Heliyon 10 (2024) e25994

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Finite element-based optimization procedure for an irregular domain with unstructured mesh

Md Shahidul Islam [*], Ali Zulkar Nayem, Kazi Naimul Hoque

Department of Naval Architecture and Marine Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

ARTICLE INFO

Keywords:
Topology optimization
Finite element method
SIMP
Unstructured mesh
Python

ABSTRACT

At present, structural optimization is a highly demanding area of research in engineering. Engineers aim to minimize material in a body while maintaining its usability and safety at the same time. Developing a user-friendly program to optimize a structure using the finite element method (FEM) is the goal of the current study. With the advent of additive manufacturing, the production of complex-shaped designs is showing promise. A detailed optimization algorithm based on solid isotropic material with penalization (SIMP) is presented in this paper. `UnTop2D`: An object-oriented Python program with a graphical user interface (GUI) has been developed, which can be applied to structures with both structured and unstructured meshes. The mesh is not required to be topologically ball and can be imported from professional meshing software. Any selected element can be frozen to prevent its removal during optimization, and wall elements can also be frozen for real-world scenarios. The optimized structure can be exported as an Abaqus input file for structural analysis and STL file for 3D printing. This paper presents several examples to demonstrate the effectiveness of the proposed procedure.

1. Introduction

The modern trend for engineers is to design safe structures with minimum cost and innovative design. Engineers use finite element analysis (FEA) to check whether a structure is safe under applied loading and boundary conditions. By the trial-and-error method, it takes a long time to optimize a structure, as each intermediate design, after the removal of materials with less developed stress, has to be checked. This technique might not lead to the desired result, as checking the result files manually is physically nearly impossible, and the contour plot usually shows banded results. Instead, structural optimization techniques are being implemented.

There are three types of structural optimization: size, shape and topology. If a block with a circular slot is taken as an example for size optimization, the slot's position remains fixed, while its size is determined by its radius. To optimize the block weight, the slot's radius can be adjusted. However, the structure's geometry remains unchanged. In the case of shape optimization, if the same block is considered, the circular shape may not be a good option. Shapes such as ovals might be more suitable. To control the slot's shape during optimization, parametric curves, such as a closed Bezier curve, can be chosen to optimize the slot's boundary. The shape design variables, in this case, would be the control point locations [1].

The question of how to arrange material within a design space to achieve optimal structural performance is a fundamental problem in engineering that topology optimization can help address. The initial configuration can be updated using this. Topology optimization

* Corresponding author.
E-mail address: shahid777@name.buet.ac.bd (M.S. Islam).

https://doi.org/10.1016/j.heliyon.2024.e25994
Received 3 June 2023; Received in revised form 3 January 2024; Accepted 6 February 2024
Available online 8 February 2024

can enhance structural efficiency by increasing stiffness while reducing the material used [2]. While this idea was first introduced for mechanical design problems, it has since been adopted across various other physical domains, such as fluids [3,4], acoustics [5], electromagnetics [6], optics [7,8] and their interdisciplinary combinations. Our current research is limited to mechanical design.

Over the past few decades, extensive research has been conducted on topology optimization [9,10]. Numerous methodologies have been developed to address this area. One such approach is the ground structure method [11–16], which employs an extensive array of truss or beam elements within the defined design domain. The design variable, in this case, is the individual cross-sectional area. When optimizing, elements with a cross-sectional area of zero or approaching the lower limit of the design variable are removed from the design. Another optimization technique is the evolutionary structural optimization (ESO) [17–21]. ESO is an optimization method that systematically eliminates inefficient material from a structure to achieve an optimal configuration. It is closely linked to the classical fully stressed design approach, where ideal structures have uniform stress levels. A rejection criterion based on local stress levels is applied, with materials experiencing low stress being deemed inefficient and removed from the design. Furthermore, a different category of topology optimization methods involves using structural boundaries as the design variables. One prominent example of this approach is the level-set method [22–25], which was first proposed by Osher and Sethian [26]. Another technique, known as the homogenization method [27–29], converts complex structural topology problems into size optimization problems by introducing a material density function in each element. This method determines the mechanical properties of materials and can be used to solve various topology optimization problems, offering mathematical bounds on theoretical structural performance.

SIMP is one of the main existing methods. This is the most used FEA-based topology optimization method that emerged as an alternative to the homogenization method in order to avoid composite materials in final domains [30]. Rozvany et al. [31] coined the name SIMP. In this method, the structure is discretized into small elements whose material properties are assumed constant across the element and are calculated as material properties multiplied by relative material densities to the power penalization number. Relative material densities are continuous, varying from 0 to 1. In practice, an infinitesimal number is used instead of 0 for numerical stability representing void material. Moreover, 1 means 100% material in an element. These relative material densities are solved through some calculus operations for a given objective (e.g., compliance minimization). In 2001, Sigmund [32] implemented an educational 99-line MATLAB code for compliance minimization problems for statically loaded structures with square elements. The 99-line MATLAB code inspired many researchers to work on the SIMP algorithm. Later, the code was enhanced to an 88-line MATLAB code [33] with improved performance and an added density filter. Talischi et al. [34] presented a MATLAB code for structural topology optimization featuring a finite element routine based on isoparametric polygonal elements. They focused on utilizing unstructured meshes within arbitrary domains, an area that had taken little attention before. Bochenek and Tajs-Zielinska [35] extended the concept of cellular automata to an unstructured grid of cells related to non-regular finite element meshes.

In our current work, we have developed a general tool named `UnTop2D` for topology optimization based on the SIMP method to optimize two-dimensional problems. Quadrilateral (Q4) plain stress element is used. The software can import any Q4 mesh, whether structured or unstructured, from professional software such as Abaqus, Ansys, Nastran or Gmsh. The mesh does not need to be topologically ball so that the design domain can be of any shape, regular or irregular. The GUI allows users to set boundary conditions, loads and optimization parameters easily. The program has a mesh check tool additionally.

During the optimization process, calculating element stiffness matrices becomes a crucial step. Unstructured meshes necessitate a higher computational cost for these calculations. To address this challenge, stiffness matrices are computed during the first optimization iteration and stored in the memory. Subsequent iterations draw data from these stored matrices, incurring a memory cost while significantly conserving computational resources. Real-time tracking of structural compliance and the maximum change in relative material density during optimization is possible with the graphs to facilitate monitoring of optimization convergence. Each optimization iteration is visualized graphically and can be saved within the project directory. This feature is particularly advantageous for research and analysis, eliminating the need for optimization calculations each time.

The tool offers the flexibility to designate specific elements as ‘frozen’, preventing their removal during optimization. Users can also choose to freeze elements - those adjacent to the design domain walls. Unlike previous literature, which often lacked emphasis on postprocessing, our approach incorporates a filtering technique. This technique eliminates insignificant elements from the structure after optimization, enabling export to Abaqus for further structural analysis or to STL format for 3D printing.

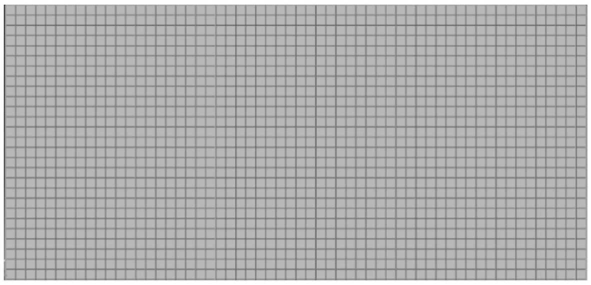


Fig. 1. A structured mesh.

2. Structured and unstructured meshes

A regular design domain is rectangular and has no holes or cuts. It can be discretized using a structured mesh. On the contrary, an irregular design domain is not rectangular and may have holes or cuts. It cannot be discretized using a structured mesh, so an unstructured mesh is typically required.

Structured meshes are characterized by a predetermined and periodic node-element connectivity pattern. This means that the same pattern of elements is repeated throughout the mesh, as shown in Fig. 1. Unstructured meshes, on the other hand, do not have a periodic connectivity pattern. The number of elements connected to a node can vary, and the internal angles of the quadrilaterals can also change. This is because unstructured meshes are designed to conform to the geometry of the domain and other constraints, as shown in Fig. 2. The size of the elements in an unstructured mesh can also vary significantly across the mesh. In the close-up view of the top-left circular opening, node A is connected to three elements, node B to five elements and node C to four elements. More details can be found in Ref. [36].

3. Optimization problem

The optimization goal is to minimize the compliance c of the design domain for a given value of volume fraction f. The volume fraction is the ratio of the structure's material volume to the design domain's volume.

The optimizer finds the elements' relative material densities x_e for the following objective in Equation (1) [32].

$$minimize \quad c(\boldsymbol{x}) = \boldsymbol{U}^T\boldsymbol{K}\boldsymbol{U} = \sum_{e=1}^{N}(x_e)^p \boldsymbol{u}_e{}^T\boldsymbol{k}_0\boldsymbol{u}_e \tag{1}$$

Here, $\boldsymbol{U}$ is the global displacement vector, $\boldsymbol{K}$ is the global stiffness matrix, $\boldsymbol{x}$ is the vector containing elements' relative material densities x_e, p is the penalization power (taken as 3 for a Poisson's ratio 1/3), $\boldsymbol{u}_e$ is the nodal displacement vector of elements, $\boldsymbol{k}_0$ is the element stiffness matrix, and N is the total element number. Tfhe maximum value of x_e is 1, which represents solid material in an element. Any value of x_e less than 1 can be considered a porous element. p accelerates the optimization by suppressing the contribution of x_e that is less than 1.

During individual iteration, the following constraints in Equations (2)–(4) are to be satisfied.

$$V(\boldsymbol{x})/_{V_0} = f \tag{2}$$

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F} \tag{3}$$

$$0 < x_{min} \leq x_e \leq 1 \tag{4}$$

Here, $V(\boldsymbol{x})$ is the material volume, V_0 is the design domain volume, and $\boldsymbol{x}_{min}$ is a very small number greater than 0 to avoid the singularity in numerical analysis. Equation (3) is solved for $\boldsymbol{U}$ using FEA.

4. Solution steps

Fig. 3 shows the optimization process based on the SIMP algorithm.

4.1. Design domain and finite element discretization

The design domain may contain openings or voids. Before performing FEA, the design domain must be discretized. This can be

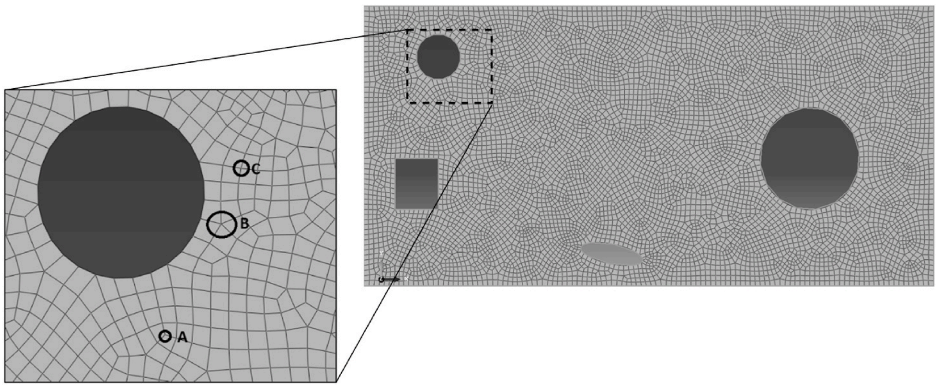


Fig. 2. An unstructured mesh.

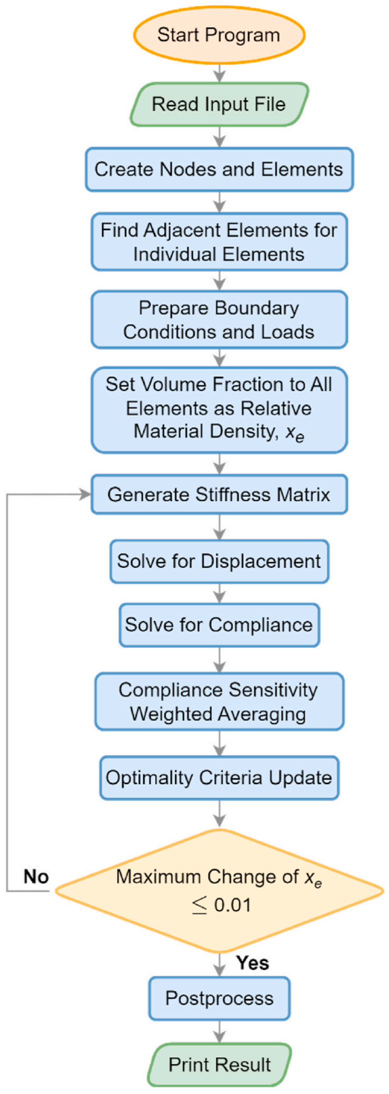

Start Program
Read Input File
Create Nodes and Elements
Find Adjacent Elements for Individual Elements
Prepare Boundary Conditions and Loads
Set Volume Fraction to All Elements as Relative Material Density, x_e
Generate Stiffness Matrix
Solve for Displacement
Solve for Compliance
Compliance Sensitivity Weighted Averaging
Optimality Criteria Update
Maximum Change of x_e ≤ 0.01
No
Yes
Postprocess
Print Result

Fig. 3. SIMP-based topology optimization flowchart.

accomplished using commercial, open-source or custom software. The mesh that is produced can be either structured or unstructured.

4.2. Nodes and elements

The developed program of this study takes an object-oriented approach. The mesh data, loads and boundary conditions are stored in a custom input file. The input file parser reads the mesh data to generate nodes and element objects. The node object has geometry,

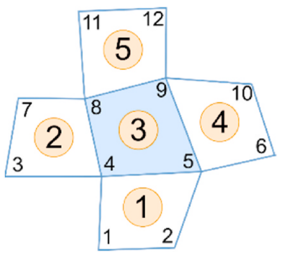


Fig. 4. Finding adjacent elements that share an edge (the encircled integers represent element numbering, and the remaining integers represent node numbering).

loads and boundary conditions attributes. Since the mesh element is quadrilateral, each element object consists of four node objects.

4.3. Finding adjacent elements

The program determines if a target element shares a common edge with other elements (having a common node can also be considered neighboring elements, but current research focuses on edge sharing). The elements with shared edges are then added as neighbors to the target element in a Python list as attributes. These lists of adjacent elements are used for mesh-independence filtering within the optimization loop.

In Fig. 4, element number 3 is surrounded by elements 1, 2, 4 and 5. Element 1 and element 3 share a common edge: nodes 4 & 5. Similarly, the edges of 2 & 3, 3 & 4 and 3 & 5 are shared.

4.4. Optimization loop

At the beginning of the optimization loop, the relative material density x_e of each element is set uniformly to the value of volume fraction f. Then, the loop begins.

4.4.1. Generate element stiffness matrices

In the plane stress formulation, element stiffness matrices $\boldsymbol{k_0}$ are generated and stored in element objects during the first iteration of the optimization loop. In subsequent iterations, the optimizer retrieves the element stiffness matrices from the element objects and accumulates them into the global stiffness matrix $\boldsymbol{K}$.

A linear quadrilateral element is used with four-point Gaussian quadrature [37] to calculate the stiffness. The element stiffness matrix is generated using Equations (5) and (6).

$$\boldsymbol{k_0} = t_e \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^T\boldsymbol{DB}\ det\,\boldsymbol{J}\ d\boldsymbol{\xi}d\boldsymbol{\eta} \tag{5}$$

$$\boldsymbol{B} = \boldsymbol{AG} \tag{6}$$

here, t_e is the thickness of the element. The definitions of other symbols can be found in Ref. [38].

The steps to evaluate the element stiffness matrix $\boldsymbol{k_0}$ are as follows (each step is considered as an individual function in the program).

- o Generate 2×2 Jacobian matrix $\boldsymbol{J}$ for each Gauss point. The parameters are ξ and η where, $\xi = \pm 0.5773$, $\eta = \pm 0.5773$ as shown in Fig. 5.
- o Evaluate determinants of Jacobian matrices $det\,\boldsymbol{J}$.
- o Generate 3×4 $\boldsymbol{A}$ matrix for each element, where the arguments are $\boldsymbol{J}$ and $det\,\boldsymbol{J}$.
- o Generate 4×8 $\boldsymbol{G}$ matrix for each element, where the parameters are ξ and η.
- o Calculate $\boldsymbol{B}$ for each Gaussian point by dot multiplication of $\boldsymbol{A}$ and $\boldsymbol{G}$.
- o Generate 3×3 $\boldsymbol{D}$ matrix, where the arguments are Young's modulus E and Poisson's ratio ν.
- o Calculate $\boldsymbol{k_i}$ for each Gaussian point by $\boldsymbol{B}^T\boldsymbol{DB}\ det\,\boldsymbol{J}$. Here, $i = 1,\ 2,\ 3,\ 4$.
- o Finally, evaluate element stiffness matrix $\boldsymbol{k_0}$ by summing up $\boldsymbol{k_i}$ at all Gaussian points and multiplied by element thickness. So, $\boldsymbol{k_0} = (\boldsymbol{k_1} + \boldsymbol{k_2} + \boldsymbol{k_3} + \boldsymbol{k_4}) \times t_e$

4.4.2. Calculate displacements

The program generates the global stiffness matrix $\boldsymbol{K}$ by accumulating element stiffness matrices $\boldsymbol{k_0}$. The size of the $\boldsymbol{K}$ matrix is $2N \times$

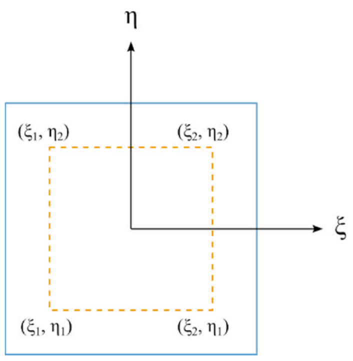


Fig. 5. Four-point Gauss quadrature in the two-dimensional space.

$2N$ for two degrees of freedom (DOF) of nodes, where N is the total node number. In the program, $\boldsymbol{K}$ is constructed as SciPy sparse matrix. During accumulation, $\boldsymbol{k_0}$ is modulated into $\boldsymbol{k_0}^{new}$ by multiplying the relative material densities x_e to power p as shown in Equation (7).

$$\boldsymbol{k_0}^{new} = \boldsymbol{k_0} x_e^{\,p} \tag{7}$$

The global displacement vector $\boldsymbol{U}$ is found by solving Equation (8). The spsolve solver from SciPy [39] is used to solve the sparse linear algebra system. spsolve utilizes UMFPACK routines, which use an unsymmetrical multifrontal method to solve the system. Before solving the system, the constraint DOFs are eliminated from the matrices using the elimination method.

$$\boldsymbol{K}[\boldsymbol{x}]\boldsymbol{U} = \boldsymbol{F} \tag{8}$$

here, the global stiffness matrix $\boldsymbol{K}[\boldsymbol{x}]$ is modulated by $\boldsymbol{x}$, and the load vector $\boldsymbol{F}$ is constructed by accumulating node object loads.

4.4.3. *Calculate compliance and compliance sensitivity*

The element displacement vectors $\boldsymbol{u_e}$ are extracted from the global displacement vector $\boldsymbol{U}$ to calculate the element compliances. According to Equation (1), the element compliance is calculated as $(x_e)^p \boldsymbol{u_e}^T \boldsymbol{k_0} \boldsymbol{u_e}$.

The compliance sensitivities of elements are mathematically expressed as Equation (9). They evaluate the effect of changing the relative material density x_e on the element compliances over optimization iterations.

$$\partial c / \partial x_e = -p(x_e)^{p-1} \boldsymbol{u}_e^T \boldsymbol{k_0} \boldsymbol{u}_e \tag{9}$$

4.4.4. *Compliance sensitivity weighted averaging*

Checkerboard patterns can occur in the final optimized structure due to poor finite element discretization [40]. To address this issue, the compliance sensitivities are modified by weighted averaging over neighboring elements, which are listed in a Python list at the beginning of the program. The weighted averaging operation is performed using Equation (10).

$$\left(\partial c / \partial x_e \right)_{modified} = \frac{1}{\sum_{f=1}^{N} H_f} \sum_{f=1}^{N} H_f x_e \partial c / \partial x_e \tag{10}$$

Here, N is the number of adjacent elements, including the target element. The weight factor H_f is taken as 0.1 for adjacent elements and 1.1 for the target element in current research.

4.4.5. *Optimality criteria update*

Element relative material densities are updated using Equation (11).

$$x_e^{new} = \begin{cases} x_1 \ if x_e B_e^{\eta} \le x_1 \\ x_e B_e^{\eta} \ if \ x_1 < x_e B_e^{\eta} < x_2 \\ x_2 \ if x_e B_e^{\eta} \ge x_2 \end{cases} \tag{11}$$

Here, $x_1 = max(x_{min}, \ x_e - m)$, $x_2 = min(1, \ x_e + m)$ and η is the numerical damping coefficient, which is set to 1/ 2. m is the positive move limit, which is chosen as 0.15. B_e is calculated using Equation (12).

$$B_e = \frac{-\partial c / \partial x_e}{\lambda} \tag{12}$$

Here, λ is the Lagrangian multiplier, which is found using the bisection method to satisfy Equation (2).

After the element relative material densities x_e are updated, the optimizer checks for the maximum change in x_e from the previous iteration. The optimization is finished if the change is smaller than a threshold value (generally 0.01, which is 1% of 1). Otherwise, the program starts a new optimization iteration, beginning from step 4.4.1.

5. The program

5.1. *Coding overview and the GUI*

The PyQt [41] module is used to build the GUI. It is a cross-platform GUI toolkit that is a Python binding for the popular software Qt. NumPy and SciPy [39] are used for numerical computations. NumPy's array data structure is used to store general matrices. It also has functions for linear algebra operations. SciPy or Scientific Python, is a widely used module in the scientific community for optimization, integration, interpolation and solving ordinary differential equations. In this research, SciPy is used to store sparse matrices and perform linear algebra operations on sparse matrices. PyVista [42] is used to visualize 3D graphs. It is a 3D plotting and mesh analysis module. Matplotlib [43] is used to generate 2D plots. The meshio [44] module is used to import meshes generated by any professional meshing tool, such as Abaqus, Ansys, Nastran or Gmsh. It is also used to export the optimized mesh to Abaqus and stl formats.

The main GUI can be divided into three sections, as shown in Fig. 6.

- The **menu bar** is located at the top of the GUI. It contains menu items and sub-menu items. Clicking on any menu item will open a dropdown menu with additional options.
- On the left side of the **control panel**, there is the 'Optimize' button, which starts the optimization process for an imported mesh structure. The right side of the control panel contains navigation buttons. These buttons allow users to navigate through all of the optimization iterations.
- The **visualization panel** shows the mesh data for the optimization domain. The color of each element represents the value of its relative material density. The value for a color can be found in the color bar, which is located at the bottom left of the panel. Boundary conditions and static loads are also annotated in this panel.

5.2. Topology optimization workflow

First, an input file needs to be generated from an external mesh file. This process is not part of the GUI program. A separate Python script named create_input_from_mesh.py is written using the meshio module. This script takes the mesh data and converts it into an input file. A custom syntax is used for the input file to store the mesh data, loads, boundary conditions and topology optimization-specific variables. The input file has specific data deceleration procedures for different entity types (e.g., nodes, elements, etc.).

The GUI software can be opened by running the gui.py file. A new optimization project can be created from a previously generated input file by clicking on the 'Create Project from Input File' submenu item under the 'File' menu item (Fig. 7(a)). The mesh is then

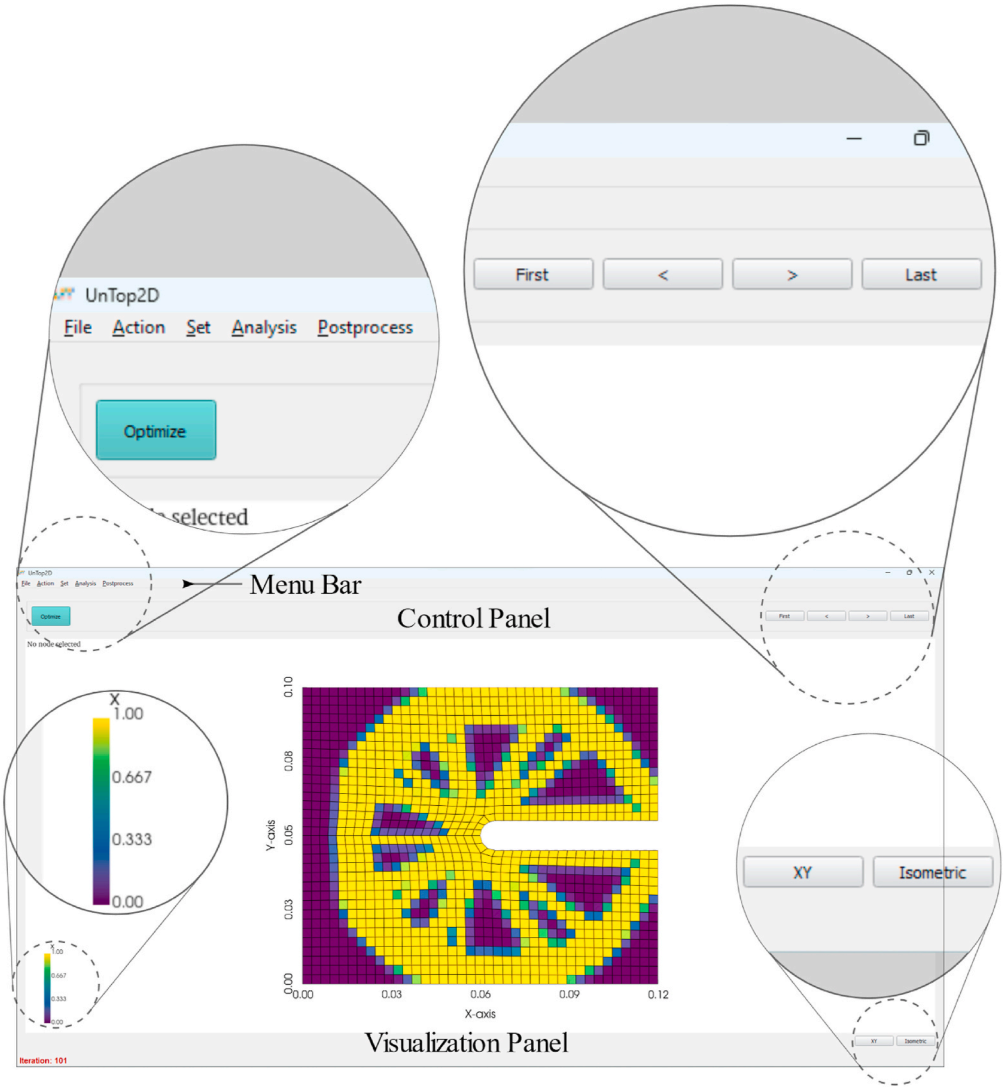
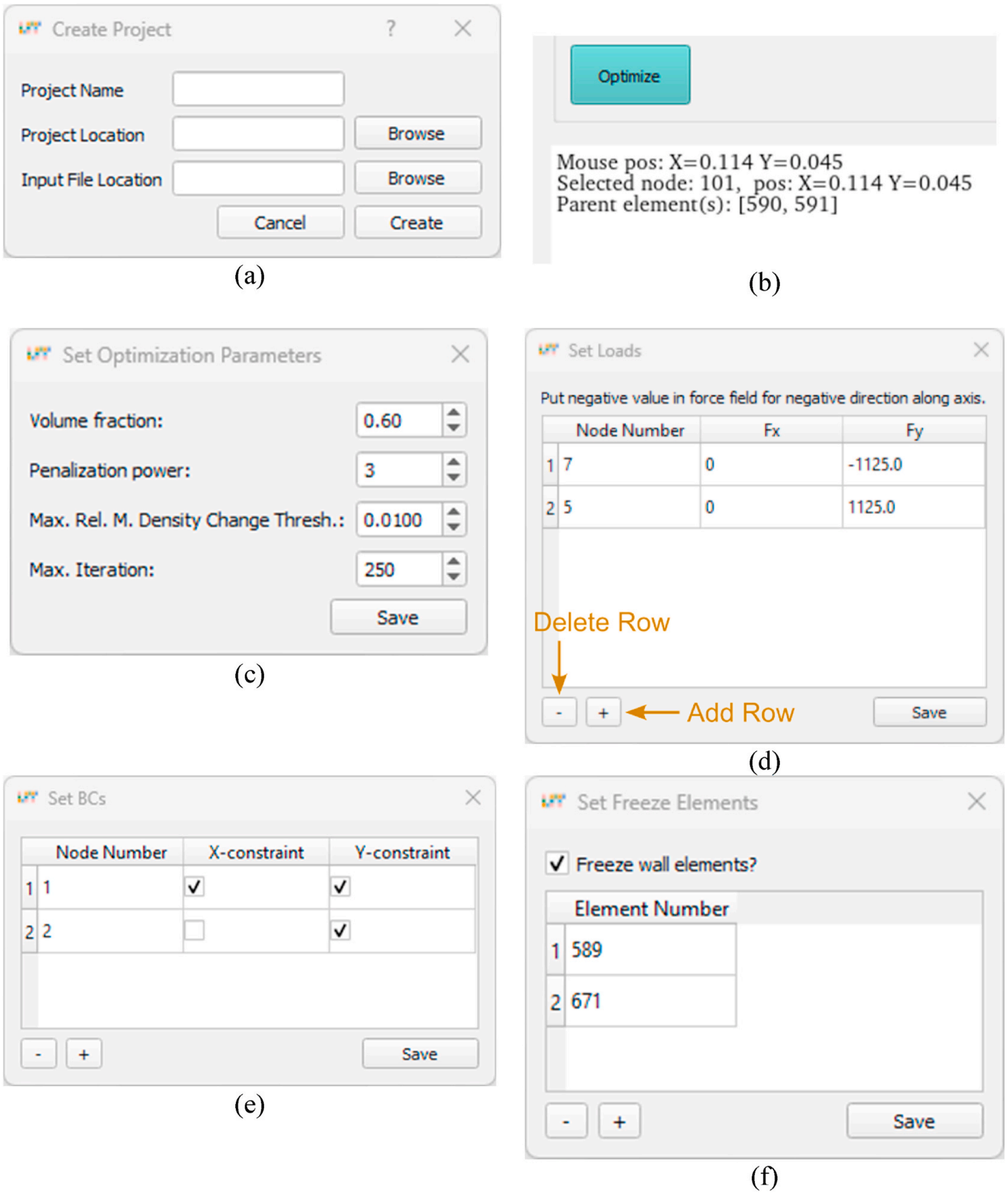


Fig. 6. The GUI of the developed program.



Project Name	
Project Location	
Input File Location	

```
Mouse pos: X=0.114 Y=0.045
Selected node: 101,  pos: X=0.114 Y=0.045
Parent element(s): [590, 591]
```

Volume fraction:	0.60
Penalization power:	3
Max. Rel. M. Density Change Thresh.:	0.0100
Max. Iteration:	250

Put negative value in force field for negative direction along axis.

	Node Number	Fx	Fy
1	7	0	-1125.0
2	5	0	1125.0

	Node Number	X-constraint	Y-constraint
1	1	✓	✓
2	2	☐	✓

✓ Freeze wall elements?

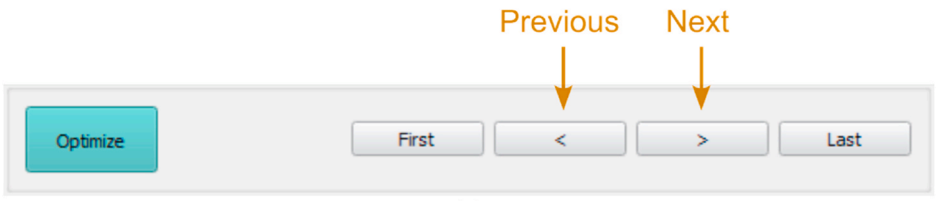
	Element Number
1	589
2	671

Fig. 7. (a) Create an optimization project from an input file, (b) View node properties by clicking on the desired node within the GUI, (c) Set the optimization parameters, (d) Set static loads on nodes. The force acting on a node is divided into x and y components, F_x and F_y. The sign convention is that forces acting along positive axes are considered positive and vice versa, (e) Set the displacement boundary conditions. A zero-displacement boundary condition is exerted on a node along the respective axis by checking the corresponding checkbox, (f) Set the frozen elements.

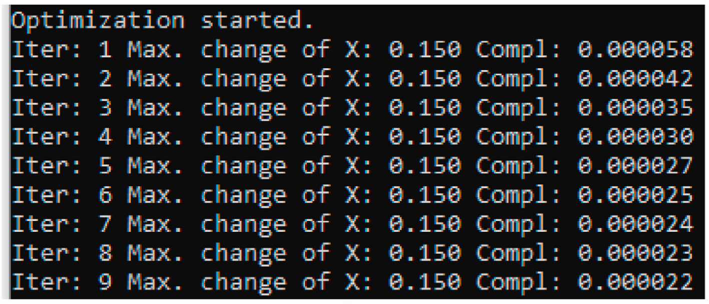
loaded into the visualization panel. By clicking on any point in the mesh geometry, the nearest node will be selected, and its details (i. e., node number, parent elements, etc.) will be displayed at the top-left corner of the visualization panel (Fig. 7(b)). Optimization parameters, static loads on nodes, displacement boundary conditions and element freezing can be set from the 'Set' menu item (Fig. 7 (c-f)). Freezing elements is a process of always keeping the elements present regardless of topology optimization.

After setting the input parameters, the topology optimization process can be started by clicking on the 'Optimize' button (Fig. 8(a)). In the GUI, the optimization iterations can be seen graphically by clicking on the navigation buttons, such as 'First', 'Last', 'Previous' and 'Next'. In the command line interface (CLI) (Windows: command prompt, Linux: terminal), compliance and the maximum relative density change of each iteration are printed (Fig. 8(b)).

Compliance plot and the maximum relative density change plot over iterations can be found under the 'Analysis' menu item (Fig. 9



(a)



```
Optimization started.
Iter: 1 Max. change of X: 0.150 Compl: 0.000058
Iter: 2 Max. change of X: 0.150 Compl: 0.000042
Iter: 3 Max. change of X: 0.150 Compl: 0.000035
Iter: 4 Max. change of X: 0.150 Compl: 0.000030
Iter: 5 Max. change of X: 0.150 Compl: 0.000027
Iter: 6 Max. change of X: 0.150 Compl: 0.000025
Iter: 7 Max. change of X: 0.150 Compl: 0.000024
Iter: 8 Max. change of X: 0.150 Compl: 0.000023
Iter: 9 Max. change of X: 0.150 Compl: 0.000022
```

(b)

Fig. 8. (a) Control panel section in GUI, (b) Optimization iteration tracking in the CLI.

(a) and (b)). While the optimization process is running, the plot can be updated by clicking the 'Refresh' button.

The software has a mesh quality checking utility, which can be found under the 'Analysis' menu item (Fig. 9(c)). The mesh analysis results include statistics of nodes and elements, as well as an analysis of the elements' area, skewness and aspect ratio.

After the optimization process is finished, the output result is shown in terms of relative material densities. However, all mesh elements are present in the output. This is not practical for additive manufacturing or further structural analysis of the optimized structure. Therefore, insignificant elements need to be filtered out.

Filtering out elements means removing the elements from the mesh whose relative material densities are less than a threshold value. The 'Filter Elements' submenu item under the 'Postprocess' menu item opens a dialog (Fig. 10(a)) for filtering elements. Before opening the dialog, the last iteration step needs to be selected by clicking on the 'Last' button in the control panel. The threshold value can be defined in the 'Filter Threshold' input field. In our current research, we use 0.8 as the default threshold value for filtering out elements, but this can be changed from the GUI. There are two export buttons in the dialog to export the filtered mesh in INP format for structural analysis in Abaqus software or in STL format for 3D printing.

After setting the filter threshold, clicking the 'Filter' button will trigger the filtering process and generate a filtered mesh. The filtered mesh can then be exported by clicking on the export buttons. A filtered mesh is shown in Fig. 10(b). The elements whose relative material densities are less than 0.8 are eliminated from the design mesh. The node and element numbering is also updated. This renumbering process is necessary because removing elements from the mesh breaks the continuity of the element and node numbering. If the mesh is not renumbered, the FEA code will raise errors.

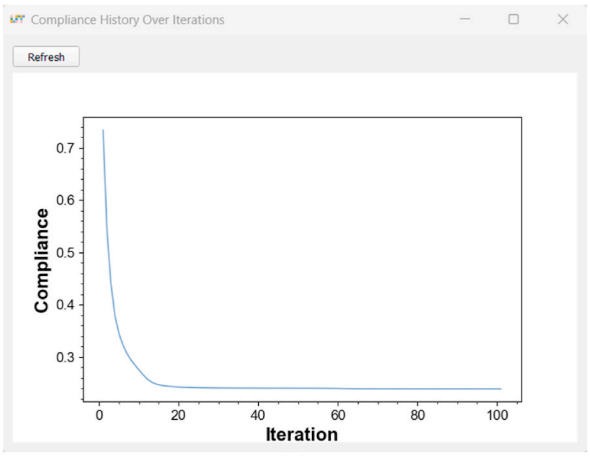
An example of the renumbering of nodes and elements during the filtering process is illustrated in Fig. 11. The elements numbered 11, 12, 13, 15, 16, 17, 18, 20, 29 and 30 are removed from the mesh (Fig. 11(a) and (b) show before and after removal respectively). This means that the element numbers 14 and 19 are renumbered to 11 and 12, respectively. In the top row, the element numbering starts from 13. The node numbering is also updated to reflect the changes in the element numbering.

6. Examples

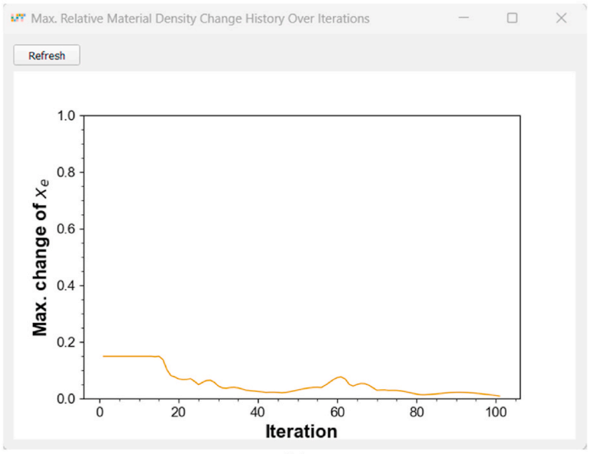
6.1. *Topology optimization of MBB-beam*

A half MBB-beam structure is shown in Fig. 12(a). It has a width of 60 units, a height of 20 units and a thickness of 1 unit. A symmetry boundary condition is applied along the left edge of the structure i.e., zero displacement boundary condition is applied along the x-axis. This makes the structure behave like a full MBB-beam. A zero-displacement boundary condition is also applied along the y-axis at the bottom-right corner. A vertical load of 1 unit is applied at the top-left corner of the half MBB-beam. Young's modulus E and Poisson's ratio ν of the material are taken as 1 unit and 0.3, respectively. The volume fraction f is set to 0.3.

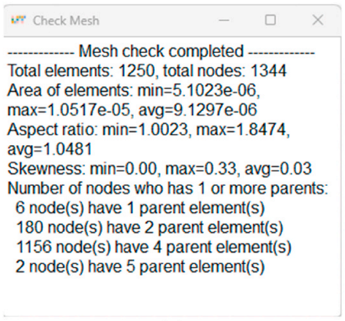
Topology optimization is conducted for two types of meshes: structured and unstructured. First, a Python script is written to generate a structured mesh for the half MBB-beam. The mesh consists of 1281 nodes and 1200 square elements, as shown in Fig. 12(b). Each element has a length of 1 unit. Second, Abaqus/CAE 6.13 is used to generate an unstructured mesh for the half MBB-beam. The domain is discretized into 1533 nodes and 1452 quadrilateral elements, as shown in Fig. 12(c). The minimum, maximum and average aspect ratios of the elements are 1.002, 2.1761 and 1.2311, respectively. The elements have maximum and average skewness of 0.34 and 0.06, respectively.



(a)



(b)

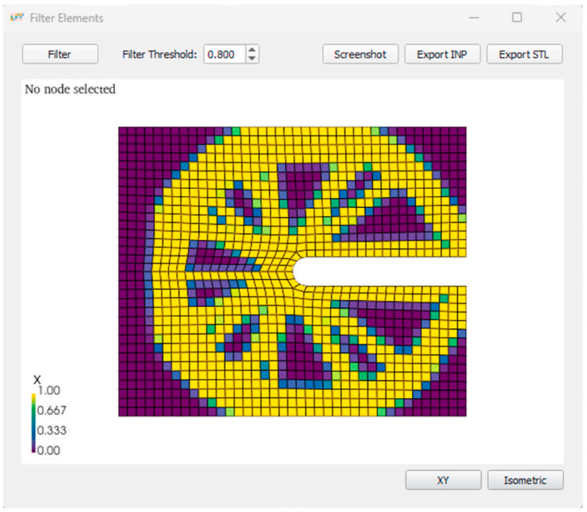


```
------------- Mesh check completed -------------
Total elements: 1250, total nodes: 1344
Area of elements: min=5.1023e-06,
max=1.0517e-05, avg=9.1297e-06
Aspect ratio: min=1.0023, max=1.8474,
avg=1.0481
Skewness: min=0.00, max=0.33, avg=0.03
Number of nodes who has 1 or more parents:
  6 node(s) have 1 parent element(s)
  180 node(s) have 2 parent element(s)
  1156 node(s) have 4 parent element(s)
  2 node(s) have 5 parent element(s)
```

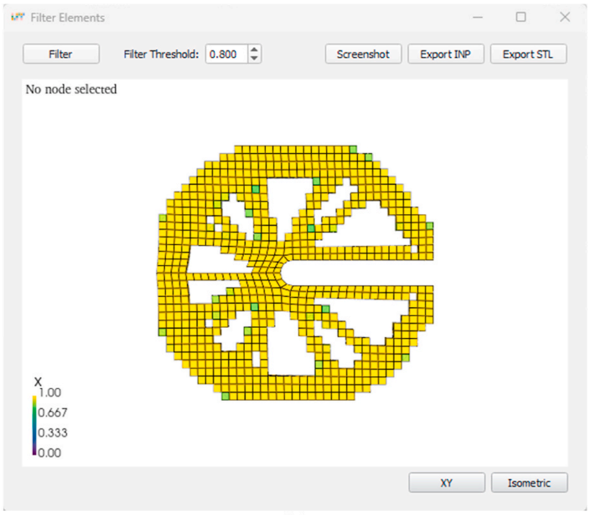
(c)

Fig. 9. (a) Compliance vs. iteration plot, (b) Maximum relative material density change vs. iteration plot, (c) Mesh analysis information.

Fig. 13 shows the final output of the topology optimization of MBB-beam for different cases. Fig. 13(a) and (b) are the results of current research for structured and unstructured meshes, respectively. Fig. 13(c) is the result of the 99-line MATLAB code [32] with the same design domain, loads and boundary conditions. In Fig. 13(a) and (b), yellow indicates a relative material density 1, and violet indicates a relative material density 0. In-between densities are shown in the color bar. In Fig. 13(c), black corresponds to relative density 1, and white corresponds to relative density 0. Fig. 13(a), which uses a structured mesh, produces the same result as Fig. 13(c). Fig. 13(b), which uses an unstructured mesh, has some scattered elements compared to Fig. 13(c), but the overall shape of the optimized structure is quite similar.



(a)



(b)

Fig. 10. (a) Filter elements and export dialog, (b) Filtered mesh after removing elements with relative material densities less than a threshold value.

6.2. Topology optimization of C-clip

A rectangular homogeneous steel structure with dimensions of 120 mm in length, 100 mm in width and 3 mm in thickness is selected as the design domain (Fig. 14(a)). A U-shaped notch runs from the right-middle of the structure to the left, with a parallel length of 55 mm, width of 10 mm and half-circle radius of 5 mm. The Young's modulus E and Poisson's ratio of the material ν are 2×10^{11} Pa and 0.3, respectively. Two static point loads of 1125 N are applied to the corners of the U-shaped notch. One is on the right-top, directed upward, and the other is on the right-bottom, directed downward. The tip of the U-notch is constrained to zero displacement in both the x- and y-directions, and the left-middle point of the structure is constrained to zero displacement in the ydirection. The volume fraction f is 0.6. The mesh of the design domain is generated using the Abaqus/CAE 6.13 meshing tool (Fig. 14(b)). The mesh has a total of 1344 nodes and 1250 elements. The minimum, maximum and average aspect ratios of the elements are 1.0023, 1.8474 and 1.0481, respectively. The maximum and average skewness of the elements are 0.33 and 0.03, respectively.

Fig. 15(a–i) shows the relative material densities of the elements in significant optimization iterations. In the initial iterations, the elements' relative material densities change drastically. However, after 15 iterations, the changes in the elements' relative material densities tend to minimize (although there are spikes in the curve, the overall trend is decreasing), as shown in Fig. 16(a).

In the first optimization iteration, compliance of the structure is found to be 0.7337. In subsequent iterations, the compliance decreases exponentially, as shown in Fig. 16(b). In the 101st iteration, the compliance of the structure reaches 0.2389. The maximum relative density change of the elements reaches 0.01 (1%), which is the threshold for finishing the optimization. Therefore, the

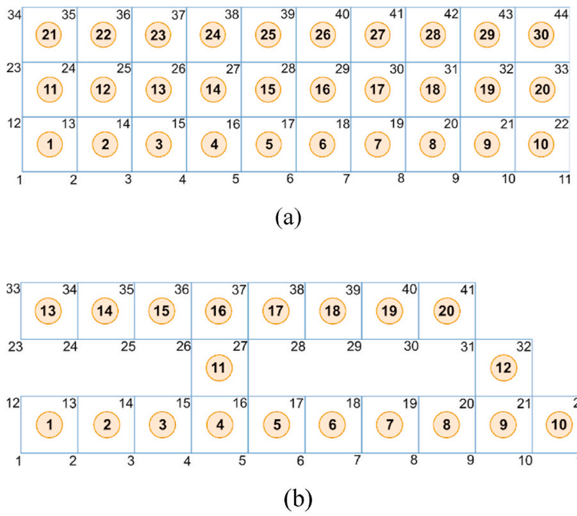


Fig. 11. Renumbering of nodes and elements (a) Before filtering out, (b) After filtering out.

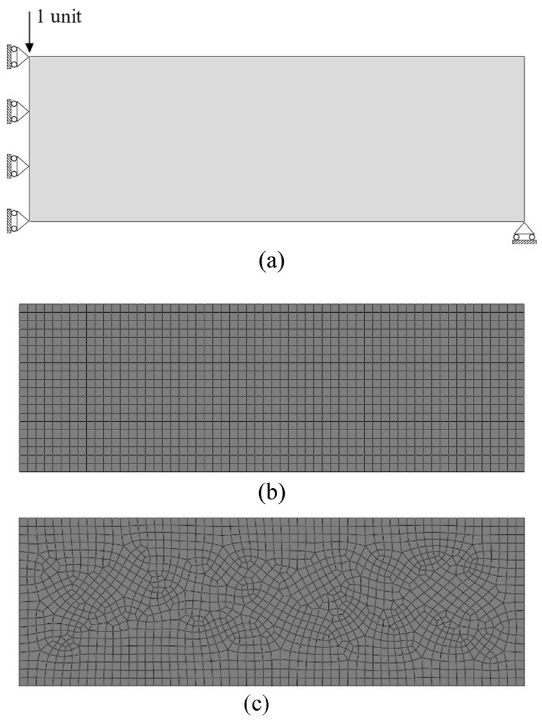


Fig. 12. MBB-beam (a) Case setup with loads and boundary conditions. $E = 1\ unit,\ \ \nu = 0.3,\ \ f = 0.3$, (b) Meshing with square Q4 elements, (c) Meshing with Q4 elements (unstructured).

optimization loop is converged. The optimization took 52.14 s.

Fig. 15(i) shows that the elements' relative densities are aggregated in certain areas and morphed into a C-clip shape.

Once the optimization is complete, the shape of the output is defined by the relative material densities. Where the relative material densities are 1 or close to 1, it can be said that material is present in those areas. However, below a certain level (i.e., less than 0.8), the material can be assumed to be absent in the corresponding elements.

Fig. 17 compares the topology optimization output from the current research (Fig. 17(a)) and Abaqus (Fig. 17(b)). It is to be noted that Abaqus uses wall smoothing while filtering (although it does not re-mesh, it simply separates regions by contours). This smoothing feature is not yet implemented in the current research. Therefore, there is room for further improvement in this area.

Fig. 18 compares the time taken for the optimization process by the current research and Abaqus. The current research takes 52.14 s

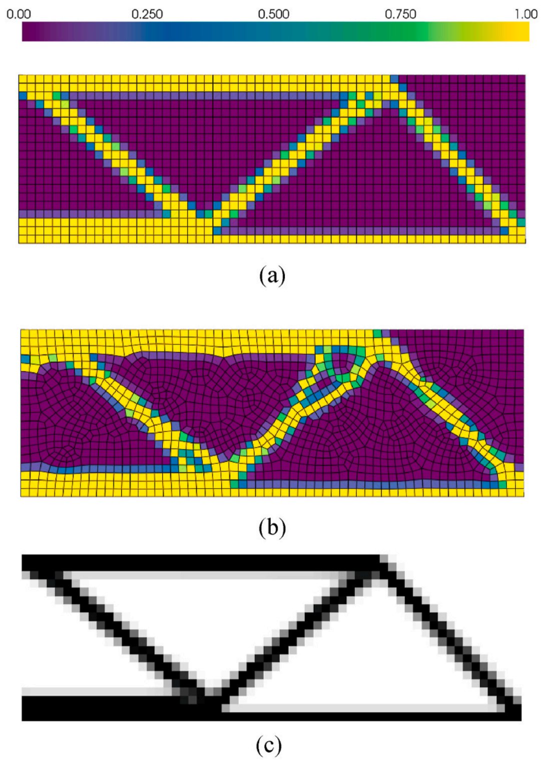


Fig. 13. Topology optimization results comparison with 99-line MATLAB program (a) With structured mesh, (b) With unstructured mesh, (c) From 99-line MATLAB program.

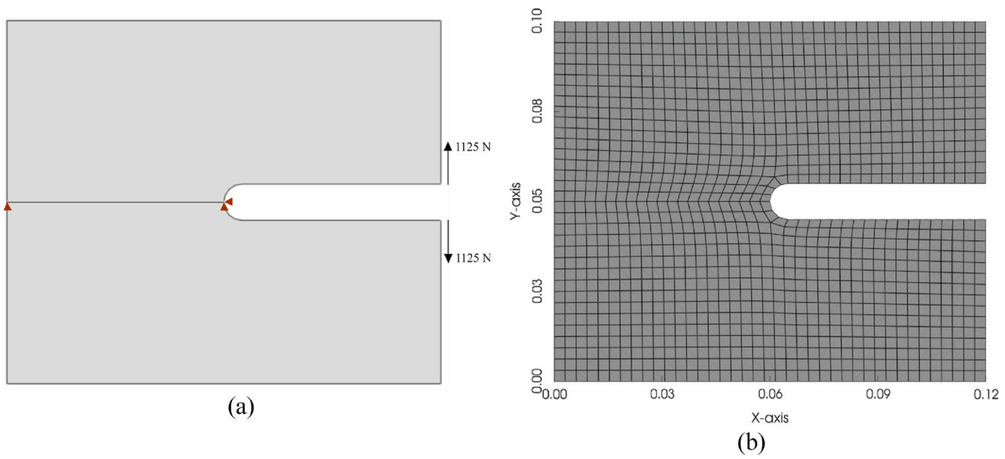


Fig. 14. C-clip (a) Case setup with loads and boundary conditions. $E = 2 \times 10^{11}\ Pa,\ \ \nu = 0.3,\ \ f = 0.6$, (b) Meshing with quadrilateral elements.

to converge, while Abaqus takes 21 min and 8 s. This represents a significant improvement in terms of computational time. The hardware configuration used was:

- 12th Gen Intel(R) Core (TM) i5-12500 up to 4.60 GHz
- 8 GB DDR4 RAM
- 240 GB SATA SSD

This comparison may not be 100% accurate, as other processes running on the machine may consume more or less computational power in different scenarios. However, for the current research, the computer environment was carefully set up to minimize the impact of these other processes. Therefore, the time comparison is very close to reality.

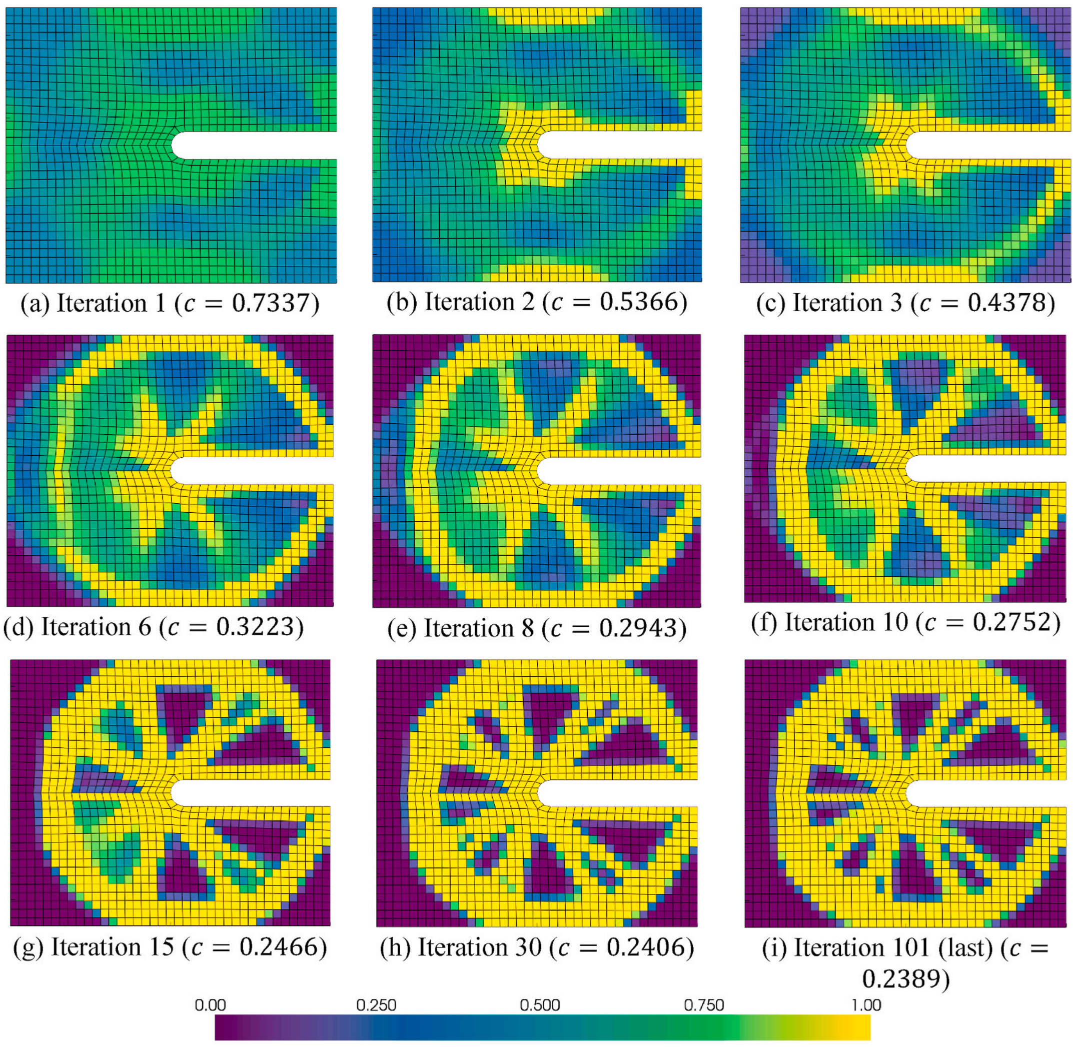

(a) Iteration 1 (c = 0.7337)
(b) Iteration 2 (c = 0.5366)
(c) Iteration 3 (c = 0.4378)
(d) Iteration 6 (c = 0.3223)
(e) Iteration 8 (c = 0.2943)
(f) Iteration 10 (c = 0.2752)
(g) Iteration 15 (c = 0.2466)
(h) Iteration 30 (c = 0.2406)
(i) Iteration 101 (last) (c = 0.2389)
0.00
0.250
0.500
0.750
1.00

Fig. 15. C-clip topology optimization (a–i) Some significant iterations.

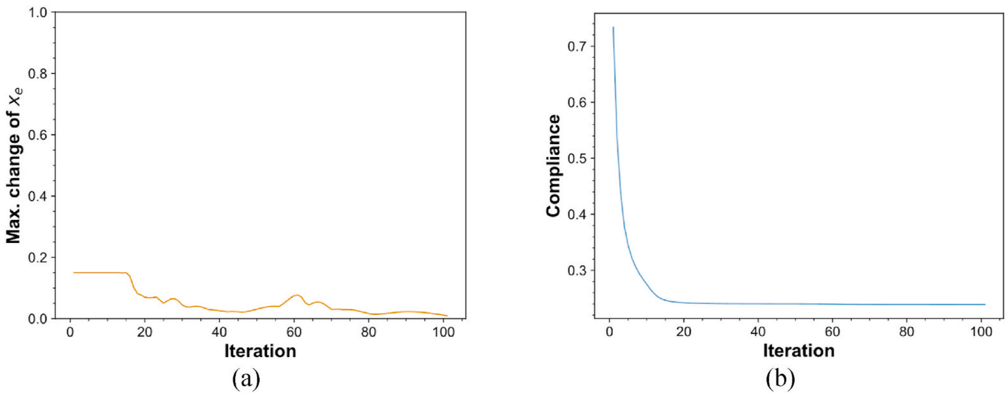


Fig. 16. C-clip optimization (a) Maximum relative material density change vs. iteration plot, (b) Compliance vs. iteration plot.

The optimized C-clip is exported in Abaqus (INP) for different volume fractions f and the Von Mises stress is analyzed in Abaqus software. The other parameters remain the same. The Von Mises stress is used to determine the onset of failure in isotropic and ductile materials. The Von Mises stress σ_{VM} for a plane stress element can be represented as follows by Equation (13) [38]:

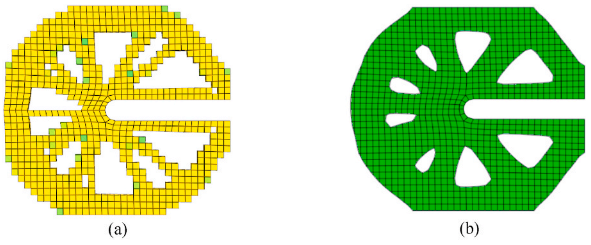


Fig. 17. C-clip (a) Filtered topology optimization output, (b) Output from Abaqus with identical case setup.

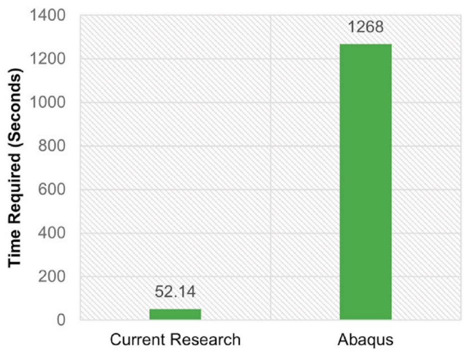


Fig. 18. Comparison of topology optimization computation time of C-clip: current research vs. Abaqus.

$$\sigma_{VM} = \sqrt{(\sigma_x + \sigma_y)^2 - 3(\sigma_x\sigma_y - \tau_{xy}{}^2)} \tag{13}$$

Table 1 shows the maximum Von Mises stress values for different volume fractions. For volume fractions of 0.4 and 0.5, the maximum Von Mises stresses exceed the yield stress (252 MPa) of the material. These structures are not safe. For the remaining volume fractions, the stress values do not exceed the material's yield stress. So, these structures are safe.

However, the structure should also be optimal in terms of material reduction. A volume fraction of 0.6 has a maximum material reduction of 40% while still being within the safe stress limit (Fig. 19). Therefore, this structure is optimal.

6.3. *Topology optimization of J-hook*

The design domain of the J-hook is shown in Fig. 20(a). The bounding box of the domain is 275 mm wide and 430 mm high. The thickness is 5 mm. The hole radius is 40 mm. The concave region has a half-circle radius of 50 mm. Three point-loads of 1200 N are applied to the structure. Young's modulus E and Poisson's ratio of the material ν are 2×10^{11} Pa and 0.3, respectively. The inner wall of the hole is constrained in the x- and y-directions. The volume fraction f is 0.5. The mesh is generated using the Abaqus/CAE 6.13 meshing tool (Fig. 20(b)). The mesh has a total of 2393 nodes and 2186 elements. The minimum, maximum and avarage aspect ratios of the elements are 1.0014, 3.4731 and 1.3612, respectively. The maximum and average skewness of the elements are 0.48 and 0.02, respectively.

To retain the J-hook shape, the walls are kept frozen in the design domain. Fig. 21 shows the wall elements that are kept frozen. In each optimization iteration, the relative material densities in these wall elements are always kept at a maximum. To achieve this, first the compliance sensitivities of all elements are calculated, as shown in Equation (9). Then, the maximum calculated compliance sensitivity is tracked, and the calculated compliance sensitivity of the frozen elements are overwrited with this maximum value, as shown in Equation (14). This ensures that the relative material densities of the wall elements are always kept at a maximum, which helps to retain the J-hook shape.

Table 1
Finding the optimal C-clip structure by analyzing Von Mises stress.

Volume Fraction	Material Reduction (%)	Max. Von Mises Stress (MPa)	Yield Stress Comparison	Comment
0.4 (40%)	60%	381.4	>252 MPa	Not safe
0.5 (50%)	50%	258.1	>252 MPa	Not safe
0.6 (60%)	40%	249.3	<252 MPa	Safe and optimal
0.7 (70%)	30%	248.5	<252 MPa	Safe
0.8 (80%)	20%	249	<252 MPa	Safe
0.9 (90%)	10%	249	<252 MPa	Safe

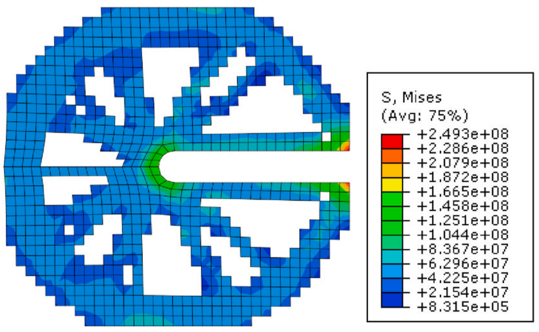


Fig. 19. Von Mises stress analysis of the optimal C-clip structure at volume fraction $f = 0.6$, maximum Von Mises stress = 249.3 MPa.

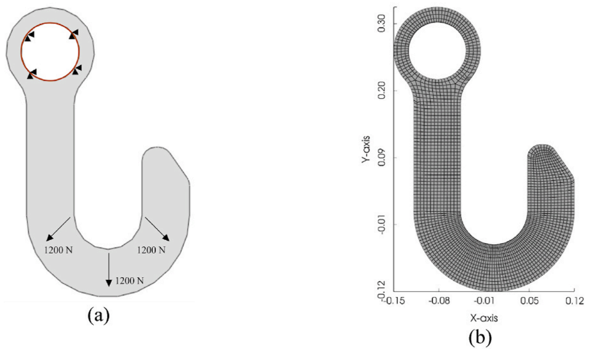


Fig. 20. J-hook (a) Case setup with loads and boundary conditions. $E = 2 \times 10^{11}\ Pa,\ \ \nu = 0.3,\ \ f = 0.5$, (b) Meshing with quadrilateral elements.

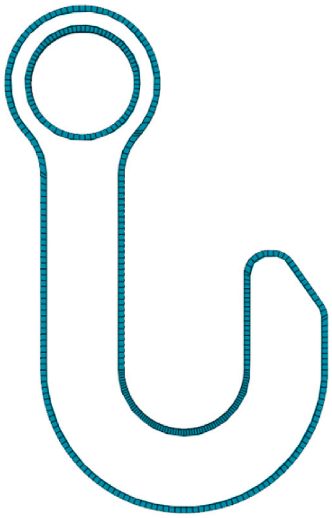


Fig. 21. J-hook wall elements (frozen).

$$\left({}^{\partial c}/_{\partial x_e} \right)_{freeze} = maximum\left(\left| {}^{\partial \boldsymbol{c}}/_{\partial \boldsymbol{x}_e} \right| \right) \tag{14}$$

Fig. 22(a–i) shows some significant optimization iterations graphically. The optimization converges in the 107th iteration (Fig. 22 (i)) when the maximum relative density change of elements reaches 0.009, which is lower than the finish threshold 0.01. The final compliance of the structure is 1.2202. Fig. 23 shows the maximum relative material density change and compliance data over iterations. It is observed from the last iteration that the wall elements are not removed. They are forcefully kept during optimization iterations.

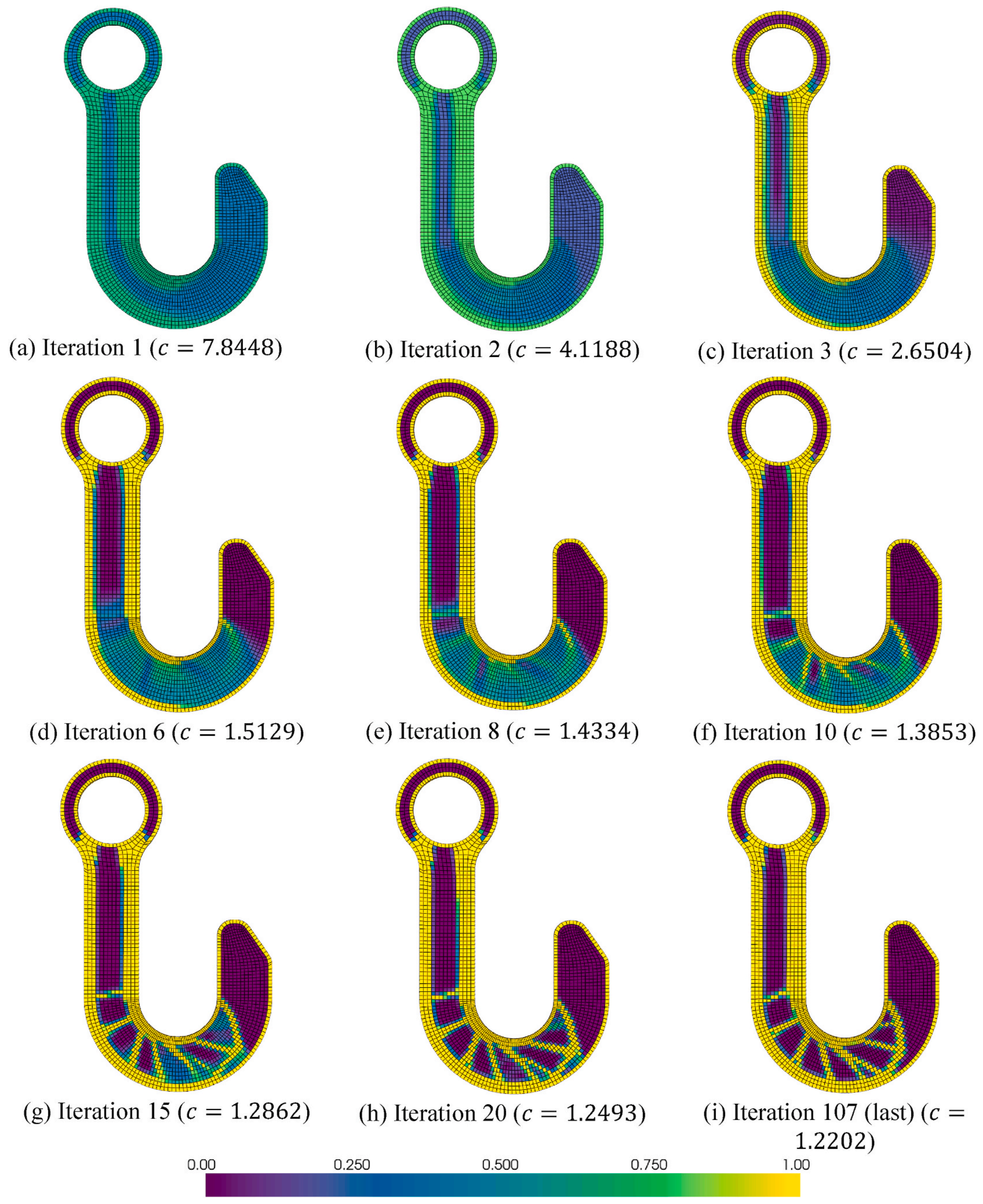

(a) Iteration 1 (c = 7.8448)
(b) Iteration 2 (c = 4.1188)
(c) Iteration 3 (c = 2.6504)
(d) Iteration 6 (c = 1.5129)
(e) Iteration 8 (c = 1.4334)
(f) Iteration 10 (c = 1.3853)
(g) Iteration 15 (c = 1.2862)
(h) Iteration 20 (c = 1.2493)
(i) Iteration 107 (last) (c = 1.2202)
0.00
0.250
0.500
0.750
1.00

Fig. 22. J-hook topology optimization (a–i) Some significant iterations.

Fig. 24 compares the topology optimization results of the J-hook using the current research (Fig. 24(a)) and Abaqus software (Fig. 24(b)). The case setup is identical for both studies. Wall elements are kept frozen for this J-hook case.

The current research takes 2 min and 8 s (128.34 s) to converge, while Abaqus takes 31 min and 1 s (1871 s) in the optimization process (Fig. 25).

Table 2 shows the maximum Von Mises stress values for different volume fractions. The structure is not safe for a volume fraction of 0.4, as the maximum Von Mises stress exceeds the yield stress (252 MPa) of the material. Other configurations are safe. Of these, the volume fraction of 0.5 has a maximum material reduction of 50% (Fig. 26). Therefore, this is the optimal structure.

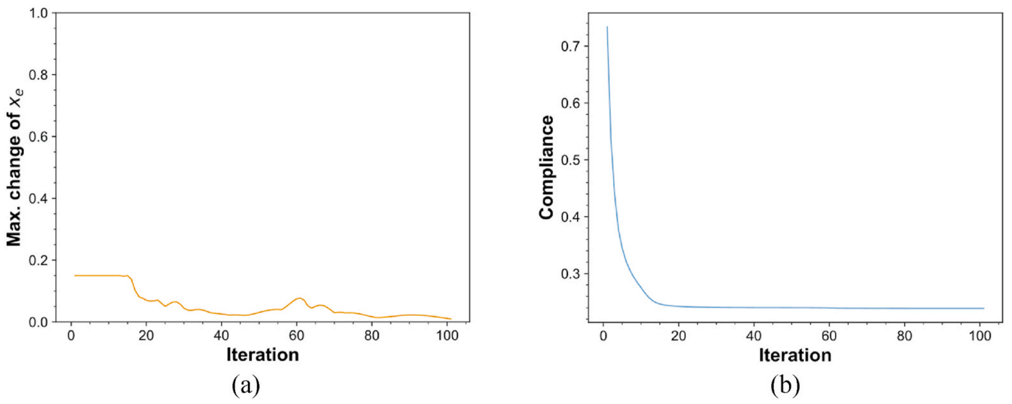


Fig. 23. J-hook optimization (a) Maximum relative material density change vs. iteration plot, (b) Compliance vs. iteration plot.

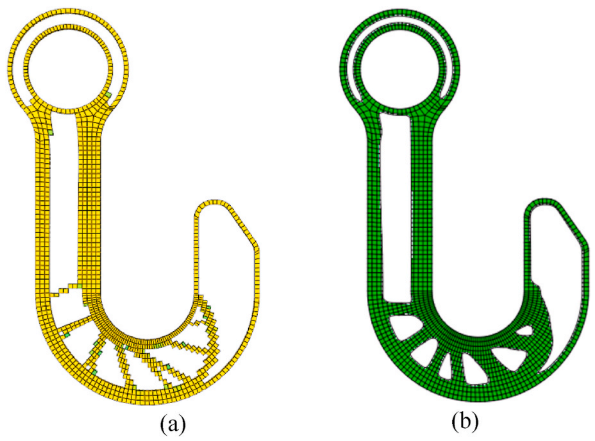


Fig. 24. J-hook (a) Filtered topology optimization output, (b) Abaqus output with identical case setup.

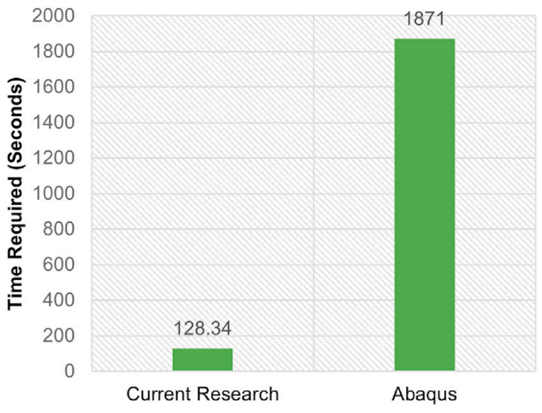


Fig. 25. Comparison of topology optimization computation time of J-hook: current research vs. Abaqus.

6.4. *Topology optimization of derrick*

Fig. 27(a) shows the design domain of the derrick. The thickness of the structure is 25 mm. Two holes are located at the top of the derrick to carry loads. In this setup, a 12000 N force is applied to the outer hole, and an 18000 N force is applied to the inner hole. The bottom of the derrick is constrained in the x- and y-directions. Young's modulus E and Poisson's ratio of the material ν are 2×10^{11} Pa and 0.3, respectively. The volume fraction f is 0.6. Fig. 27(b) shows the mesh of the derrick's design domain. The mesh was generated

Table 2
Finding optimal J-hook structure by analyzing Von Mises stress.

Volume Fraction	Material Reduction (%)	Max. Von Mises Stress (MPa)	Yield Stress Comparison	Comment
0.4 (40%)	60%	664	>252 MPa	Not safe
0.5 (50%)	50%	145.9	<252 MPa	Safe and optimal
0.6 (60%)	40%	169.2	<252 MPa	Safe
0.7 (70%)	30%	137.2	<252 MPa	Safe
0.8 (80%)	20%	127.5	<252 MPa	Safe
0.9 (90%)	10%	119.6	<252 MPa	Safe

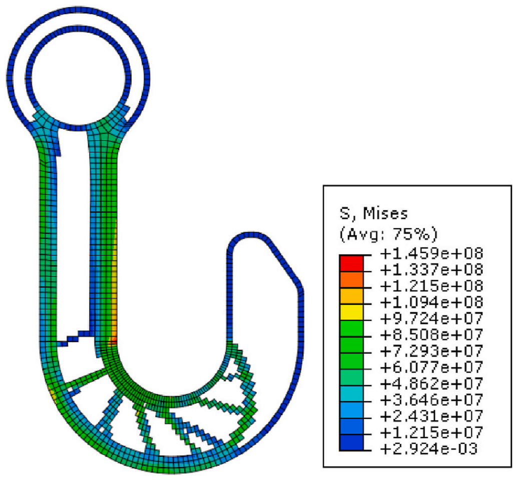


Fig. 26. Von Mises stress analysis of J-hook at optimal volume fraction $f = 0.5$, maximum Von Mises stress = 145.9 MPa.

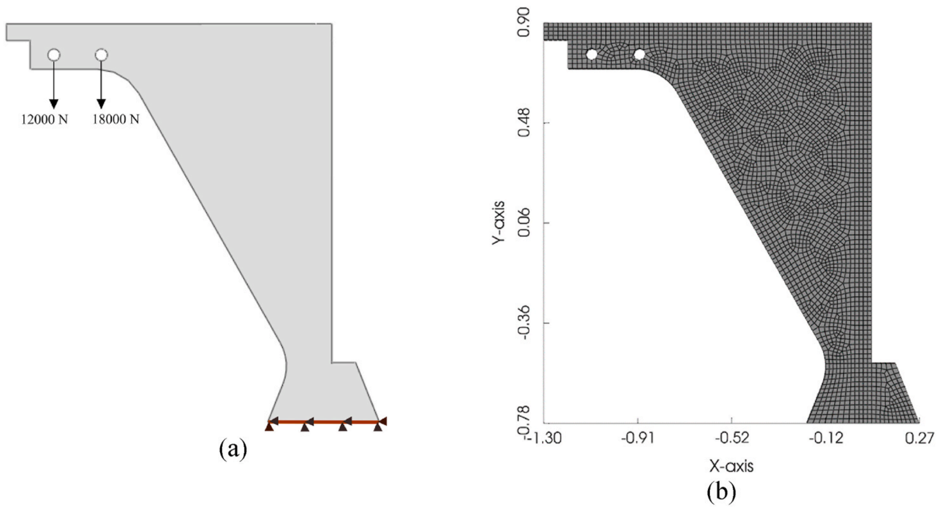


Fig. 27. Derrick (a) Case setup with loads and boundary conditions. $E = 2 \times 10^{11}$ Pa, $\nu = 0.3$, $f = 0.6$, (b) Meshing with quadrilateral elements.

using Abaqus/CAE 6.13. The total numbers of nodes and elements are 3117 and 2960, respectively. The minimum, maximum and average aspect ratio of elements are 1, 2.1340 and 1.2308, respectively. The elements have maximum and average skewness of 0.35 and 0.06, respectively.

Elements adjacent to the walls of the derrick's loading holes need to be frozen so that they are never removed during topology optimization. In Fig. 28, the frozen elements are shown in a different color.

Some significant optimization iterations are shown graphically in Fig. 29(a–i). Fig. 30(a) shows the maximum change in relative material densities over iterations. Fig. 30(b) shows the compliance vs. iteration curve. The optimization converged in the 115th iteration (Fig. 29(i)), with a final compliance of 89.1967.

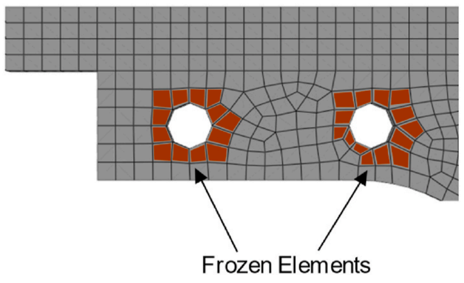


Fig. 28. Frozen elements of derrick hole walls in topology optimization.

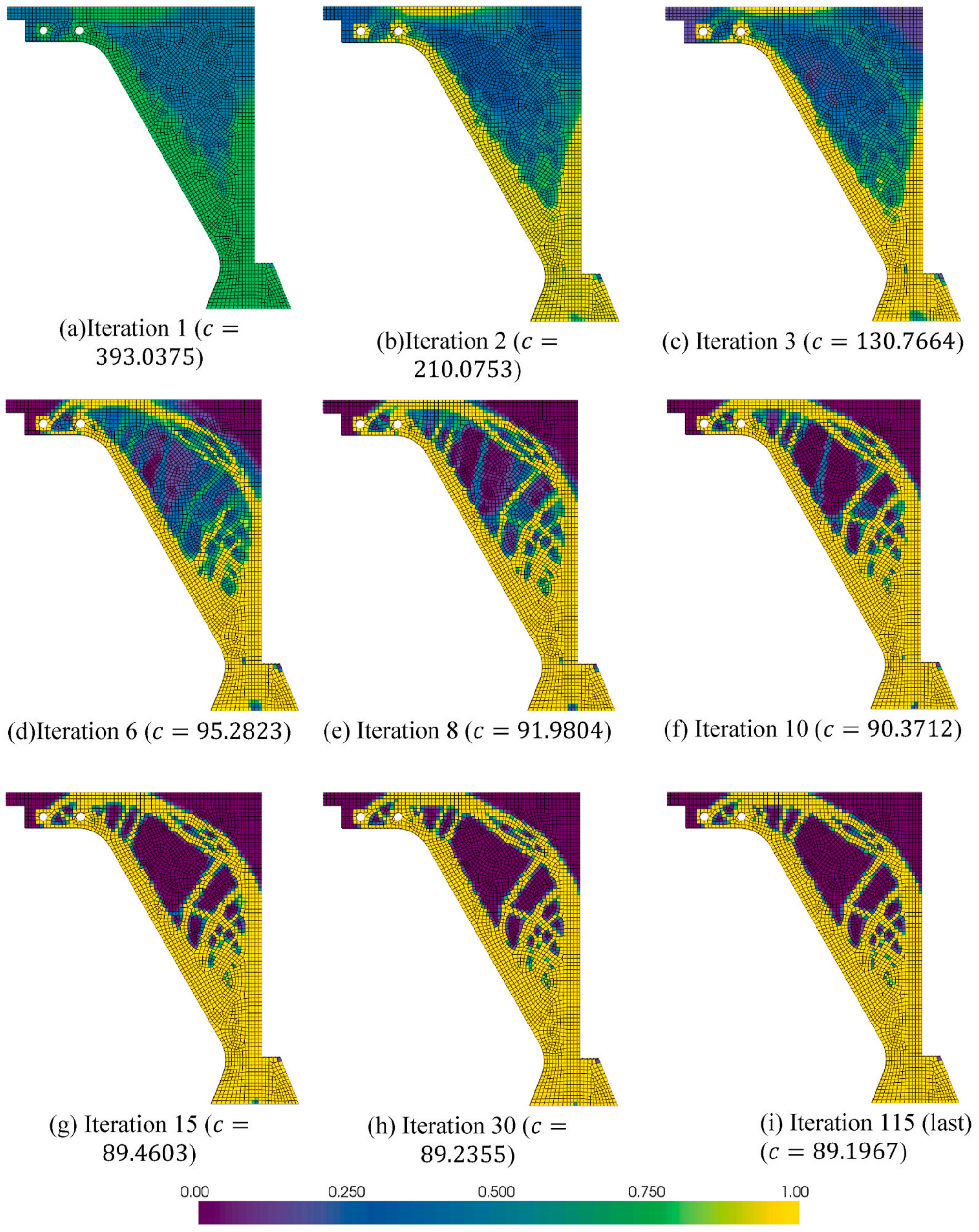

(a)Iteration 1 ($c =$ 393.0375)
(b)Iteration 2 ($c =$ 210.0753)
(c) Iteration 3 ($c = 130.7664$)
(d)Iteration 6 ($c = 95.2823$)
(e) Iteration 8 ($c = 91.9804$)
(f) Iteration 10 ($c = 90.3712$)
(g) Iteration 15 ($c =$ 89.4603)
(h) Iteration 30 ($c =$ 89.2355)
(i) Iteration 115 (last) ($c = 89.1967$)
0.00
0.250
0.500
0.750
1.00

Fig. 29. Derrick topology optimization (a–i) Some significant iterations.

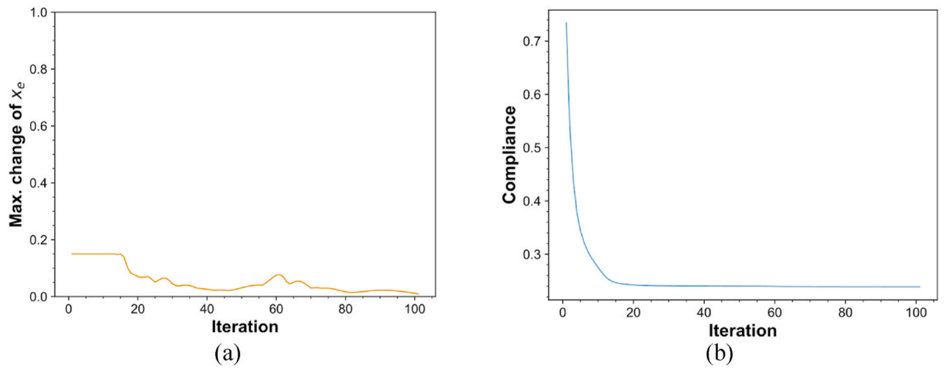


Fig. 30. Derrick optimization (a) Maximum relative material density change vs. iteration plot, (b) Compliance vs. iteration plot.

Fig. 31 shows the topology optimization comparison of derrick with Abaqus software. The case setup is identical for both current research (Fig. 31(a)) and Abaqus (Fig. 31(b)). The Abaqus software takes 23 min and 51 s to finish the optimization, while the current research takes 4 min and 23 s to converge (Fig. 32).

Table 3 shows the maximum Von Mises stress values for different volume fractions. Among the safe volume fraction configurations, the volume fraction of 0.5 has the maximum material reduction, as shown in Fig. 33. Therefore, it is the safe and optimal configuration.

7. Conclusions

We have developed a general-purpose topology optimization tool named UnTop2D for two-dimensional problems using object-oriented Python programming. The following conclusions can be made from the current research:

- UnTop2D can import any structured or unstructured Q4 mesh and optimize the design domain with improved computation time and accuracy. This has been verified by comparing the results with the renowned professional software Abaqus.
- UnTop2D also features a GUI that makes it easy to set boundary conditions, loads and optimization parameters. The visualization panel shows the optimization results with colors.
- The developed program allows users to designate specific elements as 'frozen'. This can be useful for preventing the removal of essential elements during optimization.
- The software incorporates a filtering technique to eliminate insignificant elements from the structure after optimization. This makes it easier to export the optimized structure to other software for further analysis or additive manufacturing.

There have been many advancements in the field of topology optimization. Many commercial software packages have developed advanced algorithms, but these algorithms are not always available for research purposes. In this paper, we incorporate object-oriented programming methodologies, which make it easier to optimize general design domains for practical engineering applications. We believe that this work will encourage other researchers to build upon our findings.

Replication of results

The paper provides a detailed discussion of the algorithm and the coding procedure. Corresponding examples are also given with figures. We expect a graduate student to reproduce a similar code from the provided information. The readers are welcome to contact the authors for any queries.

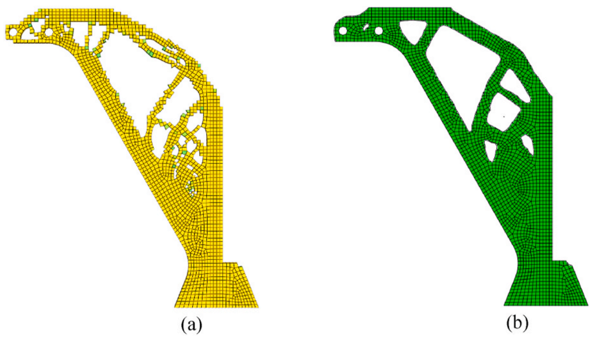


Fig. 31. Derrick (a) Filtered topology optimization output, (b) Abaqus output with identical case setup.

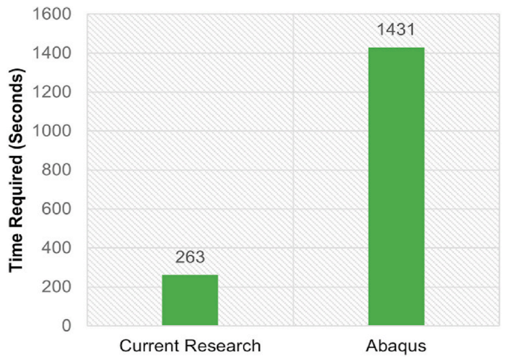


Fig. 32. Comparison of topology optimization computation time of derrick: current research vs. Abaqus.

Table 3
Finding optimal derrick structure by analyzing Von Mises stress.

Volume Fraction	Material Reduction (%)	Max. Von Mises Stress (MPa)	Yield Stress Comparison	Comment
0.4 (40%)	60%	545.2	>252 MPa	Not safe
0.5 (50%)	50%	190.1	<252 MPa	Safe and optimal
0.6 (60%)	40%	190.1	<252 MPa	Safe
0.7 (70%)	30%	190.1	<252 MPa	Safe
0.8 (80%)	20%	190.1	<252 MPa	Safe
0.9 (90%)	10%	190.1	<252 MPa	Safe

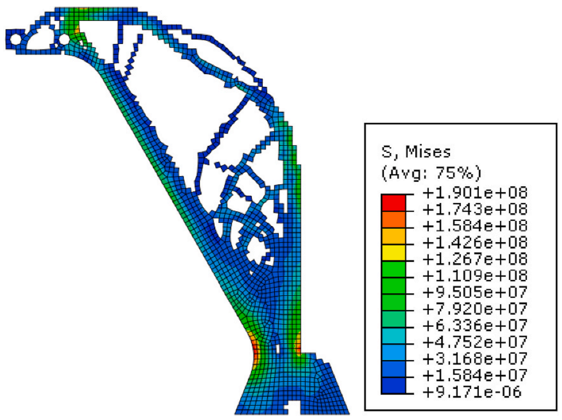


Fig. 33. Von Mises stress analysis of derrick at optimal volume fraction $f = 0.5$, maximum Von Mises stress = 190.1 MPa.

Data availability statement

All data are generated internally.

Additional information

No additional information is available for this paper.

CRediT authorship contribution statement

Md Shahidul Islam: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Ali Zulkar Nayem:** Conceptualization, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **K.N. Hoque:** Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge the high-performance computing facility at the Marine Structure Simulation Lab of the Department of Naval Architecture and Marine Engineering of Bangladesh University of Engineering and Technology. This facility is developed by University Grants Commission of Bangladesh fund HEQEP sub-project CP#3131.

References

[1] X. Zhang, B. Zhu, Topology Optimization of Compliant Mechanisms, Springer, 2017.
[2] W.J.P. Casas, E.M. Cesconeto, E.d.S. Lisboa, J.B.D. Moreira, J.E. Medeiros, T.S. Ribeiro, Topology optimization of an aircraft component as a fluid-structure system with unstructured mesh, in: Aerospace Technology Congress, Solna, Stockholm, October 2016, pp. 11–12.
[3] V.J. Challis, J.K. Guest, Level set topology optimization of fluids in Stokes flow, Int. J. Numer. Methods Eng. 79 (2009) 1284–1308.
[4] H. Li, T. Kondoh, P. Jolivet, N. Nakayama, K. Furuta, H. Zhang, B. Zhu, K. Izui, S. Nishiwaki, Topology optimization for lift–drag problems incorporated with distributed unstructured mesh adaptation, Struct. Multidiscip. Optim. 65 (2022).
[5] M.B. Duhring, J.S. Jensen, O. Sigmund, Acoustic design by topology optimization, J. Sound Vib. 317 (3–5) (2008) 557–575.
[6] F. Lucchini, R. Torchio, V. Cirimele, P. Alotto, P. Bettini, Topology optimization for electromagnetics: a survey, IEEE Access 10 (2022) 98593–98611.
[7] G. Fujii, H. Watanabe, T. Yamada, T. Ueta, M. Mizuno, Level set based topology optimization for optical cloaks, Appl. Phys. Lett. 102 (25) (2013).
[8] R.E. Christiansen, O. Sigmund, Inverse design in photonics by topology optimization: tutorial, J. Opt. Soc. Am. B 38 (2) (2021) 496–509.
[9] G.I.N. Rozvany, A critical review of established methods of structural topology optimization, Struct. Multidiscip. Optim. 37 (2009) 217–237.
[10] O. Sigmund, K. Maute, Topology optimization approaches, Struct. Multidiscip. Optim. 48 (2013) 1031–1055.
[11] W. Dorn, R. Gomory, H.J. Greenberg, Automatic design of optimal structures, J. Mec. 3 (1964) 25–52.
[12] D.S. Ramrakhyani, M.I. Frecker, Hinged beam elements for the topology design of compliant mechanisms using the ground structure approach, Struct. Multidiscip. Optim. 37 (2009) 557–567.
[13] J. Zhan, X. Zhang, Topology optimization of compliant mechanisms with geometrical nonlinearities using the ground structure approach, Chin. J. Mech. Eng. 24 (2) (2011) 257–263.
[14] T. Zegard, G.H. Paulino, Grand — ground structure based topology optimization for arbitrary 2D domains using MATLAB, Struct. Multidiscip. Optim. 50 (2014) 861–882.
[15] X. Zhang, S. Maheshwari, A.S. Ramos, G.H. Paulino, Macroelement and macropatch approaches to structural topology optimization using the ground structure method, J. Struct. Eng. 142 (11) (2016).
[16] X. Zhang, A.S. Ramos, G.H. Paulino, Material nonlinear topology optimization using the ground structure method with a discrete filtering scheme, Struct. Multidiscip. Optim. 55 (2017) 2045–2072.
[17] Y.M. Xie, G.P. Steven, Basic evolutionary structural optimization, in: Evolutionary Structural Optimization, Springer, 1997, pp. 12–29.
[18] P. Tanskanen, The evolutionary structural optimization method: theoretical aspects, Comput. Methods Appl. Mech. Eng. 191 (2002) 5485–5498.
[19] G. He, X. Huang, H. Wang, G. Li, Topology optimization of periodic structures using BESO based on unstructured design points, Struct. Multidiscip. Optim. 53 (2022) 271–275.
[20] G. Azamirad, B. Arezoo, Structural design of stamping die components using bi-directional evolutionary structural optimization method, Int. J. Adv. Manuf. Technol. 87 (2016) 969–979.
[21] L. Xia, Q. Xia, X. Huang, Y.M. Xie, Bi-directional evolutionary structural optimization on advanced structures and materials: a comprehensive review, Arch. Comput. Methods Eng. 25 (2018) 437–478.
[22] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2) (2001) 463–502.
[23] J. Cui, D. Wang, Q. Shi, Structural topology design of container ship based on knowledge-based engineering and level set method, China Ocean Eng. 29 (2015) 551–564.
[24] P.D. Dunning, B.K. Stanford, H.A. Kim, Coupled aerostructural topology optimization using a level set method for 3D aircraft wings, Struct. Multidiscip. Optim. 51 (2015) 1113–1132.
[25] P. Wei, Y. Yang, S. Chen, M.Y. Wang, A study on basis functions of the parameterized level set method for topology optimization of continuums, ASME. J. Mech. Des. 143 (4) (2020).
[26] S. Osher, J. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[27] M.P. Bendsoe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Comput. Methods Appl. Mech. Eng. 71 (2) (1988) 197–224.
[28] B. Hassani, E. Hinton, A review of homogenization and topology optimization I—homogenization theory for media with periodic structure, Comput. Struct. 69 (6) (1998) 707–717.
[29] G.A. Chechkin, A.L. Piatnitski, A.S. Shamaev, Homogenization: Methods and Applications, American Mathematical Soc., 2007.
[30] M.P. Bendsoe, Optimal shape design as a material distribution problem, Struct. Optim. 1 (1989) 193–202.
[31] G.I.N. Rozvany, M. Zhou, T. Birker, Generalized shape optimization without homogenization, Struct. Optim. 4 (1992) 250–252.
[32] O. Sigmund, A 99 line topology optimization code written in Matlab, Struct. Multidiscip. Optim. 21 (2001) 120–127.
[33] E. Andreassen, A. Clausen, M. Schevenels, B. Lazarov, O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code, Struct. Multidiscip. Optim. 43 (1) (2011) 1–16.
[34] C. Talischi, G.H. Paulino, A. Pereira, I.F.M. Menezes, PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes, Struct. Multidiscip. Optim. 45 (2012) 329–357.
[35] B. Bochenek, K. Tajs-Zielinska, Gotica - generation of optimal topologies by irregular cellular automata, Struct. Multidiscip. Optim. 55 (2017) 1989–2001.
[36] D.S.H. Lo, Finite Element Mesh Generation, CRC Press, 2015.
[37] D.L. Logan, A First Course in Finite Element Method, Thomson Learning, 2007.
[38] T. Chandrupatla, A. Belegundu, Introduction to Finite Elements in Engineering, Prentice Hall, 2012.
[39] E. Bressert, SciPy and NumPy: an Overview for Developers, O'Reilly Media, Inc., 2012.
[40] O. Sigmund, Design of Material Structures Using Topology Optimization, Department of Solid Mechanics, Technical University of Denmark, 1994. Ph.D. Thesis.
[41] B.M. Harwani, Qt5 Python GUI Programming Cookbook: Building Responsive and Powerful Cross-Platform Applications with PyQt, Packt Publishing Ltd., 2018.
[42] C.B. Sullivan, A.A. Kaszynski, PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK), J. Open Source Softw. 4 (2019).
[43] J.D. Hunter, Matplotlib: a 2D graphics environment, Comput. Sci. Eng. 9 (2007) 90–95.
[44] N. Schlomer, Meshio: Tools for Mesh Files, GitHub Repository, 2022.