



A 262-line Matlab code for the level set topology optimization based on the estimated gradient field in the body-fitted mesh

Zicheng Zhuang^{1,3} · Fengming Xu¹ · Junhong Ye¹ · Wei Tong¹ · Zeyao Chen² · Yiwei Weng¹

Received: 17 March 2024 / Revised: 8 September 2024 / Accepted: 11 September 2024
© The Author(s) 2024

Abstract

Topology optimization is an influential technique engineers and designers employ to achieve desirable material distribution within a designated domain. This educational article introduces a concise and efficient Matlab code, comprising only 262 lines, developed explicitly for the Level Set topology optimization based on the estimated Gradient Field (GFLS) in the body-fitted mesh. Unlike conventional level set methods that rely on the upwind scheme employed in the structured meshes, the proposed algorithm adopts the per-cell linear estimation of the discrete gradient vectors in the body-fitted mesh framework to obtain the velocity field and update the level set function. The Matlab code, named GFLS262, consists of a 62-line main program, 41-line finite element analysis function, and 48-line sub-functions, enabling the implementation of the GFLS method in 2D scenarios. Additionally, a 111-line function describes an improved mesh generator incorporated in the code to facilitate the generation of body-fitted meshes. The superiority of this innovative approach over the previous optimization methods with invariant meshes is demonstrated through various benchmark examples. For ease of access and further learning, the educational Matlab code is available on the website and can also be found in the Appendix section of this article.

Keywords Topology optimization · Level set method · Gradient field estimation · Body-fitted mesh · Educational Matlab code

1 Introduction

Topology optimization, a mathematical method for achieving optimized material distribution within design domains, has gained increasing popularity in the engineering and architecture communities (Dorn et al. 1964; Glowinski 1984; Goodman et al. 1986; Kikuchi et al. 1986; Lurie et al. 1982; Svanberg 1987). It aims to achieve better structural performance while satisfying various design constraints by redistributing prescribed material. Over the years, several methods have been developed to tackle this challenging

problem, including the well-known homogenization method proposed by Bendsoe and Kikuchi (1988). The homogenization method leverages the concept of homogenization theory (Bendsoe 1989; Suzuki and Kikuchi 1991), which allows for the representation of heterogeneous materials as an equivalent homogeneous material with effective properties. These methods seek to find the desirable distribution of material phases within a design domain discretized by finite elements, considering the macroscopic behavior and performance of the structure. However, the homogenization optimization method can be computationally demanding due to the requirement of solving multiple microscale problems in each iteration. The accuracy of the method is also dependent on the fidelity of the homogenization model used and the appropriate representation of material behavior. Nowadays, three popular methods in topology optimization that have shown remarkable efficacy are the Solid Isotropic Material with Penalization (SIMP) method, the Bi-directional Evolutionary Structural Optimization (BESO) method, and the level set method.

The SIMP method (Bendsoe and Sigmund 2004; Sigmund 2001; Sigmund and Maute 2013), widely recognized for its

Responsible editor: Gregoire Allaire.

✉ Yiwei Weng
yiwei.weng@polyu.edu.hk

¹ Department of Building and Real Estate, The Hong Kong Polytechnic University, Hong Kong, China

² School of Electro-Mechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China

³ Centre for Innovative Structures and Materials, School of Engineering, RMIT University, Melbourne 3001, Australia

simplicity and effectiveness, penalizes intermediate densities of material to encourage either full inclusion or complete exclusion of material in each element of the design domain. By assigning a high penalty to intermediate densities, the method drives the optimization process toward achieving either fully solid or void regions. The SIMP method has gained widespread popularity due to its ability to generate structurally efficient designs efficiently. Several versions of Matlab programs employing the SIMP method in the fixed structured mesh have been published over the past decades (Andreassen et al. 2011; Ferrari and Sigmund 2020; Liu and Tovar 2014; Sigmund 2001). The SIMP method provides a simple and intuitive framework, making it suitable for quick design iterations and preliminary studies.

The BESO method (Xie and Steven 1992; 1993; 1996), drawing inspiration from evolutionary processes, utilizes bi-directional optimization wherein both the inclusion and exclusion of material are considered simultaneously. Through an iterative procedure, the BESO method progressively removes the least significant material elements while introducing new elements to enhance the structural performance (Huang et al. 2006; Huang and Xie 2009; 2010). It has been implemented using Python code (Zuo and Xie 2015) and Matlab codes (Huang and Xie 2010; Zhuang et al. 2022a, b). The BESO method, with its evolutionary nature, excels in generating innovative designs and exploring unconventional solutions (He et al. 2023; Xiong et al. 2023).

The level set method (Osher and Sethian 1988; Sethian 1999; Sethian and Wiegmann 2000) represents the evolving geometry implicitly using a higher-dimensional level set function, widely applied in numerical analysis and image processing. In this century, the level set method has been employed in topology optimization (Allaire et al. 2002, 2004; Wang et al. 2003), where the boundaries between solid and void regions are represented by a level set function. This formulation provides a versatile framework for handling complex shape variations, including topological changes (Allaire et al. 2011; Yamada et al. 2010). The level set method enables the evolution and optimization of the level set function to obtain the desired material distribution (Li et al. 2021; Wang et al. 2022; Zhuang et al. 2021). The published Matlab codes (Challis 2010; Otomori et al. 2014; Wei et al. 2018) demonstrate level set optimization methods by solving the Hamilton–Jacobi, radial-basis, and reaction–diffusion equations, respectively. The level set method has gained significant attention in the field of structural optimization due to its ability to capture and represent complex geometries implicitly.

The level set optimization method offers a highly appealing advantage with its smooth and distinct material boundaries depicted by the zero-level contour. However, conventional practices commonly interpolate the level set function into a structured rectangular/hexahedral mesh, resulting

in zig-zag boundaries that compromise accuracy. Thus, researchers leveraged the body-fitted mesh to accurately represent the material boundaries provided by the level set optimization method (Allaire et al. 2011; 2013; 2014) and the Deformable Simplicial Complex (DSC) method (Christiansen et al. 2014). Body-fitted meshing involves creating a computational grid that conforms to the geometry of the physical system being analyzed in the last decade (Dapogny and Frey 2012; Dapogny et al. 2014; Talischi et al. 2012). This technique is particularly useful for solving problems where the geometry is complex or changing, and the solution requires a high degree of accuracy around the boundaries. Body-fitted meshes typically consist of structured or unstructured grids overlaid onto the physical domain (Baiges et al. 2019; Salazar de Troya and Tortorelli 2018; Zhang et al. 2020). The grid points are distributed across the physical domain such that they align with the boundaries, ensuring an accurate representation of features such as flow separation, boundary layer development, or material interfaces. This work generates the body-fitted mesh according to the smooth boundaries using the Delaunay triangulation and force–displacement equilibrium (Persson and Strang 2004; Zhuang et al. 2021; Zhuang et al. 2022a, b). The node positions are iteratively changed to produce the body-fitted mesh with high quality, which can be employed in level set optimization methods to solve complex problems in structural mechanics and materials science. The body-fitted meshing ensures an accurate representation of physical boundaries, while the level set topology optimization provides a flexible framework for exploring complex material layouts. This combined approach can lead to the creation of innovative and efficient structures and materials, helping to solve real-world engineering problems.

The utilization of body-fitted meshing is essential within the finite element analysis and level set optimization processes rather than merely serving as a post-processing tool for generating optimized results. This study aims to integrate body-fitted meshing with finite element analysis, sensitivity analysis, and the design variable updating procedures in level set topology optimization. However, it is crucial to acknowledge the specific limitations associated with the commonly employed upwind scheme in level set topology optimization. Primarily, the upwind scheme is limited to uniform sampling and tends to be sensitive to mesh irregularities (Museth et al. 2005). The upwind scheme and level set optimization presented in previously published Matlab code are exclusively compatible with structured rectangular meshes (Challis 2010; Wei et al. 2018). Generally, implementing the conventional upwind scheme can lead to inaccuracies and impact the smoothness of the level set function when interpolating from a structured mesh to a body-fitted mesh (van Dijk et al. 2010). The level set function may develop steep gradients in the body-fitted mesh, leading to

problems in numerical approximations using the upwind schemes (Wang et al. 2007). Meanwhile, this study adopts the bi-section method rather than the augmented method to compute the Lagrangian multiplier, ensuring quicker convergence and volume accuracy. In this context, the iterative calculation process within the upwind scheme may significantly escalate the computational costs during the optimization phase (Zhuang et al. 2024).

Thus, this paper presents an educational Matlab code that performs level set topology optimization utilizing the estimated gradient field (GFLS) within the framework of a body-fitted mesh. Building upon the 172-line code Tri-TOP172 (Zhuang et al. 2022a, b), the code GFLS262 utilizes the gradient field-based level set method rather than the BESO sensitivity ranking scheme to update design variables, maximizing the benefits of smooth boundaries. The GFLS method employs per-cell linear estimation of discrete gradient vectors, instead of the traditional upwind method, to update the level set function and design variable. It is well-known that time-dependent partial differential equations (PDEs) govern the evolution of the level set function and material densities. Through discretizing the design domain into a finite element mesh, the time-dependent issue can be transformed into a series of steady-state problems that are solvable in iterative procedures. Then, this approach allows for the step-by-step computation of the solution over a specified time interval based on the gradient field of the level set function (Azari Nejat et al. 2022; Jiang and Zhao 2020). The proposed method offers an effective means to solve time-dependent PDEs in topology optimization, which facilitates the exploration of intricate material layouts and topological changes. The level set function is updated at prescribed time intervals, enabling the material distribution to evolve toward an optimized configuration that aligns with the design objectives. This iterative process converges toward a material distribution that minimizes or maximizes a specific objective function while adhering to designated design constraints. The primary advantages of this method encompass superior mesh adaptivity, reduced computational cost, and enhanced compatibility with the bi-section Lagrangian method.

Researchers and practitioners face the hurdle of writing efficient and robust computer codes. Developing such codes demands expertise in both numerical techniques and programming skills. To enhance usability, the proposed GFLS262 code is equipped with user-friendly interfaces, allowing users to customize input parameters, including design domain dimensions, boundary conditions, and volume constraints. The code consists of 262 lines, making it easily understandable and modifiable for users. The provided Matlab program includes the main optimization loop (Lines 2–62), body-fitted mesh generator (Lines 64–173), and finite element analysis (Lines 174–214). In the main loop, the program provides visualization capabilities to display the

progressive evolution of the optimized designs. The code discretizes the governing equations and calculates the sensitivity information necessary for optimization iterations. The derivations of curvature and gradient field are displayed in Lines 215–262, together with the level set reinitiating sub-function. The code facilitates the generation of optimized structures by minimizing or maximizing the objective function subject to various constraints. This educational article will serve as a valuable tool for researchers and engineers working on topology optimization, providing fast and efficient implementation of body-fitted level set methods based on the estimated gradient field.

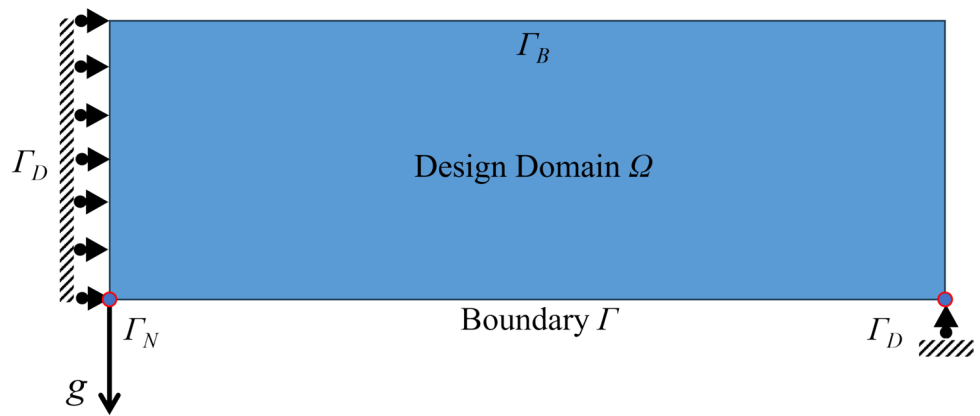
The following sections provide an outline of the contents of this paper. In Sect. 2, we introduce the optimization problem and the algorithms of the proposed GFLS method, which includes a definition of the level set function, the body-fitted mesh generator, and the derivation of the gradient field. The provided Matlab script is further explained and implemented to solve the compliance minimization problem in Sect. 3. The presented code is thoroughly validated against benchmark problems, demonstrating its reliability and accuracy. Finally, in Sect. 4, we summarize the contributions of this work. An educational Matlab program named GFLS262 is provided in the Appendix section of this paper for practical implementation. (<https://github.com/zhuanginhongkong/GFLS262>)

2 Methodology

2.1 GFLS topology optimization

Topology optimization is a design process that optimizes the distribution of material within a given design space $\Omega \subset \mathbb{R}^2$ while simultaneously minimizing an objective function and considering specific constraints, such as volume and stress constraints. The optimization problem often involves solid and void materials within the design domain $\Omega = \Omega_s \cup \Omega_v$ under the linear elasticity setting. A typical configuration for the Messerschmitt-Bölkow-Blohm (MBB) beam optimization problem is depicted in Fig. 1, where the material boundary Γ consists of three components: $\Gamma = \Gamma_N \cup \Gamma_B \cup \Gamma_D$. The Dirichlet boundary condition is imposed on Γ_D , while the Neumann boundary condition is applied on $\Gamma_N \cup \Gamma_B$. The Dirichlet boundary portion marked as Γ_D remains fixed with no displacement. The concentrated load, represented by F (50 N), is applied on the inhomogeneous Neumann boundary labeled as Γ_N . The homogeneous Neumann boundary Γ_B bears no load and is not subject to displacement constraints. Assuming zero body force is applied within the design domain, the surface traction g represents the external loads on the boundary Γ_N . In this article, the objective function

Fig. 1 A half MBB beam optimization problem with the design domain Ω and boundary Γ



represents the strain energy, where J characterizes the functional aspect of topology optimization for compliance minimization in linear elasticity.

$$\begin{aligned} \text{Min} : J(u) &= \int_{\Omega} D\varepsilon(u) \cdot \varepsilon(u) d\Omega = \int_{\Gamma_N} g \cdot u d\Gamma_N \\ \text{s.t.} : \begin{cases} (D\varepsilon(u))n|_{\Gamma_N} = g \\ u|_{\Gamma_D} = 0 \\ \int_{\Omega} d\Omega - V_{\max} \leq 0 \end{cases} \end{aligned} \tag{1}$$

The displacement field u represents the unique solution of the linearized elasticity system. The strain tensor $\varepsilon(u)$ is defined as $(\nabla u + (\nabla u)^T)/2$, while D pertains to the elasticity tensor governed by Hooke’s law. The common choice for the optimization constraint is the maximum volume fraction V_{\max} of the solid material.

The level set method is a powerful approach for solving shape and topology optimization problems, which can achieve topological changes such as merging, splitting, and erosion of material regions. This method represents the interface between different material phases using a level set function, which can be evolved using time-dependent PDEs. This work introduces a level set model as an iso-surface of a scalar function $\varphi: \mathbb{R}^2 \rightarrow \mathbb{R}$, which directly controls the exterior and interior boundary shapes of the structure. The design variable of the GFLS optimization is the boundary Γ determined by the level set function. The level set function value φ can be expressed as:

$$\begin{cases} \varphi(x) < 0 \forall x \in \Omega_s \setminus \Gamma \\ \varphi(x) = 0 \forall x \in \Gamma \\ \varphi(x) > 0 \forall x \in \Omega_v \setminus \Gamma \end{cases} \tag{2}$$

The variable x represents a space location in the design domain. The proposed method optimizes the design pattern by iteratively updating the level set function according

to the shape derivative. At the beginning of each iteration, the level set function φ is reinitialized as a signed distance function to the material boundaries. Using the pre-established level set function φ , we can depict the optimization problem in the following manner using the Heaviside function $H(\varphi)$ and Dirac function $\delta(\varphi)$.

$$\begin{aligned} \text{Min} : J(\varphi) &= \int_{\Gamma_N} g \cdot u \delta(\varphi) |\nabla \varphi| d\Gamma_N \\ \text{s.t.} : \begin{cases} \int_{\Omega} D\varepsilon(u) \cdot \varepsilon(w) H(\varphi) d\Omega = \int_{\Gamma_N} (g \cdot w) \delta(\varphi) |\nabla \varphi| d\Gamma_N \forall w \in W \\ u|_{\Gamma_D} = 0 \\ \int_{\Omega} d\Omega - V_{\max} \leq 0 \end{cases} \end{aligned} \tag{3}$$

This equation holds for any displacement field w in the space of kinematically admissible fields W . In previous level set works (Wang et al. 2003, 2007), the smoothed Dirac delta function and the Heaviside function are typically used to avoid regenerating the element mesh when the boundary is modified or updated during the iterative process. Conversely, the Dirac delta function with a near-theoretical profile has the potential to be integrated into the proposed GFLS method, as the regenerated body-fitted mesh can continuously describe the moving boundaries in the design domain. However, the final objective function values and optimized structures are similar when using the smoothed and near theoretical delta function, since the shape derivative only considers solid elements ($\varphi(x) < 0$). Elements in the void region are excluded from the finite element analysis and sensitivity computation in this work.

The shape derivative of the compliance objective can be derived as the sensitivity according to previous level set works (Allaire et al. 2002, 2004; Wang et al. 2003). For a reference domain Ω_0 , we consider domains of the type $\Omega = \Omega_0 + \psi$. The continuous function ψ belongs to the Hilbert functional space Ψ of order one, which is restricted on the Dirichlet boundary and inhomogeneous Neumann boundary.

$$\Psi = \left\{ \psi \mid \psi \in W^{1,\infty}(\mathbb{R}^2, \mathbb{R}^2) \text{ with } \psi = 0 \text{ on } \Gamma_N \& \Gamma_D \right\} \quad (4)$$

In this case, the shape derivative of the objective function $J(\Omega)$ at Ω_0 can be defined as the Fréchet derivative in Ψ at 0.

$$\left\langle \frac{\partial J}{\partial \Omega}(\Omega_0), \psi \right\rangle = \int_{\Gamma_B} \left(2 \left[\frac{\partial(g \cdot u)}{\partial n_0} + \kappa(g \cdot u) \right] - D\varepsilon(u) \cdot \varepsilon(u) \right) \psi \cdot n_0 d\Gamma \quad (5)$$

where n_0 represents the normal vector to variable boundary Γ_B . The curvature κ is part of the expression of the obtained Fréchet derivative, which can be computed according to the positions of boundary nodes in the body-fitted mesh.

The dynamic level set model specifies a surface S in an implicit form as an iso-surface that takes into account the location x and time t .

$$S(t) = \{x(t) : \varphi(x(t), t) = 0\} \quad (6)$$

Subsequently, the Hamilton–Jacobi equation can be derived by differentiating each side of the equation with respect to the time variable t while applying the chain rule.

$$\frac{\partial \varphi(x, t)}{\partial t} + \nabla \varphi(x, t) \frac{dx}{dt} = 0 \quad (7)$$

The shape derivative of the objective function delivers a vector field v_c throughout the design domain. This time-dependent PDE can be expressed using the magnitude of the velocity field v_c and gradient field of the level set function $\nabla \varphi(x, t)$ as follows:

$$\frac{\partial \varphi(x, t)}{\partial t} + v_c |\nabla \varphi(x, t)| = 0 \quad (8)$$

The discrete solution of the Hamilton–Jacobi equation is employed in the level set optimization to approximate the system variables of a continuous level set function. A robust computational method to obtain the discrete solution was developed based on the notion of weak solutions and entropy limits (Osher and Sethian 1988). To solve the Hamilton–Jacobi equation, the equation can be replaced with an approximation based on the forward difference derivative by introducing a virtual time interval Δt .

$$\varphi(x, t + \Delta t) - \varphi(x, t) \approx -v_c |\nabla \varphi(x, t)| \Delta t \quad (9)$$

The time step Δt for the finite difference scheme must satisfy the Courant–Friedrichs–Lewy (CFL) stability condition.

$$\Delta t = \alpha / \max |v_c| \quad (10)$$

where α is a user-defined parameter to control the length of the time step. A larger time step accelerates topological changes in each iteration, thereby achieving faster convergence in optimization. However, the time step must not

exceed a specific upper limit due to the CFL condition, which is essential for ensuring convergence when solving PDEs.

The time-dependent PDE can be transformed into a series of steady-state problems solved in iterative procedures. The solution of the updated level set function φ_{n+1} can be calculated in each iteration as follows.

$$\varphi_{n+1}(x, t) = \varphi_n(x, t) - v_c |\nabla \varphi_n(x, t)| \Delta t \quad (11)$$

The magnitude of the velocity v_c for the compliance minimization problem can be obtained using the shape derivative (Allaire et al. 2002; Simon 1980; Sokolowski and Zolesio 1992). In 2D scenarios, the design domain is discretized using a body-fitted triangular mesh. As a result, the value $v_{c(e)}$ of the element e in the finite element framework can be expressed as:

$$v_{c(e)} = 2 \left[\frac{\partial(g \cdot u)}{\partial n_0} + \kappa(g \cdot u) \right] - D\varepsilon(u) \cdot \varepsilon(u) = u_e^T k_e u_e + 2\kappa_e u_e^T k_e u_e$$

$$s.t. : \begin{cases} k_e = \rho_e k_{se} \\ \rho_e = 0 \text{ or } 1 \\ \sum_{e=1}^N \rho_e V_e - V_{\max} \leq 0 \end{cases} \quad (12)$$

The solid material stiffness matrix k_{se} can be determined by the vertices' coordinates of the body-fitted element e . The ρ_e represents the relative density of the mesh e , where it is either one or zero, eliminating the greyscale problem, while the volume of the mesh e is denoted as V_e . The curvature term is incorporated into the sensitivity analysis to enhance the stability of advection, whose derivation in the body-fitted mesh is included in Sect. 3.4. Meanwhile, the element displacement vector u_e and stiffness matrix k_e are determined during the finite element analysis procedure. The element sensitivity number is interpolated to the nodes to obtain v_c values using a weighted average method.

The Hamilton–Jacobi equation shown in Eq. (11) can be solved by a standard upwind scheme on a Cartesian grid (Allaire et al. 2004; Sethian 1999; Wang et al. 2003). Assuming fields $\varphi_n^{x+1}, \varphi_n^{x-1}, \varphi_n^{y+1}, \varphi_n^{y-1}$ represent the level set functions in the rectangular design domain with small offsets in x - and y - directions. The finite difference in various directions $T_n^{+x}, T_n^{-x}, T_n^{+y}, T_n^{-y}$ can be calculated as:

$$T_n^{+x} = \frac{\varphi_n^{x+1} - \varphi_n}{\Delta x} \quad (13)$$

$$T_n^{-x} = \frac{\varphi_n - \varphi_n^{x-1}}{\Delta x} \quad (14)$$

$$T_n^{+y} = \frac{\varphi_n^{y+1} - \varphi_n}{\Delta y} \tag{15}$$

$$T_n^{-y} = \frac{\varphi_n - \varphi_n^{y-1}}{\Delta y} \tag{16}$$

The effectiveness of the upwind scheme in the level set method has been demonstrated in the formwork of a fixed structured mesh. The level set function can be updated by:

$$\varphi_{n+1}(x, t) = \varphi_n(x, t) - [\max(v_c, 0)\nabla^+ + \min(v_c, 0)\nabla^-]\Delta t \tag{17}$$

$$\nabla^+ = \max(T_n^{-x}, 0)^2 + \min(T_n^{+x}, 0)^2 + \max(T_n^{-y}, 0)^2 + \min(T_n^{+y}, 0)^2 \tag{18}$$

$$\nabla^- = \max(T_n^{+x}, 0)^2 + \min(T_n^{-x}, 0)^2 + \max(T_n^{+y}, 0)^2 + \min(T_n^{-y}, 0)^2 \tag{19}$$

However, this well-established method proves to be less accurate and efficient when applied to an unstructured mesh. Unlike the structured mesh, the individual nodes of a body-fitted triangular mesh have ambiguous neighboring nodes, and therefore, the previous upwind method becomes invalid to obtain fields $\varphi_n^{x+1}, \varphi_n^{x-1}, \varphi_n^{y+1}, \varphi_n^{y-1}$, which are necessary to update the level set function. In this work, the GFLS method utilizes per-cell linear estimation of discrete gradient vectors, instead of the traditional upwind method (Osher and Sethian 1988), as a finite difference scheme to efficiently update the level set function in the body-fitted mesh. Let us consider a triangular element e in the body-fitted mesh with three vertices $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c$, and three edges $(\mathbf{v}_a - \mathbf{v}_c), (\mathbf{v}_c - \mathbf{v}_b), (\mathbf{v}_b - \mathbf{v}_a)$. The gradient vector of the triangular element $\nabla\varphi_e$ for the scalar field of the level set function φ can be expressed by:

$$\nabla\varphi_e = \frac{(\varphi_a - \varphi_c)(\mathbf{v}_c - \mathbf{v}_b)^\perp}{2V_e} + \frac{(\varphi_b - \varphi_c)(\mathbf{v}_a - \mathbf{v}_c)^\perp}{2V_e} \tag{20}$$

where the φ_a, φ_b , and φ_c are the level set function values on the vertices $\mathbf{v}_a, \mathbf{v}_b$, and \mathbf{v}_c . The $(\mathbf{v}_c - \mathbf{v}_b)^\perp$ means that the edge $(\mathbf{v}_c - \mathbf{v}_b)$ is rotated by 90 degrees. Thus, the per-vertex gradient estimation $\nabla\varphi_p$ can be obtained using the weighted average approach as follows.

$$\nabla\varphi_p = \frac{1}{V_{sum}} \sum_{e=1}^{N(p)} \nabla\varphi_e V_e \tag{21}$$

where $N(p)$ denotes all the triangles that include the node p as a vertex, and V_{sum} is the area sum of these elements. Thus, the level set function value at each node p in the body-fitted mesh can be expressed as:

$$\varphi_{p(new)} = \varphi_p - v_c \left| \nabla\varphi_p \right| \Delta t \tag{22}$$

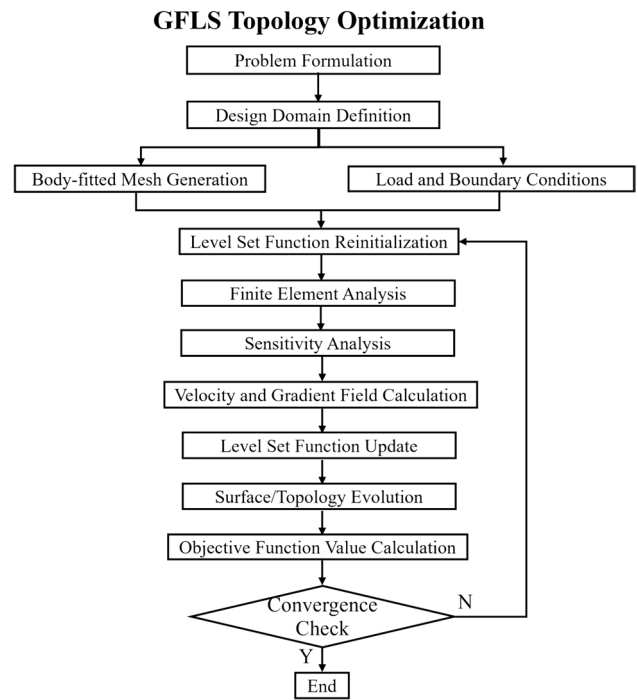


Fig. 2 Overview of the GFLS topology optimization processes

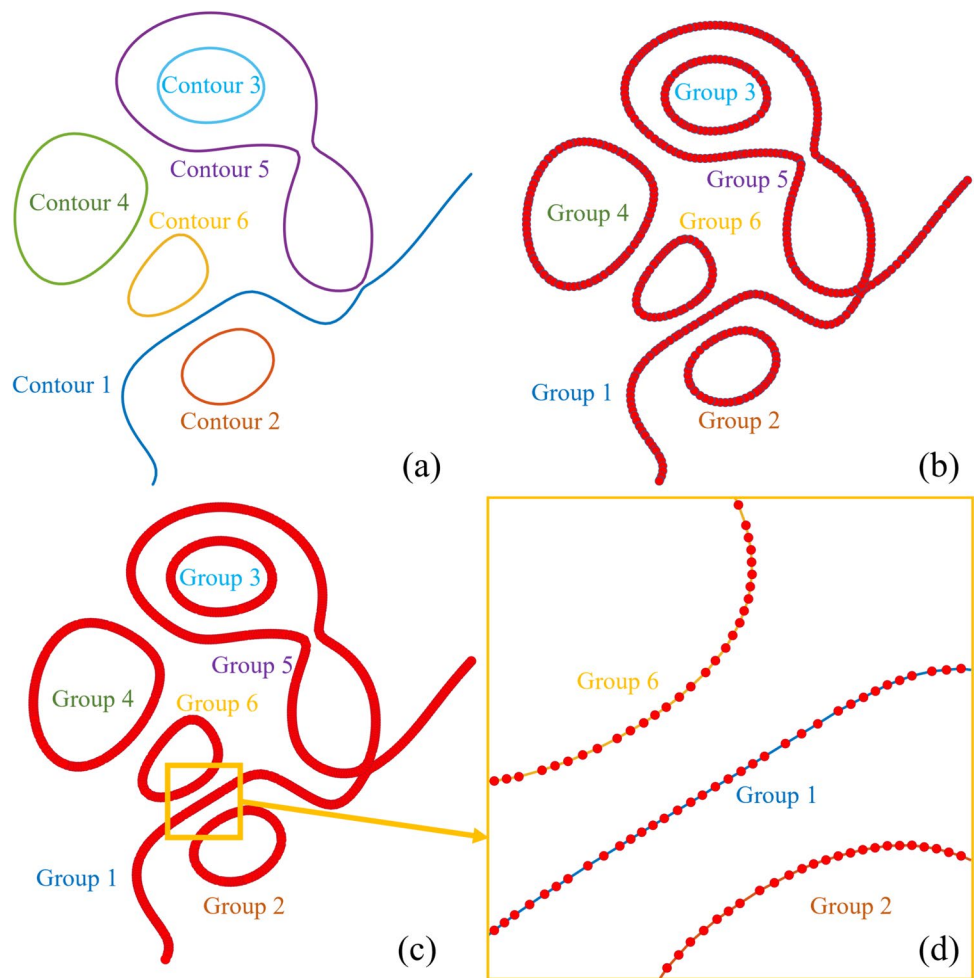
As illustrated in Fig. 2, the level set function and optimized structure can be iteratively updated by solving Eq. (11) until it meets the convergence criteria. This study adopts the bi-section Lagrangian method to ensure precise control of the volume fraction ratio and accelerate convergence.

2.2 Body-fitted meshing

This section presents the body-fitted meshing based on the Delaunay triangulation, improving the mesh generation scheme in the existing studies (Persson and Strang 2004). Firstly, a Matlab function `ContourPoints`¹ is designed to find the points on the zero iso-surface of the level set function. In this work, the `structure array` data type in Matlab is employed to save the required information by groups using data containers called fields. Figure 3 consists of four parts illustrating different stages in the process of identifying and adjusting boundary nodes. Firstly, the zero-level contours of the level set function are plotted in Fig. 3a. Then, Fig. 3b displays the boundary nodes initially identified by MATLAB along the zero-level contours determined in Fig. 3a. The nodes on the zero-level contour of the level set function are grouped based on their belongings to an open/closed

¹ Names in type-writer font refer to Matlab variables, commands, and functions.

Fig. 3 **a** The zero-level contours of the level set function; **b** the boundary nodes identified by Matlab; **c** the adjusted boundary nodes; **d** the magnified section of the adjusted nodes in Groups 1, 2, and 6



curve using the contour operator, where the repeated nodes in the closed groups must be removed. For example, Contour 1 shown in Fig. 3a is an open curve, while Contours 2, 3, 4, 5, and 6 are closed curves. However, these boundary nodes need to be adjusted from the initial positions to improve accuracy in describing the boundary geometries. As illustrated in Fig. 3c, the distances between the neighboring points are adjusted to ensure the effective control of the node distribution. Figure 3d provides a closer view of the adjusted boundary nodes in three groups, labeled as Groups 1, 2, and 6. The magnification highlights the detailed positioning and adjustments made to the boundary nodes.

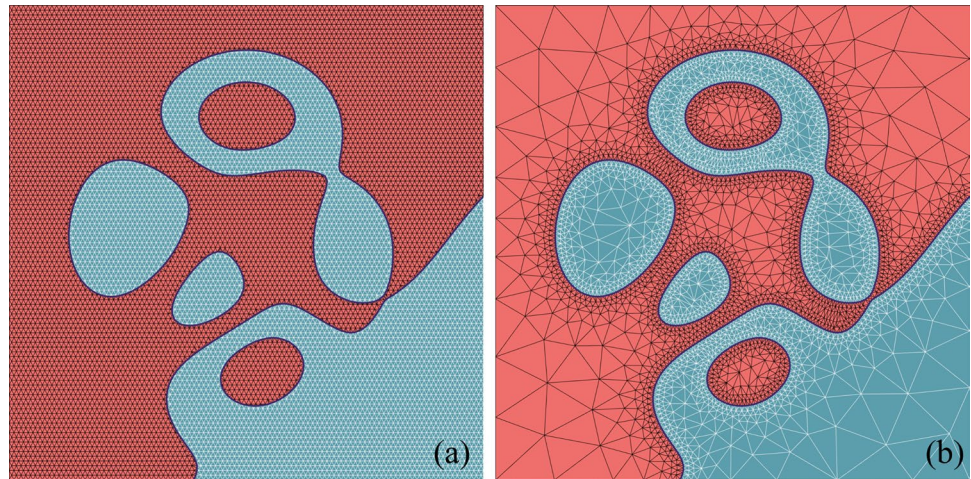
The resulting information is stored as structure array variables, which include node numbers, coordinates, open/closed group designations, and curvature values. The adjusted boundary nodes and loaded nodes are interspersed among unfixed points that are uniformly distributed across the structure. Figure 4a illustrates the body-fitted meshes generated by these nodes using Delaunay triangulation. Subsequently, the positions of these nodes undergo iterative adjustments to accurately delineate the boundaries by solving for force–displacement equilibrium. The sparseness of the

body-fitted mesh can be manually controlled by specifying a target edge length (L_0) for each element. This work aims to create a denser mesh around the boundaries to improve the analysis accuracy. A coarser mesh is generated in the low-sensitivity area to decrease the computational cost. Thus, the L_0 value is defined as proportional to the distance from the material boundaries, as illustrated in Fig. 4b. The edge length (L) of the body-fitted mesh generated by Delaunay triangulation can be calculated for each unstructured element. The difference between current L and desired bar length L_0 causes the repulsive forces within the bars according to Hooke’s law. Suppose p_s and p_e represent the coordinates of the starting and ending points of bar b . In that case, the force vector F_b can be obtained using a scaling factor (f_s) as follows:

$$F_b = (\max(f_s L_0 \sqrt{L^2/L_0^2} - L, 0)/L)(p_e - p_s) \tag{23}$$

The fixed scaling factor f_s ensures that repulsive forces exist in most bars ($F_b > 0$) (Persson and Strang 2004). Now, we consider a node connected to several bars (total number:

Fig. 4 **a** Initial regularly-distributed nodes and the meshes generated by the first Delaunay triangulation; **b** Optimized node positions and the meshes generated by the last triangulation



b_m) in the body-fitted mesh framework. The force vector on the node is evaluated by the vector summation of all the connected bars. The node position \mathbf{p} changes according to a positive parameter β after applying the repulsive forces. The parameter β is defined as 0.2 to ensure numerical stability.

$$\mathbf{p} = \mathbf{p} + \beta \sum_{b=1}^{b_m} ((\max(f_s L_0 \sqrt{L^2/L_0^2} - L, 0)/L)(\mathbf{p}_e - \mathbf{p}_s)) \quad (24)$$

Subsequently, the mesh generator iteratively performs Delaunay triangulation and adjusts node positions until the current and desired edge lengths are sufficiently close. After several iterations, high-quality body-fitted meshes can be generated, as depicted in Fig. 4b. In summary, the body-fitted technique ensures that the computational grid conforms to the material boundaries, providing accurate representation and minimizing numerical errors.

3 Matlab implementation

The main objective of this educational article is to provide a user-friendly and accessible codebase that allows engineers and researchers to experiment and explore body-fitted level set topology optimization techniques. The compact and well-documented code enables users to grasp the underlying concepts and incorporate them into their specific design problems. The numerical implementation of the GFLS optimization approach is introduced by explaining the proposed 262-line Matlab code GFLS262 line-by-line. The Matlab program can be called using the following command:

GFLS262(lx,ly,Vr,volReq,maxedge,minedge,E,nu,tau,sigma,alpha).

In this context, lx and ly specify the dimensions of the 2D design domain along the x and y-axis directions

correspondingly. Vr is utilized as the solid material removal ratio to regulate the volume decrease rate and adapt the number of iterations required for convergence. volReq denotes the maximum volume fraction that needs to be achieved. The parameters maxedge and minedge are employed to limit the range of edge lengths for the body-fitted mesh, thereby controlling the mesh sparseness throughout the design domain. E and nu represent Young's modulus and Poisson's ratio of the linearly elastic solid material. tau denotes a positive coefficient that contributes to the curvature term when calculating the sensitivity. A Gaussian filter is employed to smooth the material boundaries controlled by a positive parameter sigma. The step length Δt for the finite difference scheme is determined by the value of alpha. If this coefficient is too small, the topology evolution speed will decrease, potentially converging to a suboptimal local minimum (Challis 2010).

This code generates a visual representation of the distribution of design variables in each iteration. Deep pink and indigo blue are employed to depict solid and void materials, respectively, while the material boundaries are captured with purple curves. Figure 5 showcases the optimized configuration of a 2D MBB beam using the proposed Matlab code, executed with the specified input line.

GFLS262(120,40,0.05,0.5,10,2,2e5,0.3,4.8e-5,1.0,260)

Building upon prior research (Sigmund 2001), this approach leverages symmetry to diminish the computational expenses involved in optimization. Specifically, this example focuses solely on the right half of the MBB beam. During optimization, the smooth boundaries continuously evolve with the body-fitted mesh to improve the analysis accuracy. After 22 iterations, the optimized structure is achieved with an objective function value of 1.128 using the GFLS method. The volume fraction ratio closely matches the prescribed value of 50%, measuring at 50.01%. The Matlab program was executed using 12th Gen Intel(R) Core(TM) i7-12700H@2.30-GHz CPUs in this work.

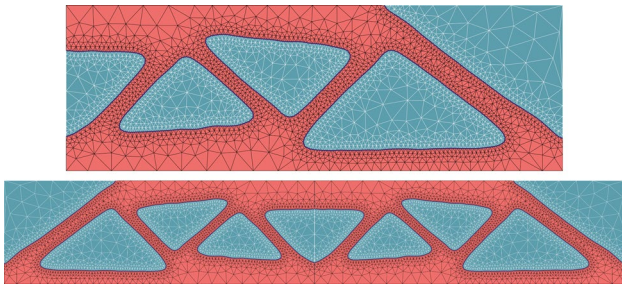


Fig. 5 The optimized configurations of the half and full MBB beam designs obtained by the proposed Matlab program GFLS262

In the subsequent sections, we present a detailed description of the code architecture, along with explanations of the numerical methods employed. The 262-line code consists of four main parts, namely the GFLS main loop, body-fitted mesh generator, finite element analysis, and gradient field derivation, each explained individually in Sects. 3.1–3.4. Section 3.5 discusses the parameter definitions, differences from previous code, and possible extensions. Additionally, Sect. 3.6 provides extra numerical examples to demonstrate the effectiveness and robustness of the proposed method.

3.1 GFLS main loop (Lines 2–62)

The code provided in the appendix implements a structural optimization algorithm based on the finite element method and the level set method. The main function of the GFLS method only contains 61 lines. First, the parameters and the initial mesh in the design domain are defined in Lines 3–9. In Line 3, a two-dimensional rectangular domain *BDY* is defined to represent the range of the optimization area. The coordinates of the regularly distributed points across the optimization area are obtained using the operator *meshgrid* in Line 4. Line 6 creates a level set function *phi* to describe the initial material distribution, which is further adjusted in Line 8 to obtain a usable boundary curve *c*. The maximum and minimum distances d_1 and d_2 balance the distance between neighboring nodes on the zero contour of the level set function *phi*. In Line 9, the *GenerateMesh* function is called to establish a body-fitted mesh using the obtained boundary curve *c*, generating a data structure that includes finite element information. The variables *p*, *t*, *t1*, *t2*, *Ve*, and *pmid* denote node coordinates, element connectivity, void element connectivity, solid element connectivity, element volume, and element centroid coordinates, respectively.

Lines 11–62 form the main loop of the GFLS optimization program, which iterates multiple times, and each iteration consists of the following steps. Line 13 performs finite element analysis for the body-fitted mesh to calculate strain energy *J*, volume fraction of the solid material *vol*, and sensitivity numbers *Ce*, which is saved for each iteration. In

Line 18, the element sensitivity numbers are interpolated to each node in the design domain. The scalar curvature values on the current boundary node are calculated in Line 15, stored in *Curvfull* and integrated into the node sensitivity field. Line 19 calculates the velocity vector *Vc*, which represents the direction to transition from the current state to the optimized state. Lines 20–21 calculate the time step *dt* and the target volume *V* of the current iteration. Inspired by the evolutionary ratio in the conventional BESO method (Huang and Xie 2010), the target volume in each iteration is determined by a solid material removal ratio *Vr* and the allowable maximum volume *volReq*. Then, the objective function and volume fraction of the optimization are output, and the optimized configuration is plotted using deep pink and indigo blue using the *patch* function (Lines 24–26). The smooth and elegant material boundaries are drawn using the contour operator in Line 27. Lines 30–32 check for convergence criteria, including the minimum iteration to convergence, volume requirement, and objective function requirement. The objective function differences between the current and previous five iterations should be less than 0.5 percent to meet the criteria and exit the Matlab program.

Then, Lines 34–37 reinitialize the level set function in each iteration and interpolate to each node. The variables *x*, *phiE*, and *phiN* store the relative density, level set values for elements, and level set values for nodes. For numerical stability, the level set values for each element are defined as a signed distance function to the nearest material boundaries (Challis 2010). In Line 38, the discrete gradient field of the level set function *phiN* is obtained as *Gradx* and *Grady* in *x*- and *y*-directions, which is used for subsequent design updates. Next, Lines 40–58 perform an optimization process using the bi-section method to update the level set function based on given constraints and target volume. It utilizes interpolation, contour extraction, and one-iteration meshing functions to estimate the current volume, adjust the Lagrange multiplier, and generate an updated mesh representation. The variables *l1* and *l2* are initialized as 0 and 1×10^9 (Line 40), representing the lower and upper bounds of the Lagrange multiplier. A while loop is initiated, which continues until the absolute difference between *l2* and *l1* divided by the absolute value of *l2* is smaller than 1×10^{-9} . In Line 42, the sensitivity number for the bi-section algorithm *Vbs* is calculated as the difference between *Vc* and the average of *l1* and *l2*. Line 43 updates the level set function by solving the time-dependent PDE according to the sensitivity and gradient values, as shown in Eq. (11). In Line 45, the function *ContourPoints* is called to extract the contour points of the updated level set function *phinew*. Next, an if-else condition checks if any contour points are found. If contour points exist, a mesh is generated to calculate the volume fraction of the solid materials in the current design. Note that the re-meshing process only operates for one

iteration (maxiter defined as 1) to save computational cost. If no contour points are found, the volume of the void space inside the design is calculated based on the mean value of the level set function evaluated at the nodes of the triangles. The level set function is then convolved with a Gaussian kernel (Zhuang et al. 2022a, b) and interpolated to the regularly distributed nodes to plot contours in Lines 44–45. Another if-else condition is used to update the Lagrange multiplier value based on the volCurr (volume fraction of the current structure) being greater or smaller than the target volume V . Line 61 is employed to generate a new body-fitted mesh based on the updated level set function. The resulting mesh data, including coordinates, connectivity, element volumes, and midpoints, are stored in the respective variables.

In conclusion, this part of the code implements an optimization-based structural design process by combining the finite element method and the GFLS algorithm to optimize the distribution of structural materials. In each iteration, the target volume of the current iteration is prescribed, and the material distribution is continuously evolved based on this value. The convergence of the system is checked during the iteration, and the optimization is terminated when the convergence criteria are met, aiming to achieve a decent design objective.

3.2 Body-fitted mesh generator (Lines 64–173)

The body-fitted mesh generator iteratively creates the mesh to capture the material boundaries (zero contours of the level set function), which mainly contains three functions: ContourPoints (Lines 65–113), GenerateMesh (Lines 115–161), and Uniquenode (Lines 163–173). The Matlab function ContourPoints is utilized to find the nodes on the contour and adjust the distance between neighboring nodes. The boundary curve c is a two-dimensional array that records the coordinates of the contour nodes, and $d1$ and $d2$ are two threshold parameters for distance. Lines 67–74 contain a while loop that divides the contour node set c into several segments, each containing several nodes. As illustrated in Fig. 3b, the points of each segment are saved in a structure array, and whether each segment is an open curve is recorded. Meanwhile, the numbering columns obtained by the Matlab function contour are removed from the contour node-set. Next, the for loop in Lines 76–112 iterates through each segment of the contour and adjusts the node spacing for further processing. Line 75 clears the variable c and Curv to store new contour points and their curvature values. Lines 77–80 retrieve the coordinate points of the current segment and calculate the distances between neighboring nodes in the current segment, saving in ndist. The for loop in lines 81–86 is designed to remove all nodes that are too close to their neighboring points. Lines 88–90 check if the distance between the first and last points of a contour is less than

$d1$, and if so, the last point is removed. Then, Lines 91–96 calculate the distances between neighboring nodes in the updated node set and store them in ndist. Lines 97–103 aim to insert midpoints between the nodes that are larger than $d2$ in distance for the current segment. Lastly, the processed nodes with balanced distance are added to the variable c , and the corresponding curvature values are stored in the variable Curv (Lines 104–110).

The Matlab function ContourPoints is a novel function proposed by this paper, which significantly improves the quality of the body-fitted mesh generated by the function GenerateMesh. Lines 116–122 generate the regularly distributed nodes pi in the design domain and calculate the minimum distances from each node to the contour points, stored in variable d . The contour points and loaded nodes are fixed in pfix, and the rejection method deletes points using a probability proportional to $1/d^2$ (Lines 123–127). After removing duplicate points, the for loop in Lines 131–151 iteratively updates the node coordinates p until the maximum iteration count is reached. First, the if statement in Lines 132–136 is used to determine whether to conduct Delaunay triangulation according to the amplitude of node movement in the last iteration. In this work, the edge length of the body-fitted mesh is defined as proportional to this distance, which is also limited by the prescribed values maxedge and minedge. The parameter fscale influences the relation between displacement and force in the body-fitted mesh generator. Lines 138–144 aim to obtain the current L and desired length $L0$ for each edge using the user-defined parameter fscale. Next, the repulsive forces within the bars are calculated using Hooke's law, explained in Eq. (24), which is employed to update the positions of the non-fixed nodes accordingly. The node coordinates are determined by solving linear force–displacement equilibrium in a truss structure, and the Delaunay triangulation iteratively updates the topology (Persson and Strang 2004; Zhuang et al. 2021). Lines 149–150 restrict the node coordinates within the prescribed boundary range. Then, the function Uniquenode is called in Line 152 to process the obtained node coordinates p and triangular mesh t , removing duplicate nodes and updating the connectivity information. The variable pmid is calculated as the average coordinates of the three vertices for each element, which represents the midpoints of the elements. The level set values of elements dE are obtained in Line 154 by interpolating the level set values of nodes dN at the positions pmid using cubic interpolation. Elements are classified into two groups based on the sign of their corresponding dE values. Elements with positive dE values are stored in $t1$ as the void domain, while elements with non-positive dE values are stored in $t2$ as the solid domain (Line 155–156). Lastly, a for loop iterates in Lines 159–161 over each element to calculate the volume Ve of each element using a determinant formula.

The third function, `Uniquenode`, removes duplicate nodes and recomputes the node indices in the triangular mesh. Lines 164–168 eliminate the elements whose volume is zero and delete repeated/vacant nodes from `p`, while Lines 169–173 update the connectivity information in `t` according to the updated node matrix `p`. Overall, this part of the code appears to be performing mesh regeneration by comparing current and desired edge length, moving node positions, updating connectivity information, and calculating curvature values of the boundary nodes.

3.3 Finite element analysis (Lines 174–214)

Collectively, this section contributes a valuable finite element analysis (FEA) tool for researchers and practitioners engaged in topology optimization using the body-fitted mesh. This section also contains three Matlab functions: `E2N` (Lines 174–180), `FEA` (Lines 181–208), and `GetmatrixKe` (Lines 209–214). The objective of the function `E2N` is to estimate the node sensitivity number `dN` using the element sensitivity `x`. It iterates over each node in `p`, calculates the dot product between the element volume and sensitivity, and then divides it by the sum of all elements. Line 178 finds all elements that contain the specific node, which determines the nodal sensitivity using the weighted average method.

Subsequently, the FEA program incorporates finite element analysis within the body-fitted mesh framework. Lines 183–186 form a loop to compute the stiffness matrix of the solid elements and combine them into a larger stiffness matrix. We only consider the solid elements in the FEA procedure to save remarkable computational costs, especially for structures with a low-volume fraction. Within the loop, the function `GetmatrixKe` is called to calculate the stiffness matrix for each element using the vertex coordinates, elastic modulus `E`, and Poisson's ratio `nu`. The detailed derivations of the element stiffness matrix for the body-fitted mesh using symbol calculation are provided by Zhuang et al. (2021). The results are then stored in the corresponding positions of the matrix `KK`. Line 159 defines an array `elemDof` to store the degree of freedom indices for each element. Based on the node indices `t`, the degrees of freedom in the `x`- and `y`-directions are assigned to each element. Inspired by published SIMP codes (Andreassen et al. 2011; Ferrari and Sigmund 2020), Lines 190–194 reconstruct `KK` into a one-dimensional array `sK` and create a symmetrized sparse matrix `NK` using the matrices `iK`, `jK`, and `sK`. After assembling the global stiffness matrix, the program defines the loads and boundary conditions across the design domain. Lines 195–197 find the indices of the fixed nodes on the boundary and store the fixed degree of freedom in `fixedDof`. Line 198 finds the indices of force nodes and stores them in `forceNodes`. Next, the displacement vector `U` and the force vector `F` are initialized in Lines 201 and 202, where the

degrees of freedom corresponding to the loaded position are subjected to a vertical force of -50 . Thus, the displacement vector `U`, the strain energy `J`, sensitivity numbers `Ce`, and solid material volume `vol` can be obtained in Lines 201–208.

While the element stiffness matrix remains constant across all elements in the structured rectangular mesh, it varies in the body-fitted triangular mesh based on the node coordinates. Lines 210 to 214 define the `GetmatrixKe` function, which is employed to calculate the stiffness matrix `Ke` for each element. The function inputs the vertex coordinates `X` and `Y` of a triangular element, as well as the material parameters to generate the elasticity matrix `D`. It first calculates the Jacobian matrix `J`, then computes the strain matrix `Be`, and finally obtains the stiffness matrix `Ke` of the element based on `Be` and the material parameters. Overall, this part of the code implements the functionality of calculating the element stiffness matrix, assembling the global stiffness matrix, solving for displacements, and computing strain energy/sensitivity in finite element analysis.

3.4 Derivation of curvature and gradient field (Lines 215–262)

The last part of the code involves three functions for level set function reinitialization, discrete gradient derivation, and curvature calculation. These functions are broken down and explained as follows. The function `pdist` is designed to reinitialize the level set function using the signed distance function at the beginning of each optimization iteration (Line 36). The node coordinates `xinds` and `yinds` are obtained for void and solid domains. Then, Lines 224–226 calculate the minimum Euclidean distance between the centroid point of each element to the material boundaries and output the values in the `dist` array. The `Mshdist` library (Dapogny and Frey 2012), an open-source tool, is highly recommended for reinitializing level set functions due to its numerical efficiency. It computes the exact distance only in the neighborhood of the boundary and then propagates inside the domain.

The function `GradDeri` (Lines 228–252) computes the discrete gradient of the level set function on a body-fitted triangular mesh. A per-cell linear estimation of the discrete gradient inspired by the previous works (Mancinelli et al. 2018; Zhuang et al. 2024) is adopted in this section. The program takes the level set function values (`f`), nodal coordinates (`p`), element connectivity matrix (`t`), element areas (`Ve`), and a node-set (`c`) as input parameters. First, Lines 229 and 230 initialize gradient components `Gradx` and `Grady` as zero arrays. The function then iterates over the nodes and finds all the elements `ts` that include the specific node (Line 233). Then, the gradient vector for the elements included in `ts` can be estimated. Within the for loop, it calculates the difference vectors `D1` and `D2` based on the nodal coordinates `p1`, `p2`, and `p3` and computes the gradient vector (`Gradu`) using the nodal level set function

values $\phi_1, \phi_2,$ and ϕ_3 . Thus, the per-vertex discrete gradient components Grad_x and Grad_y for node A_0 can be calculated using the weighted average approach in Lines 250–251. The curvature number for a boundary node can be estimated according to the coordinates of its adjacent two points on the contour. A circumscribed circle of the triangle is created by the three points $i - 1, i, i + 1$ to determine the scalar curvature values. The curvature radius R of this circle can be calculated as follows.

$$R_i = \frac{|X_{i-1} - X_{i+1}| |X_i - X_{i+1}| |X_i - X_{i-1}|}{|(X_{i-1} - X_i) \times (X_{i+1} - X_i)|} \tag{25}$$

where the $X_{i-1}, X_i,$ and X_{i+1} represent the coordinates of the three nodes. The symbol “ \times ” returns the cross product of vectors. The scalar curvature κ_i is the reciprocal of the curvature radius, which is computed as follows:

$$\kappa_i = \frac{|(X_{i-1} - X_i) \times (X_{i+1} - X_i)|}{|X_{i-1} - X_{i+1}| |X_i - X_{i+1}| |X_i - X_{i-1}|} \tag{26}$$

CalculateCurv (Lines 254–262) computes the curvature C of a curve defined by a set of contour points (X). Line 255 initializes N as the number of points in X and appends an additional column of zeros to X . It then iterates over each point in X except the first and last points and calculates the curvature radius R according to Eq. (25). Next, the curvature C can be computed as the reciprocal of R (Line 262).

In summary, this part of the code provides functions for calculating distances between points, computing the discrete gradient of the level set function on a body-fitted mesh, and evaluating the curvature values. The accessibility and ease of implementation of the code contribute to ongoing advancements in structural optimization.

3.5 Discussions

This section explores how to select appropriate values for the parameters used in the Matlab implementation of the proposed algorithm. It also outlines potential extensions that could enhance the approach to address more complex optimization problems. Table 1 presents the recommended value ranges for the nine input parameters established by testing the stiffness optimization problems. Users have the flexibility to define these parameters based on specific requirements. Input values within the recommended ranges typically yield optimization results with objective values that closely approach the optimal value. Selecting various parameters can produce a range of competitive designs, which allows architects and engineers to achieve more architectural innovation (He et al. 2023).

Furthermore, Table 2 includes definitions for other parameters within the Matlab code. The Young’s modulus and Poisson’s ratio of the solid material can be manually specified as input parameters. The remaining parameters are hardcoded in the script to streamline operations, as these

Table 1 Suggested parameter values for nine input parameters in program GFLS262

Parameter	Meaning	Suggested value range	Inappropriate values will cause:
lx & ly	Dimensions of the design domain along the x and y-axis directions	Integer values The value of $lx \times ly$ should range from 400 to 40,000	Too small: Poor mesh quality Too large: Significant computational costs
Vr	Solid material removal ratio between iterations	The value of Vr should range from 0.01 to 0.2	Too small: Very slow convergence Too large: Local optimality problem
volReq	Maximum allowable volume fraction of solid material	The value of volReq should range from 0 to 1	Too small: No solid materials in the domain Too large: No void materials in the domain
maxedge & minedge	Range of edge lengths for the body-fitted mesh	The ratio of maxedge/minedge should range from 2 to 10	Too small: Uniform mesh distribution Too large: Poor mesh quality
tau	Positive coefficient contributing to the curvature term	The value of tau should range from 1×10^{-5} to 1×10^{-3}	Too small: The effect of the curvature term is unnoticeable Too large: The curvature term dominates the level set evolution
sigma	Gaussian filtering coefficient	The value of sigma should range from 0.8 to 2	Too small: Unsmooth material boundaries Too large: Large volume fraction changes
alpha	Step length coefficient	Integer value The value of alpha should range from 200 to 300	Too small: The design changes slowly and converges to a poor solution Too large: The design changes steeply and affects accuracy

values have been found suitable for stiffness optimization problems.

The key differences between the proposed 262-line code GFLS262 and the previous 172-line code TriTOP172 (Zhuang et al. 2022a, b) are outlined below.

- (1) The proposed GFLS method employs the gradient field-based level set method instead of the BESO sensitivity ranking scheme to update the design variables, which takes advantage of the smooth boundaries to the maximum extent;
- (2) Matlab function ContourPoints is provided in this work to find the points on the zero iso-surface of the level set function. The points on each contour are found, adjusted, and stored as *structure array* variables, which significantly increases the mesh generator efficiency;

- (3) The mean curvature values of the boundary nodes are included in the sensitivity analysis rather than the non-linear diffusion term;
- (4) Gradient field derivation and level set function reinitialization are involved in updating the design variables during the optimization;
- (5) As shown in Table 3, the newly introduced 262-line code improves the optimization efficiency and objective function value (mean compliance) of the optimized configuration.

The proposed method is capable of addressing optimization problems featuring various objective functions and constraints. For instance, compliant mechanisms achieve mobility via elastic deformation rather than relying on rigid-body connections. In the optimal design of compliant

Table 2 The definitions of material property parameters and parameters hard-coded in program GFLS262

Parameter	Meaning
E (Input)	The Young’s modulus of the solid material
nu (Input)	The Poisson’s ratio of the solid material
BYD (Line 3)	The ranges of the design domain
d ₁ & d ₂ (Line 7)	Maximum and minimum allowable distances between boundary nodes
Fscale = 1.2 (Lines 9, 47, 61)	A positive scaling factor to ensure that most bars give repulsive forces
maxiter = 80 (Lines 9, 61)	Maximum iteration number during the re-meshing process
0.005 (Line 30)	Relative tolerance on the volume fraction and objective values for termination of the program
l1 = 0 & l2 = 1 × 10 ⁹ (Line 40)	Lower and upper bounds of the Lagrange multiplier
num & col (Line 66)	Group index and node index
rv = 0.5 (Line 125)	Rejection value
beta = 0.2 (Line 129)	Force–displacement parameter

Table 3 Optimization objectives and computational efficiency comparison of the GFLS, BESO, BFLS-UPWIND, and HJLS methods

Optimization Method	GFLS262 (GFLS)	TriTOP172 (BESO)	BFLS with Upwind	HJLS
Mean Compliance	1.799	1.802	1.867	1.882
Objective Function Difference	N/A	0.17%	3.64%	4.41%
Iterations to Convergence	24	22	27	74
Total Run Time (second)	89.800	142.349	122.652	100.488
Total Element Number	7,859	9,783	6,088	4,000
Total DOF	8,060	10,036	6,232	8,262
Run Time per DOF (second)	1.114 × 10 ⁻²	1.418 × 10 ⁻²	1.968 × 10 ⁻²	1.216 × 10 ⁻²
Efficiency Difference	N/A	21.44%	43.39%	8.39%
Algorithm Category	Boundary-based method	Density-based method	Boundary-based method	Boundary-based method
Using Upwind Scheme?	×	×	✓	✓
Using Body-fitted Mesh?	✓	✓	✓	×

mechanisms, the objective function focuses on displacement in a specified direction, denoted as u_{out} at the specific output port.

$$\begin{aligned} &Max : J = u_{out} \\ &s.t. : \begin{cases} u|_{\Gamma_D} = 0 \\ \int_{\Omega} d\Omega - V_{max} \leq 0 \end{cases} \end{aligned} \tag{27}$$

The sensitivity analysis of the compliant mechanism problem utilizes an adjoint load approach, where a virtual force is applied to the specified output port. A gripper optimization example demonstrating the design of a compliant mechanism using the proposed GFLS method is presented in the following section.

Stress constraints can be incorporated with topology optimization (Bruggi and Duysinx 2012; Chen et al. 2021; Fan et al. 2019; Holmberg et al. 2013; Xia et al. 2018) to address issues related to excessive structural stress. The compliance optimization problem, which concurrently considers volume and stress constraints, can be formulated as follows:

$$\begin{aligned} &Min : J(\varphi) = \int_{\Gamma_N} g \cdot u\delta(\varphi)|\nabla\varphi|d\Gamma_N \\ &s.t. : \begin{cases} u|_{\Gamma_D} = 0 \\ \int_{\Omega} d\Omega - V_{max} \leq 0 \\ 0 < \sigma_{vM}^{max} \leq \sigma_{vM}^* \end{cases} \end{aligned} \tag{28}$$

where σ_{vM}^{max} denotes the maximum von Mises stress for all body-fitted elements in the design domain. σ_{vM}^* represents the maximum allowable von Mises stress value. When dealing with multiple constraints in the optimization problem, the proposed approach can integrate the augmented Lagrangian technique (Luo et al. 2008) or incorporate an isotropy constraint (Challis et al. 2008). A T-Pier column example illustrating the multi-constraint optimization is presented in the subsequent section to showcase the advantage of the proposed GFLS method.

3.6 More numerical examples

The numerical examples presented in this section showcase the reliability and effectiveness of the proposed program in achieving outcomes within a reasonable number of iterations. The Matlab program can solve user-defined optimization problems by changing a few command lines. Different load and boundary conditions can be applied by making changes to Lines 195–202 in the finite element analysis function. The initial pattern within the design domain can be defined manually in Line 6. The two threshold parameters (d1 and d2) for the distance between adjacent nodes on the boundary curve are prescribed in Line 6. Furthermore, the parameters fscale (Lines 9, 47, 61), maxiter (Lines 9, 61), rv

(Line 125), and beta (Line 129) can also be adjusted according to the specific optimization problem. The following command is added in Line 28 to save the plotted figures during each iteration in the folder.

```
saveas(gcf,['./fig', int2str(iterNum) '.png']);
```

As illustrated in Fig. 6, a vertical downward external load F with a magnitude of 100 N is applied at the midpoint of the right edge of the rectangular design domain. The entire left edge of the 80×50 design domain is fixed on both the x- and y-directions. The boundary condition of the cantilever beam can be defined using the following command to replace the command in Lines 195–198:

```
fixedNodes = find(p(:,1) == BDY(1,1));
forceNodes = find(p(:,1) == BDY(2,1) & p(:,2) == 0);
fixedDof = [2*fixedNodes-1; 2*fixedNodes];
```

Similarly, the load condition of the cantilever beam can be defined using the following command to replace the command in Line 202:

```
F = sparse(2*forceNodes,1,-100,2*length(p),1);
```

The initial pattern is also changed in Line 6 to expedite convergence.

```
phi = -(sin(xn/BDY(2,1)*5*pi) .* cos(yn/BDY(2,1)*5*pi) + 0.5);
```

Figure 6 displays the optimized configuration of a 2D cantilever beam using the provided Matlab code, executed with the designated input line:

```
GFLS262(80,50,0.1,0.5,10,2,1e5,0.3,4.3e-5,1.1,265).
```

The GFLS method effectively produces a highly optimized structure with an objective function value of 1.799 converged in 24 iterations. The volume fraction ratio of the solid materials is measured as 49.99%, which closely matches the desired value of 50%. The body-fitted mesh generated in each iteration precisely captures the smooth material boundaries without zig-zag shapes to take advantage of the level set method.

This work compares three alternative topology optimization methods, BESO, BFLS-UPWIND, and HJLS, with the

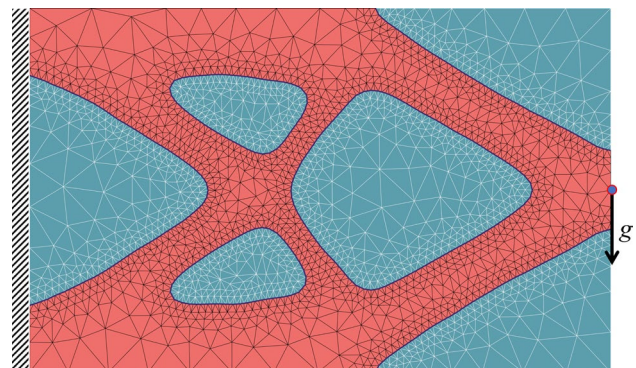





Fig. 6 The optimized configuration of the cantilever beam designs obtained by the proposed Matlab program GFLS262

optimized result shown in Fig. 7a obtained using the GFLS method. The comparison utilizes the same design domain dimensions, boundary conditions, and a similar number of nodes. Table 3 presents a comparison of the objective function values, iteration numbers, and computational efficiency of the GFLS method alongside the other methods. Regarding computational efficiency, it is important to consider the total run time required for each algorithm rather than focusing solely on iteration counts. Computational time per degree of freedom (DOF) for the Matlab programs GFLS262 and the other methods are evaluated using profile viewer commands. Additionally, iteration numbers to convergence are detailed in Table 3 to assess the performance of each algorithm. While parameter settings, initial patterns, and algorithmic enhancements influence iteration counts, this number remains crucial in evaluating the convergence speed, stability, and effectiveness of an optimization approach.

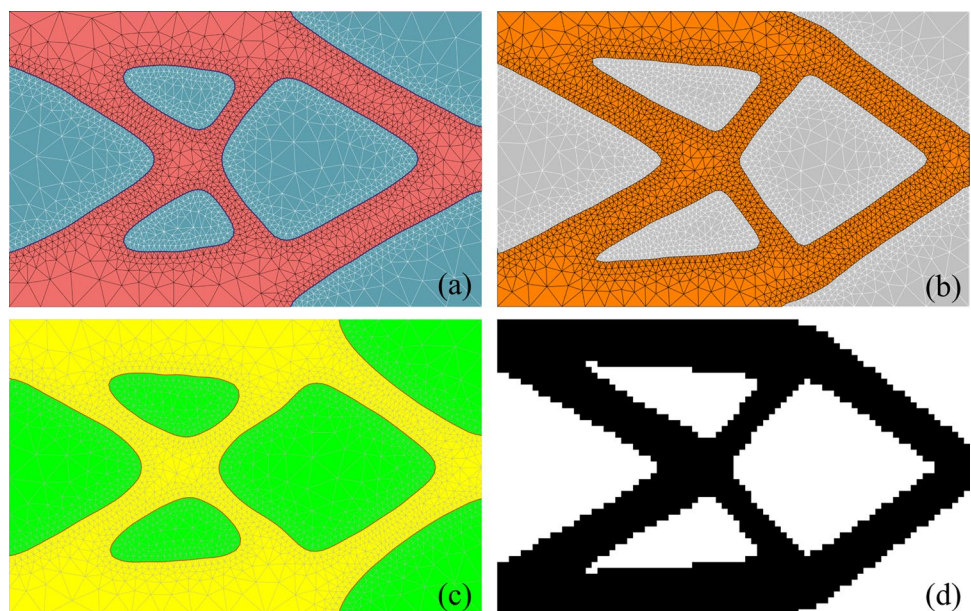
The Matlab program TriTOP172 integrates a body-fitted mesh and nonlinear diffusion regularization in the BESO method to eliminate zig-zag boundaries (Zhuang et al. 2022a, b). Figure 7b displays the optimized configuration of a cantilever beam using the TriTOP172 code. As indicated in Tables 3 and 4, the similar compliance values of the optimized structures achieved by GFLS262 and TriTOP172 are notably better than those obtained by other methods. This similarity suggests that the primary enhancement in performance stems from the capability of body-fitted mesh algorithms to adapt the mesh to the solution rather than from the specific choice between GFLS or BESO approaches. This observation underscores the significance of employing body-fitted meshes to achieve favorable outcomes in topology optimization. Moreover, the CPU computational time

Table 4 Optimization result comparison between the proposed GFLS, SIMP, BESO, and RDLS methods

Numerical Examples		Cantilever Beam	MBB Beam	L-bracket
GFLS 262	Iterations to convergence	24	22	21
	Objective function value	1.799	1.128	11.673
	Optimized Design			
BESO 101	Iterations to convergence	79	71	41
	Objective function value	1.864	1.187	11.905
	Improvement Ratio	3.49%	4.97%	1.95%
SIMP 99	Iterations to convergence	71	53	41
	Objective function value	1.917	1.250	12.223
	Improvement Ratio	6.16%	9.76%	4.50%
RDLS 88	Iterations to convergence	104	108	108
	Objective function value	1.896	1.186	11.867
	Improvement Ratio	5.12%	4.89%	1.63%

per DOF of the proposed GFLS262 code is 21.44% lower than that of the TriTOP172 code.

Fig. 7 The optimized configurations of the cantilever beam designs using the **a** GFLS method; **b** BESO method (Zhuang et al. 2022a, b); **c** BFLS-UPWIND method; and **d** HJLS method



Based on prior research (Allaire et al. 2014), this study adapts the conventional Level Set method into the Body-Fitted mesh framework (BFLS-UPWIND), incorporating upwind schemes and numerical interpolation. This approach involves interpolating level set function values to a virtual structured mesh to ensure compatibility between the traditional upwind scheme and the body-fitted mesh used in level set optimization. However, the iterative upwind schemes and excessive interpolation processes impact the algorithm accuracy and computational efficiency of the BFLS-UPWIND method during the level set function updating procedure. Figure 7c illustrates the optimized configuration achieved by the BFLS-UPWIND method, converging to an objective function value of 1.867 after 27 iterations. According to Table 3, compared to the GFLS method, the BFLS-UPWIND method achieves an objective function value 3.64% higher, with computational time being 43.39% longer for optimizing the cantilever beam.

The upwind scheme proves effective and efficient in level set topology optimization within fixed structured meshes (Allaire et al. 2004; Wang et al. 2003). Figure 7d provides the optimized structure of the Level Set method by solving the Hamilton–Jacobi equation (HJLS) under identical load and boundary conditions. The optimization adapts the 129-line Matlab code from Challis (2010) with fixed rectangular meshes. However, the resulting zig-zag boundaries do not conform well to the actual geometry, resulting in a 4.41% higher compliance value compared to the GFLS method. Despite additional tasks such as re-meshing and re-distancing that may extend total run times, the proposed GFLS method offers a computational cost savings of 8.39% over the HJLS method.

For clarity, we utilized Matlab commands to profile the execution time for the GFLS method. Figure 8 visually represents the cost distribution across different parts of the proposed Matlab program GFLS262. In the cantilever beam case study, the costs associated with remeshing, encompassing the discovery of contour nodes and Delaunay triangulation, constitute approximately 31% of the overall computational expenses during optimization. Identifying contour nodes comprises 7% of the computational costs, which is crucial for subsequent mesh generators to accurately capture material boundaries. Iterative Delaunay triangulation, accounting for 24% of the costs, transforms these nodes into a body-fitted mesh with user-defined mesh density. An efficient FEA procedure consumes 4% of the computational resources to assess structural responses under specific conditions. In the proposed GFLS method, we excluded low-sensitivity void regions from the time-consuming FEA, which means that FEA is only applied to the DOFs within the solid domain represented by SolidDof in the code, rather than the entire design space. In more complex optimization scenarios, especially those

COMPUTATIONAL COST BREAKDOWN

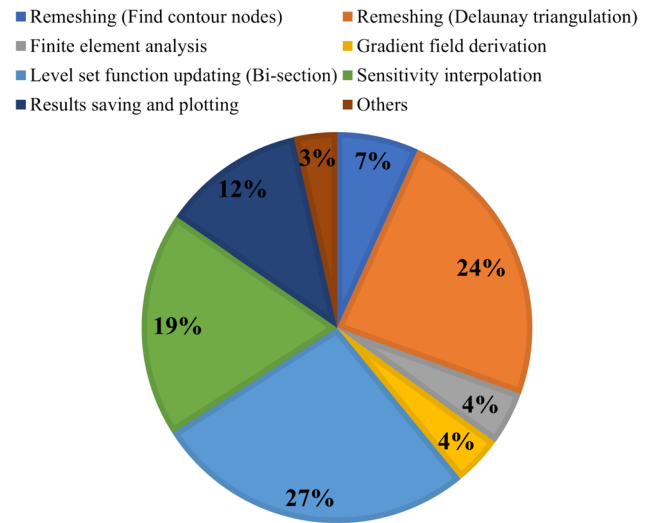


Fig. 8 The cost allocation pie chart for the proposed Matlab program GFLS262

involving extensive or non-linear FEA models, the computational cost of FEA would be considerably higher. The derivation of the gradient field represents another 4% of computational expenses, offering an efficient alternative to traditional upwind schemes. Interpolating sensitivities consume 16.8% of computational resources to obtain the sensitivity numbers on the nodes in the design domain. Updating the level set function using bi-section methods accounts for 27% of the computational costs, ensuring precise control over solid material volume and accelerating convergence. Finally, 12% of computational costs are dedicated to storing essential information such as node coordinates, element connectivity, and updated level set values in each iteration. This step is vital for analyzing optimization progress and visualizing optimized structures.

The following paragraph describes an optimization design for an L-bracket in a 100×100 rectangular design domain. The model is fixed on the top edge, and the downward loads are imposed at the bottom right corner of the passive regions. A passive region is located in a 60×60 square in the upper right corner of the design domain, where all elements should be defined as the void element. Based on the GFLS262 program, the nodes on the boundaries of the passive region can be found by adding the following command after Line 118.

```

px=0.2 * BDY(1,1); py=0.2 * BDY(1,2);
[xp,yp]=meshgrid(px,py:BDY(2,2));
[xp2,yp2]=meshgrid(px:BDY(2,1),py);
P=[xp(:),yp(:); xp2(:),yp2(:)];
Forcepts=[BDY(2,1) py; BDY(2,1)-1 py;BDY(2,1)-2
py;BDY(2,1)-3 py;BDY(2,1)-4 py; BDY(2,1)-5 py];

```


These nodes are fixed in the mesh generator by replacing the codes in Line 124.

```
pfix = [ c; P; Forcepts; BDY(2,1) BDY(1,2); BDY(1,1)
BDY(2,2); BDY(1,1) BDY(1,2); BDY(2,1) BDY(2,2)];
```

The edge length L should be limited to a specific range using the following command after Line 142:

```
L = min(max(0.1,L),8);
```

The passive elements tp , non-passive void elements tv , void elements $t1$, and solid elements $t2$ are defined using the following command to replace Lines 155–157.

```
tnp = t(~(pmid(:,1) > px & pmid(:,2) > py),:);
tp = t((pmid(:,1) > px & pmid(:,2) > py),:);
Passive = length(tp);
dEnp = dE(~(pmid(:,1) > px & pmid(:,2) > py),:);
tv = tnp(dEnp > 0,:);
t1 = [tp; tnp(dEnp > 0,:); t2 = tnp(dEnp < 0,:);
t = [t1;t2];
```

In addition, Lines 9, 24, 61, and 115 are replaced to define passive elements tp and non-passive void elements tv as the output of the function `GenerateMesh`. Line 9 is replaced by:

```
[p,t,t1,t2,Ve,pmid,tv,tp]=GenerateMesh(xn,yn,c,phi,maxedge,minedge,1.2,BDY,80);
```

Line 24 is replaced by:

```
patch('Faces',tv, 'Vertices',p, 'EdgeColor',[250 250
250]/255, 'FaceColor',[92 158 173]/255);
```

Line 61 is replaced by:

```
[p,t,t1,t2,Ve,pmid,tv,tp]=GenerateMesh(xn,yn,c,phi,maxedge,minedge,1.2,BDY,80);
```

Line 115 is replaced by:

```
function [p,t,t1,t2,Ve,pmid,tv,tp]=GenerateMesh(xn,yn,c,dN,maxedge,minedge,fscale,BDY,maxiter).
```

Next, Lines 184–185 in the finite element analysis are replaced by:

```
for i = 1:NT.
if i <= length(t1) x = 1e-5; else x = 1; end.
KK(:,6*i-5:6*i) = x* GetmatrixKe(p(t(i,:),1),p(t(i,:),2),
E,nu);
```

The following command is required to replace Line 50 during the bi-section procedure of the GFLS algorithm.

```
xnew = max(0,sign(mean(phiNnew(t),2)));
xnew(1:length(tp)) = 0;
volVoid = dot(xnew,Ve');
```

The load and boundary condition of the L-bracket design replaces Lines 195–202 as:

```
fixedNodes = find(p(:,2) == BDY(2,1));
fixedDof = [2*fixedNodes-1; 2*fixedNodes];
SolidDof = 1:2*length(p);
forceNodes = find(p(:,1) <= BDY(2,1) &
p(:,1) >= BDY(2,1)-5 & p(:,2) == 0.2*BDY(1,2));
freeDofs = setdiff(SolidDof, fixedDof);
U = zeros(2*length(p),1);
F = sparse(2*forceNodes,1,-0.1,2*length(p),1);
```

Lastly, the initial pattern of the L-bracket is defined in Line 6 as follows.

```
phi = -(sin(xn/BDY(2,1)*5*pi).*cos(yn/BDY(2,1)*5*pi)+0.5);
```

The threshold parameters for the distance between adjacent nodes on the boundary curve are changed in Line 7 as follows:

```
d1 = 0.6; d2 = 1.2;
```

The optimized configuration of an L-bracket design is illustrated in Fig. 9, obtained through the execution of the Matlab code `GFLS262` with the specified input line:

```
GFLS262(100,100,0.05,0.4,10,2.5,2,0.3,6e-4,1,250).
```

The optimized design of the L-bracket is achieved within 21 iterations using the GFLS method, resulting in a compliance value of 11.673 and a solid volume fraction ratio of 40.02%. The body-fitted mesh precisely captures the smooth and refined boundaries, significantly improving the objective function value and increasing the analysis accuracy. Compared to the previous SIMP 99-line code (Sigmund 2001), BESO 101-line code (Huang and Xie 2010), and reaction–diffusion level set (RDLS) 88-line code (Otomori et al. 2014), the proposed GFLS methodology can generate a 1.63–9.76% better compliance value for the identical numerical example, as illustrated in Table 4. Meanwhile, the iteration number required before convergence is significantly reduced, saving computational cost. The improvement is primarily attributed to the superior mesh adaptivity, high computational efficiency, and enhanced compatibility of the proposed GFLS method. The proposed GFLS method is demonstrated to be effective and robust in achieving decent

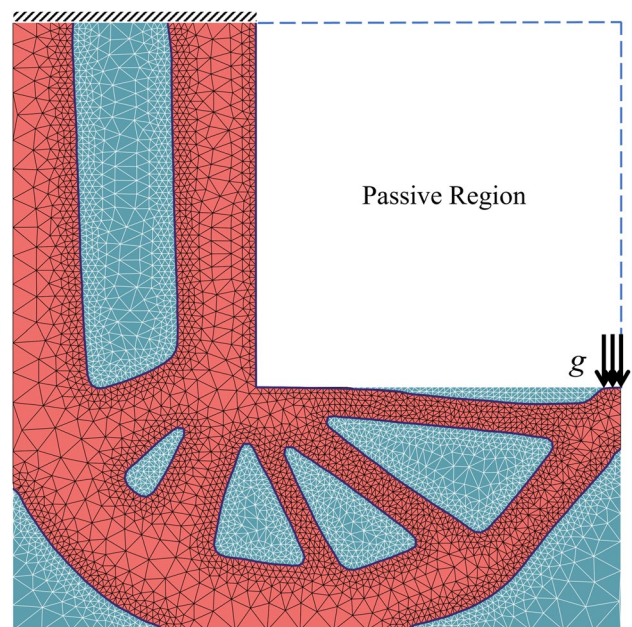


Fig. 9 The optimized configuration of the L-bracket designs obtained by the proposed Matlab program `GFLS262`

objective function values and smooth boundaries through the provided numerical examples. Figure 10 illustrates the convergence history of the optimization problems. The command can be added after Line 30 to store the body-fitted mesh and optimization history information for plotting figures.

```
save('History.mat', 'Obj', 'volt', 'p', 't', 't1', 't2');
```

The compliance values of structures optimized with body-fitted meshes are typically better due to improved mesh quality. Sharp corner details presented in conventional approaches utilizing fixed rectangular meshes can introduce numerical issues, particularly in simulations involving boundary-dependent quantities like traction loads (Andreasen et al. 2020; Soares da Costa Azevêdo et al. 2024). Body-fitted meshes are designed to conform closely to material boundaries, resulting in a more precise representation of complex shapes and smoother transitions between elements. Thus, the finite element model built on a body-fitted mesh tends to capture structural behavior more accurately during optimization processes, generating improved compliance compared to designs on the square grid. The result obtained by the SIMP method is post-processed to ensure volume-preserving thresholding, ensuring a fair comparison based on thresholded 0–1 solutions (Sigmund 2022). The proposed body-fitted meshing approach can be extended to 3D scenarios using the unstructured tetrahedral mesh (Zhuang et al. 2022a, b). Meanwhile, the open-source Mmg library, freely available on <http://www.mmgtools.org/>, addresses mesh evolution challenges in the context of free and moving boundary problems across 2D and 3D dimensions (Dapogny et al. 2014; Dobrzynski and Frey 2008).

In another instance, the focus is on achieving an optimal design of a compliant mechanism, which functions as an

efficient gripper. As shown in Fig. 11a, a predefined non-design region measuring $25 \times 26 \times 1$ is allocated within a $100 \times 100 \times 1$ design domain for workpiece gripping. This section leverages the GFLS algorithm to achieve the desired objective function of displacement at the corners of the passive region under volume constraints. An external force of magnitude g is applied horizontally at the midpoint of the left edge, with the top and bottom corners of this edge being immovably fixed. The input parameters are defined as follows: material removal ratio $V_r = 0.05$; maximum volume fraction $\text{volReq} = 0.25$; Young's modulus $E = 0.1$; Poisson's ratio $\nu = 0.3$; and curvature term coefficient $\tau = 4 \times 10^{-3}$. As illustrated in Fig. 11b, an optimized gripper is achieved with a volume fraction of 25.16% after 39 iterations. Well-defined material boundaries between solid and void domains ensure smooth transitions, resulting in a higher displacement value at the output port, specifically 1.009×10^{-2} . Figure 11c, d illustrate the optimization results using the BESO method with a fixed mesh and the RDLS method with a body-fitted mesh. The displacement values obtained by the RDLS and BESO methods are 9.833×10^{-3} and 9.180×10^{-3} , respectively, demonstrating reduction ratios of 2.55% and 9.02% compared to the GFLS method.

Furthermore, the proposed GFLS method demonstrates promising capabilities in representing smooth boundaries, which is particularly beneficial for addressing stress-constrained optimization challenges. This paper illustrates an optimization example involving a T-Pier column, where a 50% volume constraint and a maximum allowable von Mises stress (σ_{vM}^*) of 0.5 for all elements are enforced. Details regarding the design domain dimensions and boundary conditions are depicted in Fig. 12a. The T-Pier structure is fixed at the bottom of the design domain, subjected to uniform

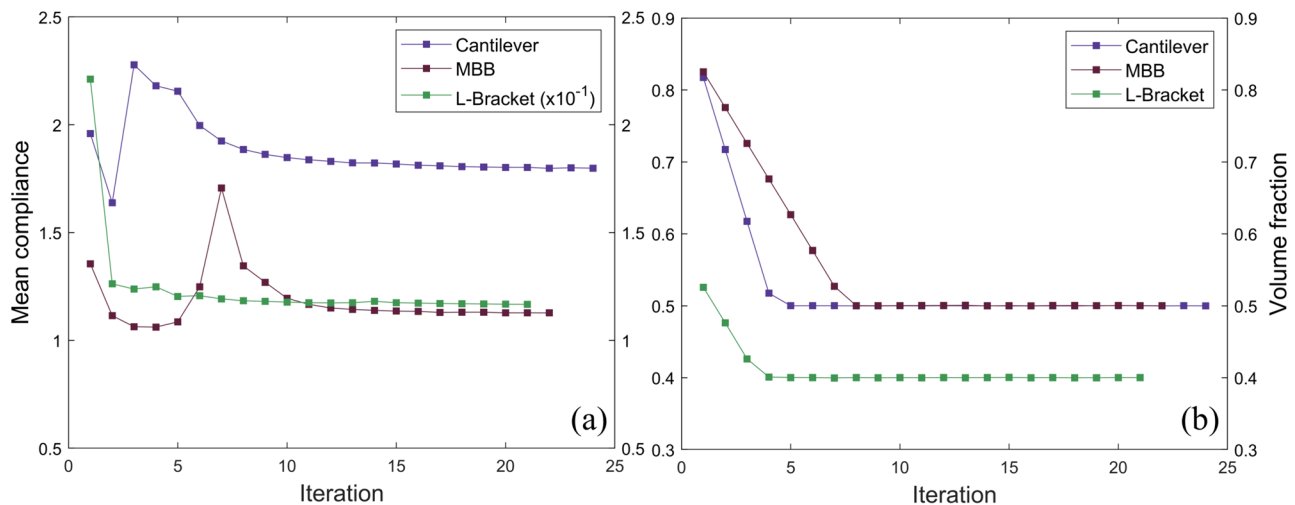
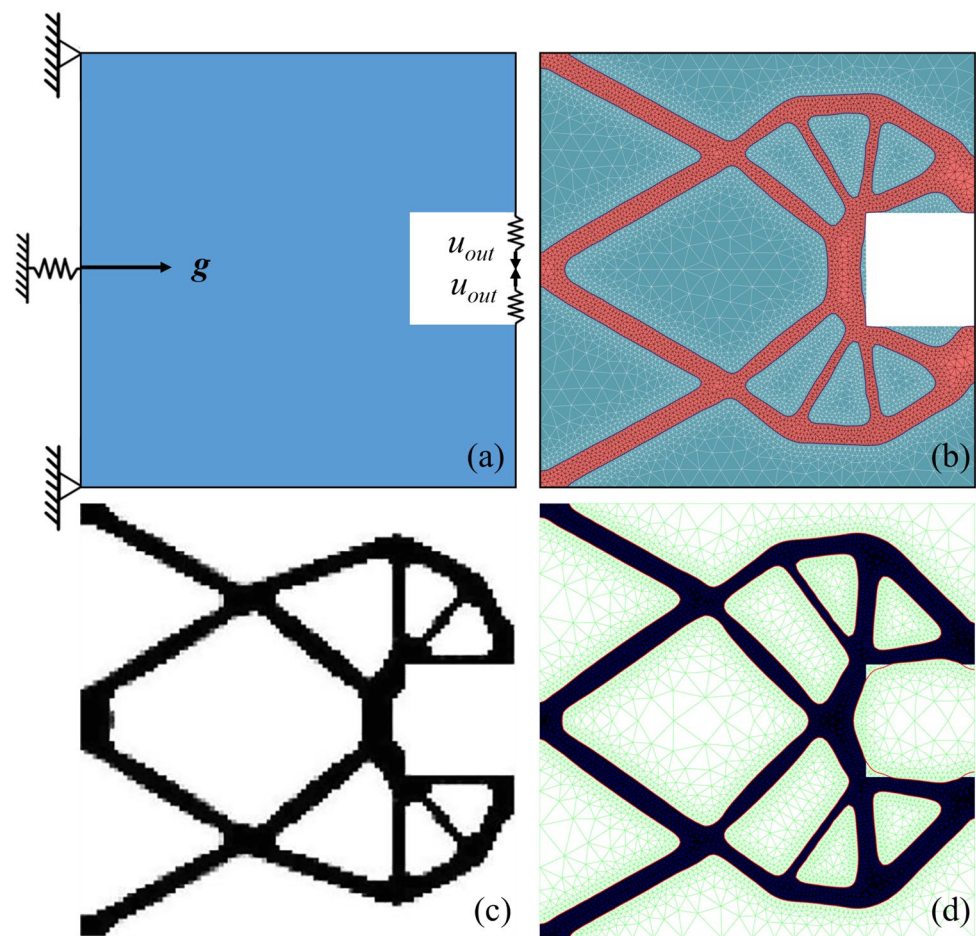


Fig. 10 Convergence history of the numerical examples using the GFLS method: **a** objective function value and **b** volume fraction of the solid materials

Fig. 11 **a**The boundary conditions and the optimization results of the gripper mechanism using the **b** proposed GFLS method with a body-fitted mesh; **c** BESO method with a fixed mesh (Li 2014); **d** RDLS method with a body-fitted mesh (Zhuang et al. 2021)



loads on its top surface, with a 240×15 rectangular non-design domain near the surface.

Figure 12b depicts the optimized configuration achieved with the proposed GFLS method employing body-fitted meshes, facilitating the precise application of boundary conditions at structural interfaces. Compared to the optimized structure using a fixed rectangular mesh displayed in Fig. 12c, there is a 2.42% improvement in the objective function value, decreasing from 629.648 to 614.439. Fixed rectangular meshes, especially when dealing with irregular or highly curved geometries, may not align well with the actual shape, leading to inaccurate representations of stress and strain distributions. As shown in Fig. 12c, the non-optimal stress concentration areas highlighted in red circles contribute to an increase in the maximum von Mises stress. Body-fitted meshes ensure accurate transmission of forces and constraints throughout the model by aligning mesh elements with the geometric contours. This alignment results in more realistic von Mises stress distributions, as demonstrated in Fig. 12c, d, where colored von Mises stress fields are compared between optimized structures using fixed rectangular and body-fitted triangular meshes with the same color map. Notably, the maximum von Mises stress within

the body-fitted elements shows an 8.55% reduction from 0.351 to 0.321.

4 Conclusions

This educational article presents an efficient Matlab program comprising 262 lines, specifically designed for the level set topology optimization. The level set method, known for its ability to represent complex geometries implicitly, has proven to be a valuable approach in structural optimization. The proposed GFLS method has great potential to solve complex problems of fluid–structure interaction, aerodynamics, and additive manufacturing with the body-fitted mesh. This approach incorporates several key improvements, including updated body-fitted meshing, the integration of the gradient field derivation for design variable updates, and the inclusion of mean curvature values in sensitivity analysis. The capability of the proposed code is showcased through illustrative examples, highlighting its efficacy in optimizing structures while adhering to various objective functions, boundary conditions, and design constraints. The simplicity and compactness of this program ensure that users can

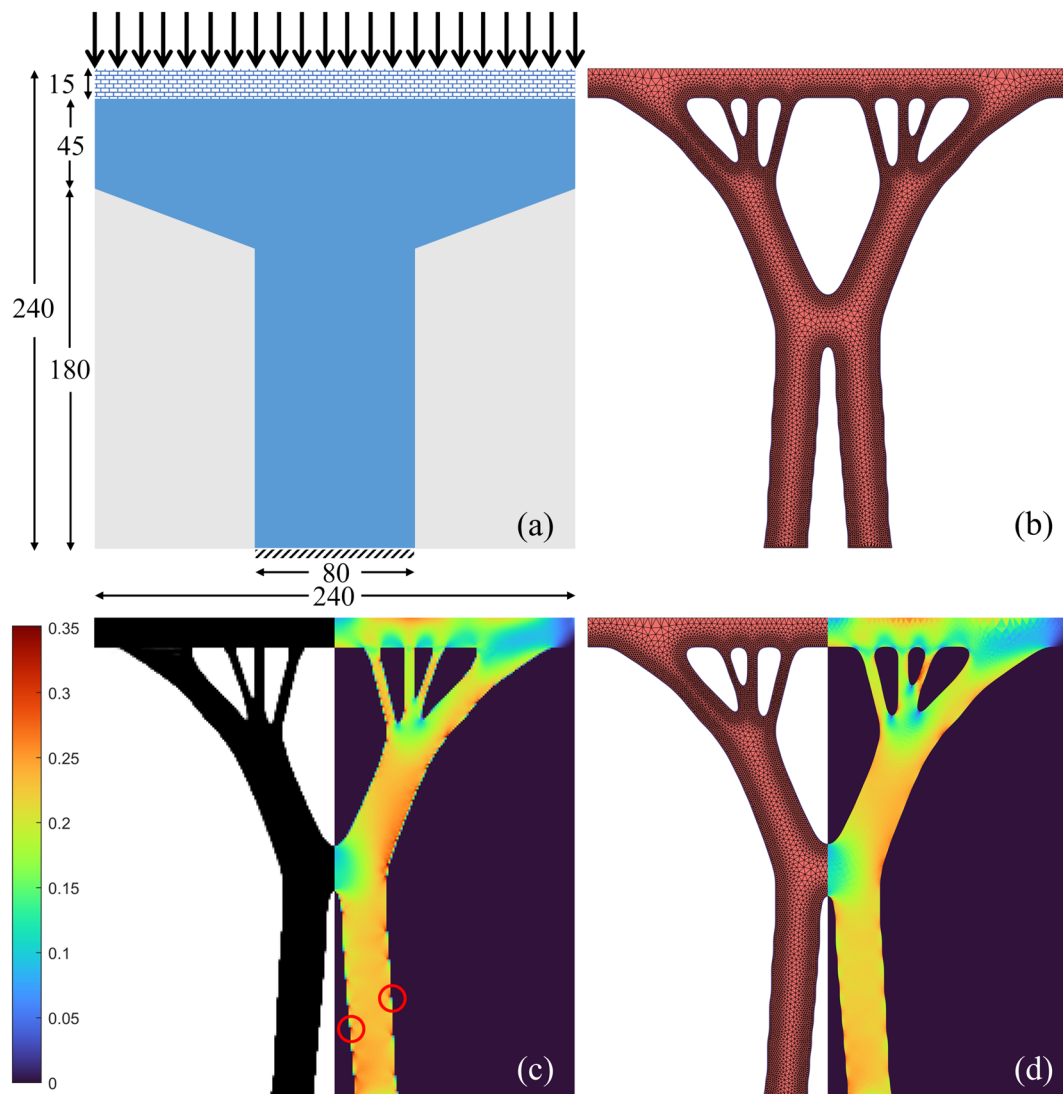


Fig. 12 **a** The design domain dimension and boundary conditions of a T-Pier column example; **b** the optimization results using the proposed GFLS method with a body-fitted mesh; **c** colored von Mises

stress field for the optimized structure using a fixed rectangular mesh; **d** colored von Mises stress field for the GFLS optimization result

readily integrate it into their existing workflows, thereby promoting further advancements in structural optimization.

Appendix–Matlab Code GFLS262

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s00158-024-03891-y>.

Acknowledgements This work was supported by the Hong Kong Polytechnic University (P0044299).

Funding Open access funding provided by The Hong Kong Polytechnic University, Hong Kong Polytechnic University, P0044299, Zicheng Zhuang

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there are no competing interests to declare that are relevant to the content of this article.

Replication of results All numeric examples within this paper were conducted using the provided Matlab program in the appendix. The information required to replicate the results has been provided in the supplementary materials.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing,

adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Allaire G, Jouve F, Toader A-M (2002) A level set method for shape optimization. *CR Math* 334:1125–1130. [https://doi.org/10.1016/S1631-073X\(02\)02412-3](https://doi.org/10.1016/S1631-073X(02)02412-3)
- Allaire G, Jouve F, Toader A-M (2004) Structural optimization using sensitivity analysis and a level-set method. *J Comput Phys* 194(1):363–393. <https://doi.org/10.1016/j.jcp.2003.09.032>
- Allaire G, Dapogny C, Frey P (2011) Topology and geometry optimization of elastic structures by exact deformation of simplicial mesh. *CR Math* 349:999–1003. <https://doi.org/10.1016/j.crma.2011.08.012>
- Allaire G, Dapogny C, Frey P (2013) A mesh evolution algorithm based on the level set method for geometry and topology optimization. *Struct Multidisc Optim* 48(4):711–715. <https://doi.org/10.1007/s00158-013-0929-2>
- Allaire G, Dapogny C, Frey P (2014) Shape optimization with a level set based mesh evolution method. *Comput Methods Appl Mech Eng* 282:22–53. <https://doi.org/10.1016/j.cma.2014.08.028>
- Andreassen CS, Elingaard MO, Aage N (2020) Level set topology and shape optimization by density methods using cut elements with length scale control. *Struct Multidisc Optim* 62(2):685–707. <https://doi.org/10.1007/s00158-020-02527-1>
- Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in MATLAB using 88 lines of code. *Struct Multidisc Optim* 43(1):1–16. <https://doi.org/10.1007/s00158-010-0594-7>
- Azari Nejat A, Held A, Trekel N, Seifried R (2022) A modified level set method for topology optimization of sparsely-filled and slender structures. *Struct Multidisc Optim* 65(3):85. <https://doi.org/10.1007/s00158-022-03184-2>
- Baiges J, Martínez-Frutos J, Herrero-Pérez D, Otero F, Ferrer A (2019) Large-scale stochastic topology optimization using adaptive mesh refinement and coarsening through a two-level parallelization scheme. *Comput Methods Appl Mech Eng* 343:186–206. <https://doi.org/10.1016/j.cma.2018.08.028>
- Bendsoe MP (1989) Optimal shape design as a material distribution problem. *Structural Optimization* 1(4):193–202. <https://doi.org/10.1007/BF01650949>
- Bendsoe 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](https://doi.org/10.1016/0045-7825(88)90086-2)
- Bendsoe MP and Sigmund O (2004) Topology optimization: Theory, methods, and applications. <https://doi.org/10.1007/978-3-662-05086-6>
- Bruggi M, Duysinx P (2012) Topology optimization for minimum weight with compliance and stress constraints. *Struct Multidisc Optim* 46(3):369–384. <https://doi.org/10.1007/s00158-012-0759-7>
- Challis VJ (2010) A discrete level-set topology optimization code written in Matlab. *Struct Multidisc Optim* 41(3):453–464. <https://doi.org/10.1007/s00158-009-0430-0>
- Challis VJ, Roberts AP, Wilkins AH (2008) Design of three dimensional isotropic microstructures for maximized stiffness and conductivity. *Int J Solids Struct* 45(14):4130–4146. <https://doi.org/10.1016/j.ijsolstr.2008.02.025>
- Chen A, Cai K, Zhao Z-L, Zhou Y, Xia L, Xie M (2021) Controlling the maximum first principal stress in topology optimization. *Struct Multidisc Optim* 63(1):327–339. <https://doi.org/10.1007/s00158-020-02701-5>
- Christiansen AN, Nobel-Jørgensen M, Aage N, Sigmund O, Bærentzen JA (2014) Topology optimization using an explicit interface representation. *Struct Multidisc Optim* 49(3):387–399. <https://doi.org/10.1007/s00158-013-0983-9>
- da Costa S, Azevêdo A, Li H, Ishida N, Siqueira LO, Cortez RL, Nelli Silva EC, Nishiwaki S, Picelli R (2024) Body-fitted topology optimization via integer linear programming using surface capturing techniques. *Int J Numer Meth Eng* 125(13):e7480. <https://doi.org/10.1002/nme.7480>
- Dapogny C, Frey P (2012) Computation of the signed distance function to a discrete contour on adapted triangulation. *Calcol* 49(3):193–219. <https://doi.org/10.1007/s10092-011-0051-z>
- Dapogny C, Dobrzynski C, Frey P (2014) Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *J Comput Phys* 262:358–378. <https://doi.org/10.1016/j.jcp.2014.01.005>
- Dobrzynski C and Frey P (2008) Anisotropic delaunay mesh adaptation for unsteady simulations. *Proceedings of the 17th International Meshing Roundtable*:177–194.
- Dorn WS, Gomory RE, Greenberg HJ (1964) Automatic design of optimal structures. *J De Mecanique* 3(1):25–52
- Fan Z, Xia L, Lai W, Xia Q, Shi T (2019) Evolutionary topology optimization of continuum structures with stress constraints. *Struct Multidisc Optim* 59(2):647–658. <https://doi.org/10.1007/s00158-018-2090-4>
- Ferrari F, Sigmund O (2020) A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D. *Struct Multidisc Optim* 62(4):2211–2228. <https://doi.org/10.1007/s00158-020-02629-w>
- Glowinski R (1984) Numerical simulation for some applied problems originating from continuum mechanics, in trends in applications of pure mathematics to mechanics. *symp., Palaiseau/France 1983. Lect Notes Phys* 195:96–145
- Goodman J, Kohn RV, Reyna L (1986) Numerical study of a relaxed variational problem from optimal design. *Comput Methods Appl Mech Eng* 57(1):107–127. [https://doi.org/10.1016/0045-7825\(86\)90073-3](https://doi.org/10.1016/0045-7825(86)90073-3)
- He Y, Zhao Z-L, Lin X, Xie YM (2023) A hole-filling based approach to controlling structural complexity in topology optimization. *Comput Methods Appl Mech Eng* 416:116391. <https://doi.org/10.1016/j.cma.2023.116391>
- Holmberg E, Torstenfelt B, Klarbring A (2013) Stress constrained topology optimization. *Struct Multidisc Optim* 48(1):33–47. <https://doi.org/10.1007/s00158-012-0880-7>
- Huang X, Xie YM (2009) Bidirectional evolutionary topology optimization of continuum structures with one or multiple materials. *Comput Mech* 43:393–401. <https://doi.org/10.1007/s00466-008-0312-0>
- Huang X, Xie YM, Burry MC (2006) A new algorithm for bi-directional evolutionary structural optimization. *JSM Int j, Ser C* 49(4):1091–1099. <https://doi.org/10.1299/jsmec.49.1091>
- Huang X and Xie YM (2010) Evolutionary topology optimization of continuum structures: methods and applications. <https://doi.org/10.1002/9780470689486.ch2>

- Jiang Y, Zhao M (2020) Topology optimization under design-dependent loads with the parameterized level-set method based on radial-basis functions. *Comput Methods Appl Mech Eng* 369:113235. <https://doi.org/10.1016/j.cma.2020.113235>
- Kikuchi N, Chung KY, Torigaki T, Taylor JE (1986) Adaptive finite element methods for shape optimization of linearly elastic structures. *Comput Methods Appl Mech Eng*. https://doi.org/10.1007/978-1-4615-9483-3_6
- Li H, Yamada T, Jolivet P, Furuta K, Kondoh T, Izui K, Nishiwaki S (2021) Full-scale 3D structural topology optimization using adaptive mesh refinement based on the level-set method. *Finite Elem Anal Des* 194:103561. <https://doi.org/10.1016/j.finel.2021.103561>
- Li Y (2014) Topology optimization of compliant mechanisms based on the BESO method.
- Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. *Struct Multidisc Optim* 50(6):1175–1196. <https://doi.org/10.1007/s00158-014-1107-x>
- Luo J, Luo Z, Chen S, Tong L, Wang MY (2008) A new level set method for systematic design of hinge-free compliant mechanisms. *Comput Methods Appl Mech Eng* 198(2):318–331. <https://doi.org/10.1016/j.cma.2008.08.003>
- Lurie KA, Cherkav AV, Fedorov AV (1982) Regularization of optimal design problems for bars and plates, part 2. *J Optim Theory Appl* 37(4):523–543. <https://doi.org/10.1007/BF00934954>
- Mancinelli C, Livesu M, Puppo E (2018) Gradient field estimation on triangle meshes. *Smart Tools Apps Graphics*. <https://doi.org/10.2312/stag.20181301>
- Museth K, Breen DE, Whitaker RT, Mauch S, Johnson D (2005) Algorithms for interactive editing of level set models. *Comput Graphics Forum* 24(4):821–841. <https://doi.org/10.1111/j.1467-8659.2005.00904.x>
- Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J Comput Phys* 79(1):12–49. [https://doi.org/10.1016/0021-9991\(88\)90002-2](https://doi.org/10.1016/0021-9991(88)90002-2)
- Otomori M, Yamada T, Izui K, Nishiwaki S (2014) Matlab code for a level set-based topology optimization method using a reaction diffusion equation. *Struct Multidisc Optim* 51:1159–1172. <https://doi.org/10.1007/s00158-014-1190-z>
- Persson P-O, Strang G (2004) A simple mesh generator in MATLAB. *SIAM Rev* 46:329–345. <https://doi.org/10.1137/S0036144503429121>
- Salazar de Troya MA, Tortorelli DA (2018) Adaptive mesh refinement in stress-constrained topology optimization. *Struct Multidisc Optim* 58(6):2369–2386. <https://doi.org/10.1007/s00158-018-2084-2>
- Sethian JA, Wiegmann A (2000) Structural boundary design via level set and immersed interface methods. *J Comput Phys* 163(2):489–528. <https://doi.org/10.1006/jcph.2000.6581>
- Sethian JA (1999) Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.
- Sigmund O (2001) A 99 line topology optimization code written in Matlab. *Struct Multidisc Optim* 21(2):120–127. <https://doi.org/10.1007/s001580050176>
- Sigmund O (2022) On benchmarking and good scientific practise in topology optimization. *Struct Multidisc Optim* 65(11):315. <https://doi.org/10.1007/s00158-022-03427-2>
- Sigmund O, Maute K (2013) Topology optimization approaches. *Struct Multidisc Optim* 48(6):1031–1055. <https://doi.org/10.1007/s00158-013-0978-6>
- Simon J (1980) Differentiation with respect to the domain in boundary value problems. *Numer Funct Anal Optim* 2(7–8):649–687. <https://doi.org/10.1080/01630563.1980.10120631>
- Sokolowski J and Zolesio J-P (1992) Introduction to shape optimization. 10(<https://doi.org/10.1137/1.9780898718690>)
- 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](https://doi.org/10.1016/0045-7825(91)90245-2)
- Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. *Int J Numer Meth Eng* 24:359–373
- Talisch C, Paulino GH, Pereira A, Menezes IFM (2012) PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab. *Struct Multidisc Optim* 45(3):309–328. <https://doi.org/10.1007/s00158-011-0706-z>
- van Dijk NP, Yoon GH, van Keulen F, Langelaar M (2010) A level-set based topology optimization using the element connectivity parameterization method. *Struct Multidisc Optim* 42(2):269–282. <https://doi.org/10.1007/s00158-010-0485-y>
- 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](https://doi.org/10.1016/S0045-7825(02)00559-5)
- Wang SY, Lim KM, Khoo BC, Wang MY (2007) An extended level set method for shape and topology optimization. *J Comput Phys* 221(1):395–421. <https://doi.org/10.1016/j.jcp.2006.06.029>
- Wang C, Xie YM, Lin X, Zhou S (2022) A reaction diffusion-based B-spline level set (RDBLS) method for structural topology optimization. *Comput Methods Appl Mech Eng* 398:115252. <https://doi.org/10.1016/j.cma.2022.115252>
- 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 Multidisc Optim* 58(2):831–849. <https://doi.org/10.1007/s00158-018-1904-8>
- Xia L, Zhang L, Xia Q, Shi T (2018) Stress-based topology optimization using bi-directional evolutionary structural optimization method. *Comput Methods Appl Mech Eng* 333:356–370. <https://doi.org/10.1016/j.cma.2018.01.035>
- Xie YM and Steven GP (1992) Shape and layout optimization via an evolutionary procedure. *Proceedings of the International Conference on Computational Engineering Science*.
- 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](https://doi.org/10.1016/0045-7949(93)90035-C)
- Xie YM, Steven GP (1996) Evolutionary structural optimization for dynamic problems. *Comput Struct* 58(6):1067–1073. [https://doi.org/10.1016/0045-7949\(95\)00235-9](https://doi.org/10.1016/0045-7949(95)00235-9)
- Xiong Y, Zhao Z-L, Lu H, Shen W, Xie YM (2023) Parallel BESO framework for solving high-resolution topology optimisation problems. *Adv Eng Softw* 176:103389. <https://doi.org/10.1016/j.advengsoft.2022.103389>
- Yamada T, Izui K, Nishiwaki S, Takezawa A (2010) A topology optimization method based on the level set method incorporating a fictitious interface energy. *Comput Methods Appl Mech Eng* 199(45–48):2876–2891. <https://doi.org/10.1016/j.cma.2010.05.013>
- Zhang S, Gain AL, Norato JA (2020) Adaptive mesh refinement for topology optimization with discrete geometric components. *Comput Methods Appl Mech Eng* 364:112930. <https://doi.org/10.1016/j.cma.2020.112930>
- Zhuang Z, Xie YM, Zhou S (2021) A reaction diffusion-based level set method using body-fitted mesh for structural topology optimization. *Comput Methods Appl Mech Eng* 381:113829. <https://doi.org/10.1016/j.cma.2021.113829>
- Zhuang Z, Xie YM, Li Q, Zhou S (2022a) A 172-line Matlab code for structural topology optimization in the body-fitted mesh. *Struct Multidisc Optim* 66(1):11. <https://doi.org/10.1007/s00158-022-03464-x>
- Zhuang Z, Xie YM, Li Q, Zhou S (2022b) Body-fitted bi-directional evolutionary structural optimization using nonlinear diffusion

- regularization. *Comput Methods Appl Mech Eng* 396:115114. <https://doi.org/10.1016/j.cma.2022.115114>
- Zhuang Z, Weng Y, Xie YM, Wang C, Zhang X, Zhou S (2024) A node moving-based structural topology optimization method in the body-fitted mesh. *Comput Methods Appl Mech Eng* 419:116663. <https://doi.org/10.1016/j.cma.2023.116663>
- 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>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.