Japan Journal of Industrial and Applied Mathematics (2023) 40:877–905
https://doi.org/10.1007/s13160-023-00563-0

ORIGINAL PAPER

Inertial projected gradient method for large-scale topology optimization

Akatsuki Nishioka[1] · Yoshihiro Kanno[1,2]

Received: 9 May 2022 / Revised: 16 November 2022 / Accepted: 6 January 2023 /
Published online: 16 February 2023

Abstract
We present an inertial projected gradient method for solving large-scale topology optimization problems. We consider the compliance minimization problem, the heat conduction problem and the compliant mechanism problem of continua. We use the projected gradient method to efficiently treat the linear constraints of these problems. Also, inertial techniques are used to accelerate the convergence of the method. We consider an adaptive step size policy to further reduce the computational cost. The proposed method has a global convergence property. By numerical experiments, we show that the proposed method converges fast to a point satisfying the first-order optimality condition with high accuracy compared with the existing methods. The proposed method has a low computational cost per iteration, and is thus effective in a large-scale problem.

Keywords Topology optimization · First-order optimization method · Projected gradient method · Inertial algorithm · Accelerated gradient method

Mathematics Subject Classification 65K10 · 90C26 · 90C90

1 Introduction

Topology optimization is a method to obtain an optimal structural design depending on the objective by mathematical programming. The extensive study of topology optimization dates back to the seminal work by Bendsøe and Kikuchi [6] in 1988. Since then, a wide range of applications have been suggested in fluid, heat, elec-

✉ Akatsuki Nishioka
akatsuki_nishioka@mist.i.u-tokyo.ac.jp

1 Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Hongo, 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan

2 Mathematics and Informatics Center, The University of Tokyo, Hongo, 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan

tromagnetic, acoustic, and aerospace engineering [7, 13]. A topology optimization problem of continua is formulated as an infinite-dimensional optimization problem. We can discretize the problem by the finite element method and obtain a conventional finite-dimensional optimization problem [7, 9]. The discretized problem is a large-scale nonconvex optimization problem with some constraints. Moreover, it requires the *finite element analysis* (FEA), which is a solution of a linear equation, for calculating the objective function value and the gradient of the objective function at each iteration. This property makes the computational cost of topology optimization even larger.

There are various types of approaches to reduce the computational cost of topology optimization. In topology optimization, most of the computational cost is spent on FEA. Accordingly, there are many studies on reducing the computational cost of FEA [11, 34–36, 38, 39].

In this paper, we attempt to reduce the computational cost by reducing the number of iterations using an efficient and faster optimization algorithm. In a large-scale topology optimization problem, common nonlinear optimization algorithms such as the interior-point method and the sequential quadratic programming are often impractical because of the huge iteration cost (computational cost per iteration). Therefore, algorithms designed specifically for structural (topology) optimization such as the optimality criteria method [6] and the method of moving asymptotes [31, 32] are commonly used. See [27] for a comparative study on the optimization algorithms for topology optimization. Some studies on faster optimization algorithms for topology optimization are found in [20, 28].

Recently, first-order optimization methods which only require the first-order derivatives of the objective (and constraint) functions have been attracting much attention in the machine learning community. First-order methods are suited for large-scale problems because of their low iteration cost in time and memory storage. Second-order methods such as the Newton method and the interior-point method require the (approximated) second-order derivative and solution of linear equations. The iteration cost grows drastically as the problem size increases, and thus second-order methods are impractical for a large-scale problem. Although the convergence of first-order methods is basically slower than that of second-order methods, there are studies on accelerating the convergence of first-order methods. For an unconstrained convex optimization problem where the objective function f has Lipschitz continuous gradient (also called that the objective function is L-smooth), Nesterov's accelerated gradient method [23] achieves the convergence rate at $f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*) \leq O(1/k^2)$, while the steepest descent method converges with rate $O(1/k)$ where k is the iteration counter and $\boldsymbol{x}^*$ is the optimal solution. The above convergence rate is often equivalently described by the iteration complexity: $O(1/\epsilon^{1/2})$ iterations to acheive $f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*) \leq \epsilon$. Beck and Teboulle [4] combined Nesterov's acceleration technique with the proximal gradient method for convex optimization problems, which is a generalization of the projected gradient method, to treat simple nondifferentiable functions and constraints.

The accelerated gradient method has also been extended to an optimization problem with a nonconvex objective function with Lipschitz continuous gradient. In unconstrained nonconvex optimization, the optimality measure $f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*)$ used in convex optimization is not appropriate since there may exist multiple local minima. Therefore, the number of iterations k required to acheive $\|\nabla f(\boldsymbol{x}^k)\| \leq \epsilon$ (the iteration complexity) is considered. The steepest descent method has $O(1/\epsilon^2)$ iteration complexity. The accelerated gradient methods by [15, 18] have the same order of iteration complexity as the steepest descent method in nonconvex case, but have the accelerated convergence rate in convex case same as Nesterov's accelerated gradient method. Although they do not have theoretically improved convergence rates in nonconvex optimization, their empirical performance is expected to be better than that of first-order methods without acceleration since the iteration complexity is worst-case under all L-smooth functions f. With the additional assumption of Lipschitz continuity of the Hessian of the objective function, Carmon et al. [10] acheived an improved iteration complexity of $O(\epsilon^{-7/4}\log(1/\epsilon))$ and Li and Lin [19] acheived $O(\epsilon^{-7/4})$. The former method requires more complicated update scheme. The methods by [10, 19] only treat unconstrained problems. The accelerated gradient methods for nonconvex optimization are still developing. See [3, 12, 22] for more details in first-order methods and their acceleration.

Although the accelerated proximal gradient method has been applied to optimization problems in computational plasticity [17, 29, 30], there are very few applications to topology optimization. Li and Zhang [21] applied the accelerated mirror descent method to a robust topology optimization problem under stochastic load uncertainty. They used stochastic optimization techniques to efficiently obtain a robust design. However, as they applied a convex optimization algorithm to nonconvex optimization problems, the convergence of the method is not guaranteed.

In [24], the authors applied the accelerated projected gradient method based on [15] to the compliance minimization problem in topology optimization. Although we call the proposed method the "accelerated" projected gradient method, the convergence rate is not improved theoretically from the classical projected gradient method for a nonconvex objective function. Moreover, to guarantee the convergence, the method requires additional FEAs at each iteration.[1] Therefore, in this paper, we adopt an inertial projected gradient method based on iPiano [25] instead. The main advantage of this algorithm is that it contains no auxiliary variables and requires a smaller number of FEAs than [15] to guarantee global convergence. It has an inertial term in its update formula to accelerate the convergence. Although the theoretical convergence rate is the same as that of the projected gradient method (and that of the method in [15]), practical performance is expected to be better than the projected gradient method. We consider an adaptive step size policy to further reduce the computational cost. The proposed method is easy to implement and guaranteed to converge to a stationary point which satisfies the first-order optimality condition. This convergence guarantee is important to properly stop the algorithm and obtain a high-quality solution. We also extend the results to the heat conduction problem and the compliant mechanism problem. We show that the projection onto the feasible set can be easily calculated for each of the equality and inequality constraints on the structural volume, and thus the inertial

[1] The statements about the faster convergence rate and the guaranteed global convergence of the proposed method in [24] are mistakes, and the authors correct it here.

projected gradient method can be efficiently applied to the topology optimization problems considered in this paper.

In numerical experiments, we consider the compliance minimization problem, the heat conduction problem and the compliant mechanism problem, and compare the proposed method with the optimality criteria method [6], the *method of moving asymptotes* (MMA) [31–33], and the MATLAB `fmincon` (interior-point method and sequential quadratic programming). We show that the proposed method has a low iteration cost and converges fast. Moreover, the solution obtained by the proposed method satisfies the first-order optimality condition with higher accuracy than those obtained by the existing method.

This paper is organized as follows: In Sect. 2, we provide the fundamentals of topology optimization and the problem formulation. In Sect. 3, we briefly discuss the projected gradient method and the projection onto the feasible set of topology optimization problems. Then, we explain the proposed method, the inertial projected gradient method and its step size policy. In Sect. 4, we show the results of numerical experiments. Finally, some concluding remarks are provided in Sect. 5.

All of the norms $\|\cdot\|$ in this paper are the Euclidean norm of a vector. The inner product is denoted by $\langle\cdot,\cdot\rangle$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the vectors with all components equal to 0 and 1, respectively. Moreover, $\max\{0,\cdot\}$ and $\min\{1,\cdot\}$ are the componentwise operations acting on a vector.

2 Problem formulation

We consider three topology optimization problems with simple linear constraints: the compliance minimization problem, the heat conduction problem and the compliant mechanism problem. The problem setting in this paper is based on [1, 7].

Consider a topology optimization problem the design domain of which is discretized by the conventional finite element method. An example of discretization of the design domain is shown in Fig. 1. For simplicity, we divide the design domain into n identical square finite elements with unit volume. The design variable of the optimization problem is the density vector $\boldsymbol{x} \in \mathbb{R}^n$, the eth component x_e of which denotes the density of the eth finite element. Each density x_e takes the value in $[0, 1]$. When $x_e = 0$, the element e is regarded as void and when $x_e = 1$, the element e is regarded as material.

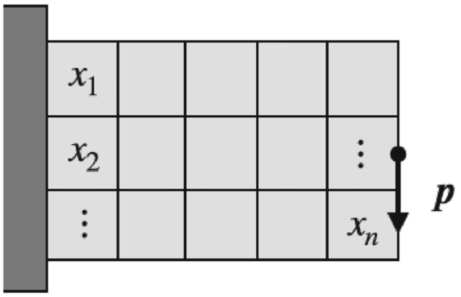

Fig. 1 A finite element discretization of the design domain of a topology optimization problem. The eth component x_e $(e = 1, \ldots, n)$ of the the density vector $\boldsymbol{x}$ corresponds to the density of the eth finite element

Thus $\boldsymbol{x}$ corresponds to a design of the structure. We use the SIMP (solid isotropic material with penalization) method [5] to penalize the intermediate values in (0, 1).

2.1 Compliance minimization problem

Consider the compliance minimization problem shown in Fig. 2. The top figure in Fig. 2 describes an example of problem setting and the bottom figure describes the optimal solution of the discretized problem with uniform square finite elements in the way shown in Fig. 1. The aim is to maximize the stiffness of structure when the external force is applied. In the SIMP method, the global stiffness matrix can be defined as

$$K(\boldsymbol{x}) = \sum_{e=1}^{n} (E_{\min} + (E_0 - E_{\min}) x_e^p) K_e, \tag{1}$$

where $p > 1$ is the penalty parameter, $E_0 \gg E_{\min} > 0$ are constants and K_e is the local stiffness matrix which is a constant symmetric matrix. In addition, we use the density filter [9] to prevent mesh dependency; refining the mesh leads to a different optimal structural design, not a refined structural design. The density filter is a linear operator acting on the density vector $\boldsymbol{x}$. Therefore, by using a constant matrix $H \in \mathbb{R}^{n\times n}$, the filtered density vector can be written as $\tilde{\boldsymbol{x}} = H\boldsymbol{x}$.

The compliance minimization problem is defined as follows:

$$\begin{array}{ll} \underset{\boldsymbol{x}\in\mathbb{R}^n, \tilde{\boldsymbol{x}}\in\mathbb{R}^n, \boldsymbol{u}\in\mathbb{R}^m}{\text{Minimize}} & \boldsymbol{p}^{\mathrm{T}}\boldsymbol{u} \\ \text{subject to} & K(\tilde{\boldsymbol{x}})\boldsymbol{u} = \boldsymbol{p}, \\ & \mathbf{1}^{\mathrm{T}}\tilde{\boldsymbol{x}} = V_0, \\ & \tilde{\boldsymbol{x}} = H\boldsymbol{x}, \\ & \mathbf{0} \le \boldsymbol{x} \le \mathbf{1}. \end{array} \tag{2}$$

Here, $\boldsymbol{p} \in \mathbb{R}^m$ is the constant load vector, $\boldsymbol{u} \in \mathbb{R}^m$ is the global nodal displacement vector, m is the number of degrees of freedom of the nodal displacements, and $V_0 \in$

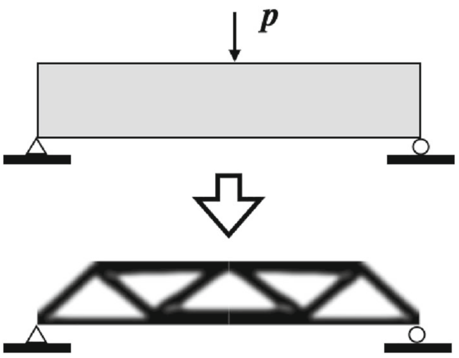

Fig. 2 Problem setting of the compliance minimization problem (top) and its optimal design (bottom)

$(0, n)$ is the upper limit of the structural volume. Problem (2) can be rewritten as the following optimization problem with a nonconvex objective function and linear constraints:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{Minimize}} \quad & f(\boldsymbol{x}) := \boldsymbol{p}^{\mathrm{T}} K(H\boldsymbol{x})^{-1}\boldsymbol{p} \\ \text{subject to} \quad & \mathbf{1}^{\mathrm{T}} H\boldsymbol{x} = V_0, \\ & \mathbf{0} \le \boldsymbol{x} \le \mathbf{1}. \end{aligned} \tag{3}$$

Note that, in practice, we do not calculate the inverse matrix of the global stiffness matrix, rather we solve the equilibrium equation $K(\tilde{\boldsymbol{x}})\boldsymbol{u} = \boldsymbol{p}$ (corresponding to FEA) at each iteration. Subsequently we use the solution $\boldsymbol{u}$ of FEA to calculate the objective function $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{u}$. Also, the gradient of the objective function is calculated by substituting $\boldsymbol{u}$ into the following formula:

$$\frac{\partial f}{\partial x_e} = -\sum_{i=1}^{n} h_{ie} p \tilde{x}_i^{p-1}(E_0 - E_{\min})\boldsymbol{u}^{\mathrm{T}} K_i \boldsymbol{u} \quad (e = 1, \dots, n), \tag{4}$$

where h_{ij} is the (i, j)th entry of H. Each component of the gradient $\nabla f(\boldsymbol{x})$ is non-positive, because K_e $(e = 1, \dots, n)$ is positive semidefinite. This is why we consider equality volume constraint in (2).

2.2 Heat conduction problem

The heat conduction problem aims to maximize the heat conduction from the entire design domain under uniformly distributed heating to the designated region where the temperature is constant T (lower than that of the entire design domain) as shown in Fig. 3. It can be formulated in the same way as the compliance minimization problem (2). The vector $\boldsymbol{u}$, the stationary solution of the discretized heat equation, consists of the temperature of each node. Note that the steady state heat equation and the equilibrium equation of linear elasticity are both described by the Poisson equation. We put $\boldsymbol{p} = c_{\mathrm{T}}\mathbf{1}$ using a scaling parameter c_{T}. Then the problem (2) corresponds to the minimization problem of the average temperature of the design domain (the problem

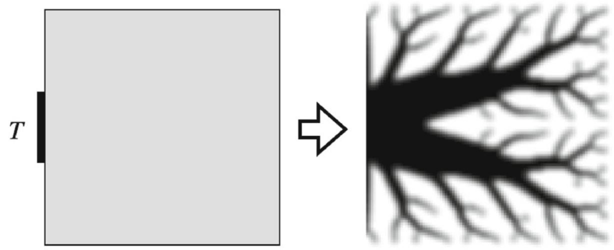

Fig. 3 Problem setting of the heat conduction problem (left) and its optimal design (right)

to find the optimal shape of the heat conductor to minimize the average temperature of the design domain). See [7] for details.

2.3 Compliant mechanism problem

A compliant mechanism transmits the force and motion through elastic body deformation. In Fig. 4, we aim to design a compliant mechanism which maximizes the displacement in the direction of vector $\boldsymbol{u}_{\text{out}}$ when the force is applied in the direction of vector $\boldsymbol{p}$. The spring with stiffness k_{in} and k_{out} are added to the points at which $\boldsymbol{p}$ is applied and $\boldsymbol{u}_{\text{out}}$ is measured, respectively. The compliant mechanism problem is defined as follows [7]:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n,\tilde{\boldsymbol{x}}\in\mathbb{R}^n,\boldsymbol{u}\in\mathbb{R}^m}{\text{Minimize}} \quad & \boldsymbol{q}^{\mathrm{T}}\boldsymbol{u} \\ \text{subject to} \quad & K(\tilde{\boldsymbol{x}})\boldsymbol{u} = \boldsymbol{p}, \\ & \mathbf{1}^{\mathrm{T}}\tilde{\boldsymbol{x}} \le V_0, \\ & \tilde{\boldsymbol{x}} = H\boldsymbol{x}, \\ & \mathbf{0} \le \boldsymbol{x} \le \mathbf{1}. \end{aligned} \tag{5}$$

The main difference from the compliance minimization problem is the objective function. The coefficient in the objective function $\boldsymbol{q}$ is different from the right hand side of the equilibrium equation $\boldsymbol{p}$. The gradient of the objective function is calculated by

$$\frac{\partial f}{\partial x_e} = -\sum_{i=1}^{n} h_{ie} p \tilde{x}_i^{p-1}(E_0 - E_{\min})\boldsymbol{u}^{\mathrm{T}} K_i \bar{\boldsymbol{u}} \quad (e = 1, \dots, n), \tag{6}$$

where $\bar{\boldsymbol{u}}$ is the solution of so-called adjoint equation $K(\tilde{\boldsymbol{x}})\bar{\boldsymbol{u}} = \boldsymbol{q}$. This equation can be efficiently solved using the Cholesky decomposition, because the coefficient matrix is the same as the equilibrium equation. Note that a component of the gradient is not necessarily non-positive in this case, and thus the volume constraint is imposed as an inequality constraint.

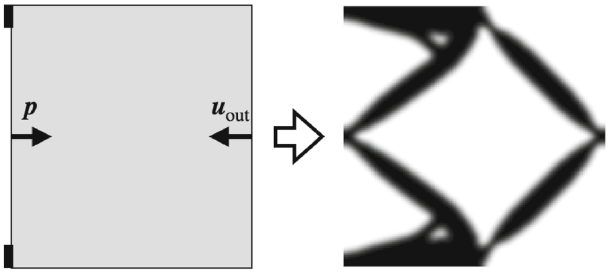

Fig. 4 Problem setting of the compliant mechanism problem (left) and its optimal design (right)

Problems (2) and (5) can be written as follows:

$$\underset{\boldsymbol{x}\in S}{\text{Minimize}}\ f(\boldsymbol{x}), \tag{7}$$

where S is the feasible set defined by

$$S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} = V_0,\ \mathbf{0} \le \boldsymbol{x} \le \mathbf{1}\} \tag{8}$$

or

$$S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} \le V_0,\ \mathbf{0} \le \boldsymbol{x} \le \mathbf{1}\}. \tag{9}$$

Although we have $\boldsymbol{v} = H^{\mathrm{T}}\mathbf{1}$ in this paper, the coefficient of the volume constraint is in general not necessarily equal to $\mathbf{1}$ (as the volume of each element can differ from each other). In the next section, we present algorithms to solve problem (7).

3 Inertial projected gradient method for topology optimization

3.1 Projected gradient method

The projected gradient method [16] is a classical optimization algorithm for an optimization problem in the form of (7) with a smooth objective function $f : \mathbb{R}^n \to \mathbb{R}$ and a closed convex feasible set $S \subset \mathbb{R}^n$. It is a special case of the proximal gradient method which has been attracting much attention in recent years [3, 4, 22]. The projected gradient method finds a solution by repeating the following formula starting from the initial point $\boldsymbol{x}^0 \in S$:

$$\boldsymbol{x}^{k+1} = \Pi_S(\boldsymbol{x}^k - \alpha_k \nabla f(\boldsymbol{x}^k)). \tag{10}$$

Here, $\alpha_k > 0$ is the step sizes, and $\Pi_S(\boldsymbol{w}) \in S$ is the projection of a given vector $\boldsymbol{w} \in \mathbb{R}^n$ onto S defined as follows:

$$\Pi_S(\boldsymbol{w}) := \underset{\boldsymbol{x}\in S}{\arg\min}\|\boldsymbol{x} - \boldsymbol{w}\|. \tag{11}$$

That is, $\Pi_S(\boldsymbol{w})$ is the closest point in S from $\boldsymbol{w}$. The projected gradient method coincides with the steepest descent method when $S = \mathbb{R}^n$.

3.2 Projection onto the feasible set

To use the projected gradient method effectively, the projection onto the feasible set must be easily calculated. We present an easy way to calculate the projection in our problems, for each of the equality volume constraint cases in (8) and the inequality volume constraint case in (9). The algorithm of the projection in our problems is similar to that of the projection onto the probability simplex (see e.g. [26]).

3.2.1 Case of equality volume constraint

Consider S in (8). The projection $\Pi_S(\boldsymbol{w})$ is equal to the unique optimal solution of the following problem:

$$\operatorname*{Minimize}_{\boldsymbol{x}\in S} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{w}\|^2. \tag{12}$$

This is a convex optimization problem, and the following KKT (Karush–Kuhn–Tucker) condition is the necessary and sufficient condition for optimality:

$$\begin{aligned}
&\boldsymbol{x}-\boldsymbol{w}-\boldsymbol{\lambda}+\boldsymbol{\nu}+\mu\boldsymbol{v}=\boldsymbol{0},\\
&\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}=V_0,\quad \boldsymbol{0}\le\boldsymbol{x}\le\boldsymbol{1},\\
&\boldsymbol{\lambda}\ge\boldsymbol{0},\quad \boldsymbol{\nu}\ge\boldsymbol{0},\\
&\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{x}=0,\quad \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{1})=0,
\end{aligned} \tag{13}$$

where $\boldsymbol{\lambda}$, $\boldsymbol{\nu}\in\mathbb{R}^n$ and $\mu\in\mathbb{R}$ are the Lagrange multipliers.

We find the unique point $\boldsymbol{x}$ which satisfies (13) for given $\boldsymbol{w}$, and it is the projection $\Pi_S(\boldsymbol{w})$. By the first equality in (13), we obtain $\boldsymbol{x}=\boldsymbol{w}+\boldsymbol{\lambda}-\boldsymbol{\nu}-\mu\boldsymbol{v}$. To satisfy $\boldsymbol{0}\le\boldsymbol{x}\le\boldsymbol{1},\boldsymbol{\lambda}\ge\boldsymbol{0},\boldsymbol{\nu}\ge\boldsymbol{0},\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{x}=0$ and $\boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{1})=0$, we set $\boldsymbol{\lambda}=\max\{0,-(\boldsymbol{w}-\mu\boldsymbol{v})\}$ and $\boldsymbol{\nu}=\max\{0,\boldsymbol{w}-\mu\boldsymbol{v}-\boldsymbol{1}\}$. Then, for given $\boldsymbol{w}$, $\boldsymbol{x}$ is a function depending only on μ:

$$\boldsymbol{x}(\mu;\boldsymbol{w}):=\max\{0,\min\{1,\boldsymbol{w}-\mu\boldsymbol{v}\}\}. \tag{14}$$

Therefore, we need to find μ^* such that $\boldsymbol{x}(\mu^*;\boldsymbol{w})$ satisfies the rest of the condition (13): the volume constraint $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(\mu^*;\boldsymbol{w})=V_0$. As $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(\mu;\boldsymbol{w})$ is a monotonically decreasing piecewise linear function of μ and $0<V_0<n$, μ^* exists in $[\min\{w_1/v_1,\dots,w_n/v_n\}-1,\max\{w_1/v_1,\dots,w_n/v_n\}]$.

In practice, all we need to do is to find the solution μ^* of

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(\mu;\boldsymbol{w})=V_0 \tag{15}$$

by e.g. the bisection method (in the numerical experiments, we use MATLAB `fzero` function). The projection is then calculated by

$$\Pi_S(\boldsymbol{w})=\boldsymbol{x}(\mu^*;\boldsymbol{w}). \tag{16}$$

3.2.2 Case of inequality volume constraint

In the case that the volume constraint is an inequality constraint, the projection can be calculated in a manner similar to Sect. 3.2.1. The KKT condition is as follows:

$$\begin{aligned}
&\boldsymbol{x} - \boldsymbol{w} - \boldsymbol{\lambda} + \boldsymbol{\nu} + \mu \boldsymbol{v} = \boldsymbol{0}, \\
&\boldsymbol{v}^{\mathrm{T}} \boldsymbol{x} \leq V_0, \quad \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1}, \\
&\boldsymbol{\lambda} \geq \boldsymbol{0}, \quad \boldsymbol{\nu} \geq \boldsymbol{0}, \quad \mu \geq 0, \\
&\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{x} = 0, \quad \boldsymbol{\nu}^{\mathrm{T}} (\boldsymbol{x} - \boldsymbol{1}) = 0, \quad \mu(\boldsymbol{v}^{\mathrm{T}} \boldsymbol{x} - V_0) = 0.
\end{aligned} \tag{17}$$

If $\boldsymbol{x}(0; \boldsymbol{w})$ satisfies the volume constraint $\boldsymbol{v}^{\mathrm{T}} \boldsymbol{x}(0; \boldsymbol{w}) \leq V_0$, then $\boldsymbol{x}(0; \boldsymbol{w})$ satisfies the KKT condition (17), and hence

$$\Pi_S(\boldsymbol{w}) = \boldsymbol{x}(0; \boldsymbol{w}). \tag{18}$$

If $\boldsymbol{v}^{\mathrm{T}} \boldsymbol{x}(0; \boldsymbol{w}) > V_0$, then there exists $\mu^* \in [0, \max\{w_1/v_1, \dots, w_n/v_n\}]$ such that $\boldsymbol{v}^{\mathrm{T}} \boldsymbol{x}(\mu^*; \boldsymbol{w}) = V_0$. Then the projection is written as

$$\Pi_S(\boldsymbol{w}) = \boldsymbol{x}(\mu^*; \boldsymbol{w}). \tag{19}$$

The constraints of the topology optimization problems in this paper are expressed as the intersection of box constraints ($\boldsymbol{a} \leq \boldsymbol{x} \leq \boldsymbol{b}$) and a single linear constraint. Therefore, we only need to find the scalar Lagrange multiplier μ which can be efficiently calculated by e.g. the bisection method. Note that, in a problem with general linear constraints, the calculation of projection becomes convex quadratic programming and computationally expensive in a large-scale problem.

3.3 Inertial projected gradient method

The projected gradient method has a low iteration cost and is suited for a large-scale optimization problem such as a topology optimization problem. However, the convergence of the projected gradient method is not very fast. Recently, the acceleration techniques of the projected gradient method and, more generally, the proximal gradient method have been attracting much attention.

There are several different kinds of acceleration techniques for the projected gradient method for nonconvex optimization. We adopt iPiano (inertial proximal algorithm for nonconvex optimization) [25] to solve topology optimization problems. Its simple update scheme is suited for topology optimization. It does not require additional evaluations of the objective value. Most accelerated projected gradient methods [15, 18] require evaluations of the objective value more than once at each iteration to update the design variable or to guarantee the convergence. Moreover, some methods [37] require FEA at an infeasible point where the global stiffness matrix may become singular. Although iPiano does not have a faster convergence rate than the projected gradient method, in the numerical examples we show that it is practically faster than the projected gradient method for topology optimization problems.

In iPiano, the design variable is updated as follows:

$$\boldsymbol{x}^{k+1} = \Pi_S(\boldsymbol{x}^k - \alpha_k \nabla f(\boldsymbol{x}^k) + \beta_k (\boldsymbol{x}^k - \boldsymbol{x}^{k-1})), \tag{20}$$

where $\alpha_k > 0$ and $\beta_k \geq 0$ are step size parameters discussed in Sect. 3.4. The term $\beta_k(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})$ in (20) is a so-called inertial (or momentum) term which accelerates the convergence. When $\beta_k \equiv 0$, (20) coincides with the classical projected gradient method (10).

3.4 Step size policy

To achieve faster and guaranteed convergence to a stationary point, the choice of the step size parameters is crucial. One choice is to use constant step size parameters. To choose constant step size parameters of a first-order method, the Lipschitz constant of the gradient of the objective function is often used as a guideline. For a differentiable function $\Psi : \mathbb{R}^n \to \mathbb{R}$, we call $L \geq 0$ the Lipschitz constant of $\nabla\Psi$ over $D \subset \mathbb{R}^n$ if it satisfies

$$\|\nabla\Psi(\boldsymbol{x}) - \nabla\Psi(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \quad (\forall \boldsymbol{x}, \boldsymbol{y} \in D). \tag{21}$$

If such an L exists, we say that $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth over D. Many first-order methods including the proposed method assume the L-smoothness of the objective function. In the topology optimization problems in Sect. 2, the objective functions are L-smooth over S, since they are rational functions and twice continuously differentiable on $[0, 1]^n$. See [3] for more details on the L-smoothness. In [25], the condition for constant step size parameters of iPiano (20) to guarantee the convergence is introduced as follows:

$$\alpha_k \equiv \alpha < \frac{2(1-\beta)}{L}, \quad \beta_k \equiv \beta \in [0, 1). \tag{22}$$

Although this constant step size policy is simple, there are two drawbacks. One is that it requires a good estimation of L, the Lipschitz constant of the gradient of the objective function. This estimation is difficult in topology optimization. The other is that a constant step size cannot benefit from a smaller local value of L. The Lipschitz constant over $D' \subset D$ can possibly be much less than the Lipschitz constant over D, and the points generated by an algorithm can be restricted to a smaller subset of D as iterations progress. In this case, the acceptable step size for the convergence guarantee becomes greater than (22) as iterations progress. For faster convergence, it is better to adjust the step size parameters at each iteration.

In case that the step size parameters change at each iteration, they must satisfy the following conditions to guarantee the convergence [25]:

$$\alpha_k = 2(1-\beta_k)/(2a_2 + L_k), \quad \beta_k = (b-1)\Big/\left(b - \frac{1}{2}\right), \tag{23}$$

where $b = \left(a_1 + \frac{L_k}{2}\right) / \left(a_2 + \frac{L_k}{2}\right)$ and $a_1 \geq a_2 > 0$ are constant parameters, and L_k is a parameter satisfying the following *descent condition*:

$$f(\boldsymbol{x}^{k+1}) \leq f(\boldsymbol{x}^k) + \langle \nabla f(\boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \rangle + \frac{L_k}{2} \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2. \qquad (24)$$

Note that if $L_k \geq L$, (24) is always satisfied (see e.g. [3] for the proof). However, too large L_k leads to a small step size and hence slow convergence. Also, too small L_k leads to a large step size and numerical instability or even divergence. Thus, we need to choose L_k appropriately for fast and stable convergence.

Backtracking is a popular way to choose the step size parameter L_k in a first-order method (see e.g. [3]). In the backtracking procedure, we start with a sufficiently small initial value s for L_k and repeat multiplying $\eta > 1$ until the descent condition (24) is satisfied, i.e. we set $L_k = s\eta^l$ where l is the smallest nonnegative integer such that $L_k = s\eta^l$ satisfies (24). However, the backtracking procedure requires evaluations of the objective value many times to check if the descent condition (24) is satisfied (Note that if we change the value of L_k, the objective value $f(\boldsymbol{x}^{k+1})$ of the left-hand side of (24) changes). This means we need to perform FEA many times to decide the step size parameters, which is computationally expensive.

Therefore, we estimate the initial value for the backtracking procedure by

$$L_k = \max\left\{L_{\min}, \frac{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|}\right\}, \qquad (25)$$

where $L_{\min}$ is a small positive constant to avoid the numerical instability, and we choose sufficiently large L_0 for the first iteration of the inertial projected gradient method. If L_k in (25) does not satisfy (24), then we update $L_k \leftarrow \eta L_k$ in the same way as the conventional backtracking procedure. The estimate (25) is motivated by the definition of L in (21). By definition, we see that L_k in (25) is no smaller than $L_{\min}$ and no greater than L. Although $L_k \geq L$ is a sufficient condition to satisfy (24), in numerical examples, L_k in (25) satisfies the descent condition (24) in most cases, and no additional FEAs are needed. By this step size policy, we can automatically adjust the step size regardless of the problem setting (e.g. the design domain, the boundary conditions and the number of finite elements).

Remark The step size (25) has a relationship with the Barzilai–Borwein step sizes [2]. For the simplicity of notation, we set $\boldsymbol{s}^{k-1} = \boldsymbol{x}^k - \boldsymbol{x}^{k-1}$ and $\boldsymbol{y}^{k-1} = \nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})$. The relationship

$$\frac{|\langle \boldsymbol{s}^{k-1}, \boldsymbol{y}^{k-1}\rangle|}{\|\boldsymbol{s}^{k-1}\|^2} \leq \frac{\|\boldsymbol{y}^{k-1}\|}{\|\boldsymbol{s}^{k-1}\|} \leq \frac{\|\boldsymbol{y}^{k-1}\|^2}{|\langle \boldsymbol{s}^{k-1}, \boldsymbol{y}^{k-1}\rangle|} \qquad (26)$$

immediately follows from the Cauchy–Schwarz inequality. The right-hand side and the left-hand side of (26) are inverses of the Barzilai–Borwein step sizes. The Barzilai–Borwein step sizes are derived from an approximation to the secant equation underlying the quasi-Newton method, and converge fast for convex quadratic programming. We can use the inverse of the Barzilai–Borwein step sizes instead of $\|\boldsymbol{y}^{k-1}\|/\|\boldsymbol{s}^{k-1}\|$ in (25). However, a large step size leads to many FEAs, and a small step size leads to slow convergence, thus we use (25).

Based on all of the above discussions, the algorithm of the inertial projected gradient method for topology optimization is described in Algorithm 1. The stopping criteria are discussed in the next section. Each iteration of Algorithm 1 consists of vector additions, scalar multiplications and projections other than FEA, thus computationally cheap even if the problem size is very large.

Algorithm 1 Inertial projected gradient method for topology optimization

```
input: $\boldsymbol{x}^0 \in S$, $L_0 > 0$, $L_{\min} > 0$, $\eta > 1$, $a_1 \geq a_2 > 0$, $k = 0$
 1: while Stopping criterion is not satisfied do
 2:   if $k = 0$ then
 3:     Conduct FEA at $\boldsymbol{x}^k$ to calculate $\nabla f(\boldsymbol{x}^k)$
 4:   else
 5:     Calculate $\nabla f(\boldsymbol{x}^k)$ using the result of FEA at line 12 of the previous iteration
 6:     Calculate step size parameter $L_k$ by (25)
 7:   end if
 8:   while true do
 9:     Set $\alpha_k$ and $\beta_k$ by (23)
10:     $\hat{\boldsymbol{x}}^{k+1} \leftarrow \boldsymbol{x}^k - \alpha_k \nabla f(\boldsymbol{x}^k) + \beta_k(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})$
11:     $\boldsymbol{x}^{k+1} \leftarrow \Pi_S(\hat{\boldsymbol{x}}^{k+1})$ (by a procedure shown in Section 3.2.)
12:     Conduct FEA at $\boldsymbol{x}^{k+1}$ to calculate $f(\boldsymbol{x}^{k+1})$
13:     if Descent condition (24) is satisfied then
14:       Break while
15:     else
16:       $L_k \leftarrow \eta L_k$
17:     end if
18:   end while
19:   $k \leftarrow k + 1$
20: end while
```

4 Numerical examples

We conduct the numerical experiments in three examples: the compliance minimization problem, the heat conduction problem and the compliant mechanism problem. We compare the proposed method with popular optimization algorithms for topology optimization: the optimality criteria method (OC) [1, 6] and the globally convergent version of the method of moving asymptotes (GCMMA) [32, 33]. As GCMMA is designed for an inequality-constrained optimization problem, for the numerical experiments on GCMMA we change the volume constraints in the compliance minimization problems and the heat conduction problems to the inequality-volume constraint $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} \leq V_0$.[2] We also make comparisons with the general nonlinear optimization algorithms: the interior-point method (IPM) and the sequential quadratic programming (SQP) of MATLAB `fmincon`. We use the limited-memory-BFGS (L-BFGS) formula for the Hessian approximation in IPM and SQP. The L-BFGS formula has a low iteration cost and is more suited for a large-scale problem than the BFGS formula or the exact Hessian.

[2] This does not cause any issues as the volume constraint is always active at the optimal solution. Note that using $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} \leq V_0$ and $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} \geq V_0$ to treat the equality constraint may lead to numerical instabilities (a matrix used in a subproblem in GCMMA may become close to singular).

The experiments have been run on iMac (Intel(R) Core i9, 3.6 GHz CPU, 128 GB RAM) and MATLAB R2020b. The MATLAB code of topology optimization is based on [1, 7, 14]. The following values are common in all the experiments: $E_0 = 1$, $E_{\min} = 10^{-3}$ and $p = 3$. The Poisson ratio in the local stiffness matrix K_e $(e = 1, \ldots, n)$ is 0.3. The filter radius used for the density filter is 0.05 times the number of elements in the horizontal direction. The initial point of each algorithm is $\boldsymbol{x}^0 = (V_0/n)\mathbf{1}$. The parameters of the proposed method are follows: $L_0 = 10$, $L_{\min} = 10^{-3}$, $\eta = 1.5$, $a_1 = 0.1$ and $a_2 = 10^{-6}$.

4.1 Optimality measure and stopping criterion

The proposed method aims to find a stationary point of problem (7), i.e. the point satisfying the first-order optimality condition (see e.g. [8]):

$$\langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0 \quad (\forall \boldsymbol{x} \in S). \tag{27}$$

For a given differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, a convex set $S \subset \mathbb{R}^n$ and $\alpha > 0$, define the gradient mapping $G_\alpha : \mathbb{R}^n \to \mathbb{R}^n$ by

$$G_\alpha(\boldsymbol{x}) := \frac{1}{\alpha}(\boldsymbol{x} - \Pi_S(\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x}))). \tag{28}$$

As a first-order optimality measure, we use the Euclidean norm of the gradient mapping $\|G_\alpha(\boldsymbol{x})\|$. We can easily see $G_\alpha(\boldsymbol{x})$ coincides with the gradient $\nabla f(\boldsymbol{x})$ when $S = \mathbb{R}^n$. Thus, the gradient mapping is a generalization of the gradient. Moreover, $\|G_\alpha(\boldsymbol{x})\|$ is a continuous function of $\boldsymbol{x}$, and $\|G_\alpha(\boldsymbol{x})\| = 0$ if and only if $\boldsymbol{x}$ is a stationary point of the problem in the form of (7) (see [3] for the proof). Thus, we can use the gradient mapping $\|G_\alpha(\boldsymbol{x})\|$ as a first-order optimality measure of $\boldsymbol{x}$. We set $\alpha = 1$ for simplicity. Note that $G_1(\boldsymbol{x}^k)$ corresponds to the proximal residual defined in [25]. This optimality measure can be used for any algorithms. We calculate $\|G_1(\boldsymbol{x}^k)\|$ at each iteration of each algorithm independent of the update of the design variables $\boldsymbol{x}^k$ so that we can equally measure the first-order optimality of each point generated by each algorithm. We use $\|G_1(\boldsymbol{x}^k)\| < \epsilon$ as a stopping criterion for sufficiently small $\epsilon > 0$.

The reason why we adopt the gradient mapping for comparison is that other choices are inaccurate or unable to equally compare the optimality of points generated by different algorithms. As the projected gradient method and OC do not calculate the Lagrange multipliers at each iteration, it is difficult to adopt the KKT residual norm, which is used in GCMMA and MATLAB `fmincon`. Also, the change of the objective function value $f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^k)$ or the design variable $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|$ can be strongly influenced by a step size, i.e. if we choose an arbitrary small step size, these values become arbitrarily small, and the algorithm terminates with a very small number of iterations even though the current point is not optimal.

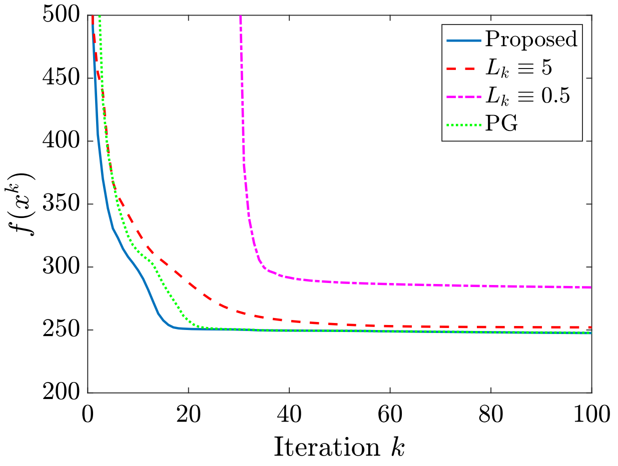

Fig. 5 Objective function value

4.2 Compliance minimization problem

We consider the compliance minimization problem of the MBB beam shown in Fig. 2. Note that, by utilizing the symmetry, we consider only the right half of the entire design domain. The upper limit of the volume is $V_0 = 0.5n$. The magnitude of the external force is 1.

4.2.1 Effectiveness of acceleration and step size policy

We compare the proposed method with the original (non-inertial) projected gradient method (PG) to show the effectiveness of the acceleration by the inertial term. Also, to show the effectiveness of the proposed step size policy, we compare it with the constant step size policy (22).

The objective function value and the norm of the gradient mapping at each iteration of 500 iterations for $n = 2700$ are shown in Figs. 5 and 6, respectively. Note that we omit after 100 iterations in Fig. 5 because of small changes. Also, we omit the figures of the obtained solutions, as not much difference is seen.

Although the proposed method does not have an improved convergence rate theoretically, both the objective function value and the norm of the gradient mapping decrease faster than PG as shown in Figs. 5 and 6. Moreover, when we use a constant step size parameter $L_k = 10$ $(\forall k)$ or $L_k = 0.5$ $(\forall k)$, the convergence gets slow. In particular, the result of $L_k = 0.5$ shows that too large step size leads to numerical instability (large objective values at the first few iterations). In contrast, the step size parameter L_k of the proposed method changes drastically at the first few iterations as shown in Fig. 7. This shows that the proposed step size policy effectively adjusts the step size for faster convergence.

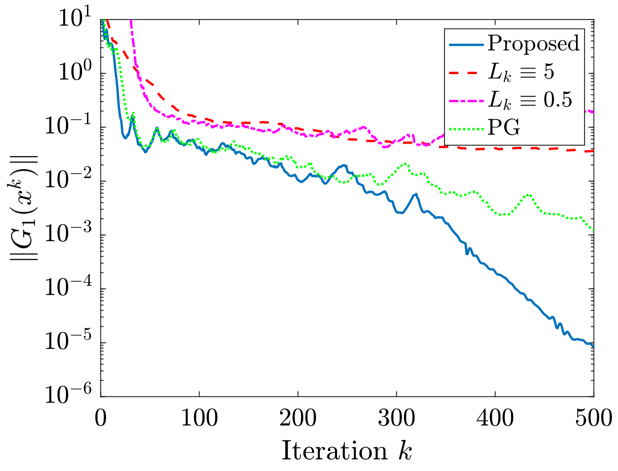

Fig. 6 Optimality measure

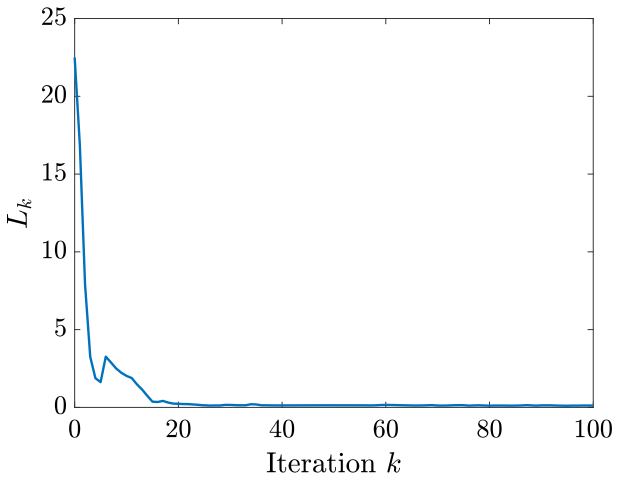

Fig. 7 L_k by the proposed step size policy

4.2.2 Comparison with existing methods

We compare the proposed method with OC, GCMMA, IPM and SQP. We use the same stopping criterion $\|G_1(\boldsymbol{x}^k)\| < 10^{-3}$ for all the algorithms. The maximum number of iterations is 2000. The total computational time and the computational time per iteration in seconds versus the number of finite elements n are shown in Figs. 8 and 9, respectively. Note that the graphs of Proposed and OC are overlapped in Fig. 9. The computational time of SQP is shown only for small values of n because it increases rapidly as n becomes large. To see how the objective value and the optimality measure decrease, we show these values of the proposed method, OC, MMA and IPM with 2000 iterations when $n = 10{,}800$ in Figs. 10 and 11. Note that the proposed method

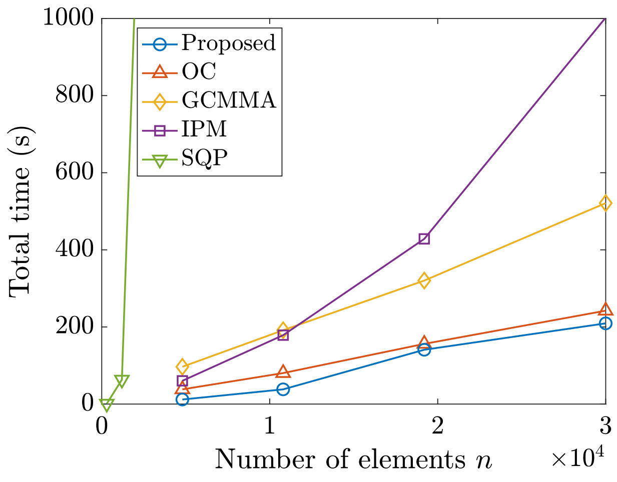

Fig. 8 Total computational time (compliance minimization problem)

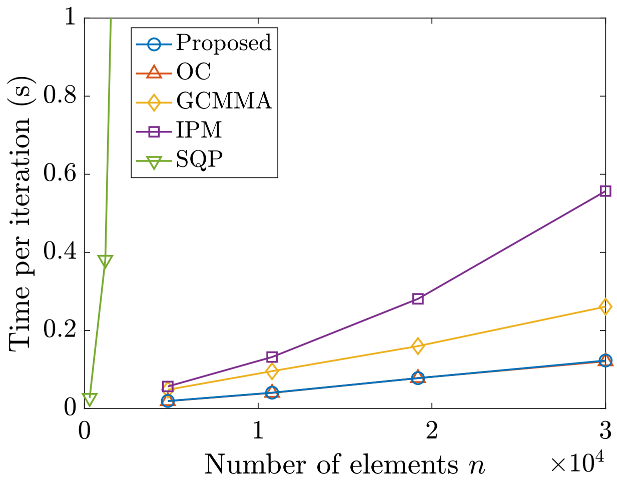

Fig. 9 Computational time per iteration (compliance minimization problem)

satisfies the stopping criterion before 2000 iterations. The obtained designs are shown in Fig. 12. Table 1 lists the detailed results: the number of iterations "iter.", the number of FEA, the total computational time t, the computational time per iteration t_{it}, and the objective value $f(\boldsymbol{x})$ and the Euclidean norm of the gradient mapping $\|G_1(\boldsymbol{x})\|$ at the last iteration. Note that IPM automatically stops before 2000 iterations for $n =$ 10,800, 19,200 and 30,000, although the obtained solution does not satisfy the stopping criterion. This is because the MATLAB `fmincon` stops automatically if the step size becomes too small.

From Figs. 8 and 9, we see that the computational cost per iteration of IPM and SQP increases drastically as the number of finite elements n increases. SQP has a particularly high iteration cost as it solves a quadratic programming problem at each

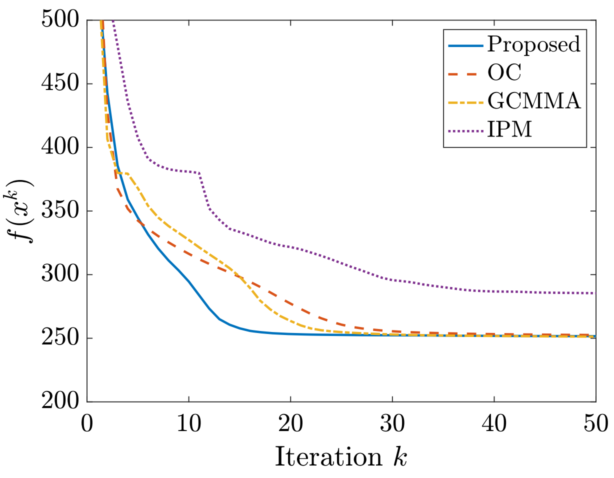

Fig. 10 Objective function value (compliance minimization problem, $n = 10{,}800$)

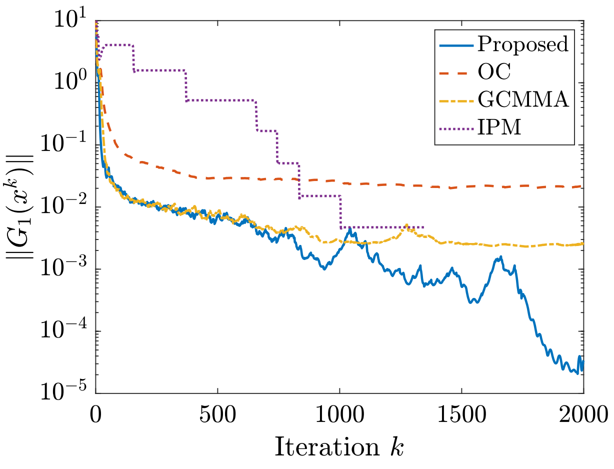

Fig. 11 Optimality measure (compliance minimization problem, $n = 10{,}800$)

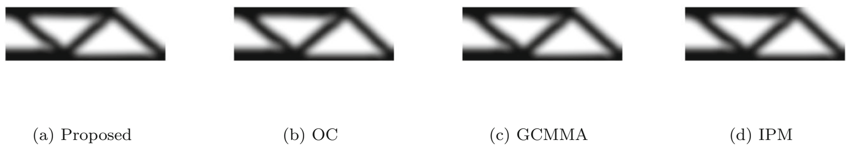

Fig. 12 Solutions by each algorithm (compliance minimization problem, $n = 10{,}800$)

Table 1 Numerical results of the compliance minimization problem

$n = 4800$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	609	609	11.84	1.94×10^{-2}	247.46	9.98×10^{-4}
OC	2000	2000	38.16	1.91×10^{-2}	248.87	4.38×10^{-2}
GCMMA	2000	2004	96.70	4.83×10^{-2}	247.44	3.23×10^{-3}
IPM	1066	1066	60.32	5.66×10^{-2}	247.42	6.26×10^{-4}
$n = 10{,}800$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	937	937	38.16	4.07×10^{-2}	248.84	9.91×10^{-4}
OC	2000	2000	80.40	4.02×10^{-2}	250.51	2.18×10^{-2}
GCMMA	2000	2005	191.54	9.58×10^{-2}	248.77	2.48×10^{-3}
IPM	1349	1349	178.68	1.32×10^{-1}	248.75	4.72×10^{-3}
$n = 19{,}200$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	1808	1808	140.85	7.79×10^{-2}	249.84	9.96×10^{-4}
OC	2000	2000	156.12	7.81×10^{-2}	251.55	1.88×10^{-2}
GCMMA	2000	2003	320.33	1.60×10^{-1}	249.84	4.24×10^{-3}
IPM	1523	1523	428.39	2.81×10^{-1}	249.79	2.55×10^{-3}
$n = 30{,}000$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	1707	1707	209.32	1.23×10^{-1}	250.68	9.75×10^{-4}
OC	2000	2000	242.28	1.21×10^{-1}	252.31	1.31×10^{-2}
GCMMA	2000	2005	521.19	2.61×10^{-1}	250.63	6.46×10^{-3}
IPM	1800	1800	1002.26	5.57×10^{-1}	250.59	4.10×10^{-3}

iteration. Therefore, these nonlinear programming solvers are impractical in a large-scale optimization problem. We omit IPM and SQP in numerical experiments hereafter, as they are particularly slow. GCMMA also has a high iteration cost compared to the proposed method and OC as it solves a convex subproblem at each iteration. The proposed method and OC consist of the vector addition, the scalar multiplication and the bisection method (solution of a single variable equation), and hence have low iteration costs. The number of FEA of the proposed method in Table 1 shows that the proposed step size policy effectively estimates an appropriate step size for stable convergence because almost no additional FEA is needed. As shown in Table 1, OC and GCMMA do not stop until the 2000 iteration. In fact, Fig. 11 shows that the optimality measures of OC and GCMMA do not decrease sufficiently. Note that OC is a heuristic algorithm and the convergence to a stationary point is not guaranteed. In contrast, the proposed method stops at fewer iterations than the other algorithms, and hence the proposed method has a shorter computational time as observed in Fig. 8. Figure 10 shows that the objective function value of the proposed method also decreases faster than those of the other algorithms. Moreover, the solution of the proposed method satisfies the optimality condition with higher accuracy as shown in Fig. 11, which

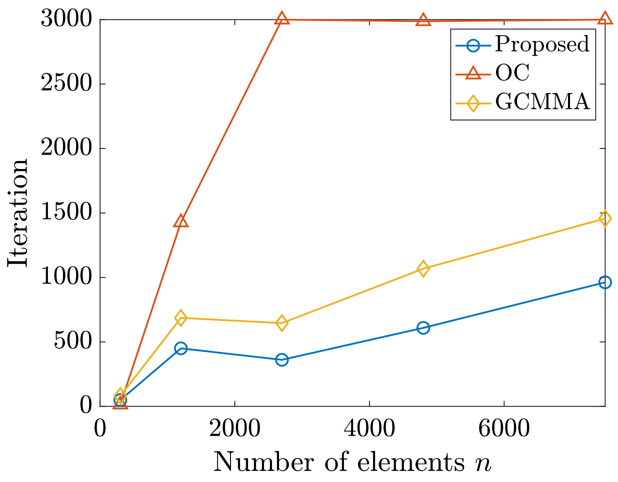

Fig. 13 Number of iterations (small-scale problem)

means that the solution is a more reliable optimal solution. In Fig. 12, the obtained solutions are slightly different (compare the angles of the right inclined bars). There is no guarantee that the solution obtained by OC is a local optimum since OC is heuristic (the objective function value is larger as shown in Table 1). In contrast, the solution obtained by the proposed method can be considered at least a stationary point.

4.2.3 Large-scale problems

To show the effectiveness of the proposed method in large-scale problems, we make a comparison with OC and GCMMA. To see the practical performance, we use different stopping criteria, which are commonly used for the three algorithms. The stopping criterion of the proposed method and OC is $\|G_1(\boldsymbol{x}^k)\| < 10^{-3}$ [14, 27] and $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| < 10^{-3}$ [25], respectively. GCMMA stops when the Euclidean norm of the KKT residual [32, 33] is less than 10^{-3}. The maximum number of iterations is 3000.

We show the number of iterations until the stopping criteria are satisfied for small-scale problems in Fig. 13. OC satisfies the stopping criteria for only small values of n as shown in Fig. 13. As OC does not satisfy the stopping criteria until the maximum number of iterations, the computational time of OC becomes huge when n gets large. Therefore, we omit OC for large-scale problems. A convergence guarantee is important to properly stop the algorithm and obtain a high-quality solution.

The total computational time and the optimality measure $\|G_1(\boldsymbol{x}^k)\|$ of the solutions in large-scale problems are shown in Figs. 14 and 15, respectively. We also add the results of the proposed method with the stopping criterion $\|G_1(\boldsymbol{x}^k)\| < 10^{-2}$.

Figure 14 shows that the proposed method and GCMMA converge at a moderate amount of time for large-scale problems. However, the solutions of GCMMA satisfy the optimality condition only with low accuracy compared with the proposed method with $\|G_1(\boldsymbol{x}^k)\| < 10^{-3}$, as shown in Fig. 15. The proposed method with the stopping criterion $\|G_1(\boldsymbol{x}^k)\| < 10^{-2}$ can obtain the solutions satisfying the optimality condition

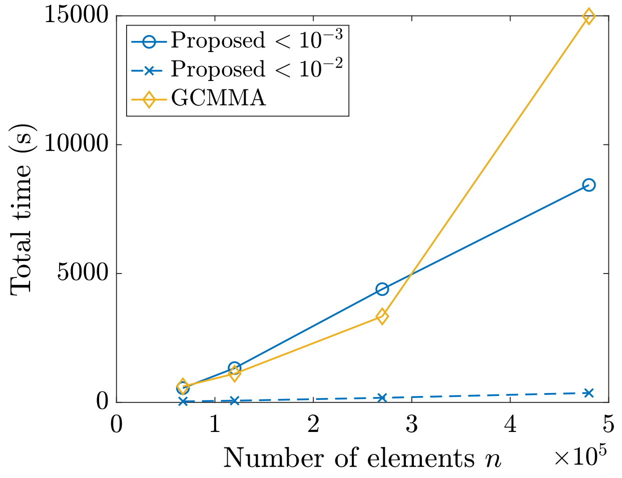

Fig. 14 Total computational time (large-scale problem)

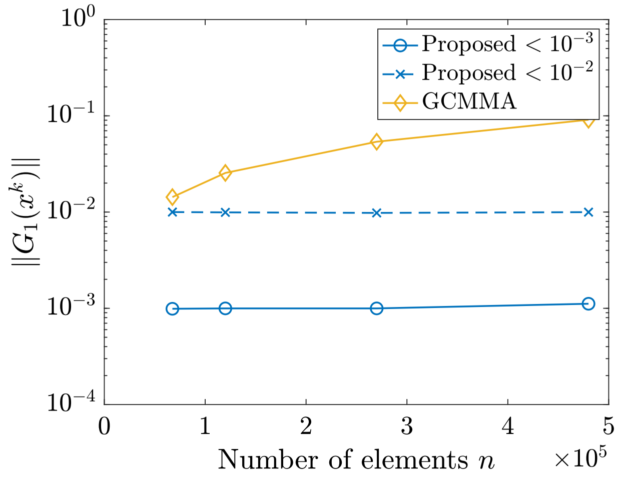

Fig. 15 Optimality measure when each algorithm stops (large-scale problem)

with similar or higher accuracy than those of GCMMA for much shorter computational time. This shows the effectiveness of the proposed method for obtaining an optimal solution with moderate accuracy in a large-scale problem.

4.3 Heat conduction problem

In this section, we consider the heat conduction problem shown in Fig. 3. We use the following parameters: $V_0 = 0.4n$ and $\boldsymbol{p} = (10/n)\mathbf{1}$.

We compare the proposed method with OC and GCMMA. The stopping criterion of the algorithms is $\|G_1(\boldsymbol{x}^k)\| < 10^{-3}$ and the maximum number of iterations is 2000. The total computational time versus the number of finite elements n is shown

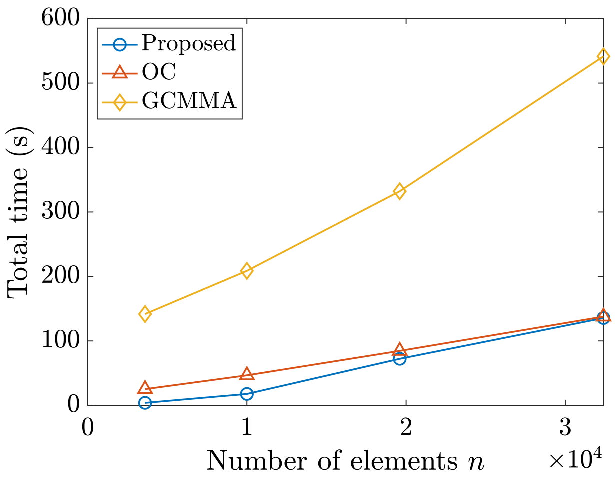

Fig. 16 Total computational time (heat conduction problem)

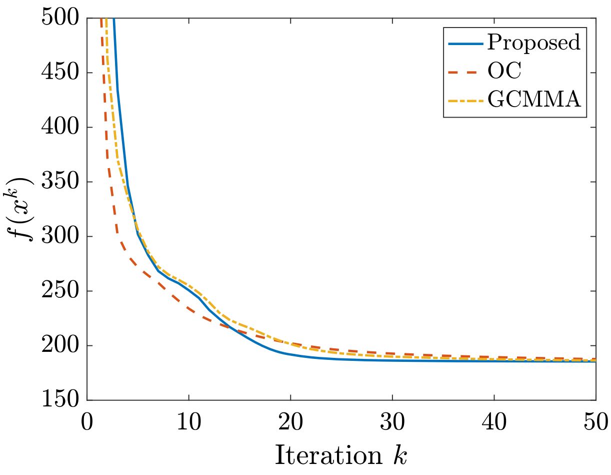

Fig. 17 Objective function value (heat conduction problem, n = 10,000)

in Fig. 16. Note that OC and GCMMA reach the maximum number of iterations. The objective value and the optimality measure until 2000 iterations for n = 10,000 are shown in Figs. 17 and 18, respectively. Note that the proposed method satisfies the stopping criterion before 2000 iterations. The designs obtained by the three algorithms are shown in Fig. 19. Table 2 lists the detailed results in the same manner as Table 1 for the compliance minimization problem.

Figure 16 and Table 2 show a trend similar to the compliance minimization problem; the proposed method has a low iteration cost, achieves faster convergence and satisfies the optimality condition with higher accuracy than the other methods. Figure 19 shows that the designs obtained by the three algorithms are different from each other. This

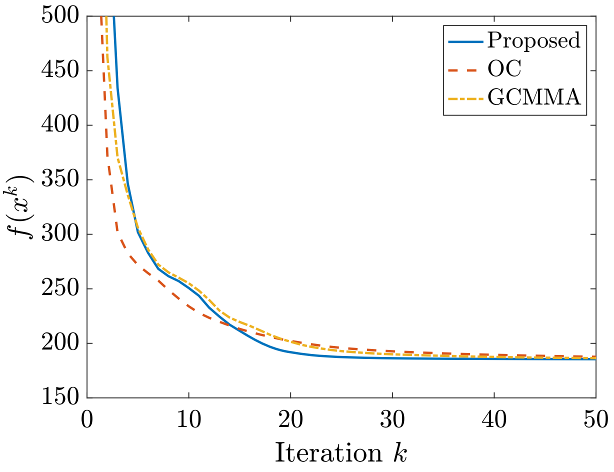

Fig. 18 Optimality measure (heat conduction problem, $n = 10{,}000$)

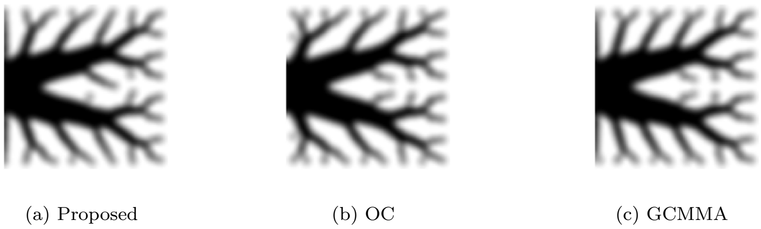

Fig. 19 Solutions obtained by the three algorithms (heat conduction problem, $n = 10{,}000$)

suggests that the heat conduction problem has more local optimal solutions than the compliance minimization problem.

4.4 Compliant mechanism problem

In this section, we consider the compliant mechanism problem shown in Fig. 4. Note that, by utilizing the symmetry, we consider only the lower half of the entire design domain. We use the following parameters: $V_0 = 0.3n$ and $k_{\mathrm{in}} = k_{\mathrm{out}} = 0.01$. The magnitude of the external force is 1.

We compare the proposed method with OC and GCMMA. The stopping criteria of the algorithms are $\|G_1(\boldsymbol{x}^k)\| < 10^{-3}$ and $f(\boldsymbol{x}^k) < -0.1$. The latter criterion is added to obtain a meaningful solution. The direction of the vector $\boldsymbol{u}_{\mathrm{out}}$ in Fig. 4 is the negative direction of the nodal displacement in the global coordinate system. Therefore, we seek to find a solution with a negative objective value. The maximum number of iterations is 2000. The total computational time versus the number of finite elements n is shown in Fig. 20. The objective value and the optimality measure until 2000 iterations for $n = 9800$ are shown in Figs. 21 and 22, respectively. The designs

Table 2 Numerical results of the heat conduction problem

$n = 3600$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\Vert G_1(\boldsymbol{x})\Vert$
Proposed	287	300	2.14	7.44×10^{-3}	189.65	8.54×10^{-4}
OC	2000	2000	13.25	6.62×10^{-3}	188.62	1.54×10^{-2}
GCMMA	2000	2193	69.32	3.47×10^{-2}	189.21	1.23×10^{-3}
$n = 10{,}000$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\Vert G_1(\boldsymbol{x})\Vert$
Proposed	987	994	16.50	1.67×10^{-2}	184.09	9.89×10^{-4}
OC	2000	2000	32.90	1.65×10^{-2}	184.26	1.49×10^{-2}
GCMMA	2000	2028	148.70	7.44×10^{-2}	183.81	3.87×10^{-3}
$n = 19{,}600$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\Vert G_1(\boldsymbol{x})\Vert$
Proposed	1735	1740	51.34	2.96×10^{-2}	180.69	9.54×10^{-4}
OC	2000	2000	58.73	2.94×10^{-2}	181.85	6.70×10^{-3}
GCMMA	2000	2023	230.27	1.15×10^{-1}	181.56	7.54×10^{-3}
$n = 32{,}400$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\Vert G_1(\boldsymbol{x})\Vert$
Proposed	2000	2004	99.78	4.99×10^{-2}	179.41	1.86×10^{-3}
OC	2000	2000	101.65	5.08×10^{-2}	180.56	5.93×10^{-3}
GCMMA	2000	2019	364.67	1.82×10^{-1}	180.29	1.24×10^{-2}

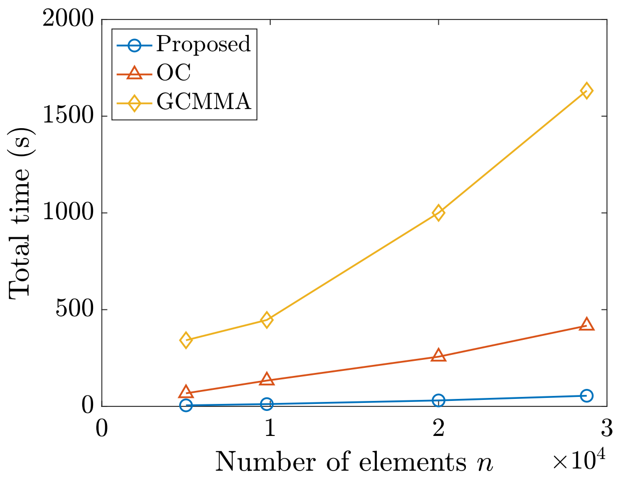

Fig. 20 Total computational time (compliant mechanism problem)

obtained by the three algorithms are shown in Fig. 23. Table 3 lists the detailed results in the same manner as Table 1 for the compliance minimization problem.

Figure 20 and Table 3 show a trend similar to the compliance minimization problem; the proposed method has a low iteration cost, achieves faster convergence and satisfies the optimality condition with higher accuracy than the other methods. However, as

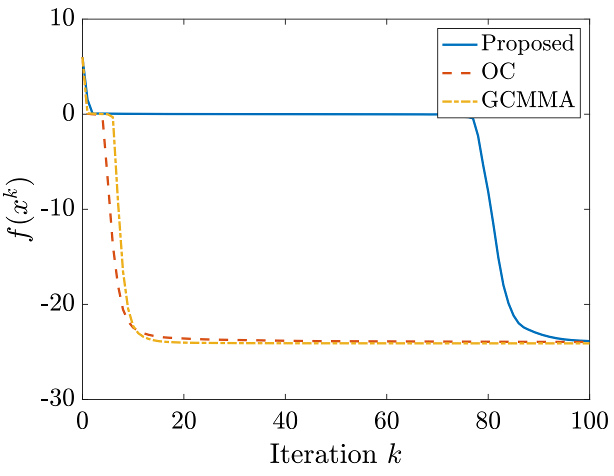

Fig. 21 Objective function value (compliant mechanism problem, $n = 9800$)

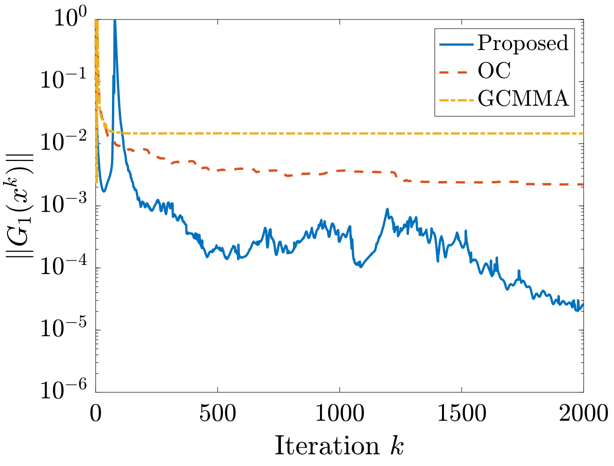

Fig. 22 Optimality measure (compliant mechanism problem, $n = 9800$)

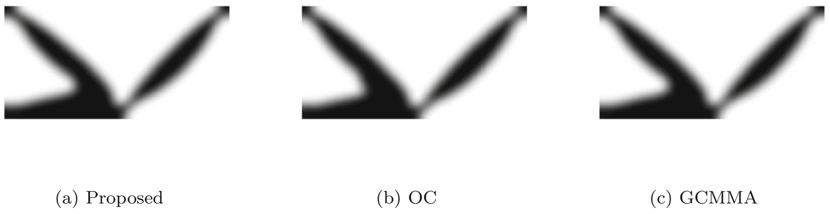

Fig. 23 Solutions by each algorithm (compliant mechanism problem, $n = 9800$)

Table 3 Numerical results of the compliant mechanism problem

$n = 5000$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	142	142	5.19	3.65×10^{-2}	−24.28	9.70×10^{-4}
OC	2000	2000	67.58	3.38×10^{-2}	−24.27	3.69×10^{-3}
GCMMA	2000	2032	341.75	1.71×10^{-1}	−24.29	7.45×10^{-3}
$n = 9800$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	181	182	11.70	6.46×10^{-2}	−24.09	9.63×10^{-4}
OC	2000	2000	133.31	6.67×10^{-2}	−24.08	2.21×10^{-3}
GCMMA	2000	2031	446.59	2.20×10^{-1}	−24.10	1.46×10^{-2}
$n = 20000$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	244	245	30.67	1.26×10^{-1}	−23.77	9.89×10^{-4}
OC	2000	2000	256.77	1.28×10^{-1}	−23.75	2.32×10^{-3}
GCMMA	2000	2035	1000.40	5.00×10^{-1}	−23.80	2.10×10^{-2}
$n = 28800$	Iter.	FEA	t (s)	t_{it} (s)	$f(\boldsymbol{x})$	$\|G_1(\boldsymbol{x})\|$
Proposed	276	278	54.79	1.99×10^{-1}	−23.58	9.97×10^{-4}
OC	2000	2000	416.88	2.08×10^{-1}	−23.54	2.08×10^{-3}
GCMMA	2000	2028	1632.02	8.216×10^{-1}	−23.62	2.26×10^{-2}

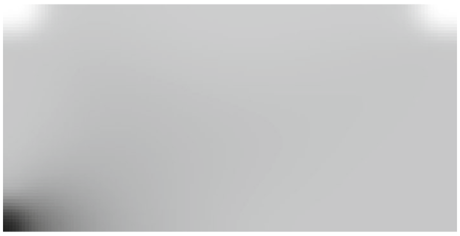
Fig. 24 Design at the 20th iteration of the proposed method ($\|\nabla f(\boldsymbol{x}^k)\| = 3.10 \times 10^{-3}$)

shown in Fig. 21, the decrease of the objective value of the proposed method slows down in the region where the sign of the objective value changes (the direction of the displacement of the output node changes). A typical design of that region is shown in Fig. 24. In that region, the norm of the gradient of the objective function is small. GCMMA is also slowed down in that region because it required 25 evaluations of the objective values at the first 5 iterations. Figure 23 shows that the design obtained by the three algorithms are similar to each other.

5 Conclusion

In this paper, we have presented an inertial projected gradient method for solving large-scale topology optimization problems. For a topology optimization problem under a single linear equality or inequality constraint with a box constraint, we have shown that the projection onto the feasible set can be efficiently computed, and hence the projected gradient method can be applied effectively. We have proposed to use an inertial version of the projected gradient method by Ochs et al. [25] to accelerate the convergence. We have also considered an adaptive step size policy to further reduce the computational cost. The proposed method is easy to implement. Moreover, the proposed method has the global convergence property.

In numerical examples, we have shown that the iteration cost of the proposed method is as low as that of the optimality criteria method. It has been demonstrated that the conventional algorithms used for topology optimization (the optimality criteria method and the method of moving asymptotes) achieve the first-order optimality condition with low accuracy. In contrast, the proposed method converges fast to a point satisfying the first-order optimality condition with higher accuracy. The proposed method is also effective for large-scale problems. We have shown that, for a topology optimization problem with simple linear constraints such as the compliance minimization problem, it is more efficient to use the proposed method than to use a general-purpose nonlinear programming solver such as the interior-point method and the method of moving asymptotes, because the proposed method takes advantage of a simple problem structure.

We have dealt with topology optimization problems with only linear constraints. To deal with large-scale optimization problems with nonlinear constraints, other first-order algorithms are to be considered. Large-scale optimization is rapidly growing especially in machine learning and data science communities. There may be some efficient large-scale optimization techniques that can be useful for developing new topology optimization algorithms.

Acknowledgements The work of the second author is partially supported by JST CREST Grant No. JPMJCR1911, Japan, the Kajima Foundation's Research Grant, and JSPS KAKENHI 21K04351.

Funding Open access funding provided by The University of Tokyo.

References

1. Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B.S., Sigmund, O.: Efficient topology optimization in MATLAB using 88 lines of code. Struct. Multidiscip. Optim. **43**(1), 1–16 (2011)
2. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA J. Numer. Anal. **8**(1), 141–148 (1988)
3. Beck, A.: First-order Methods in Optimization. SIAM, Philadelphia (2017)
4. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imag. Sci. **2**(1), 183–202 (2009)

5. Bendsøe, M.P.: Optimal shape design as a material distribution problem. Struct. Optimiz. **1**(4), 193–202 (1989)
6. Bendsøe, M.P., Kikuchi, N.: Generating optimal topologies in structural design using a homogenization method. Comput. Methods Appl. Mech. Eng. **71**(2), 197–224 (1988)
7. Bendsøe, M.P., Sigmund, O.: Topology Optimization: Theory, Methods, and Applications, 2nd edn. Springer, Berlin (2004)
8. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Massachusetts (1999)
9. Bourdin, B.: Filters in topology optimization. Int. J. Numer. Meth. Eng. **50**(9), 2143–2158 (2001)
10. Carmon, Y., Duchi, J.C., Hinder, O., Sidford, A.: Accelerated methods for nonconvex optimization. SIAM J. Optim. **28**(2), 1751–1772 (2018)
11. Chandrasekhar, A., Suresh, K.: TOuNN: topology optimization using neural networks. Struct. Multidiscip. Optim. **63**(3), 1135–1149 (2021)
12. d'Aspremont, A., Scieur, D., Taylor, A.: Acceleration methods. Found. Trends Optimiz. **5**(1–2), 1–245 (2021)
13. Deaton, J.D., Grandhi, R.V.: A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct. Multidiscip. Optim. **49**(1), 1–38 (2014)
14. Ferrari, F., Sigmund, O.: A new generation 99 line MATLAB code for compliance topology optimization and its extension to 3D. Struct. Multidiscip. Optim. **62**(4), 2211–2228 (2020)
15. Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math. Program. **156**, 59–99 (2016)
16. Goldstein, A.A.: Convex programming in Hilbert space. Bull. Am. Math. Soc. **70**(5), 709–710 (1964)
17. Kanno, Y.: Accelerated proximal gradient method for bi-modulus static elasticity. Optim. Eng. **23**, 453–477 (2022)
18. Li, H., Lin, Z.: Accelerated proximal gradient methods for nonconvex programming. Adv. Neural. Inf. Process. Syst. **28**, 379–387 (2015)
19. Li, H., Lin, Z.: Restarted nonconvex accelerated gradient descent No more polylogarithmic factor in the $O(\epsilon^{-7/4})$ complexity. In: Proceedings of the 39th International Conference on Machine Learning vol. 162, pp. 12901–12916 (2022)
20. Li, W., Suryanarayana, P., Paulino, G.H.: Accelerated fixed-point formulation of topology optimization: application to compliance minimization problems. Mech. Res. Commun. **103**, 103469 (2020)
21. Li, W., Zhang, X.S.: Momentum-based accelerated mirror descent stochastic approximation for robust topology optimization under stochastic loads. Int. J. Numer. Meth. Eng. **122**(17), 4431–4457 (2021)
22. Lin, Z., Li, H., Fang, C.: Accelerated Optimization for Machine Learning. Springer, Singapore (2020)
23. Nesterov, Y.E.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Math. Doklady **269**, 543–547 (1983)
24. Nishioka, A., Kanno, Y.: Accelerated projected gradient method with adaptive step size for compliance minimization problem. JSIAM Lett. **13**, 33–36 (2021)
25. Ochs, P., Chen, Y., Brox, T., Pock, T.: iPiano: inertial proximal algorithm for nonconvex optimization. SIAM J. Imaging Sci. **7**(2), 1388–1419 (2014)
26. Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends Optimiz. **1**(3), 127–239 (2014)
27. Rojas-Labanda, S., Stolpe, M.: Benchmarking optimization solvers for structural topology optimization. Struct. Multidiscip. Optim. **52**(3), 527–547 (2015)
28. Rojas-Labanda, S., Stolpe, M.: An efficient second-order SQP method for structural topology optimization. Struct. Multidiscip. Optim. **53**(6), 1315–1333 (2016)
29. Shimizu, W., Kanno, Y.: Accelerated proximal gradient method for elastoplastic analysis with von mises yield criterion. Jpn. J. Ind. Appl. Math. **35**(1), 1–32 (2018)
30. Shimizu, W., Kanno, Y.: A note on accelerated proximal gradient method for elastoplastic analysis with tresca yield criterion. J. Oper. Res. Soc. Jpn. **63**(3), 78–92 (2020)
31. Svanberg, K.: The method of moving asymptotes—a new method for structural optimization. Int. J. Numer. Meth. Eng. **24**(2), 359–373 (1987)
32. Svanberg, K.: A class of globally convergent optimization methods based on conservative convex separable approximations. SIAM J. Optim. **12**(2), 555–573 (2002)
33. Svanberg, K.: Svanberg matematisk optimering och IT AB (2022). http://www.smoptit.se. Accessed 21 Jan
34. Tango, S., Azegami, H.: Acceleration of shape optimization analysis using model order reduction by Karhunen–Loève expansion. Jpn. J. Ind. Appl. Math. **39**(1), 385–401 (2022)

35. Ulu, E., Zhang, R., Kara, L.B.: A data-driven investigation and estimation of optimal topologies under variable loading configurations. Comput. Methods Biomech. Biomed. Eng. Imaging Vis. **4**(2), 61–72 (2016)
36. Wang, S., de Sturler, E., Paulino, G.H.: Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. Int. J. Numer. Meth. Eng. **69**(12), 2441–2468 (2007)
37. Wen, B., Chen, X., Pong, T.K.: Linear convergence of proximal gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems. SIAM J. Optim. **27**(1), 124–145 (2017)
38. Yamasaki, S., Yaji, K., Fujita, K.: Data-driven topology design using a deep generative model. Struct. Multidiscip. Optim. **64**(3), 1401–1420 (2021)
39. Yano, M., Huang, T., Zahr, M.J.: A globally convergent method to accelerate topology optimization using on-the-fly model reduction. Comput. Methods Appl. Mech. Eng. **375**, 113635 (2021)