highlights
- A 137-line MATLAB code for topological optimization of geometrically nonlinear structure is constructed.
- The presented code is easier to implement and understand.
- The iterative curves converge to constant values stably, and the convergence rate is fast.

✉ Bin Xu
xubin@nwpu.edu.cn

[1] Institute of Structural Health Monitoring and Control, School of Mechanics, Civil Engineering & Architecture, Northwestern Polytechnical University, Xi'an 710072, China

Published online: 15 February 2021

educational purposes, have paved the way for the flourished development of topology optimization. Meanwhile, these codes have been helpful for newcomers to understand the underlying idea of structural topology optimization (Zhu et al. 2020b). Although a detailed collection of such programs can be found in the review paper (Zhu et al. 2020a) and the educational paper (Zhu et al. 2020b), some of these programs are also introduced in Table 1. In the table, CDT represents the canonical dual theory, which is adopted by Liang and Cheng (2020). Their method is based on the CDT and integer variables. TOBS denotes topology optimization of binary structures, which is adopted by Picelli et al. (2020). Their method is based on the binary variables and integer programming. LSM stands for level set method, which is one of the most widely utilized approaches in topology optimization methods. Readers who are interested in this approach can refer to `dLSM`, `FEniCS`, `108-line`, or `62-line` to quickly become familiar with the classical level set methods, which normally obtain the optimized topologies by solving the Hamilton-Jacobi equation. In addition, one can use the `88-line` level set codes, which are programmed based on the parameterization level set methods (Luo et al. 2007; Wei et al. 2018).

Several computer programs have been proposed based on the SIMP methods for intermediate densities, which is one of the most widely utilized approaches in density-based topology optimization methods. Readers can refer to the `99-line` code and its improved versions, such as the `88-line` program (Andreassen et al. 2011). Computer programs that focus on using this approach to three-dimensional topology optimization have also been published, such as `169-line 3D` and `AC# 3D` codes. Zhu et al. (2020b) presented a code for geometrically nonlinear topology optimization written in FreeFEM. A new generation 99-line MATLAB code (`top99neo`) for compliance topology optimization and its extension to 3D (`top3D125`) is constructed by Ferrari and Sigmund (2020). Meanwhile, there have been several available codes published by using the ESO and BESO methods. These codes can be used to solve either two-dimensional (`soft-kill` and `SERA`) or three-dimensional (`PYTHON 3D`) structural topology optimization problems. Furthermore, Talischi et al. (2012) presented a `PolyTop` code that can be employed for topology optimization using unstructured polygonal finite element meshes in arbitrary domains. Xia and Breitkopf 2015 presented a MATLAB code `topX` for the optimized topology design of materials with extreme

Table 1 Educational computer programs for structural topology optimization

Names and authors	Environment	Method	Assumption
`99-line` (Sigmund 2001)	MATLAB	SIMP	Linear
`SOFT-KILL` (Huang and Xie 2010a, 2010b)	MATLAB	BESO	Linear
`dLSM` (Challis 2010)	MATLAB	LSM	Linear
`88-line` (Andreassen et al. 2011)	MATLAB	SIMP	Linear
`PolyTop` (Talischi et al. 2012)	MATLAB	SIMP	Linear
`169-line 3D` (Liu and Tovar 2014)	MATLAB	SIMP	Linear
`115-line` (Tavakoli and Mohseni 2014)	MATLAB	SIMP	Linear
`PYTHON 3D` (Zuo and Xie 2015)	Python, Abaqus	ESO	Linear
`topX` (Xia and Breitkopf 2015)	MATLAB	BESO	Linear
`MMC188` (Zhang et al. 2016)	MATLAB	MMC	Linear
`SERA` (Loyola et al. 2018)	MATLAB	ESO	Linear
`FEniCS` (Laurain 2018)	FEniCS	LSM	Linear
`88-line` (Wei et al. 2018)	MATLAB	LSM	Linear
`esoL esoX` (Xia et al. 2018)	MATLAB	BESO	Linear
`185-line` (Laurain 2018)	FEniCS	LSM	Linear
`213-line` (Chen et al. 2019)	*MATLAB, ANSYS*	*SIMP*	*Nonlinear*
`AC# 3D` (Lagaros et al. 2019)	SAP2000	SIMP	Linear
`128-line` (Liang and Cheng 2020)	MATLAB	CDT	Linear
`108-line` (Kim et al. 2020)	FreeFEM	LSM	Linear
`62-line` (Yaghmaei et al. 2020)	MATLAB	LSM	Linear
`89-line` (Zhu et al. 2020b)	*FreeFEM*	*SIMP*	*Nonlinear*
`top99neo` (Ferrari and Sigmund 2020)	MATLAB	SIMP	Linear
`top3D125` (Ferrari and Sigmund 2020)	MATLAB	SIMP	Linear
`tobs101` (Picelli et al. 2020)	MATLAB	TOBS	Linear

Entries in italic indicates nonlinear work

properties, such as bulk modulus, shear modulus, and negative Poisson's ratio. In a comprehensive review of the BESO method on advanced structures and materials, MATLAB codes `esoL` and `esoX` can be referred (Xia et al. 2018).

In mathematical optimization methods such as SIMP, the material density is penalized so that intermediate material densities invaluably appear in the topologies. As the topology develops, large displacements may cause the tangent stiffness matrix in low-density elements to become indefinite or even negative definite. Then some additional scheme must be devised to circumvent the problem such as removing low-density elements or relaxing the convergence criterion. It seems that BESO method with discrete formulation has the potential to overcome these problems.

To our knowledge, and from Table 1, one can easily observe that most of the current published codes are constructed by using the linear assumption. In regard to nonlinear topology optimization, one can hardly find convenient and compact published code. However, the structures in engineering practice are usually nonlinear structures. In pursuing more realistic designs, nonlinear structural topology optimization has been recognized as a challenging problem and has been continuously attracting research interests (Xu et al. 2020). Few educational papers (Chen et al. 2019; Zhu et al. 2020b) have been published dealing with compliance-based topology optimization of nonlinear structures, compared to the linear structures. The code (Chen et al. 2019) is built based on the commercial software MATLAB and ANSYS. The data transfer between these two platforms can cause several unexpected issues, such as special attention is needed to obtain the sensitivity information (Chen et al. 2019). Notably, in this paper, we present a BESO-based MATLAB code for geometrical nonlinear topology optimization. It only contains 137 lines in which 19 lines are comments. The remainder of the paper is organized as follows. In Section 2, the theoretical background of topology optimization considering geometrical nonlinearity, and the sensitivity numbers for the geometrically nonlinear topology optimization are introduced. Section 3 briefly states the BESO procedure. In Section 4, a detailed explanation of the presented code is provided. In Section 5, some examples are presented to show how to extend the proposed code to design other problems. Finally, meaningful conclusions are obtained in Section 6.

2 Problem formulation

2.1 Optimization problem

With regard to the geometrically nonlinear topology optimization problem, the mathematical model for the minimization of the compliance subject to material usage constraint can be described as follows:

$$\begin{aligned} &\text{Find}: \mathbf{x} = (x_1, x_2, \cdots, x_i, \cdots, x_{nel})^T \\ &\text{Minimize}: c(\boldsymbol{x}) = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} = \frac{1}{2}\sum_{i=1}^{nel} x_i^p \mathbf{u}_i^T \mathbf{k}_i \mathbf{u}_i = \frac{1}{2}(\mathbf{f}^{ext})^T\mathbf{u} \\ &\text{Subject to}: \mathbf{R}(\mathbf{u}) = \mathbf{f}^{\text{int}} - \mathbf{f}^{ext} = \mathbf{0} \\ &V(\mathbf{x}) = \sum_{i=1}^{nel} x_i v_i = V_{req} \\ &x_i \in \{x_{min}, 1\}, i = 1, 2, \cdots, nel \end{aligned} \tag{1}$$

where c and $\mathbf{x}$ denote the compliance and the design variable vector, respectively. $\mathbf{u}$ and $\mathbf{K}$ are the global displacement vector and the global stiffness matrix, respectively. $\mathbf{u}_i$ and $\mathbf{k}_i$ are the element displacement vector and the element stiffness matrix, respectively. $\mathbf{R}$ can be defined as the residual of the equilibrium equation. $\mathbf{f}^{int}$ represents the internal force vector, and $\mathbf{f}^{ext}$ denotes the external force vector. $x_{\min}$, nel, and p denote a minimum relative densities (non-zero to avoid singularity), the number of elements used to discretize the design domain, and the penalization power, respectively. v_i is the volume of the ith element; $V(\mathbf{x})$ and V_{req} are the total and required material volumes, respectively. The only difference between the optimization model and a standard linear topology optimization problem is that the equilibrium condition $\mathbf{R}(\mathbf{u}) = \mathbf{0}$ must be found using an iterative scheme to obtain the displacement field $\mathbf{u}$.

2.2 Geometric nonlinear finite element analysis

The geometric nonlinear analysis has been summarized in detail by Xu and Zhu (Xu et al. 2020; Zhu et al. 2020b). This is briefly stated in this subsection.

2.2.1 Displacement-strain conversion matrix

Element design variables and quadrangle elements are adopted to discretize the design domain. Lagrangian coordinate function $\mathbf{u}$ can be interpolated by nodal displacement as:

$$\mathbf{X} = \mathbf{N}\mathbf{X}^e \quad \mathbf{u} = \mathbf{N}\mathbf{u}_e \tag{2}$$

where $\mathbf{X} = \{X_1 \quad X_2 \quad X_3\}^T$ and $\mathbf{X}^e = \{X_1^T, X_2^T, \cdots, X_n^T\}^T$ represent the coordinate vector of any material point and the coordinate vector of elemental node, respectively. n denotes the number of element nodes. $\mathbf{N}$ and $\mathbf{u}_e$ denote the shape function and elemental node displacement vector, respectively.

In the case of large deformation, the Green strain $\mathbf{E}$ (Crisfield 1991) can be written as linear and nonlinear parts as follows:

$$\mathbf{E} = \mathbf{E}_L + \mathbf{E}_N \tag{3}$$

where

$$\mathbf{E}=\{E_{11}\quad E_{22}\quad E_{33}\quad 2E_{23}\quad 2E_{31}\quad 2E_{12}\}^T$$
$$\mathbf{E}_L=\{E_{11}^L\quad E_{22}^L\quad E_{33}^L\quad 2E_{23}^L\quad 2E_{31}^L\quad 2E_{12}^L\}^T$$
$$\mathbf{E}_N=\{E_{11}^N\quad E_{22}^N\quad E_{33}^N\quad 2E_{23}^N\quad 2E_{31}^N\quad 2E_{12}^N\}^T$$

The relationship between Green strain and displacement can be obtained as:

$$E_{ij}^L=\frac{1}{2}\left(\frac{\partial u_i}{\partial X_j}+\frac{\partial u_j}{\partial X_i}\right)\quad E_{ij}^N=\frac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial X_i}\right)^T\frac{\partial \mathbf{u}}{\partial X_j} \tag{4}$$

Substituting (2)–(4), one can obtain:

$$\mathbf{E}_L=\mathbf{L}\mathbf{N}\mathbf{u}_e=\mathbf{B}_{L0}\mathbf{u}_e \tag{5}$$

$$\mathbf{E}_N=\frac{1}{2}\mathbf{A}\cdot\mathbf{G}\cdot\mathbf{u}_e=\widetilde{\mathbf{B}}_{N0}\mathbf{u}_e \tag{6}$$

where

$$\mathbf{B}_{L0}=[\mathbf{B}_{L01},\mathbf{B}_{L02},\cdots,\mathbf{B}_{L0k},\cdots,\mathbf{B}_{L0n}]$$
$$\mathbf{G}=[\mathbf{G}_1,\mathbf{G}_2,\cdots,\mathbf{G}_k,\cdots,\mathbf{G}_n]$$
$$\mathbf{A}^T=\begin{bmatrix}\frac{\partial \mathbf{u}}{\partial X_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{u}}{\partial X_3} & \frac{\partial \mathbf{u}}{\partial X_2}\\ \mathbf{0} & \frac{\partial \mathbf{u}}{\partial X_2} & \mathbf{0} & \frac{\partial \mathbf{u}}{\partial X_3} & \mathbf{0} & \frac{\partial \mathbf{u}}{\partial X_1}\\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{u}}{\partial X_3} & \frac{\partial \mathbf{u}}{\partial X_2} & \frac{\partial \mathbf{u}}{\partial X_1} & \mathbf{0}\end{bmatrix}$$
$$\mathbf{B}_{L0k}=\begin{bmatrix}\frac{\partial N_k}{\partial X_1} & 0 & 0 & 0 & \frac{\partial N_k}{\partial X_3} & \frac{\partial \mathrm{N}_k}{\partial X_2}\\ 0 & \frac{\partial N_k}{\partial X_2} & 0 & \frac{\partial N_k}{\partial X_3} & 0 & \frac{\partial N_k}{\partial X_1}\\ 0 & 0 & \frac{\partial N_k}{\partial X_3} & \frac{\partial N_k}{\partial X_2} & \frac{\partial N_k}{\partial X_1} & 0\end{bmatrix}$$
$$\mathbf{G}_k=\left[\frac{\partial N_k}{\partial X_1}\mathbf{I}_{3\times 3}\frac{\partial N_k}{\partial X_2}\mathbf{I}_{3\times 3}\frac{\partial N_k}{\partial X_3}\mathbf{I}_{3\times 3}\right]^T$$

where $\mathbf{L}$ is a differential operator. $\mathbf{B}_{L0}$ and $\widetilde{\mathbf{B}}_{N0}$ denote the linear and nonlinear transformation matrixes between node displacement and element strain, respectively. $\mathbf{A}$ and $\mathbf{G}$ denote the derivative matrix of nodal displacement with respect to coordinates and the derivative matrix of the shape function with respect to coordinates, respectively.

After substituting (5) and (6) to (3), the Green strain can be further written as:

$$\mathbf{E}=\widetilde{\mathbf{B}}_0\mathbf{u}_e \tag{7}$$

$$\widetilde{\mathbf{B}}_0=\mathbf{B}_{L0}+1/2\widetilde{\mathbf{B}}_{N0}=\mathbf{B}_{L0}+1/2\mathbf{A}\mathbf{G} \tag{8}$$

where $\widetilde{\mathbf{B}}_0$ denotes the transformation matrix between node displacement $\mathbf{u}_e$ and element strain $\mathbf{E}$.

Because $\mathbf{A}$ contains the derivative of the displacement, the relationship between Green strain vector $\mathbf{E}$ and the nodal displacement vector $\mathbf{d}_e$ is nonlinear. From (7) and (8), the relationship between the increment of Green strain $\delta\mathbf{E}$ and the increment of the nodal displacement $\delta\mathbf{u}_e$ is as follows:

$$\begin{aligned}\delta\mathbf{E}&=\delta\mathbf{E}_L+\delta\mathbf{E}_N=\mathbf{B}_{L0}\delta\mathbf{u}_e+\frac{1}{2}(\delta\mathbf{A})\mathbf{G}\mathbf{u}_e+\frac{1}{2}\mathbf{A}\mathbf{G}\delta\mathbf{u}_e\\&=\mathbf{B}_{L0}\delta\mathbf{u}_e+\mathbf{A}\mathbf{G}\delta\mathbf{u}_e\end{aligned} \tag{9}$$

Equation (9) can be rewritten as:

$$\delta\mathbf{E}=\mathbf{B}_0\delta\mathbf{u}_e \tag{10}$$

$$\mathbf{B}_0=\mathbf{B}_{L0}+\mathbf{A}\cdot\mathbf{G}=\mathbf{B}_{L0}+\widetilde{\mathbf{B}}_{N0} \tag{11}$$

where $\mathbf{B}_0$ is defined as the transformation matrix between $\delta\mathbf{E}$ and $\delta\mathbf{u}_e$.

2.2.2 Residual of the equilibrium equation

The second type of Piola-Kirchhoff stress vector is denoted as $\mathbf{S}$. According to Ref. Xu et al. (2020)), the equilibrium equation of the design domain can be expressed as:

$$\mathbf{R}(\mathbf{u})=\int_{V_0}\mathbf{B}_0{}^T\mathbf{S}dV-\mathbf{f}^{ext}=\mathbf{f}^{\mathrm{int}}-\mathbf{f}^{ext}=\mathbf{0} \tag{12}$$

where $\mathbf{f}^{int}$ represents the internal force vector, and $\mathbf{f}^{ext}$ denotes the external force vector, i.e.,

$$\mathbf{f}^{ext}=\int_{V_0}\mathbf{N}^T\mathbf{p}_0dV+\int_{A_{0t}}\mathbf{N}^T\mathbf{q}_0dA \tag{13}$$

The physical force and surface load are denoted by $\mathbf{p}_0$ and $\mathbf{q}_0$, respectively.

2.2.3 Tangential stiffness matrix

According to the definition of tangential stiffness matrix and assuming that external load is independent of nodal displacement, then one can obtain:

$$\delta\mathbf{R}=\int_{V_0}\left(\mathbf{B}_0{}^T\delta\mathbf{S}\right)dV+\int_{V_0}\delta\left(\mathbf{B}_0{}^T\right)\mathbf{S}dV=\mathbf{K}_T\delta\mathbf{u} \tag{14}$$

where $\mathbf{K}_T$ represents the tangential stiffness matrix.

The first integral of (14) can be written as:

$$\int_{V_0}\left(\mathbf{B}_0{}^T\delta\mathbf{S}\right)dV=\int_{V_0}\mathbf{B}_0{}^T\mathbf{D}\mathbf{B}_0dV\delta\mathbf{u}=\mathbf{K}_M\delta\mathbf{u} \tag{15}$$

where $\mathbf{K}_M$ is the tangential stiffness matrix related to the constitutive tensor, which can be expressed as:

$$\mathbf{K}_M=\mathbf{K}_L+\mathbf{K}_N \tag{16}$$

where

$$\mathbf{K}_L=\int_{V_0}\mathbf{B}_{L0}{}^T\mathbf{D}\mathbf{B}_{L0}dV \tag{17}$$

$$\begin{aligned}\mathbf{K}_N=&\int_{V_0}\mathbf{B}_{L0}{}^T\mathbf{D}\widetilde{\mathbf{B}}_{L0}dV+\int_{V_0}\widetilde{\mathbf{B}}_{L0}{}^T\mathbf{D}\mathbf{B}_{L0}dV\\&+\int_{V_0}\widetilde{\mathbf{B}}_{L0}{}^T\mathbf{D}\widetilde{\mathbf{B}}_{L0}dV\end{aligned} \tag{18}$$

where $\mathbf{K}_L$ is the linear stiffness matrix under the assumption of

small displacement, and $\mathbf{K}_N$ is caused by large displacement.

The second integral of (14) can be written as:

$$\int_{V_0}\delta\left(\mathbf{B}_0{}^T\right)\mathbf{S}dV=\int_{V_0}\mathbf{G}^T\delta\mathbf{A}^T\mathbf{S}dV \tag{19}$$

Based on matrix transformation, (19) can be expressed as:

$$\left(\int_{V_0}\mathbf{G}^T\mathbf{M}\mathbf{G}dV\right)\delta\mathbf{d}=\mathbf{K}_S\delta\mathbf{u} \tag{20}$$

where $\mathbf{K}_S$ is the tangential stiffness matrix caused by the stress state. And the matrix $\mathbf{M}$ is composed of the components of the stress vector $\mathbf{S}$.

Then, the tangential stiffness matrix of the system can be written as:

$$\mathbf{K}_T=\mathbf{K}_L+\mathbf{K}_N+\mathbf{K}_S \tag{21}$$

2.3 Sensitivity analysis

The objective function in (1) can be further written as:

$$c(\mathbf{x})=\lim_{n\to+\infty}\left[\frac{1}{2}\sum_{h=1}^{n}\Delta\left(\mathbf{f}^{ext}\right)^T\left(\mathbf{u}_h+\mathbf{u}_{h-1}\right)\right] \tag{22}$$

where h is the incremental number of the load vector and n is the total number of load increments.

The sensitivity number can be easily derived using the adjoint method. The sensitivity of the complementary work with respect to the design variable x_i is

$$\frac{\partial c(\mathbf{x})}{\partial x_i}=\lim_{n\to+\infty}\left[\frac{1}{2}\sum_{h=1}^{n}\left(\left(\mathbf{f}_h^{ext}\right)^T-\left(\mathbf{f}_{h-1}^{ext}\right)^T\right)\left(\frac{\partial\mathbf{u}_h}{\partial x_i}+\frac{\partial\mathbf{u}_{h-1}}{\partial x_i}\right)\right] \tag{23}$$

An adjoint equation is introduced by adding a series of vectors of Lagrangian multipliers ξ_h into the objective function as:

$$c(\mathbf{x})=\lim_{n\to+\infty}\frac{1}{2}\sum_{h=1}^{n}\left[\left(\left(\mathbf{f}_h^{ext}\right)^T-\left(\mathbf{f}_{h-1}^{ext}\right)^T\right)\left(\mathbf{u}_h+\mathbf{u}_{h-1}\right)+\xi_h^T\left(\mathbf{R}_h+\mathbf{R}_{h-1}\right)\right] \tag{24}$$

where $\mathbf{R}_h$ and $\mathbf{R}_{h-1}$ are the residual forces in load increment steps h and $h-1$.

Thus,

$$\mathbf{R}_h+\mathbf{R}_{h-1}=\mathbf{f}_h^{ext}-\mathbf{f}_h^{\text{int}}+\mathbf{f}_{h-1}^{ext}-\mathbf{f}_{h-1}^{\text{int}}=\mathbf{0} \tag{25}$$

The sensitivity of the modified objective function is:

$$\begin{aligned}\frac{\partial c(\mathbf{x})}{\partial x_i}=\lim_{n\to+\infty}\frac{1}{2}\sum_{h=1}^{n}\Bigg\{&\left(\left(\mathbf{f}_h^{ext}\right)^T-\left(\mathbf{f}_{h-1}^{ext}\right)^T\right)\left(\frac{\partial\mathbf{u}_h}{\partial x_i}+\frac{\partial\mathbf{u}_{h-1}}{\partial x_i}\right)\\&+\xi_h^T\left(\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h}\frac{\partial\mathbf{u}_h}{\partial x_i}+\frac{\partial\mathbf{R}_{h-1}}{\partial\mathbf{u}_{h-1}}\frac{\partial\mathbf{u}_{h-1}}{\partial x_i}+\frac{\partial\left(\mathbf{R}_h+\mathbf{R}_{h-1}\right)}{\partial x_i}\right)\end{aligned} \tag{26}$$

Note that the derivative of ξ_h is not included in (26) because it would be multiplied by $(\mathbf{R}_h+\mathbf{R}_{h-1})$ which is zero. It is assumed that there is a linear force-displacement relationship in a small load increment. Therefore,

$$\frac{\partial\mathbf{R}_h}{\partial\mathbf{u}_h}=\frac{\partial\mathbf{R}_{h-1}}{\partial\mathbf{u}_{h-1}}=-\mathbf{K}_h^t \tag{27}$$

where $\mathbf{K}_h^t$ is the tangential stiffness matrix in the hth step.

By substituting the relationship (27) into (26), the sensitivity of the modified objective function can be rewritten as:

$$\frac{\partial c(\mathbf{x})}{\partial x_i}=\lim_{n\to+\infty}\frac{1}{2}\sum_{h=1}^{n}\left[\left(\left(\mathbf{f}_h^{ext}\right)^T-\left(\mathbf{f}_{h-1}^{ext}\right)^T-\xi_h^T\mathbf{K}_h^t\right)\left(\frac{\partial\mathbf{u}_h}{\partial x_i}+\frac{\partial\mathbf{u}_{h-1}}{\partial x_i}\right)+\xi_h^T\frac{\partial\left(\mathbf{R}_h+\mathbf{R}_{h-1}\right)}{\partial x_i}\right] \tag{28}$$

In order to eliminate the unknowns $\partial\mathbf{u}_h/\partial x_i+\partial\mathbf{u}_{h-1}/\partial x_i$, ξ_h can be chosen as:

$$\mathbf{K}_h^t\xi_h=\mathbf{f}_h^{ext}-\mathbf{f}_{h-1}^{ext} \tag{29}$$

This equation defines the adjoint system. From the assumption of linear force-displacement relationship in a small increment, the increment of the force can be approximately expressed by:

$$\mathbf{K}_h^t\left(\mathbf{u}_h-\mathbf{u}_{h-1}\right)=\mathbf{f}_h^{ext}-\mathbf{f}_{h-1}^{ext} \tag{30}$$

Comparing (29) with (30), ξ_h can be obtained as:

$$\xi_h=\mathbf{u}_h-\mathbf{u}_{h-1} \tag{31}$$

By substituting ξ_h into (28) and utilizing (25), the sensitivity of the objective function can be expressed as:

$$\frac{\partial c(\mathbf{x})}{\partial x_i}=-\lim_{n\to+\infty}\frac{1}{2}\sum_{h=1}^{n}\left(\mathbf{u}_h^T-\mathbf{u}_{h-1}^T\right)\left(\frac{\partial\mathbf{f}_h^{\text{int}}}{\partial x_i}+\frac{\partial\mathbf{f}_{h-1}^{\text{int}}}{\partial x_i}\right) \tag{32}$$

In the evolutionary structural optimization method, a structure can be optimized by removing and adding elements. That is to say that, the element itself is treated as the design variable. Thus, when one element is totally removed from the system, the variation of the objective function is:

$$\Delta c(\mathbf{x})=-\lim_{n\to+\infty}\left[\frac{1}{2}\sum_{h=1}^{n}\left(\mathbf{u}_h^T-\mathbf{u}_{h-1}^T\right)\left(\Delta\mathbf{f}_h^{\text{int}}+\Delta\mathbf{f}_{h-1}^{\text{int}}\right)\right] \tag{33}$$

From (12), the variation of internal force due to removing one element can be written as:

$$\Delta\mathbf{f}_h^{\text{int}}=\mathbf{c}^T\mathbf{f}_h^e \tag{34}$$

Substituting (34) into (33), the variation of the objective function can be expressed as:

$$\begin{aligned}\frac{\partial c(\mathbf{x})}{\partial x_i}\approx\Delta c(\mathbf{x})&=-\lim_{n\to+\infty}\left[\frac{1}{2}\sum_{h=1}^{n}\left(\mathbf{u}_h^T-\mathbf{u}_{h-1}^T\right)\left(\mathbf{C}^{eT}\mathbf{f}_h^e+\mathbf{C}^{eT}\mathbf{f}_{h-1}^e\right)\right]\\&=-\lim_{n\to+\infty}\sum_{h=1}^{n}\left(E_h^e-E_{h-1}^e\right)=-E_n^e\end{aligned} \tag{35}$$

where E_n^e is the total strain energy of the removed element. Equation (35) means that the decrease of the total external

work due to removing one element is equal to the total strain energy of the element in its final deformed state and irrelevant to the size of displacement intervals.

The sensitivity of the end compliance with respect to the design variable can be expressed as (Huang and Xie 2008; Xu et al. 2020):

$$\alpha_i = E_n^i = \frac{1}{2}\mathbf{u}_i^T \mathbf{f}_i^{\text{int}} \tag{36}$$

where E_n^i is the total strain energy of the removed element. $\mathbf{f}_i^{\text{int}}$ represents the internal force vector of the ith element.

3 BESO procedure and numerical techniques

The update scheme of design variables in the framework of BESO method has been proposed by Huang and Xie (2007) and summarized in detail by Xu et al. (2020). This is briefly stated in this subsection.

The target volume $V^{(l)}$ at the current topological iteration (lth) is calculated by:

$$V^{(l)} = \max\left\{V_{req}, (1-c_{er})V^{(l-1)}\right\} \tag{37}$$

where the evolutionary ratio c_{er} determines the percentage of material to be removed from the design of the previous iteration. Once the final required material volume usage V_{req} is reached, the optimization algorithm alters only the topology but keeps the material volume constant.

Checkerboard and mesh-dependency problems are typical numerical issues in common topology optimization approaches. Mesh-independence filter scheme (Huang and Xie 2009) is employed to smooth the element sensitivities using a low-pass filter of radius. The filtering process contains two steps: First, the raw element sensitivity is equally distributed to its nodes as the nodal sensitivity that does not hold a physical meaning; then, the nodal sensitivities are converted back to nearby elements using a weighting function w based on the distance between the element center and the jth node. The modified element sensitivity is calculated in the following:

$$\alpha_i = \frac{\sum_{j=1}^{N} w_{ij}^{sen}\alpha_j}{\sum_{j=1}^{N} w_{ij}^{sen}} \tag{38}$$

where w_{ij}^{sen} is a linear weight factor

$$w_{ij}^{sen} = \max\{0, r_{sen} - \Delta(i,j)\} \tag{39}$$

determined according to the prescribed filter radius r_{sen} and the element center-to-center distance $\Delta(i,j)$ between the ith and the jth elements.

Due to the discrete nature of the BESO material model, the current sensitivity numbers are averaged with their historical information to improve the convergence:

$$\alpha_i^{(l)} \leftarrow \frac{\alpha_i^{(l)} + \alpha_i^{(l-1)}}{2}, \quad \text{for} \quad l > 1 \tag{40}$$

where the superscript denotes the number of design iteration.

All elements are sorted into a vector according to their sensitivity number values. Then, the update of the topological design variables is realized by means of two threshold parameters α_{del}^{th} and α_{add}^{th} for material removal and addition, respectively (Huang and Xie 2007):

$$x_i^{(l+1)} = \begin{cases} 0 & \alpha_i \le \alpha_{del}^{th}, x_i^{(l)} = 1 \\ 1 & \alpha_{add}^{th} < \alpha_i, x_i^{(l)} = 0 \\ x_i^{(l)} & otherwise \end{cases} \tag{41}$$

The present scheme indicates that solid elements are removed when their sensitivity numbers are less than α_{del}^{th} and void elements are recovered when their sensitivity numbers are greater than α_{add}^{th}.

The optimization procedure is terminated if the convergence criterion is satisfied, which can be expressed as:

$$error = \frac{\left|\sum_{i=1}^{N_{itr}} (c_{k-i+1} - c_{k-N-i+1})\right|}{\sum_{i=1}^{N_{itr}} c_{k-i+1}} \le \tau \tag{42}$$

where c represents the value of objective function. τ and N_{itr} denote the allowable convergence factor and an integral number which are usually set to be 0.001 and 5, respectively.

4 MATLAB implementation

The MATLAB code (see Appendix 1) is built up as a standard topology optimization code. The main program is called from the MATLAB prompt by the line:

```
TOP_Geo_Non(nelx,nely,volfrac,er,rmin)
```

where `nelx` and `nely` are the numbers of elements in the horizontal and vertical directions, respectively; `volfrac` is the volume fraction; `er` is the evolutionary ratio; and `rmin` is the filter size (divided by element size). Other variables as well as boundary conditions are defined in the MATLAB code itself and can be edited if needed.

The MATLAB code can be constructed with the aforementioned formulations (see Appendix 1) by solving a standard cantilever beam design problem. The design domain (L = 4H) of the cantilever beam is illustrated in Fig. 1. The left boundary of the design domain is fixed, and the vertical load is applied at the center of the right side. Figure 2 shows the resulting density distribution obtained by the code given in Appendix 1 called with the input line:

```
TOP_Geo_Non(200,50,0.5,0.02,3)
```

It can be clearly seen that the nonlinear topology becomes asymmetric with a sufficiently large input force. The external load has a certain effect on the internal force of the geometrically nonlinear structure. The force-displacement curve of geometrically nonlinear structures is no longer a straight line. Geometrically nonlinear structures produce large displacement (large deformation). Unlike linear structures, the element stiffness matrix of geometrically nonlinear structures is no longer constant. The element stiffness matrix under different stress states is different, so the optimized geometric nonlinear topological structure is asymmetric. It can be seen that the optimized topological structure converges after 56 iterations. Compared with the work by Zhu et al. (2020b), which shows that the final iteration step is 100, the presented MATLAB code has better and fast convergence. A detailed explanation of the MATLAB code is presented in the following subsections.

4.1 Lines 1–37: main program

The parameter initialization is implemented in lines 2–5. `L` and `H` represent the dimensions of design domain. `dx` and `dy` represent the dimensions of discrete elements. To make the element size to be unit length, the size of the design domain is set to the number of elements in the horizontal and vertical directions. `a` represents half of the element size and is used to obtain the linear strain matrix (line 78). The Young modulus `E` and the Poisson ratio `nu` are set to be 3×10^9 and 0.4, respectively. `xmin` is equal to 1×10^{-3}. The main program starts by distributing the solid material in the design domain (line 4). After some other initializations, the main loop starts with the determination of the target volume `vol` at the current iteration. Then, carry out the nonlinear finite element subroutine (line 11) which returns the displacement vector `U`, sensitivity numbers `dc`, and element stiffness matrix `KE_matrix`. Different from the linear problem, the element stiffness matrix is no longer consistent for the geometrically nonlinear problem. Sensitivity analysis is also included in the subroutine. Following this, a loop over all elements (lines 12–24) determines objective function. The objective function is followed by a call to the sensitivities filter (line 26) and sensitivities average (line 28). The topological variables are updated according to the BESO optimizer (line 30). The convergence factor is calculated in line 32. The current compliance and other parameters are printed by line 34, and the resulting density distribution is plotted (line 36). The main loop is terminated if the convergence factor (`change` determined in line 32) is less than 0.001. Otherwise, the above steps are repeated.

4.2 Lines 38–49: topological variables update

The topological variables update schema of the BESO method has been proposed by the paper of Huang and Xie (2007)) and widely adopted. The updated design variables are found by the BESO optimizer (lines 38–49). The value of the threshold parameters that satisfies the volume constraint can be found by a bi-sectioning algorithm (lines 41–49). The bi-sectioning algorithm is initialized by a lower `l1` and an upper `l2` bound for the threshold parameter (line 40). The interval which bounds the threshold parameter is repeatedly halved until its size is less than the convergence criteria (line 41).

4.3 Lines 50–65: mesh-independency filtering

Mesh-independency filter has been proposed by Sigmund (2001)) and widely adopted. Lines 50–65 represent the MATLAB implementation of the mesh-independency filter.

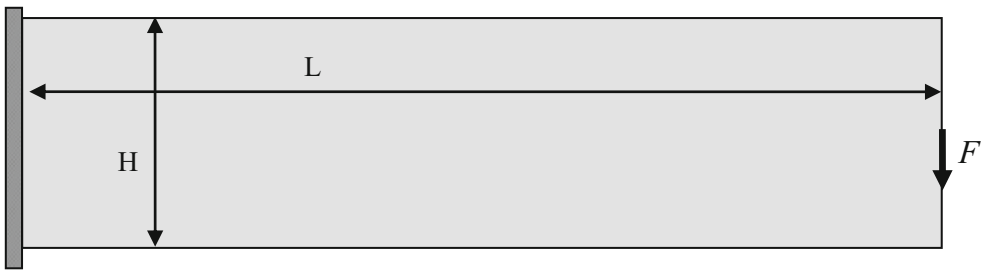

Fig. 1 Design domain, boundary conditions, and load of the cantilever beam

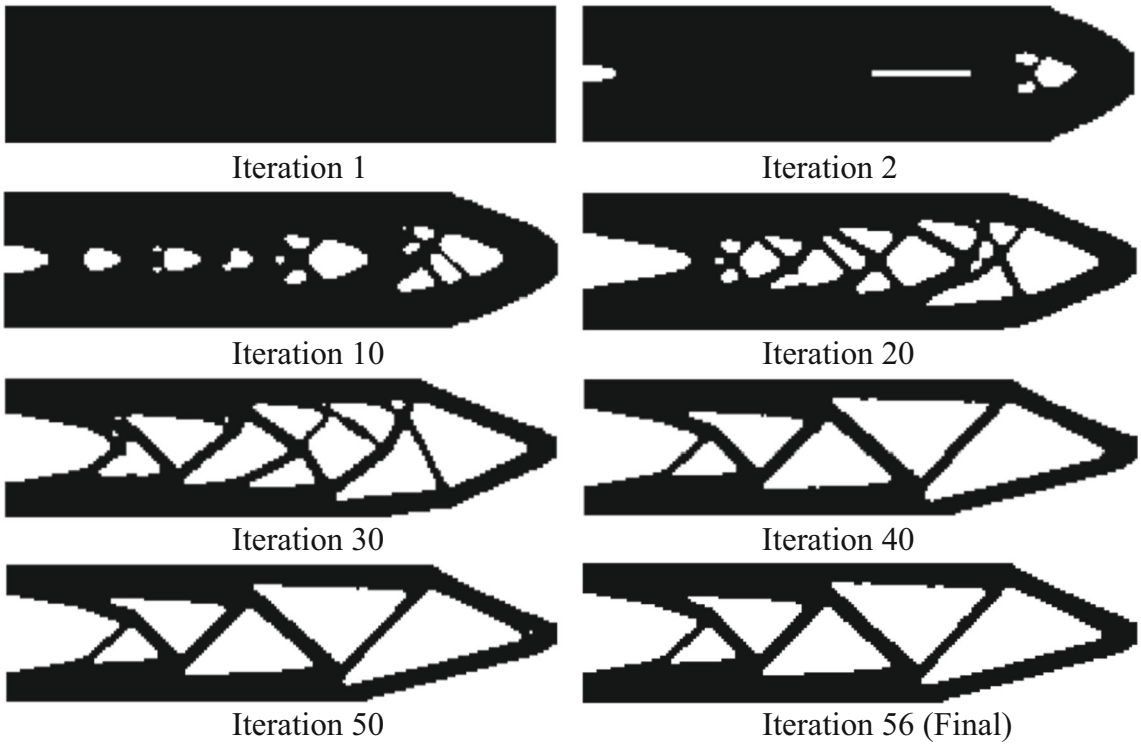

Fig. 2 The nonlinear topology optimization process of the cantilever

4.4 Lines 66–122: nonlinear finite element analysis

The geometrically nonlinear finite element code is written in lines 66–122. Similarly, both nodes and elements are numbered columnwise from left to right (Sigmund 2001; Huang and Xie 2007). Each node has two degrees of freedom (horizontal and vertical); thus, the command (line 69) applies a vertical force (-10^5) at the midpoint of the right boundary. Supports are implemented (lines 70–73) by eliminating fixed degrees of freedom from the linear equations. The initializations of global tangential stiffness matrix `Kt`, internal force `Fint`, and displacement vectors `U` are implemented in lines 74–77. The geometrically nonlinear equilibrium condition `R` must be found using an iterative scheme to obtain the displacement field `U`. The iterative scheme is carried out in lines 83–122. Element strain `estrain` and stress `estress` vectors are obtained at lines 93–94. Line 95 `Fe` represents the elemental internal force vector, and it can be integrated into the global internal force vector `Fint` in line 96. Element strain components (`ex`, `ey`, `exy`) and stress components (`sx`, `sy`, `sxy`) are obtained at lines 97–98. Parameters **A**, **G**, and **M** in (6) and (20) are implemented in lines 99–102, respectively. The linear and nonlinear

Fig. 3 The optimized cantilever beams obtained by using the code in Appendix 1 with different load magnitudes

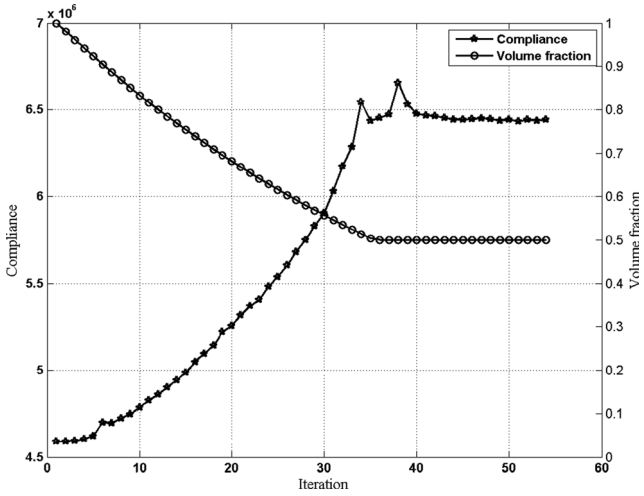

Fig. 4 Evolutionary histories of the volume fraction and the compliance under the load $F = -10^7$

transformation matrixes (`BL`, `BN`) between nodal displacement and element strain in (5)–(11) are implemented in lines 103–104, respectively. Stiffness matrixes $\mathbf{K_L}$, $\mathbf{K_S}$, and $\mathbf{K_N}$ in (16)–(21) are implemented in lines 105–107, respectively. Following this, lines 108–109 determine the element stiffness matrix and integrate it into the global tangential stiffness matrix `Kt`. Different from the linear problem, the element strain matrix `B` and element stiffness matrix `KE` are no longer consistent for the geometrically nonlinear problem. Therefore, the matrixes `B` and `KE` need to be stored (`B_matrix`, `KE_matrix`) in lines 110–111 for sensitivity analysis. Then, the global tangential stiffness matrix `Kt` can be obtained in line 115. The displacement increment vector `Uk` in the current sub-iteration step can be obtained by:

line 118 `Uk(freedofs,:)=Kt(freedofs,freedofs)\R(freedofs,:);`

Then, the displacement vector `U` in the current sub-iteration step can be obtained by:

line 120 `U=U+Uk;`

The 2-norm displacement vector is used as the convergence condition for solving geometrically nonlinear equations as:

line 121 `detaU=norm(Uk,2)/norm(U,2);`

The final displacement vector `U` can be obtained by iterative solution, and the sensitivity analysis (22) is implemented in lines 123–137.

5 Extensions

The MATLAB code given in the Appendix 1 solves the problem of optimizing the material distribution in the cantilever beam (Fig. 1) such that its compliance is minimized. In this

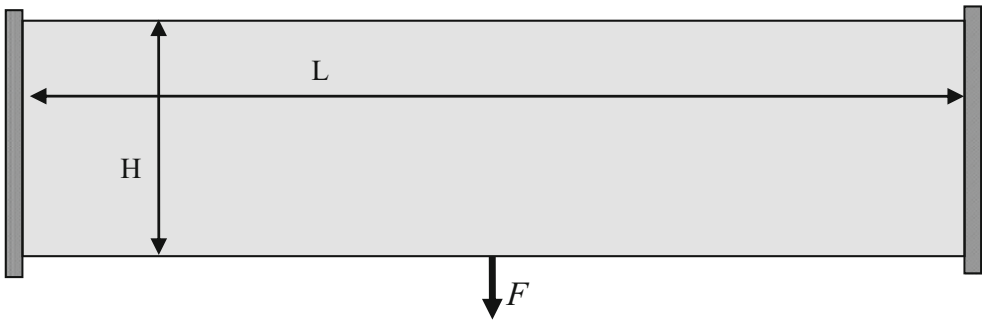

Fig. 5 Design domain, boundary conditions, and load of the clamped beam

Linear design (Final step 71) Nonlinear design (Final step 70)

Fig. 6 The optimized clamped beams under linear and nonlinear designs

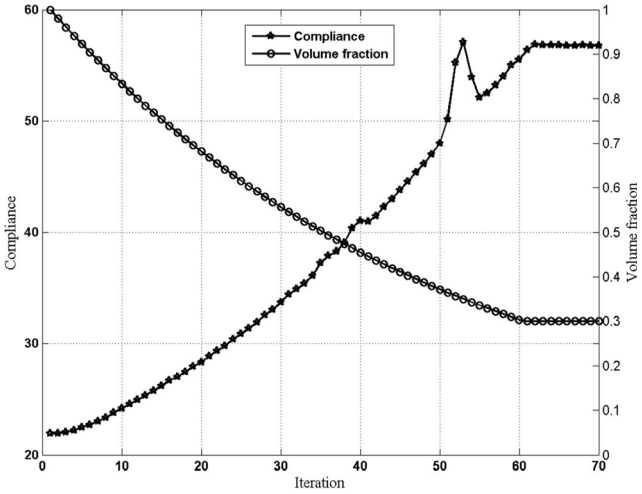

Fig. 7 Evolutionary histories of the volume fraction and the compliance under the load $F = -10^5$

section, cases with different loads will be discussed first. A number of extensions and changes in the program will be illustrated in the following subsections.

5.1 Cantilever beam with different load magnitudes

The topological design of the cantilever beam shown in Fig. 1 is further solved for four different loads by only changing the corresponding code in line 69 of the code. The corresponding optimized topologies are shown in Fig. 3.

The optimized topologies obtained from a small load are symmetric and similar to the results obtained using the linear assumption (Huang and Xie 2007). As the load increases, the topologies become asymmetric. It is worth mentioning that the code could fail to find a meaningful solution under a very large load. This phenomenon is caused by excessive load, which causes excessive deformation of geometric nonlinear structure so that collision (contact) occurs inside the topological structure, which leads to unpredictable changes in stress distribution of the optimized topological structure and then leads to singularity (indefinite or even negative definite) of structural tangential stiffness matrix.

Evolution histories of the volume fraction and the compliance under the load $F = -10^7$ are shown in Fig. 4. It can be seen that both volume fraction and the compliance stably converge after 54 iterations. Compared with the previous work by Zhu et al. (2020b), which shows that the final number of the convergence steps is 100, the presented MATLAB code has better and fast convergence.

5.2 Other boundary conditions and load cases

It is very simple to change boundary conditions and load cases in order to solve other optimization problems. In order to solve the clamped beam problem (the volume fraction constraint is set to 30% and suppose all the material parameters are the same as the cantilever design problem in Section 4) as shown in Fig. 5, only lines 69 and 71 must be changed to:

```
69 F=sparse(2*(nelx/2+1)*(nely+1),1,-100000,2*(nely+1)*(nelx+1),1);

71 fixeddofs=union([1:2*(nely+1)],[2*nelx*(nely+1)+1:2*(nely+1)*(nelx+1)]);
```

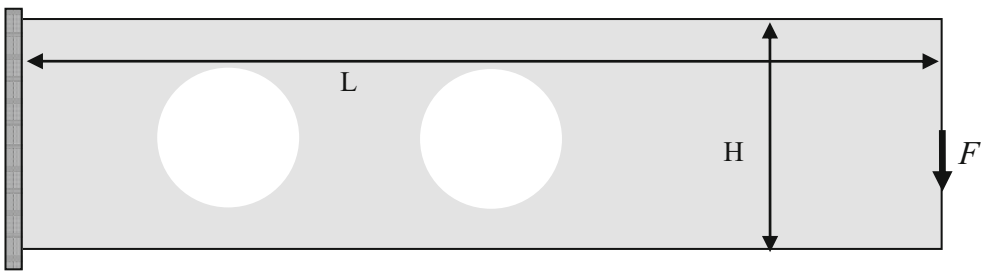

Fig. 8 Design domain, boundary conditions, and load of the cantilever beam with passive regions

Linear design (Final step 55) Nonlinear design (Final step 51)

Fig. 9 The optimized cantilever beams with passive elements under different loads

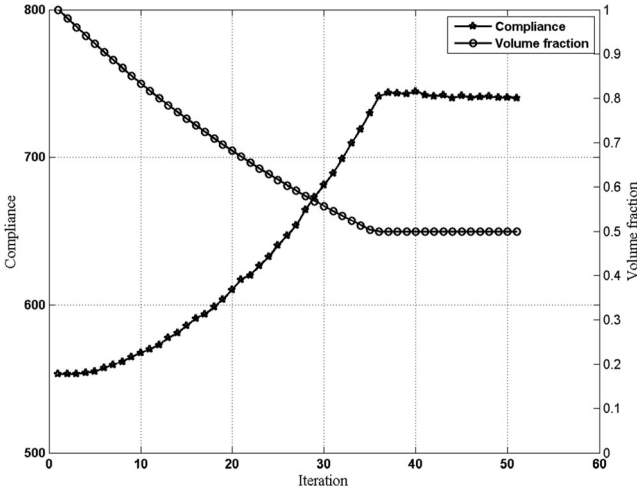

Fig. 10 Evolutionary histories of the volume fraction and the compliance under the load $F = -10^5$

With these changes, the input line for the case shown in Fig. 5 is:

```
TOP_Geo_Non(200,50,0.3,0.02,3)
```

The optimized nonlinear design is shown in Fig. 6. Meanwhile, to make a clear comparison, the result obtained by using a linear assumption with the same load is also presented. It can be observed the difference when using nonlinear topology optimization to solve the design problem. Linear and nonlinear programs are different, mainly in the finite element analysis, element stiffness matrix, etc. For completeness, the linear optimization design code is also appended to the manuscript (see Appendix 2: Topology optimization of linear structure).

Evolution histories of the volume fraction and the compliance under the load $F = -10^5$ are shown in Fig. 7. It can be seen that both volume fraction and the compliance stably converge after 70 iterations.

5.3 Design with passive elements

In some cases, some areas of the design domain may be required to remain void (some of the elements may be required to take the minimum density value), e.g., a hole for a pipe. The design problem shown in Fig. 8 is discussed, where two holes are located inside of the design domain. The size and location of the holes are defined by:

$$\left(elx-\frac{L}{4}\right)^2+\left(ely-\frac{H}{2}\right)^2=\left(\frac{3H}{10}\right)^2 \tag{43}$$

$$\left(elx-\frac{L}{2}\right)^2+\left(ely-\frac{H}{2}\right)^2=\left(\frac{3H}{10}\right)^2 \tag{44}$$

Same as the design of the cantilever beam as illustrated in Section 4, it is assumed that the volume fraction constraint is set as 50% and all the material parameters are the same as those in the cantilever design problem in Section 4. To solve this problem, lines 30 and 34 must be changed to:

```
30 if loop>1; [x]=ADDDEL(nelx,nely,vol,dc,x,xmin,passive,void_num); end
34 fprintf('It.:%4i Obj.:%10.4f Vol.:%6.3f ch.:%6.3f\n',loop,c(loop),
    sum(sum(x))/(nelx*nely-void_num),change);
```

The added line

```
43b x(find(passive))=xmin;
```

in the BESO optimizer subroutine looks for passive elements and sets their density equal to the minimum density (0.001).

Lines 39 and 44 in the BESO optimizer subroutine must be changed to:

```
39 function [x]=ADDDEL(nelx,nely,volfra,dc,x,xmin,passive,void_num)
44 if sum(sum(x))-volfra*(nelx*nely-void_num) > 0;
```

To set up and count the passive elements, the following code needs to be added between lines 5 and 6:

```
for ely = 1:nely
    for elx = 1:nelx
        if sqrt((ely-nely/2.)^2+(elx-nelx/4.)^2) < 3*nely/10.
            passive(ely,elx) = 1;
        elseif sqrt((ely-nely/2.)^2+(elx-2*nelx/4.)^2) < 3*nely/10.
            passive(ely,elx) = 1;
        else
            passive(ely,elx) = 0;
        end
    end
end
x(find(passive))=xmin;
void_num=0;
for ely=1:nely
    for elx=1:nelx
        if x(ely,elx)==xmin
            void_num=void_num+1;
        end
    end
end
```

With these changes, the input line for the case shown in Fig. 8 is:

```
TOP_Geo_Non(200,50,0.5,0.02,3)
```

The optimized nonlinear design is shown in Fig. 9. Similarly, to make a clear comparison, the result obtained by using a linear assumption with the same load is also presented. It can be observed the difference when using nonlinear topology optimization to solve the design problem.

Evolution histories of the volume fraction and the compliance under the load $F = -10^5$ are shown in Fig. 10. It can be seen that both volume fraction and the compliance stably converge after 51 iterations.

6 Conclusions

This paper has constructed a very simple implementation of a 137-line MATLAB code for geometrically nonlinear structural topology optimization. To simplify the code, the end compliance is selected as the objective function. Nineteen lines are used for comments, and 118 lines are needed to accomplish the parameter initialization, design problem definition, geometrically nonlinear finite element analysis, sensitivities analysis and sensitivities average, and design variable updating based on the BESO optimizer. Compared with other published nonlinear topology optimization code (Chen et al. 2019; Zhu et al. 2020b) for educational purpose, the presented code is easier to implement and understand. The code presented in this paper does not require the aid of other software, such as ANSYS, and the code converges very well. Future work intends to extend the code constructed in this paper to strengthen design of nonlinear structure.

Funding This work was sponsored by the National Natural Science Foundation of China (11872311).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results All the necessary data to reproduce the results reported here are provided in the Appendix. Readers can also contact us to obtain the codes by email: hanys0407@mail.nwpu.edu.cn

Appendix 1: Topology optimization of geometrically nonlinear structure

```
function TOP_Geo_Non(nelx,nely,volfrac,er,rmin)
% INITIALIZE
L=nelx;   H=nely;  dx=L/nelx;  dy=H/nely;  a=dx/2;
E=3e9;    nu=0.4;  xmin=1e-3;  x(1:nely,1:nelx)=1.;
penal=1;  vol=1;   loop=0;     change=1;
% START iTH ITERATION
while change>0.001 && loop<150
   loop=loop+1;
   if loop>1; olddc=dc; vol=max(vol*(1-er),volfrac); end
    % FE-ANALYSIS AND SENSITIVITY ANALYSIS
    [U,dc,KE_matrix]=FEA(nelx,nely,x,penal,E,nu,dx,dy,a);
    % OBJECTIVE FUNCTION
    c(loop)=0.;
    hv4=1;
    for ely=1:nely
        for elx=1:nelx
            n1=(nely+1)*(elx-1)+ely;
            n2=(nely+1)* elx   +ely;
     Ue=U([2*n1-1;2*n1;2*n2-1;2*n2;2*n2+1;2*n2+2;2*n1+1;2*n1+2],1);
            KE=KE_matrix(:,:,hv4);
            hv4=hv4+1;
            c(loop)=c(loop)+0.5*x(ely,elx)^penal*Ue'*KE*Ue;
        end
    end
    % FILTERING OF SENSITIVITIES
    [dc]=check(nelx,nely,rmin,dc);
    % STABLIZATION OF EVOLUTIONARY PROCESS
    if loop>1; dc=(dc+olddc)/2.; end
    % BESO DESIGN UPDATE
    if loop>1; [x]=ADDDEL(nelx,nely,vol,dc,x,xmin); end
    % CONVERGENCE FACTOR
    if loop>10;
    change=abs(sum(c(loop-9:loop-5))-sum(c(loop-4:loop)))/sum(c(loop
    -4:loop)); end
    % PRINT RESULTS
    fprintf('It.:%4i Obj.:%10.4f Vol.:%6.3f
    ch.:%6.3f\n',loop,c(loop),sum(sum(x))/(nelx*nely),change);
    % PLOT DENSITIES
    colormap(gray); imagesc(1-x); caxis([0,1]); axis equal; axis off;
    drawnow;
end
% BESO UPDATE
function [x]=ADDDEL(nelx,nely,volfra,dc,x,xmin)
```

```
l1=min(min(dc)); l2=max(max(dc));
while ((l2-l1)/l2 > 1.0e-5)
   th=(l1+l2)/2.0;
   x=max(xmin,sign(dc-th));
   if sum(sum(x))-volfra*(nelx*nely) > 0;
      l1=th;
   else
      l2=th;
   end
end
% MESH-INDEPENDENCY FILTER
function [dcf]=check(nelx,nely,rmin,dc)
dcf=zeros(nely,nelx);
for i=1:nelx
   for j=1:nely
      sum=0.0;
      for k=max(i-floor(rmin),1):min(i+floor(rmin),nelx)
         for l=max(j-floor(rmin),1):min(j+floor(rmin),nely)
            fac=rmin-sqrt((i-k)^2+(j-l)^2);
            sum=sum+max(0,fac);
            dcf(j,i)=dcf(j,i)+max(0,fac)*dc(l,k);
         end
      end
      dcf(j,i)=dcf(j,i)/sum;
   end
end
% GEOMETRICALLY NONLINEAR FE-ANALYSIS
function [U,dc,KE_matrix]=FEA(nelx,nely,x,penal,E,nu,dx,dy,a)
% Loads
F=sparse(2*(nelx+1)*(nely+1)-nely,1,-100000,2*(nely+1)*(nelx+1),1);
% Supports
fixeddofs=1:2*(nely+1);
alldofs  =1:2*(nely+1)*(nelx+1);
freedofs =setdiff(alldofs,fixeddofs);
% Initial Global Stiffness Matrix, Load and Displacement Vectors
Kt=sparse(2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
U=sparse(2*(nely+1)*(nelx+1),1);
Fint=sparse(2*(nely+1)*(nelx+1),1);
BL=1/(4*a)*[-1 0 1 0 1 0 -1 0; 0 -1 0 -1 0 1 0 1; -1 -1 -1 1 1 1 1 -1];
BN0=zeros(3,8);
B=BL+BN0;
DE=E/(1-nu^2)*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
A0=dx*dy;  detaU=1.;  loop=0;
% Iteratively solve the geometric nonlinear FE balance equation
while detaU > 0.001 && loop<50           % Convergence condition
   loop=loop+1;
   hv1=1;  hv3=1;
   for ely=1:nely
      for elx=1:nelx
         n1=(nely+1)*(elx-1)+ely;
         n2=(nely+1)* elx   +ely;
     edof=[2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
         Ue=U(edof);
         estrain=x(ely,elx)*B*Ue;
         estress=x(ely,elx)*DE*B*Ue;
         Fe=B'*estress*A0;
```

```
96          Fint(edof)=Fint(edof)+Fe;
97          ex=estrain(1);  ey=estrain(2);  exy=estrain(3);
98          sx=estress(1);  sy=estress(2);  sxy=estress(3);
99          A=[ex 0.5*exy 0 0; 0 0 0.5*exy ey; 0.5*exy ey ex 0.5*exy];
100         G=1/(4*a)*[-1 0 1 0 1 0 -1 0; 0 -1 0 1 0 1 0 -1;
101                    -1 0 -1 0 1 0 1 0; 0 -1 0 -1 0 1 0 1];
102         M=[sx 0 sxy 0; 0 sx 0 sxy; sxy 0 sy 0; 0 sxy 0 sy];
103         BN=A*G;
104         B=BL+BN;
105         KL=BL'*DE*BL*A0;
106         KS=G'*M*G*A0;
107         KN=(BL'*DE*BN+BN'*DE*BL+BN'*DE*BN)*A0;
108         KE=KL+KN+KS;
109         Kt(edof,edof) = Kt(edof,edof) +x(ely,elx)^penal*KE;
110         B_matrix(:,:,hv1)=B;
111         KE_matrix(:,:,hv3)=KE;
112         hv1=hv1+1;   hv3=hv3+1;
113     end
114  end
115  Kt=(Kt+Kt')/2;
116  % Displacement field
117  R=F-Fint;
118  Uk(freedofs,:)=Kt(freedofs,freedofs)\R(freedofs,:);
119  Uk(fixeddofs,:)=0;
120  U=U+Uk;
121  detaU=norm(Uk,2)/norm(U,2);
122 end
123 % Sensitivity analysis
124 hv2=1;
125 for ely=1:nely
126   for elx=1:nelx
127      n1=(nely+1)*(elx-1)+ely;
128      n2=(nely+1)* elx   +ely;
129   edof=[2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
130      Ue=U(edof);
131      B=B_matrix(:,:,hv2);
132      estress=x(ely,elx)*DE*B*Ue;
133      Fint=B'*estress*A0;
134      dc(ely,elx)=0.5*Ue'*Fint;
135      hv2=hv2+1;
136   end
137 end
138 %=====================================================================
139 % Bidirectional Evolutionary Structural Optimization (BESO)
140 % Stiffness Design with Geometric Nonlinearity
141 % 2D Cantilever Beam
142 % Yongsheng Han
143 % School of Mechanics, Civil Engineering and Architecture,
144 % Northwestern Polytechnical University, Xi'an, Shaanxi, China.
145 % Email: hanys0407@mail.nwpu.edu.cn
146 % September 25, 2020
147 %=====================================================================
```

Appendix 2: Topology optimization of linear structure

```
%%%% A SOFT-KILL BESO CODE BY X. HUANG and Y.M. Xie
%%%% Huang X, Xie YM (2010) Topology optimization of continuum
%%%% structures: methods and applications, Wiley, Chichester.
%%%% doi: 10.1002/9780470689486
function softkill(nelx,nely,volfrac,er,rmin)
% INITIALIZE
E=3e9;  nu=0.4;  xmin=1e-3;  x(1:nely,1:nelx)=1.;
vol=1;  loop=0;  change=1.;  penal=1;
% START iTH ITERATION
while change > 0.001 && loop<150
   loop = loop + 1;
   if loop > 1; olddc = dc; vol = max(vol*(1-er),volfrac); end
   % FE-ANALYSIS
   [U] = FEA(nelx,nely,x,penal,E,nu);
   % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
   [KE] = lk(E,nu);
   c(loop) = 0.;
   for ely = 1:nely
      for elx = 1:nelx
         n1 = (nely+1)*(elx-1)+ely;
         n2 = (nely+1)* elx   +ely;
   Ue =U([2*n1-1;2*n1;2*n2-1;2*n2;2*n2+1;2*n2+2;2*n1+1;2*n1+2],1);
         c(loop) = c(loop) + 0.5*x(ely,elx)^penal*Ue'*KE*Ue;
         dc(ely,elx) = 0.5*penal*x(ely,elx)^penal*Ue'*KE*Ue;
      end
   end
   % FILTERING OF SENSITIVITIES
   [dc] = check(nelx,nely,rmin,dc);
   % STABLIZATION OF EVOLUTIONARY PROCESS
   if loop > 1; dc = (dc+olddc)/2.; end
   % BESO DESIGN UPDATE
   if loop > 1; [x] = ADDDEL(nelx,nely,vol,dc,x,xmin); end
   % DETERMINE CONVERGENCE FACTOR
   if loop>10
   change=abs(sum(c(loop-9:loop-5))-sum(c(loop-4:loop)))/sum(c(loop
   -4:loop));
   end
   % PRINT RESULTS
```

```
97 KE=E/(1-nu^2)*[k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
98                k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
99                k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
100               k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
101               k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
102               k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
103               k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
104               k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
105 %================================================================
```

References

Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in Matlab using 88 lines of code. Struct Multidiscip Optim 43(1):1–16. https://doi.org/10.1007/s00158-010-0594-7

Bendsøe MP (1989) Optimal shape design as a material distribution problem. Struct Optim 1(4):193–202. https://doi.org/10.1007/BF01650949

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

Bendsøe MP, Sigmund O (2003) Topology optimization: theory, methods and applications. Springer, Berlin

Challis VJ (2010) A discrete level-set topology optimization code written in Matlab. Struct Multidiscip Optim 41(3):453–464. https://doi.org/10.1007/s00158-009-0430-0

Chen Q, Zhang X, Zhu B (2019) A 213-line topology optimization code for geometrically nonlinear structures. Struct Multidiscip Optim 59(5):1863–1879. https://doi.org/10.1007/s00158-018-2138-5

Crisfield MA (1991) Non-linear finite element analysis of solids and structures. Wiley, New York

Deaton JD, Grandhi RV (2014) A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct Multidiscip Optim 49(1):1–38. https://doi.org/10.1007/s00158-013-0956-z

Ferrari F, Sigmund O (2020) A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D. Struct Multidiscip Optim 62(4):2211–2228. https://doi.org/10.1007/s00158-020-02629-w

Guo X, Zhang W, Zhang J, Yuan J (2016) Explicit structural topology optimization based on moving morphable components (MMC) with curved skeletons. Comput Methods Appl Mech Eng 310:711–748. https://doi.org/10.1016/j.cma.2016.07.018

Han YS, Xu B, Zhao L, Xie YM (2019) Topology optimization of continuum structures under hybrid additive-subtractive manufacturing constraints. Struct Multidiscip Optim 60(6):2571–2595. https://doi.org/10.1007/s00158-019-02334-3

Huang X, Xie YM (2007) Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method. Finite Elem Anal Des 43(14):1039–1049. https://doi.org/10.1016/j.finel.2007.06.006

Huang X, Xie YM (2008) Topology optimization of nonlinear structures under displacement loading. Eng Struct 30:2057–2068. https://doi.org/10.1016/j.engstruct.2008.01.009

Huang X, Xie YM (2009) Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials. Comput Mech 43(3):393–401. https://doi.org/10.1007/s00466-008-0312-0

Huang X, Xie YM (2010a) A further review of ESO type methods for topology optimization. Struct Multidiscip Optim 41(5):671–683. https://doi.org/10.1007/s00158-010-0487-9

Huang X, Xie YM (2010b) Topology optimization of continuum structures: methods and applications. Wiley, Chichester. https://doi.org/10.1002/9780470689486

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

Lagaros ND, Vasileiou N, Kazakis G (2019) Ac# code for solving 3d topology optimization problems using sap2000. Optim Eng 20(1):1–35. https://doi.org/10.1007/s11081-018-9384-7

Laurain A (2018) A level set-based structural optimization code using FEniCS. Struct Multidiscip Optim 58(3):1311–1334. https://doi.org/10.1007/s00158-018-1950-2

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(1):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 Multidiscip Optim 50(6):1175–1196. https://doi.org/10.1007/s00158-014-1107-x

Loyola RA, Querin OM, Jiménez AG et al (2018) A sequential element rejection and admission (SERA) topology optimization code written in Matlab. Struct Multidiscip Optim 58(3):1297–1310. https://doi.org/10.1007/s00158-018-1939-x

Luo Z, Tong L, Wang MY, Wang S (2007) Shape and topology optimization of compliant mechanisms using a parameterization level set method. J Comput Phys 227(1):680–705. https://doi.org/10.1016/j.jcp.2007.08.011

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

Qian X (2013) Topology optimization in B-spline space. Comput Methods Appl Mech Eng 265:15–35. https://doi.org/10.1016/j.cma.2013.06.001

Rong JH, Xie YM, Yang XY (2001) An improved method for evolutionary structural optimisation against buckling. Comput Struct 79(3):253–263. https://doi.org/10.1016/S0045-7949(00)00145-0

Rozvany GIN (2001) Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. Struct Multidiscip Optim 21(2):90–108. https://doi.org/10.1007/s001580050174

Sethian JA, Wiegmann A (2000) Structural boundary design via level set and immersed interface methods. Int J Numer Methods Eng 163(2):489–528. https://doi.org/10.1006/jcph.2000.6581

Sigmund O (2001) A 99 line topology optimization code written in Matlab. Struct Multidiscip Optim 21(2):120–127. https://doi.org/10.1007/s001580050176

Sigmund O, Maute K (2013a) Topology optimization approaches-a comparative review. Struct Multidiscip Optim 48(6):1031–1055. https://doi.org/10.1007/s00158-013-0978-6

Sigmund O, Maute K (2013b) Topology optimization approaches-a comparative review. Struct Multidiscip Optim 48(6):1031–1055. https://doi.org/10.1007/s00158-013-0978-6

Suzuki K, Kikuchi N (1991) A homogenization method for shape and topology optimization. Comput Methods Appl Mech Eng 93(3): 291–318. https://doi.org/10.1016/0045-7825(91)90245-2

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(3):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 Multidiscip Optim 49(4):621–642. https://doi.org/10.1007/s00158-013-0999-1

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 R, Zhang X, Zhu B (2019) Imposing minimum length scale in moving morphable component (mmc)-based topology optimization using an effective connection status (ecs) control method. Comput Methods Appl Mech Eng 351:667–693. https://doi.org/10.1016/j.cma.2019.04.007

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. 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 Multidiscip Optim 52:1229–1241. https://doi.org/10.1007/s00158-015-1294-0

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

Xie YM, Steven GP (1993) A simple evolutionary procedure for structural optimization. Comput Struct 49(5):885–896. https://doi.org/10.1016/0045-7949(93)90035-C

Xie YM, Steven GP (1994) Optimal design of multiple load case structures using an evolutionary procedure. Eng Comput 11(4):295–302. https://doi.org/10.1108/02644409410799290

Xu B, Han YS, Zhao L (2020) Bi-directional evolutionary topology optimization of geometrically nonlinear continuum structures with stress constraints. Appl Math Model 80:771–791. https://doi.org/10.1016/j.apm.2019.12.009

Yaghmaei M, Ghoddosian A, Khatibi MM (2020) A filter-based level set topology optimization method using a 62-line Matlab code. Struct Multidiscip 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 Multidiscip Optim 53(6):1243–1260. https://doi.org/10.1007/s00158-015-1372-3

Zhou M, Rozvany GIN (1991) The COC algorithm, part II: topological, geometrical and generalized shape optimization. Comput Methods Appl Mech Eng 89(1):309–336. https://doi.org/10.1016/0045-7825(91)90046-9

Zhu J, Zhang W, Xia L (2016) Topology optimization in aircraft and aerospace structures design. Arch Comput Methods Eng 23(4): 595–622. https://doi.org/10.1007/s11831-015-9151-2

Zhu B, Zhang X, Zhang H, Liang J, Zang H, Li H, Wang R (2020a) 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 (2020b) An 89-line code for geometrically nonlinear topology optimization written in FreeFEM. Struct Multidiscip Optim. https://doi.org/10.1007/s00158-020-02733-x

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