Advances in Engineering Software 196 (2024) 103712

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

An efficient isogeometric topology optimization based on the adaptive damped geometric multigrid method

Shijie Luo [a], Feng Yang [a], Yingjun Wang [a,b,*]

[a] National Engineering Research Center of Novel Equipment for Polymer Processing, The Key Laboratory of Polymer Processing Engineering of the Ministry of Education (South China University of Technology), Guangdong Provincial Key Laboratory of Technique and Equipment for Macromolecular Advanced Manufacturing, South China University of Technology Guangzhou 510641, PR China

[b] State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, PR China

ARTICLE INFO

Keywords:
Isogeometric topology optimization
Geometric multigrid
Adaptive damped jacobi
Preconditioned power method

ABSTRACT

The efficiency of solving sparse linear equations in isogeometric topology optimization (ITO) can be improved by the multigrid algorithm due to its excellent convergence rate. However, its convergence rate heavily relies on the smoother's parameters. To address this problem, a new h-refinement multigrid conjugate gradient method with adaptive damped Jacobi (ADJ-hMGCG) has been developed. By analyzing the eigenvalues of the stiffness matrix, the damping coefficient of the smoother that achieves the fastest convergence rate has been determined. Due to the significant computational resources required to compute eigenvalues in the stiffness matrix, this paper also presents a preconditioned power method based on ITO and geometric multigrid characteristics to improve the efficiency of adaptive damping solutions. The results of 2D and 3D numerical examples show that the ADJ-hMGCG method successfully improves the solution speed and robustness while meeting the accuracy requirements of topology optimization, and the total computational cost can be reduced by up to 59 % compared to traditional solvers for large-scale problems.

1. Introduction

Topology Optimization (TO) is essentially a mathematical method aiming to obtain the optimal material distribution under the given load and constraint conditions [1,2]. A series of influential TO methods have been proposed over the past few decades, such as the solid isotropic material with penalization (SIMP) approach [3-5], the level set method (LSM) [6-8], the evolutionary structural optimization (ESO) [9,10], the moving morphable components (MMC) method [11-13]. In recent years, isogeometric analysis (IGA) [14,15] has been utilized in TO due to its advantages in accuracy and continuity. Seo et al. [16] pioneered the combination of IGA and TO, in which the control points of spline surfaces and trimmed curves were used as design variables for the TO, thereby eliminating the post-processing effort for converting to CAD model. Subsequently, Hassani et al. [17] proposed an isogeometric topology optimization (ITO) method based on the NURBS basis function to approximate the expression of the design and internal continuous density function, which avoids the dependence of optimization results on meshes and ensures the accuracy of geometry while effectively reducing the degree of freedoms (DOFs). Nowadays, ITO has been extended to be applied in many interesting fields such as heat transfer [18], electromagnetics [19], and acoustics [20].

In the field of ITO research, Wang et al. [21] utilized NURBS basis functions to replace the radial basis functions in the traditional LSM without constructing additional level set functions. Gao et al. [22] applied a density distribution function to represent the continuum structure, realizing an efficient ITO method. Significantly, the solution of sparse linear equations takes a significant portion of the whole time, and therefore improving solution efficiency has become a hot topic in academic research [23].

At present, improving the efficiency of solving equations mainly include the following two directions: firstly, using parallel computing to decompose the total task of the algorithm into multiple subtasks and then process these subtasks on multiple processors or computers [24]. The second is to algorithmically improve the computational efficiency and reduce the time complexity to fundamentally improve the solving efficiency [25,26].

In the field of parallel computing, Wadbro et al. [27] applied GPU parallel computing to the large-scale TO, and a heat conduction model with more than 4 million variables was adopted to verify its efficiency. A

* Corresponding author.
E-mail address: wangyj84@scut.edu.cn (Y. Wang).

https://doi.org/10.1016/j.advengsoft.2024.103712
Received 16 January 2024; Received in revised form 22 May 2024; Accepted 20 June 2024
Available online 13 July 2024

parameterized level set ITO method based on GPU parallel strategy was proposed by Xia et al. [28], and the work also analyzed the computational complexity and parallelism of different steps in ITO. In the field of high-efficient algorithms, Amir et al. [29] proposed the reanalysis method, which utilizes the characteristic of small changes in adjacent iterative stiffness matrix to approximate linear equations at a lower cost. Long et al. [30] reduced the DOFs of finite element equations by using a new TO minimum volume model based on the reanalysis method.

Alternatively, the multigrid method, as an accelerated algorithm improves the efficiency of solving large-scale linear equations by prolonging the exact solution of the coarse grid to the fine grid. Yin et al. [31] proposed the Multigrid Assisted Reanalysis method to reduce the computer cost in heat sinks. Amir et al. [32] accelerated the solution of large-scale linear equations by using multigrid preconditioned gradients (MGCG) solver. Wang et al. [33] proposed a high-efficiency ITO method by combining multilevel mesh, MGCG and local-update strategy and successfully reduced 37 %∼93 % computational time compared to the traditional ITO. For more details about the multigrid algorithm, please refer to papers [34,35].

Note that the convergence rate of the multigrid algorithm depends on the smoother parameters [36]. Thus, it is necessary to choose a suitable smoother, such as the damped Jacobi method and successive over-relaxation method, to obtain an efficient solution with fewer iterations [37]. Nevertheless, these smoothers require the pre-designed damping coefficients in advance. The optimal damping coefficient is different when facing different problems or models. Therefore, how to select the optimal damping coefficient is the key issue to reduce the number of iterations of the multigrid solver.

This paper presents an adaptive damped strategy for the damped Jacobi method based on the eigenvalues of the stiffness matrix. The adaptive damped Jacobi method enables the traditional multigrid algorithm to have better computational efficiency and robustness performance. Simultaneously, we present a preconditioned power method based on two strategies, namely vertical and horizontal, to accelerate the computation of adaptive damping coefficients. Compared with traditional solvers, the proposed solver achieves better convergence performance. Finally, the effectiveness of this method is verified by four numerical examples.

The remainder of this paper is organized as follows. Section 2 presents an overview of the NURBS and the formulation of ITO. Section 3 presents the theory and algorithm implementation of the ADJ-hMGCG. Section 4 tests three 2D examples and one 3D example to demonstrate the efficiency of the ADJ-hMGCG in ITO. Finally, conclusions and some prospects are provided in Section 5.

2. Theory of ITO

The basic theories of this paper include NURBS, IGA and SIMP-based ITO. These theories will be briefly described in this section, and more details can be found in the references [3,15,38].

2.1. B-splines and non-uniform rational B-splines (NURBS)

Computer-aided design (CAD) and computer graphics commonly employ NURBS to express complex curves and surfaces. The NURBS basis functions, as an extension of B-spline basis functions, construct intricate curves or surfaces through knot vectors, control points vectors and weights. The B-spline basis function $B_{i,p}(\xi)$ used in the construction of the NURBS basis functions can be defined by the Cox-de Boor recursion formula as:

$$B_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \le \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi) \quad (p > 0) \tag{2}$$

where we define the convention $0/0 = 0$, ξ_i is a knot in the non-decreasing sequence knot vectors of B-spline, the knot vectors can be denoted by $\boldsymbol{\Xi} = [\xi_1, \xi_2, \ldots, \xi_{nc+p+1}]$, p represents the degree of the basis function and nc is the number of control points.

The NURBS basis function can be expressed by assigning a weight w_i as follows:

$$N_{i,p}(\xi) = \frac{B_{i,p}(\xi) w_i}{\sum_{j=1}^{nc} B_{j,p}(\xi) w_i} \tag{3}$$

Subsequently, a p-degree NURBS curve $C(\xi)$ can be expressed as a function related to the control point $\boldsymbol{Q}_i$ and the NURBS basis function $N_{i,p}$ as:

$$C(\xi) = \sum_{i=1}^{nc} N_{i,p}(\xi) \boldsymbol{Q}_i \tag{4}$$

Based on the construction of the NURBS curves, NURBS surfaces enables to be defined as the product of p-degree NURBS curves in the ξ-direction and q-degree NURBS curves in the η-direction. And the equations for NURBS surfaces can be constructed as follows:

$$S(\xi, \eta) = \sum_{i=1}^{nc} \sum_{j=1}^{mc} N_{i,p}(\xi) N_{j,q}(\eta) \boldsymbol{Q}_{i,j} \tag{5}$$

where $\boldsymbol{Q}_{i,j}$ represents the control points grid in both ξ- and η-directions, $w_{i,j}$ denotes the weight factor, $N_{i,p}(\xi)$ and $N_{j,q}(\eta)$ refer to the NURBS basis functions defined in both two directions.

2.2. Isogeometric analysis

The NURBS basis functions are utilized as the shape functions for IGA, realizing the same interpolation of geometric and displacement fields [39]. Thus, the variable $\boldsymbol{x}$ (such as coordinates, force, and displacement) in parameter coordinates ξ can be obtained from nearby control points for variables as:

$$x(\xi) = \sum_{i=1}^{nc} N_i(\xi) x_i \tag{6}$$

where N_i is the NURBS basis function of control point i, and x_i is the function value of control point i. The discrete equilibrium equation for the elasticity problem can be expressed as $\boldsymbol{F} = \boldsymbol{KU}$, the global stiffness matrix can be constructed by the element stiffness matrix and the element density. In the SIMP-based ITO, the element stiffness matrix $\boldsymbol{k}_e$ can be expressed as follows:

$$\boldsymbol{k}_e = \int_{\widehat{\Omega}_e} \boldsymbol{B}^T \boldsymbol{D}_0 \boldsymbol{B} |\boldsymbol{J}_1| d\widehat{\Omega} = \int_{\overline{\Omega}_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D}_0 \boldsymbol{B} |\boldsymbol{J}_1| |\boldsymbol{J}_2| d\overline{\Omega} \tag{7}$$

where B is the strain-displacement matrix, $\boldsymbol{D}_0$ is elasticity matrix, $\widehat{\Omega}_e$ and $\overline{\Omega}_e$ represent the paracentric domain in NURBS parametric space parametric Ξ and the integration parametric space $\overline{\Xi}$, $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are the Jacobian matrices which denote the transformation relation from NURBS parametric space to the physical space and the integration parametric space to the NURBS parametric space, respectively.

Taking the 2D case as an example, the knot vector $\boldsymbol{\Xi}$ of NURBS including $\boldsymbol{\Xi}^{\xi} = [\xi_1, \xi_2, \ldots, \xi_{nc+p+1}]$ in ξ-direction and $\boldsymbol{\Xi}^{\eta} = [\eta_1, \eta_2, \ldots, \eta_{mc+q+1}]$ in η-direction, and the strain-displacement matrix $\boldsymbol{B}$ is represented as:

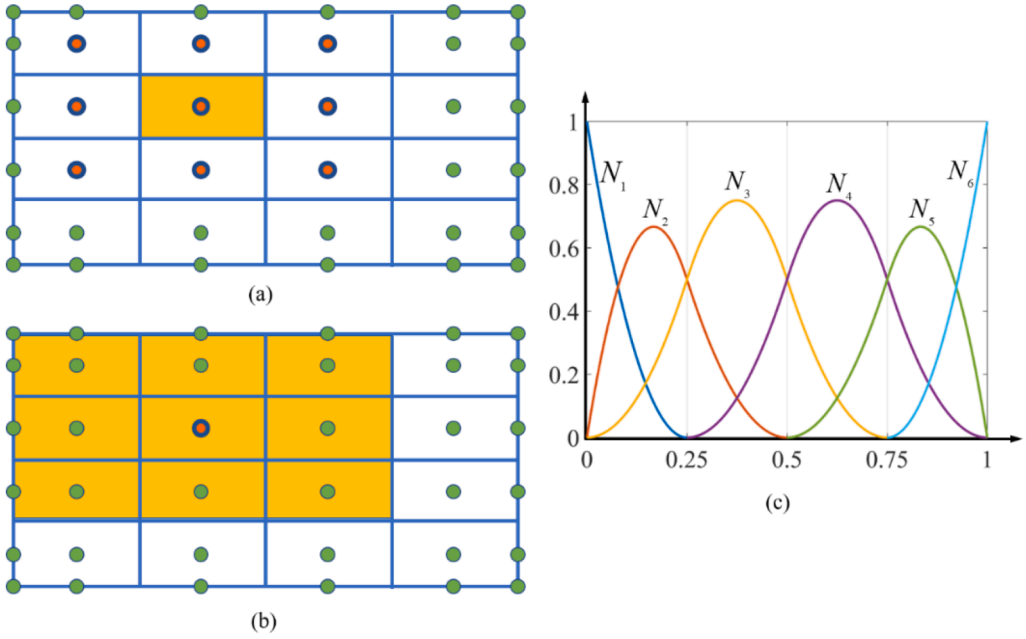


Fig. 1. The relation between control points and elements (p, $q = 2$): (a) the physical parameters of i th element are affected by the nearby $(p + 1) \times (q + 1)$ control points; (b) the i th control point and its corresponding elements; (c) the basis function of NURBS.

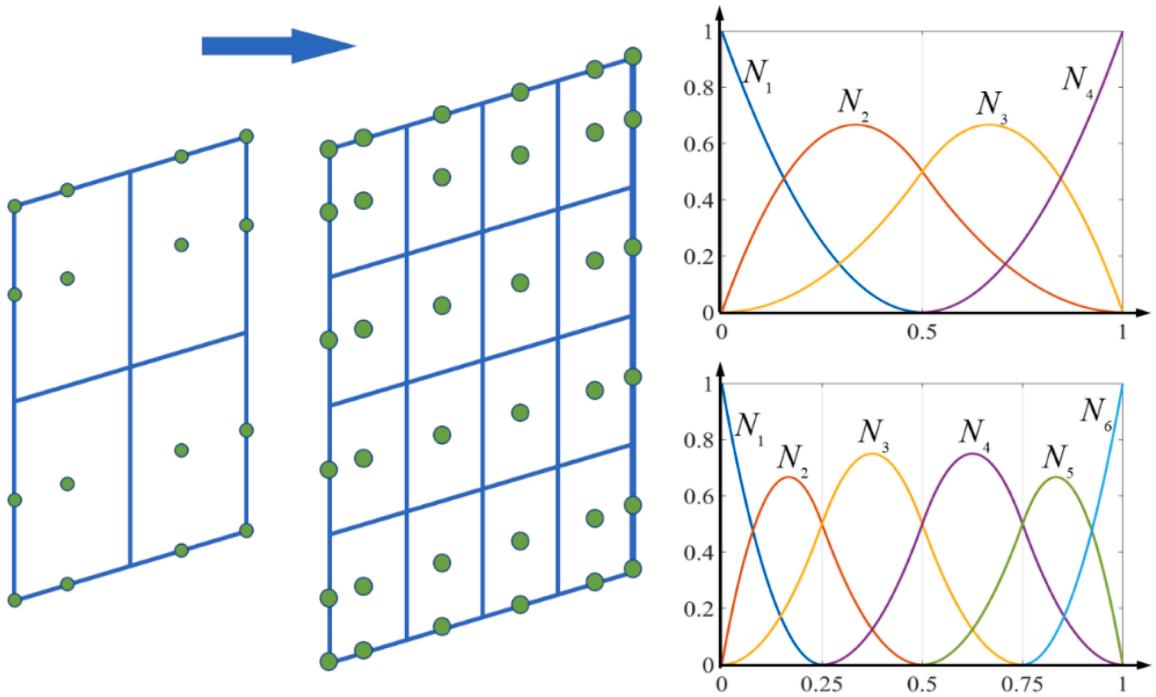


Fig. 2. The h-refinement in generating finer grid.

$$\boldsymbol{B} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \cdots & \frac{\partial N_{nmc}}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & \cdots & 0 & \frac{\partial N_{nmc}}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \cdots & \frac{\partial N_{nmc}}{\partial y} & \frac{\partial N_{nmc}}{\partial x} \end{bmatrix} \tag{8}$$

where x and y represent the position parameter coordinates of the 2D case, nmc represents the product of the control points in Ξ, indicating that $nmc = nc \times mc$.

The Jacobian matrix $\boldsymbol{J}_1$ can be calculated as:

$$\boldsymbol{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{9}$$

When the Gauss orthogonal domain [-1, 1] is transformed into the NURBS parameter domain $[\xi_i, \xi_{i+1}) \times [\eta_j, \eta_{j+1})$ in a linear fashion, the

Table 1
The algorithm implementation for hMG.

Algorithm 1 hMG
Input:
Global stiffness matrix $\boldsymbol{K}_l$, load vector $\boldsymbol{F}_l$, displacement vector $\boldsymbol{U}_l$, prolongation matrix $\boldsymbol{P}_l$ and restriction matrix $\boldsymbol{R}_l$.
Output:
The exact iteration solution $\boldsymbol{U}_l$
Function: $\boldsymbol{U}_l = hMG\_Solve(\boldsymbol{F}_l, \boldsymbol{U}_l, \boldsymbol{K}_l, \boldsymbol{P}_l)$
1. $\boldsymbol{U}_l$ = Damped Jacobi($\boldsymbol{K}_l, \boldsymbol{U}_l, \boldsymbol{F}_l, \omega$) // Pre-smoother
2. $\boldsymbol{F}_{l+1} = \boldsymbol{R}_l \cdot (\boldsymbol{F}_l - \boldsymbol{K}\boldsymbol{U}_l)$ // Restriction
3. If (The grid of K_{l+1} is coarse enough)
4. $\boldsymbol{U}_{l+1} = \boldsymbol{K}_{l+1}^{-1}\boldsymbol{F}_{l+1}$ // Direct solution
5. else
6. $\boldsymbol{U}_{l+1} = hMG\_Solve(\boldsymbol{F}_{l+1}, \boldsymbol{U}_{l+1}, \boldsymbol{K}_{l+1}, \boldsymbol{P}_{l+1})$
7. end
8. $\boldsymbol{U}_l = \boldsymbol{U}_l + \boldsymbol{P}_l \cdot \boldsymbol{U}_{l+1}$ //Prolongation
9. $\boldsymbol{U}_l = DampedJacobi(\boldsymbol{K}_b \boldsymbol{U}_b \boldsymbol{F}_b \omega)$ //Post-smoother
10. Return $\boldsymbol{U}_l$

Table 2
Algorithm implementation for the hMGCG.

Algorithm 2 hMGCG
Input:
Global stiffness matrix $\boldsymbol{K}$, load vector $\boldsymbol{F}$, prolongation matrix $\boldsymbol{P}_l$ and restriction matrix $\boldsymbol{R}_l$
Output:
The exact iteration solution of displacement vector $\boldsymbol{U}_k$.
Function: $\boldsymbol{U} = hMGCG(\boldsymbol{K}, \boldsymbol{U}, \boldsymbol{F}, \boldsymbol{P}_b \boldsymbol{R}_l)$
1. Initialize paramenters: maximum iteration *maxiter* and tolerance *tol*.
2. $k = 0$ //Iteration counter of Function
3. $\boldsymbol{r}_0 = \boldsymbol{F} - \boldsymbol{K}\boldsymbol{U}_k$ //Calculating residual
4. $\boldsymbol{p}_0, \boldsymbol{z}_0 = hMG\_Solve(\boldsymbol{F}, \boldsymbol{U}_k, \boldsymbol{K}, \boldsymbol{P}_l, \boldsymbol{R}_l)$ //Using hMG as the preconditioner
5. while $k <$ *maxiter*:
6. $\boldsymbol{\alpha}_k = \boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{z}_k / \boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{K}\boldsymbol{p}_k$ //Calculating step length
7. $\boldsymbol{U}_{k+1} = \boldsymbol{U}_k + \boldsymbol{\alpha}_k \boldsymbol{p}_k$ //Updating solution vector
8. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_{k+1} - \boldsymbol{\alpha}_k \boldsymbol{K}\boldsymbol{p}_k$ // Updating residual
9. if $\|\boldsymbol{r}_{k+1}\| / \|\boldsymbol{r}_0\| < tol$ //Checking convergence criterion
10. | Break
11. end
12. $\boldsymbol{z}_{k+1} = hMG\_Solve(\boldsymbol{F}, \boldsymbol{U}_k, \boldsymbol{K}, \boldsymbol{P}_l, \boldsymbol{R}_l)$ // Updating preconditioner
13. $\boldsymbol{\beta}_k = \boldsymbol{r}_{k+1}^T \boldsymbol{z}_{k+1} / \boldsymbol{r}_k^T \boldsymbol{z}_k$ // Updating search direction
14. $\boldsymbol{p}_{k+1} = \boldsymbol{z}_{k+1} + \boldsymbol{\beta}_k \boldsymbol{p}_k$
15. $k = k + 1$ //Updating iteration counter
16. end
17. Return $\boldsymbol{U}_k$

Table 3
Algorithm implementation of the adaptive damped Jacobi method.

Algorithm 3 Adaptive damped Jacobi method
Input:
Real matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, the initial guess solution $\boldsymbol{x}_0$, the optimal damping cofficient $\omega_{opt}$the iteration of adaptive damped Jacobi *iter*, the tolerance *tol*.
Output:
The approximate solution $\boldsymbol{x}_0$
Function $\boldsymbol{x}_0 = damped\_Jacobi(\boldsymbol{A}, \boldsymbol{x}_0, \boldsymbol{b}, \omega_{opt}, iter, tol)$
1. $k = 0$ //Iteration counter
2. $\boldsymbol{D} = 1./diag(\boldsymbol{A})$ //Obtain the inverse of the diagonal
3. $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A} \cdot \boldsymbol{x}$ // Calculate the error
4. $err = norm(\boldsymbol{r}, 2)$
5. while $err < tol$ & $k <$ *iter*:
6. $\boldsymbol{x} = \boldsymbol{x}_0 + \omega_{opt} \cdot \boldsymbol{D}^{\mathrm{T}} \cdot \boldsymbol{r}$ //Update the approximate solution
7. $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A} \cdot \boldsymbol{x}$ // Update the err
8. $err = norm(\boldsymbol{r}, 2)$ // Check convergence criterion
9. $\boldsymbol{x}_0 = \boldsymbol{x}$
10. $k = k + 1$
11. end
12. Return $\boldsymbol{x}_0$

Table 4
Algorithm implementation of the power method.

Algorithm 3 Power method
Input:
Real matrix $\boldsymbol{A}$, the initial guess for eigenvector x_0.
Output:
The corresponding eigenvector $\boldsymbol{x}_0$ and the largest eigenvalue λ of $\boldsymbol{A}$
Function $[\lambda, \boldsymbol{x}_0] = Power\_method(\boldsymbol{A}, \boldsymbol{x}_0)$
1. Initialize parameters: maximum iteration maxiter and tolerance for iteration *tol*.
2. $k = 0$ //Iteration counter
3. while $err < tol$:
4. $\quad \boldsymbol{b} = \max(\boldsymbol{x}_0)$
5. $\quad \boldsymbol{y} = \boldsymbol{x}_0 / b$ // Normalize the eigenvector estimate
6. $\quad \boldsymbol{x}_0 = \boldsymbol{A} \cdot \boldsymbol{y}$ // Update the eigenvector estimate
7. $\quad err = |\max(\boldsymbol{x}_0) - \boldsymbol{b}|$ // Check convergence criterion
8. $\quad \lambda = \max(\boldsymbol{x}_0)$
9. $\quad k = k + 1$
10. end
11. Return λ, $\boldsymbol{x}_0$

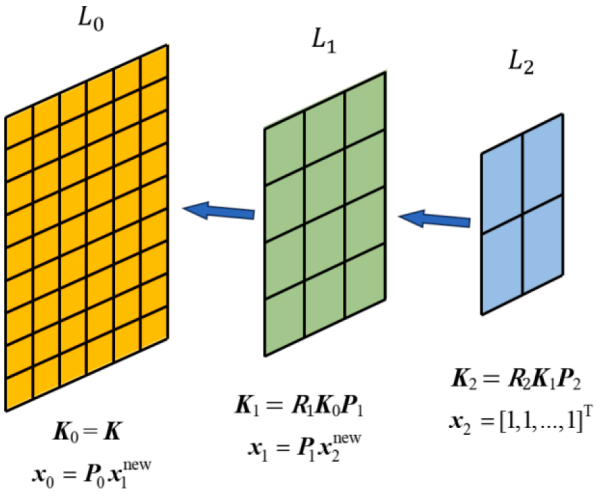


Fig. 3. The illustration of vertical strategy: using the eigenvector: using the eigenvector sequence of the coarse grid as the initial solution for the fine grid.

mapping relationship can be expressed as follows:

$$\begin{cases} \xi = \dfrac{\xi_{i+1} - \xi_i}{2}(\bar{\xi} - 1) + \xi_i \\ \eta = \dfrac{\eta_{j+1} - \eta_j}{2}(\bar{\eta} - 1) + \eta_j \end{cases} \tag{10}$$

where $\bar{\xi}$and $\bar{\eta}$ are the parameters defined in the Gauss orthogonal domain. Therefore, the Jacobi matrix $\boldsymbol{J}_2$ can be calculated as:

$$\boldsymbol{J}_2 = \begin{bmatrix} \dfrac{\partial \xi}{\partial \bar{\xi}} & \dfrac{\partial \eta}{\partial \bar{\xi}} \\ \dfrac{\partial \xi}{\partial \bar{\eta}} & \dfrac{\partial \eta}{\partial \bar{\eta}} \end{bmatrix} = \begin{bmatrix} \dfrac{\xi_{i+1} - \xi_i}{2} & 0 \\ 0 & \dfrac{\eta_{j+1} - \eta_j}{2} \end{bmatrix} \tag{11}$$

2.3. SIMP-based ITO

TO aims to identify the most efficient material distribution by minimizing strain energy while adhering to specified constraints. In the SIMP-based ITO, each element is allocated an element density ranging from 0 to 1. The relationship between the element density ρ_e and the elastic modulus E_e is presented as:

$$E_e(\rho_e) = E_{min} + \rho_e^{\beta}(E_0 - E_{min}) \quad \rho_e \varepsilon [0, 1] \tag{12}$$

where E_{min} refers to the minimum value of the elastic modulus, which is a small positive parameter that ensures the non-singularity of the

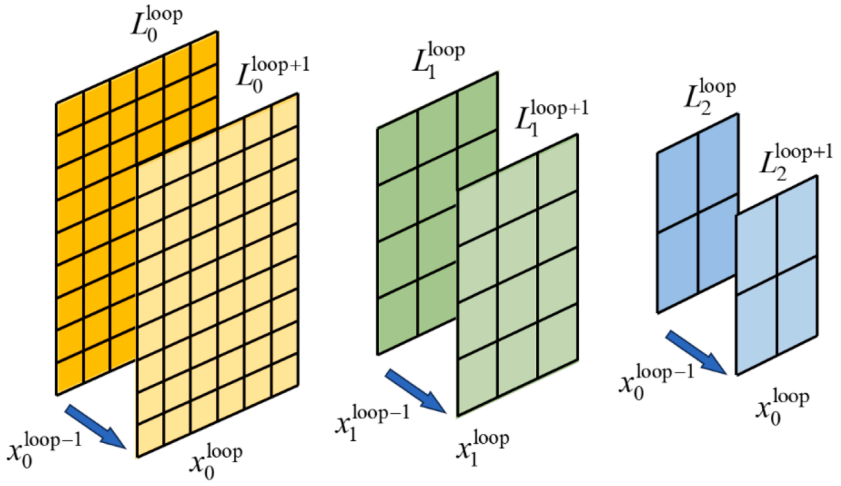


Fig. 4. The illustration of the horizontal strategy: using the eigenvector sequence of the stiffness matrix as the initial solution for the next iteration's stiffness matrix at the same level.

Table 5
Algorithm implementation for the preconditioned power method.

Algorithm 4 Preconditioned power method
Input:
Iterations of ITO *loop*, real matrix $\boldsymbol{A}_l$ in all levels, prolongation matrix $\boldsymbol{P}_l$ in all levels, the multigrid level l.
Output:
The maximum eigenvalue λ_l in different levels of $\boldsymbol{A}_l$
Function $[\lambda_1, ..., \lambda_n] = Pre\_Power(\boldsymbol{A}_l, \boldsymbol{P}_l, loop)$
1. Initialize parameters: maximum iteration *maxiter* and tolerance for iteration *tol*
2. If *loop*==1: // Adopting vertical accelerating strategy
3. $\boldsymbol{x}_n^{loop} = [1, 1, ..., 1]^T$ //n is the total number of levels,
4. for $i = n$: −1: 1
5. $\left[\lambda_i, \boldsymbol{x}_i^{loop}\right] = Power\_method\left(\boldsymbol{A}_i, \boldsymbol{x}_i^{loop}\right)$
6. $\boldsymbol{x}_{i-1}^{loop} = \boldsymbol{P}_i \cdot \boldsymbol{x}_i^{loop}$
7. end
8. else // Adopting horizontal acceleration strategy
9. for $i = n$: −1: 1
10. $\left[\lambda_i, \boldsymbol{x}_i^{loop}\right] = Power\_method\left(\boldsymbol{A}_i, \boldsymbol{x}_i^{loop-1}\right)$
11. end
12. end
13.$k = k + 1$
14. Return λ_1,..., λ_n

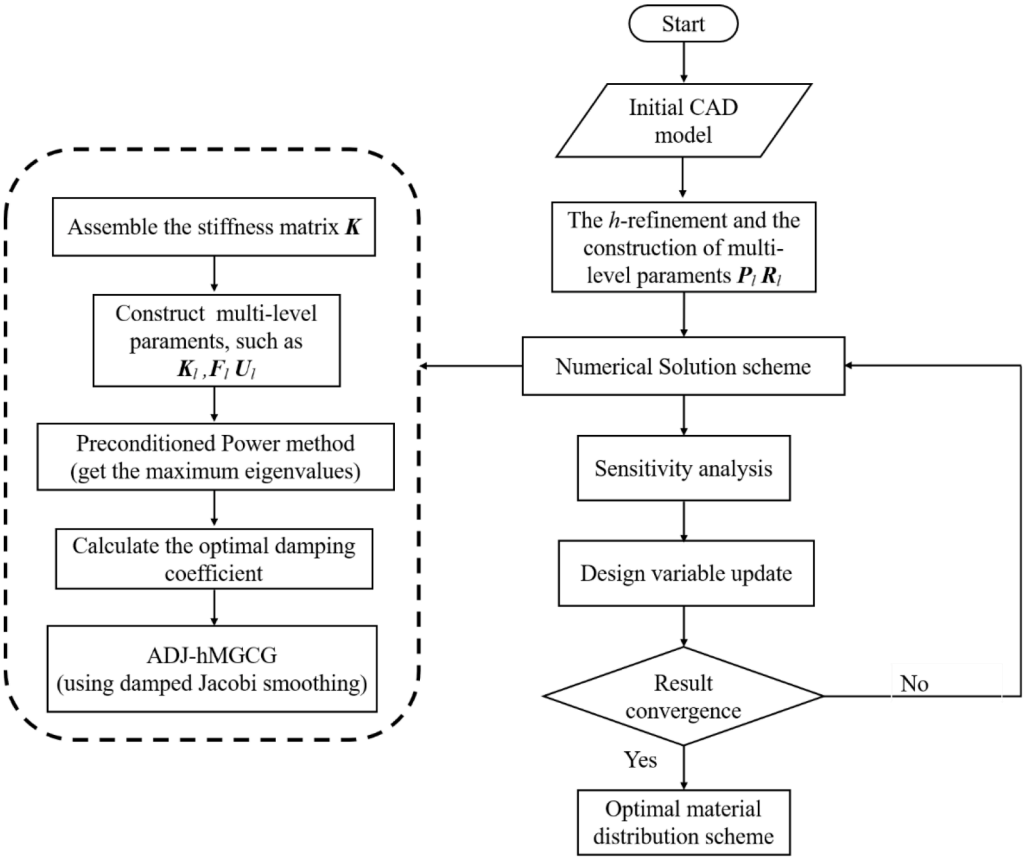


Fig. 5. Algorithm flowchart of the ADJ- hMGCG in ITO.

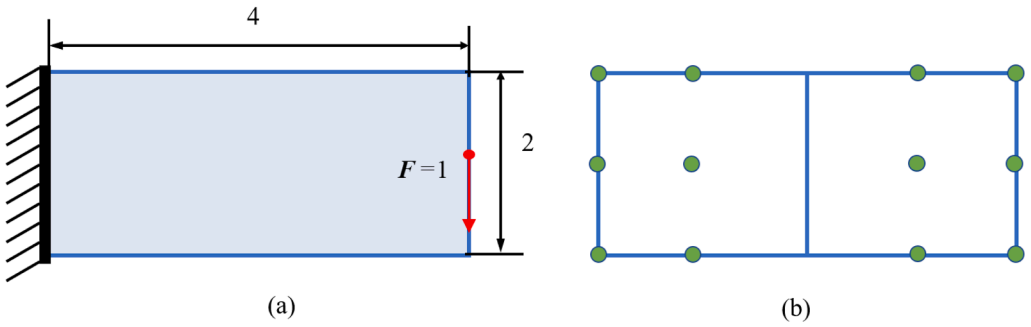


Fig. 6. The cantilever beam benchmark: (a) design domains and boundary conditions; (b) initial control points (p, q = 2).

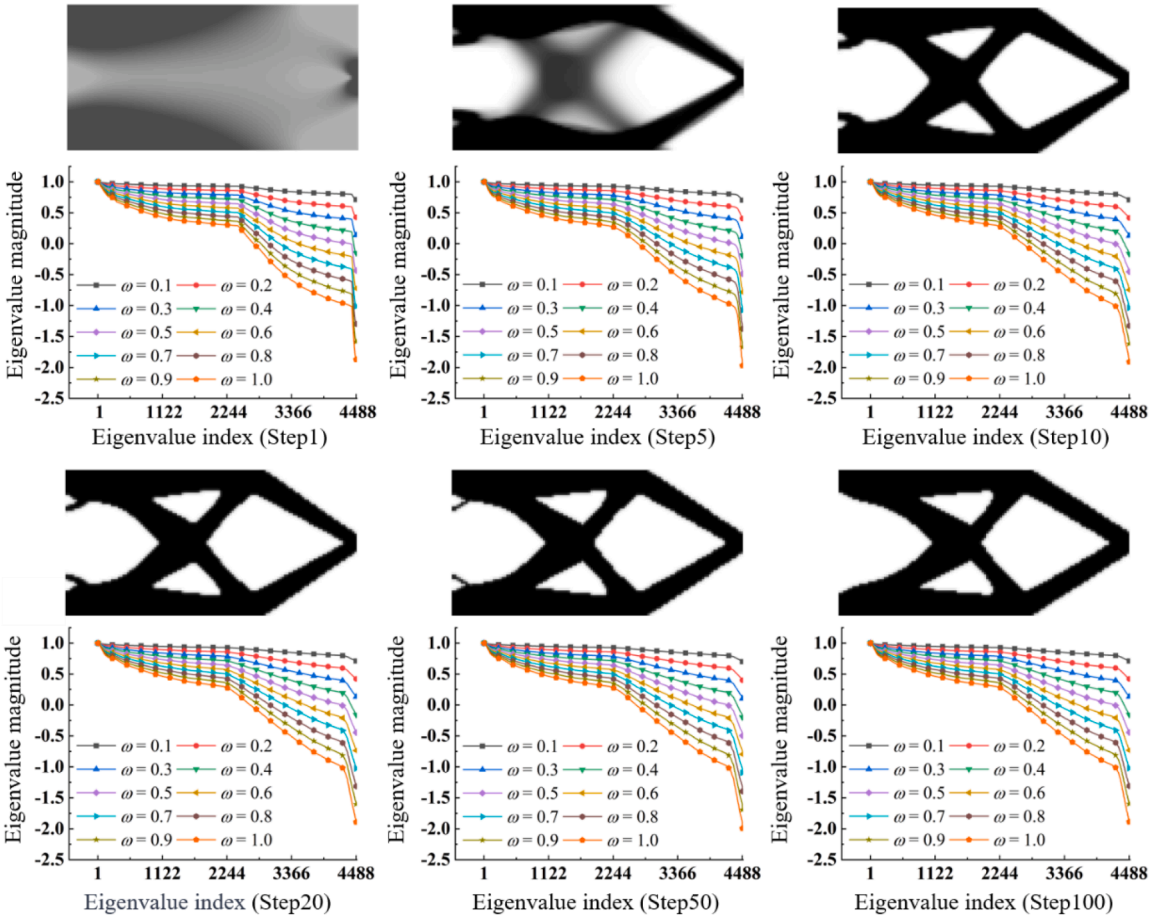


Fig. 7. The eigenvalues in descending order of stiffness matrix under different steps of ITO.

Table 6
The number of iterations of the GMGCG under different damping coefficients.

steps\ω	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	AD
1	40	33	28	25	22	20	18	78	142	—	16
5	41	33	28	25	22	20	19	80	146	—	19
10	43	35	30	27	24	22	21	106	184	—	22
20	46	39	33	28	24	22	22	102	159	—	22
50	49	40	34	30	27	24	22	99	170	—	23
100	47	39	33	28	25	23	22	93	177	—	22

stiffness matrix. E_0 denotes the elastic modulus of the solid element. The penalty coefficient, β, is utilized to achieve binarization of intermediate density and is usually set to $\beta = 3$. The minimum compliance optimization model can be expressed as:

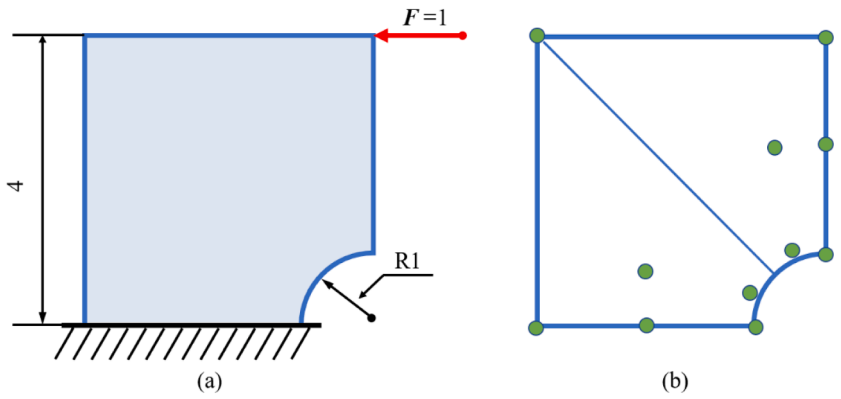


Fig. 8. The plate with holes benchmark: (a) design domains and boundary conditions; (b) initial control points (p, q = 2).

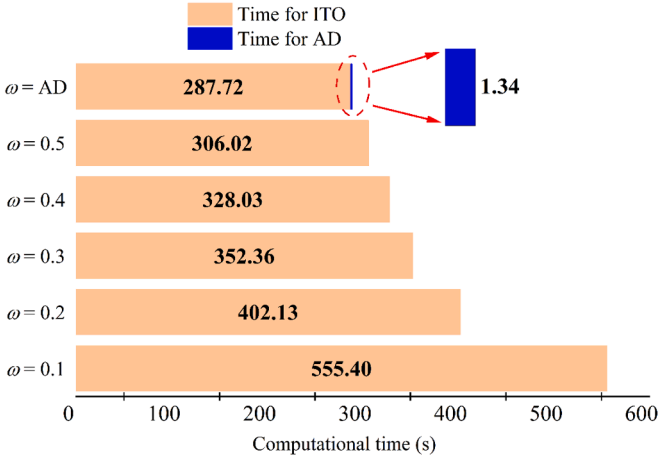


Fig. 9. Computational time of 200 iterations in different damping coefficients.

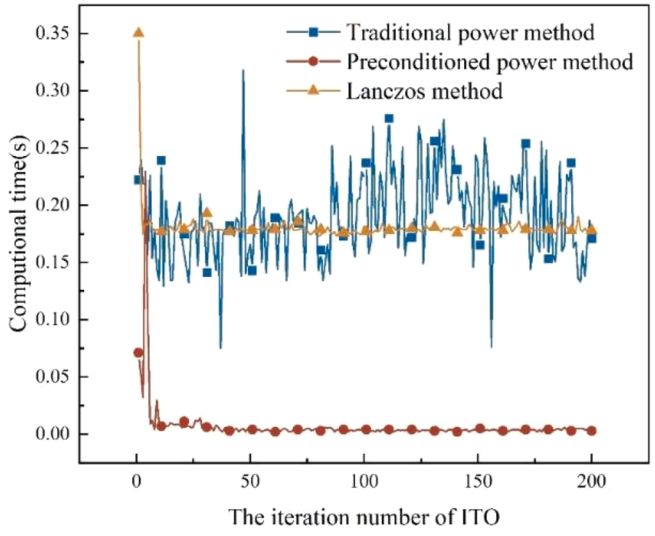


Fig. 10. The computational time of solving damping coefficients with the traditional power method, preconditioned power method and Lanczos method.

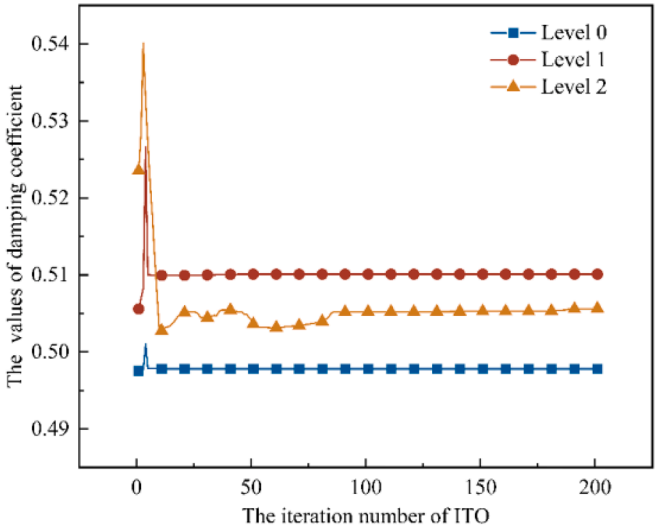


Fig. 11. The damping coefficients of ADJ-hMGCG in ITO.

$$\begin{aligned} &find\boldsymbol{\rho} = [\rho_1, \rho_2, \rho_3, \cdots, \rho_{ne}]^{\mathrm{T}} \\ &\min C(\boldsymbol{\rho}) = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{U} = \sum_{e=1}^{ne} E_e(\rho_e)\boldsymbol{u}_e^{\mathrm{T}}\boldsymbol{k}_e\boldsymbol{u}_e \\ &\quad s.t. \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F} \\ &\quad V(\boldsymbol{\rho})/V_0 \le V_F \\ &\quad 0 \le \rho_{\min} \le \rho_e \le 1 \end{aligned} \tag{13}$$

where $C(\boldsymbol{\rho})$ represents the compliance of the structure; $\boldsymbol{\rho}$ is design variable density; ne is the total number of elements; $\boldsymbol{U}$ refers to the displacement vector of the structure; $\boldsymbol{F}$ denotes the load vector of the structure; $\boldsymbol{K}$ is the global stiffness matrix of the structure; $\boldsymbol{u}_e$ represents the displacement vector of elements; V_0 and $V(\boldsymbol{\rho})$ represent the volume of the design domain and material usages; V_F is the volume fraction. The numerical calculations in ITO are carried out at the control points within a given physical field. Thus, the density of i th element ρ_{ei} can be represented by the control point density as:

$$\rho_{ei} = \sum_{j \in c_i} N_{ij}(ic)\rho_{nj} \tag{14}$$

where c_i represents the set of all control points that affect i th element, ic denotes the center of the i th element and $N_{ij}(ic)$ is the NURBS basis function of control points c_i corresponding to the center of the i th element, ρ_{nj} is the density of j-th control point. Therefore, the connection between control points density and elements density is depicted in

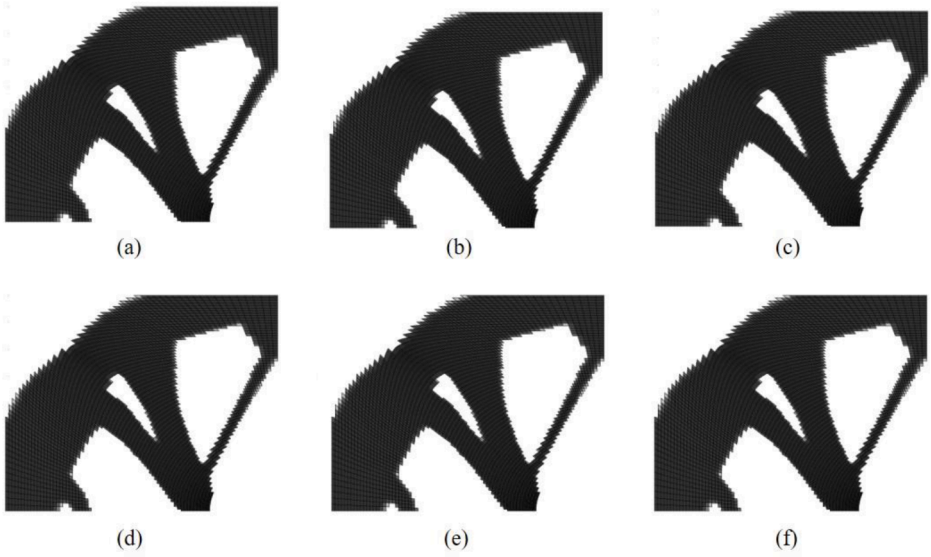


Fig. 12. The final optimized structures of different damping coefficients: (a) hMGCG with $\omega = 0.1$; (b) hMGCG with $\omega = 0.2$; (c) hMGCG with $\omega = 0.3$; (d) hMGCG with $\omega = 0.4$; (e) hMGCG with $\omega = 0.5$; (f) ADJ-hMGCG.

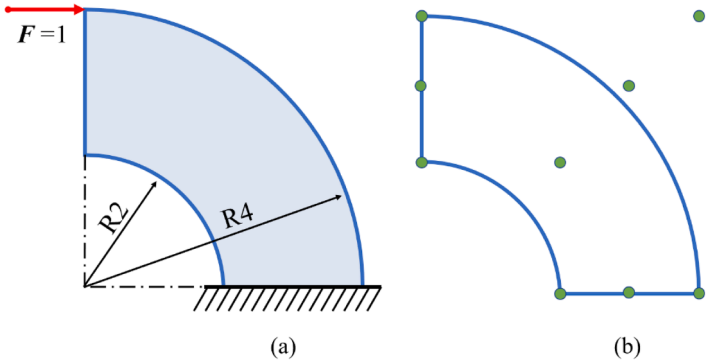


Fig. 13. The plate with holes benchmark: (a) Design domains and boundary conditions; (b) Initial control points (p, $q = 2$).

Table 7
The average time in each ITO iteration with different solvers.

case	ADJ-hMGCG (s)	LMGCG with damped Jacobi (s)	hMGCG with SSOR (s)	Jacobi-PCG (s)
64 × 64	0.554	0.675	0.872	0.417
128 × 128	3.17	3.742	3.953	3.006
256×256	13.620	16.824	18.765	26.465

Fig. 1, where the NURBS parameter space is [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1] × [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1], the degree p, $q = 2$ and the weight factor $w_{i,j} = 1$.

The control points can affect the element density of the multiple surrounding elements, so the checkerboard phenomenon will not appear in the process. The sensitivity of the objective function $C(\boldsymbol{\rho})$ in SIMP-based ITO is formulated as

$$\frac{\partial C}{\partial \rho_{nj}} = \sum_{j \varepsilon C_i} \frac{\partial C}{\partial \rho_{ei}} \frac{\partial \rho_{ei}}{\partial \rho_{nj}} = \sum_{j \varepsilon C_i} -\beta \rho_{ei}^{\beta-1} (E_0 - E_{\min}) \boldsymbol{u}_{ei}^{\mathrm{T}} \boldsymbol{k}_{ei} \boldsymbol{u}_{ei} \frac{\partial \rho_{ei}}{\partial \rho_{nj}} \tag{15}$$

And the sensitivity of the constraint function can be derived by the chain rule as follows:

$$\frac{\partial f}{\partial \rho_{nj}} = \sum_{j \in c_i} \frac{\partial f}{\partial \rho_{ei}} \frac{\partial \rho_{ei}}{\partial \rho_{nj}} = \sum_{j \in c_i} \frac{V_{ei}}{V_0} \frac{\partial \rho_{ei}}{\partial \rho_{nj}} \tag{16}$$

where f represents the constraint function, which is defined as $f = \frac{V(\rho)}{V_0} - V_F$, V_{ei} is the volume of i th element. The derivative of the density of element ρ_{ei} over the density of control points ρ_{nj} is

$$\frac{\partial \rho_{ei}}{\partial \rho_{nj}} = \sum_{j \varepsilon c_i} N_{ij}(ic) \tag{17}$$

3. ADJ-hMGCG method

3.1. Multilevel based on h-refinement

In the applications of ITO, an accurate physical model requires the fine grid, and the h-refinement is a method to insert multiple knots into a NURBS curve to get a finer grid. It is important to note that the h-refinement just changes the vector space without altering the curve.

Assuming that $C(\xi) = \sum_{i=1}^{nc} R_{i,p}(\xi) \boldsymbol{Q}_i$ is defined on the knot vector $\boldsymbol{\Xi} = [\xi_1, \xi_2, \ldots, \xi_{nc+p+1}]$, if we need to insert a knot $\xi'(\xi' \in [\xi_t, \xi_{t+1}])$ to the knot vector Ξ, the new knot vector becomes $\boldsymbol{\Xi}^{new} = \left[\xi_1, \xi_2, \cdots, \xi_t, \xi', \xi_{t+1}, \xi_{nc+p+1}\right]$, and the new control points sequence $\boldsymbol{Q}^{\mathrm{new}}$ can be generated by the old control points sequence $\boldsymbol{Q}^{\mathrm{old}}$ through:

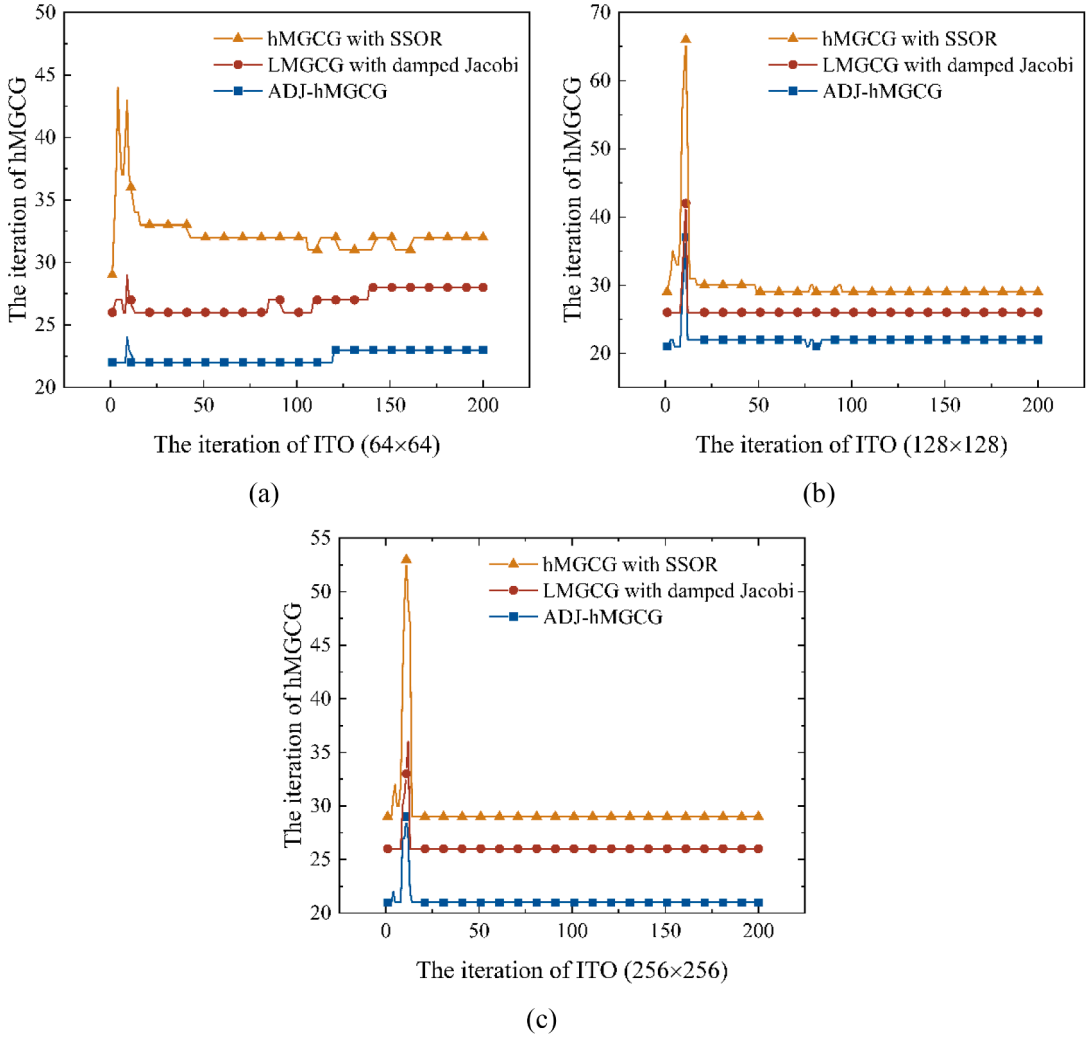


Fig. 14. The iteration of hMGCG in each iteration of ITO: (a) 64 × 64; (b) 128 × 128;(c) 256 × 256.

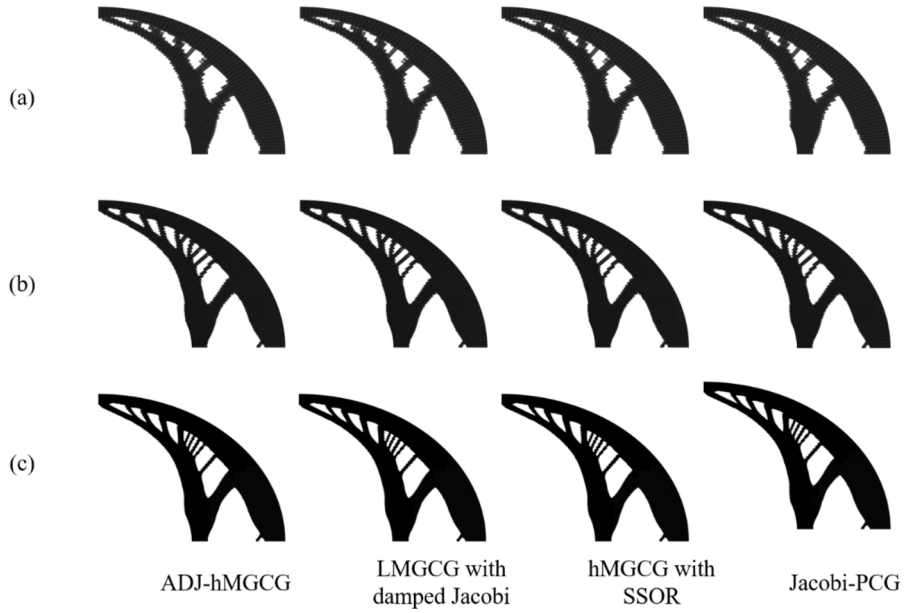


Fig. 15. The final structures of different solvers: (a) The different solvers in 64 × 64; (b) The different solvers in 128 × 128; (c) The different solvers in 256 × 256.

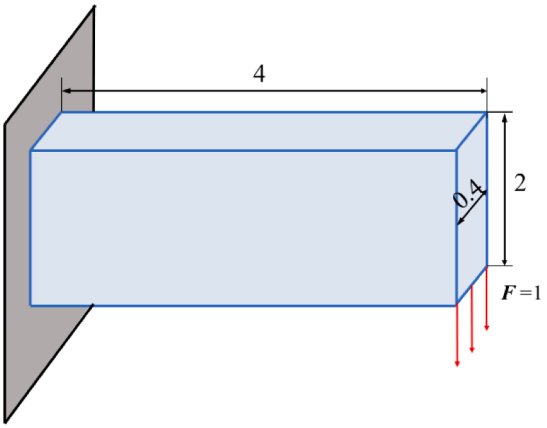


Fig. 16. The design domains and boundary conditions of a 3D cantilever beam.

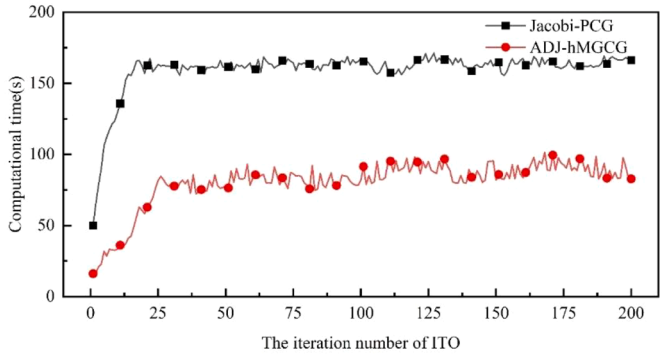


Fig. 17. The computational time of solving large-scale sparse linear equations with Jacobi -PCG and ADJ-hMGCG.

$$\boldsymbol{Q}_i^{\text{new}} = \alpha_i \boldsymbol{Q}_i^{\text{old}} + (1-\alpha_i)\boldsymbol{Q}_{i-1}^{\text{old}} \tag{18}$$

where the interpolation coefficient α_i can be represented by

$$\alpha_i = \begin{cases} 1 & 1 \le i \le t-p \\ \dfrac{\xi' - \xi_i}{\xi_{i+p} - \xi_i} & t-p+1 \le i \le t \\ 0 & t+1 \le i \le nc+p+2 \end{cases} \tag{19}$$

Therefore, the h-refinement can be represented as a matrix multiplication, specifically $\boldsymbol{Q}^{\text{new}} = \boldsymbol{P} \cdot \boldsymbol{Q}^{\text{old}}$, where $\boldsymbol{P}$ is the projection coefficient matrix maps the old control points sequence to the new control points sequence. The matrix $\boldsymbol{P}$ can be further expressed as:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I}_{k-p} & 0 & 0 \\ 0 & \boldsymbol{\alpha} & 0 \\ 0 & 0 & \boldsymbol{I}_{nc+p+2} \end{bmatrix} \tag{20}$$

where $\boldsymbol{I}$ is the unit diagonal matrix. When both the ξ- and η-directions of a NURBS surface require h-refinement, the control points can be inserted in the η-direction by first fixing the ξ-direction and using h-refinement. Then, fixing the η-direction, refinement can be performed in the ξ-direction. The projection coefficient matrix from the old to the new control points can be expressed as:

$$\boldsymbol{P}_{\xi\eta} = \boldsymbol{P}_{\xi} \cdot \boldsymbol{P}_{\eta} \tag{21}$$

Therefore, in the 2D case, new control points can be represented as:

$$\boldsymbol{Q}^{\text{new}} = \boldsymbol{P}_{\xi\eta} \cdot \boldsymbol{Q}^{\text{old}} \tag{22}$$

The process of the h-refinement in a 2D example can be found in Fig. 2. The knot vector of the initial can be represented as $\Xi_{\text{old}} = [0, 0, 0, 0.5, 1, 1, 1] \times [0, 0, 0, 0.5, 1, 1, 1]$, while the new knot vector after the h-refinement is $\Xi_{\text{new}} = [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1] \times [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1]$. Compared to the 2D example, the h-refinement in 3D requires fixing two directions and inserting the control points in the remaining direction.

3.2. Basis theory of multigrid method

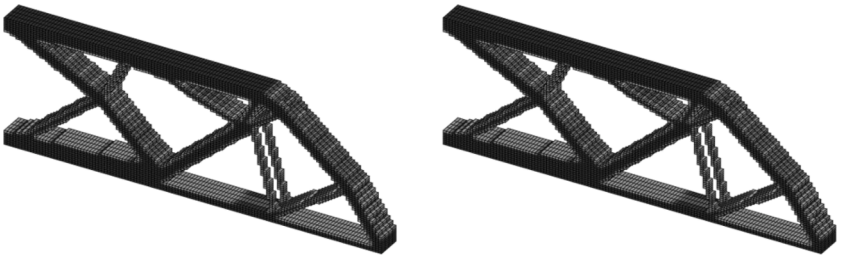
The multigrid method, which is classified into geometric multigrid (GMG) and algebraic multigrid (AMG), is a high-efficient technique for solving large-scale linear equations by performing the basis transformation for the corresponding coefficient vectors between the coarse and fine grids. The GMG method uses the geometric relationship between different levels to solve the equations and the h-refinement utilizes the projection coefficient matrix to express the geometric relationship between the levels. Therefore, the GMG method in ITO does not require an extra algorithm to construct the prolongation matrix. The GMG method using h-refinement is called hMG. The hMG method can be divided into the preprocessing process and the solving process. In the preprocessing process, the restriction matrix $\boldsymbol{R}$ transforming a fine grid to a coarse grid and the prolongation matrix $\boldsymbol{P}$ transforming the coarse grid to the fine grid are constructed. Therefore, the coarse stiffness matrix $\boldsymbol{K}_{l+1}$ can be expressed by the fine stiffness matrix $\boldsymbol{K}_l$ as:

$$\boldsymbol{K}_{l+1} = \boldsymbol{R}_l \boldsymbol{K}_l \boldsymbol{P}_l \tag{23}$$

where l represents the levels of multigrid. Besides, the Galerkin coarsening method, which sets the restriction matrix $\boldsymbol{R}$ as $\boldsymbol{R} = \boldsymbol{P}^{\text{T}}$, is proved to be an effective solving method.

The solving process consists of five steps, namely pre-smoothing, restriction, direct solution, prolongation, and post-smoothing. The algorithm implementation for hMG is depicted in Table 1.

Besides, the hMG method and the conjugate gradient method are both efficient in solving large-scale sparse linear equations but exhibit



(a) (b)

Fig. 18. The optimized structure of the 3D cantilever beam after 500 iterations with (a) ADJ-hMGCG and (b) Jacobi-PCG.

slower convergence in some specific mathematical models [40,41]. Utilizing the hMG as the preconditioner of conjugate gradient, namely geometric multigrid conjugate gradients (hMGCG), has a better performance in solving linear equations. The algorithm implementation for the hMGCG is shown in Table 2.

3.3. Adaptive damped Jacobi method

Traditional multigrid method typically employs the damped Jacobi iteration, over-relaxation iteration, or any other smoothers to solve large-scale equations of form $\boldsymbol{F} = \boldsymbol{KU}$ [42,43]. During the iteration process of the Jacobi method, the calculations for each variable are independent, allowing for effective parallelization. Therefore, this section focuses on analyzing the optimal damping coefficient of the damped Jacobi iteration method. The iterative equation utilized in the damped Jacobi iteration method is:

$$\boldsymbol{U}_{k+1} = \boldsymbol{U}_k + \omega \boldsymbol{D}^{-1}(\boldsymbol{F} - \boldsymbol{K}\boldsymbol{U}_k) \tag{24}$$

where k is the number of iterations in the damped Jacobi method, ω represents the damping coefficient and $\boldsymbol{D}^{-1}$ denotes the inverse of the diagonal of the stiffness matrix $\boldsymbol{K}$.

The convergence criterion of the damped Jacobi method is

$$\boldsymbol{\rho}_d = \left|1 - \omega\lambda_{\max}\left(\boldsymbol{D}^{-1}\boldsymbol{K}\right)\right| < 1 \tag{25}$$

where ρ_d represents the spectral radius of the iteration matrix and λ_{max} denotes the maximum eigenvalues. If $\rho_d > 1$, the smoother will fail to converge. To address this issue, many researchers try using a low relaxation damping coefficient, such as using $\omega = 2/3$ [44], to constrain the larger eigenvalues. Although it is an effective method for achieving convergence for specific models, it cannot be suitable for each model. Thus, it is worth exploring an adaptive damping coefficient to make the multigrid more efficient and robust.

According to the convergence criterion, $0 < \omega \leq 2/\lambda_{max}(\boldsymbol{D}^{-1}\boldsymbol{K})$ is the convergence interval. In ITO, the stiffness matrix is a positive definite symmetric matrix, and the eigenvalues $\lambda(\boldsymbol{D}^{-1}\boldsymbol{K})$ are non-negative. When $\lambda(\boldsymbol{D}^{-1}\boldsymbol{K})$ approaches 0, the correlation between ω and ρ_d becomes low, and this part of eigenvalues is close to 1, which is still within the convergence range. Hence, ω is mainly utilized to minimize the larger part of $\lambda(\boldsymbol{D}^{-1}\boldsymbol{K})$, and the optimal solution for ω_{opt} is:

$$\boldsymbol{\omega}_{opt} = \frac{2}{\lambda_{\max}} \tag{26}$$

Therefore, the solution of the optimal damping coefficient is transformed into the solution of the maximum eigenvalue. The adaptive damped Jacobi method is shown in Table 3.

3.4. Preconditioned power method

As mentioned above all, the optimal damping coefficient for the ADJ-hMGCG requires the calculation of the maximum eigenvalue. Eigenvalue solutions require much computational time, and with the scale increasing, the computational complexity increases exponentially. There are many algorithms for solving the eigenvalues of the matrix, including the power method, inverse iteration method, QR iteration method, and Lanczos method. More details about these methods can be found in [45-47]. The time complexity of the inverse iteration method and QR iteration method is $O(n^3)$. For large-scale problem, both the power method and the Lanczos method are commonly used iterative approaches for solving maximum eigenvalue. The power method just involves simple matrix-vector multiplications and vector normalizations in each iteration, making it computationally efficient and amenable to parallelization. Table 4 illustrates the algorithm implementation of the power method.

Besides, the convergence speed of the power method is dependent on the initial vector selection, and faster convergence can be obtained when the initial vector is closer to the maximum eigenvector. This paper proposes a preconditioned power method based on the properties of hMG and ITO. Firstly, a vertical strategy is presented by utilizing the characteristics of hMG, which prolongs the iterative solutions of the eigenvector sequence from the coarse level to the fine level. An all-ones vector, denoted as $\boldsymbol{x}_2$, is used to compute the eigenvalues of the stiffness matrix at L_2. After the iterations of the power method, the initial vector $\boldsymbol{x}_2$ is transformed into an approximate eigenvalue sequence $\boldsymbol{x}_2^{\text{new}}$ of the stiffness matrix $\boldsymbol{K}_2$. Then the $\boldsymbol{x}_2^{\text{new}}$ is prolonged to L_1 as the initial iteration vector for the Power method, with $\boldsymbol{x}_1 = \boldsymbol{P}_2\boldsymbol{x}_2^{\text{new}}$. Lastly, $\boldsymbol{x}_0$ is solved as the previous step as L_2 to L_1. Details of the vertical strategy algorithm can be found in Fig. 3.

Secondly, a horizontal strategy is employed by leveraging the similarity of optimized structure shapes in adjacent iterations. Specifically, the eigenvector sequence of the stiffness matrix in $loop1$ is utilized as the initial value for the stiffness matrix in $loop1 + 1$. When the compliance changes slowly, this initial solution approximates the exact solution. The algorithm is illustrated in Fig. 4.

There is no reference initial value available for the first calculation of the maximum eigenvalue of the stiffness matrix for the hMGCG, a vertical strategy can be employed by using the eigenvalues of the lower-dimensional stiffness matrix as a preconditioner. With the iterations of ITO, the stiffness matrix changes with the variation of element density, but its structure changes small. Hence, the horizontal strategy is a more effective way to compute the maximum eigenvalue. To achieve the highest computational efficiency, the vertical strategy is employed in $loop = 1$ and the horizontal strategy is utilized in $loop > 1$, where $loop$ represents the iteration number in ITO. Therefore, Table 5 illustrates the algorithm implementation of the preconditioned power method.

3.5. Algorithm implementation

The efficient and robust ITO, implemented in MATLAB, has been developed in this work to solve the minimum compliance problems. Fig. 5 illustrates the flowchart of the ITO with the ADJ-hMGCG. The preconditioned power method makes use of the characteristics of hMG and the ITO to accelerate the maximum eigenvalues calculation. Then the optimal damping coefficients can be chosen as Eq. (26). The ADJ-hMGCG is utilized to solve large-scale sparse linear equations. The effectiveness of the algorithm will be shown in Section 4, along with four classical examples in ITO.

4. Numerical examples

To verify the convergence and efficient performance of the ADJ-hMGCG, three two-dimensional examples and one three-dimensional example are presented in this section. All examples are run on a desktop computer with Intel Gold 5218 CPU of 2.29 GHz and MATLAB software.

Section 4.1 presents a cantilever beam example to analyze the impact of damping coefficients and adaptive damping coefficients in the hMGCG. In Section 4.2, a plate with a circle hole is proposed to prove the adaptive damping coefficients is more efficient than the traditional method using a constant damping coefficient. Section 4.3 tests a quadratic annulus to verify the ADJ-hMGCG is more efficient than other solving algorithms. The ADJ-hMGCG was implemented in a three-dimensional model, to further verify the efficiency and robustness of the algorithm in Section 4.4. All the examples assume Young's modulus $E_0 = 1$ and Poisson's coefficient $\mu = 0.3$. And all the examples use second-degree NURBS element. Two iterations are chosen for smoother, which results in the fastest solution speed. The convergence criterion of ITO is adopted for all numerical examples: the iterations are stopped when the maximum change in the design variables is less than 1 % or the number of iterations is more than 200.

4.1. Cantilever beam

In order to verify the impact of damping coefficients in the hMGCG, this section analyzes the stiffness matrix and iteration of several steps of a cantilever beam. The design domain, boundary conditions, and control points of the cantilever beam are shown in Fig. 6. The optimization objective of the example is the minimum compliance ITO model under the volume constraints, and the volume constraint ratio $V_F = 0.5$.

To obtain a more accurate discretization, we conducted 6 levels of h-refinement on the initial control points in both ξ- and η-directions. As a result, there are 128 $\times$ 64 elements in the ξ- and η-directions. Additionally, we constructed three levels of the grid for the hMGCG, where the stiffness matrix in different levels is 17160 $\times$ 17160, 4488 $\times$ 4488, 1224 $\times$ 1224 respectively. Fig. 7 shows the eigenvector sequence distribution of the stiffness matrix in the coarsest mesh under different optimized structures during the ITO. Table 6 represents the iterations of the hMGCG for the cantilever beam under different damping coefficients.

When the damping coefficient is $\omega = 1$, the smoother degenerates into the Jacobi iteration method, and the spectral radius is far greater than 1, which does not converge during the solution process. When $\omega \in [0.8, 0.9]$, the spectral radius of the iteration matrix under this damping coefficient is greater than 1, but the spectral radius of these damping coefficients is less than 1 on the finer level. Thus, the hMGCG in these damping coefficients converges but requires higher iteration numbers. When $\omega \in [0.1, 0.7]$, the spectral radius of the iteration matrix is less than 1, enabling rapid convergence of the multigrid solver. ω_{opt} is closer to the damping coefficients of 0.6 and 0.7. Consequently, setting the damped coefficient to 0.6 or 0.7 significantly enhances the performance of the damped Jacobi. Besides, using ADJ-hMGCG can accelerate the iteration performance by 1.03 times compared to the hMGCG method using $\omega = 0.7$.

4.2. Plate with a circle hole

In order to verify the efficiency and robustness of the ADJ-hMGCG, a plate with a circle hole example was used in this section. This model is referred to in the paper [14]. The design domain, boundary conditions and initial control points of this optimization model are shown in Fig. 8. The volume constraint ratio is $V_F = 0.5$.

To achieve a more precise discrete domain, we discretized the design domain with 128 $\times$ 64 elements through h-refinement, resulting in the scale of the finest grid stiffness matrix is 17160 $\times$ 17160. In addition, we established three layers of the grid for the hMGCG and the stiffness matrix is 17160 $\times$ 17160, 4488 $\times$ 4488, 1224 $\times$ 1224 respectively. In Fig. 9, we present the computational time of the hMGCG under varying damping coefficients of ITO. The hMGCG converges when the damping coefficient is below 0.6, and a faster solution speed can be achieved when $\omega = 0.5$, the ADJ-hMGCG outperforms the hMGCG with $\omega = 0.5$. Besides, the proposed preconditioned power method in this paper requires less time as well. Fig. 10 demonstrates the efficiency of the preconditioned power method compared to other algorithms. The iterations count of the Lanczos method is $j = 2\sqrt{n}$, where n is the row number of the stiffness matrix. More details can refer the paper [47]. The preconditioned power method acceleration ratio is 28.27 compared to traditional power method, 26.78 times that of Lanczos method.

The optimal damping coefficient varies for different models. For instance, the optimal damping coefficient for the cantilever beam in Section 4.1 is $\omega = 0.7$, while for the model in this section, it is $\omega = 0.5$. Thus, the ADJ-hMGCG algorithm exhibits superior robustness and enhances the efficiency of ITO.

Fig. 11 illustrates the values of the damping coefficient used in ITO. With iterations of ITO increase, the compliance changes slightly and the variation in the adaptive damping coefficient is also relatively small. Fig. 12 depicts the final optimization results using different solvers for the ITO. The final compliance falls within the acceptable error range, and the optimization outcomes are accurate.

4.3. Quadratic annulus

This section aims to demonstrate the efficiency of the ADJ-hMGCG and its greater applicability in problems with different scales. To this end, a quarter annulus example is employed to compare this algorithm with the conventional solution method. The design domain and boundary conditions and initial control point of the quarter annulus are presented in Fig. 13. The optimization objective is to obtain the minimum compliance ITO model subject to a volume constraint with a constraint ratio of $V_F = 0.5$.

To generate grid elements with varying levels of discretization, we discretized the design domain into scales of 64 $\times$ 64, 128 $\times$ 128, and 256 $\times$ 256 elements and construct three levels of the grid for hMGCG. Traditional large-scale equations solving methods include preconditioned conjugate gradient method, multigrid method, and so on. The convergence speed of the multigrid method is related to the projection coefficient matrix and the smoother. Thus, the linear interpolation multigrid conjugate gradients (LMGCG) with damped Jacobi [32], the hMGCG with SSOR and the Jacobi preconditioned conjugate gradient method (Jacobi-PCG) is used to verify the high-performance of ADJ-hMGCG. $\omega = 0.3$ is the better one in the convergence range, which is chosen for LMGCG and $\omega = 0.25$ is the best for hMGCG with SSOR. Table 7 provides a comparison of the solution speeds between the adaptive damping multigrid method and the traditional finite element method.

The ADJ-hMGCG algorithm outperforms traditional solvers in large-scale problems, achieving an acceleration ratio of 1.94 compared to Jacobi-PCG, 1.23 times that of LMGCG with damped Jacobi, and 1.377 times that of hMGCG with SSOR in 256 $\times$ 256. Fig. 14 illustrates the optimized structure under various grid sizes and algorithms.

The final compliance falls within an acceptable range of error, indicating accurate optimization results. The final structures are shown in Fig. 15. Thus, the ADJ-hMGCG is not hindered by the grid scale and is better for solving large-scale equations.

4.4. 3D cantilever beam

To further demonstrate the efficiency and robustness of the ADJ-hMGCG, we introduce a three-dimensional cantilever beam in this section, along with its design domain, boundary conditions, as depicted in Fig. 16. The objective of the optimization model is to minimize compliance while satisfying a volume constraint, where the volume constraint ratio is set at $V_F = 0.3$.

The 3D cantilever beam is discretized into second-degree NURBS elements of size 128 $\times$ 64 $\times$ 4 and the stiffness matrix of the finest grid is 154440 $\times$ 154440. Fig. 17 displays the computational time required for each iteration using different methods. After 200 iterations, Jacobi-PCG requires a total time of 31,917.44 s, while the ADJ-hMGCG requires a total time of 16,066.50 s, including adaptive coefficients solving time of 85.44 s. Compared to the Jacobi-PCG algorithm, ADJ-hMGCG achieves a speedup ratio of 1.98 times. Notably, the geometric multigrid-CG method has not been compared with the ADJ-hMGCG method, since the former fails to converge in three-dimensional large-scale scenarios in ITO. Fig. 18 shows the final optimal structure of ITO, and the final compliance is within an acceptable range of error, indicating the accuracy of the optimization results. Therefore, the ADJ-hMGCG is also applicable to 3D examples.

5. Conclusion

This paper proposes an efficient and robust ADJ-hMGCG algorithm for solving large-scale sparse linear equations in ITO. The proposed method includes determining the most appropriate damping coefficient in the finite element equation and achieving efficient implementation in

the field of ITO. The effectiveness and versatility of the ADJ-hMGCG are verified through the numerical examples of two-dimensional calculations, including cantilever beams, plate with a circle hole and quadratic annulus, as well as three-dimensional cantilever beam. The results demonstrate that the ADJ-hMGCG can solve the problem of selecting damping coefficients under different models and accelerate the equations solving process of ITO. Therefore, this method could be an efficient and robust solver for complex ITO models [48,49]. Besides, we program the solver in a general way without taking full use of the computational characteristics of MATLAB, so the solver cannot achieve its best computational efficiency. When using languages such as C/C++ or Fortran, the 3D multigrid solver implemented in Section 4.4 should take less than 1 second on a single selected CPU core [43]. This method is not limited to ITO but is also applicable to the TO based on finite elements. In the future, we will investigate the field of multigrid smoothers, including the Chebyshev smoother [50] and the block Jacobi smoother [51], to develop more efficient and versatile solvers. Furthermore, this method can be combined with other TO methods and extended to efficient and robust solutions for structural optimization problems in other fields such as heat and fluid.

Replication of results

All the data used to support the findings of this study are available from the corresponding author upon request.

CRediT authorship contribution statement

Shijie Luo: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Feng Yang:** Methodology, Data curation, Conceptualization. **Yingjun Wang:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

Data will be made available on request.

Acknowledgment

This work has been supported by National Natural Science Foundation of China [No. 52075184], Guangdong Basic and Applied Basic Research Foundation [No. 2024A1515011786]. These supports are gratefully acknowledged.

References

[1] Bendsøe MP, Kikuchi N. Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng. 1988;71:197–224. https://doi.org/10.1016/0045-7825(88)90086-2.
[2] Zheng N, Zhai X, Chen F. Topology optimization of self-supporting porous structures based on triply periodic minimal surfaces. Comput-Aided Des 2023;161:103542. https://doi.org/10.1016/j.cad.2023.103542.
[3] Sigmund O. A 99 line topology optimization code written in Matlab. Struct Multidiscip Optim. 2001;21:120–7. https://doi.org/10.1007/s001580050176.
[4] Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O. Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidiscip Optim. 2010;43:1–16. https://doi.org/10.1007/s00158-010-0594-7.
[5] Bendsøe MP, Sigmund O. Material interpolation schemes in topology optimization. Arch Appl Mech. 1999;69:635–54. https://doi.org/10.1007/s004190050248.
[6] Allaire G, Jouve F, Toader A-M. Structural optimization using sensitivity analysis and a level-set method. J Comput Phys. 2004;194:363–93. https://doi.org/10.1016/j.jcp.2003.09.032.
[7] Sethian JA, Wiegmann A. Structural boundary design via level set and immersed interface methods. J Comput Phys. 2000;163:489–528. https://doi.org/10.1006/jcph.2000.6581.
[8] Myśliński A. Piecewise constant level set method for topology optimization of unilateral contact problems. Adv Eng Soft 2015;80:25–32. https://doi.org/10.1016/j.advengsoft.2014.09.020.
[9] Huang X, Xie YM. A further review of ESO type methods for topology optimization. Struct Multidiscip Optim. 2010;41:671–83. https://doi.org/10.1007/s00158-010-0487-9.
[10] Xie YM, Steven GP. A simple evolutionary procedure for structural optimization. Comput Struct. 1993;49:885–96. https://doi.org/10.1016/0045-7949(93)90035-C.
[11] Guo X, Zhang W, Zhang J, Yuan J. Explicit structural topology optimization based on moving morphable components (MMC) with curved skeletons. Comput Methods Appl Mech Eng. 2016;310:711–48. https://doi.org/10.1016/j.cma.2016.07.018.
[12] Zhang S, Da D, Wang Y. TPMS-infill MMC-based topology optimization considering overlapped component property. Int J Mech Sci. 2022;235:107713. https://doi.org/10.1016/j.ijmecsci.2022.107713.
[13] Zheng S, Fan H, Zhang Z, Tian Z, Jia K. Accurate and real-time structural topology prediction driven by deep learning under moving morphable component-based framework. Appl Math Modell. 2021;97:522–35. https://doi.org/10.1016/j.apm.2021.04.009.
[14] Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T. Isogeometric analysis: an overview and computer implementation aspects. Math Comput Simul 2015;117:89–116. https://doi.org/10.1016/j.matcom.2015.05.008.
[15] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Eng. 2005;194:4135–95. https://doi.org/10.1016/j.cma.2004.10.008.
[16] Seo Y-D, Kim H-J, Youn S-K. Isogeometric topology optimization using trimmed spline surfaces. Comput Methods Appl Mech Eng. 2010;199:3270–96. https://doi.org/10.1016/j.cma.2010.06.033.
[17] Hassani B, Khanzadi M, Tavakkoli SM. An isogeometrical approach to structural topology optimization by optimality criteria. Struct Multidiscip Optim. 2011;45:223–33. https://doi.org/10.1007/s00158-011-0680-5.
[18] An Z, Yu T, Bui TQ, Wang C, Trinh NA. Implementation of isogeometric boundary element method for 2-D steady heat transfer analysis. Adv Eng Softw 2018;116:36–49. https://doi.org/10.1016/j.advengsoft.2017.11.008.
[19] Nishi S, Yamada T, Izui K, Nishiwaki S, Terada K. Isogeometric topology optimization of anisotropic metamaterials for controlling high-frequency electromagnetic wave. Int J Numer Methods Eng. 2019;121:1218–47. https://doi.org/10.1002/nme.6263.
[20] Chen L, Lu C, Lian H, Liu Z, Zhao W, Li S, Chen H, Bordas SPA. Acoustic topology optimization of sound absorbing materials directly from subdivision surfaces with isogeometric boundary element methods. Comput Methods Appl Mech Eng. 2020;362:112806. https://doi.org/10.1016/j.cma.2019.112806.
[21] Wang Y, Benson DJ. Isogeometric analysis for parameterized LSM-based structural topology optimization. Comput Mech. 2016;57:19–35. https://doi.org/10.1007/s00466-015-1219-1.
[22] Gao JY, Gao L, Luo Z, Li P. Isogeometric topology optimization for continuum structures using density distribution function. Int J Numer Methods Eng. 2019;119:991–1017. https://doi.org/10.1002/nme.6081.
[23] Mukherjee S, Lu D, Raghavan B, Breitkopf P, Dutta S, Xiao M, Zhang W. Accelerating large-scale topology optimization: state-of-the-art and challenges. Arch Comput Methods Eng. 2021;28:4549–71. https://doi.org/10.1007/s11831-021-09544-3.
[24] Träff EA, Rydahl A, Karlsson S, Sigmund O, Aage N. Simple and efficient GPU accelerated topology optimisation: codes and applications. Comput Methods Appl Mech Eng. 2023;410:116043. https://doi.org/10.1016/j.cma.2023.116043.
[25] Wang Y, Zheng W, Zheng Y, Da D. A new three-level mesh method to accelerate the structural topology optimization. Appl Math Modell. 2022;109:374–400. https://doi.org/10.1016/j.apm.2022.05.012.
[26] Yin J, Wang H, Li S, Guo D. An efficient topology optimization method based on adaptive reanalysis with projection reduction. Eng Comput. 2024;40:213–34. https://doi.org/10.1007/s00366-023-01783-1.
[27] Wadbro E, Berggren M. Megapixel topology optimization on a graphics processing unit. SIAM Rev 2009;51:707–21. https://doi.org/10.1137/070699822.
[28] Xia Z, Wang Y, Wang Q, Mei C. GPU parallel strategy for parameterized LSM-based topology optimization using isogeometric analysis. Struct Multidiscip Optim. 2017;56:413–34. https://doi.org/10.1007/s00158-017-1672-x.
[29] Amir O, Sigmund O, Lazarov BS, Schevenels M. Efficient reanalysis techniques for robust topology optimization. Comput Methods Appl Mech Eng. 2012;245-246:217–31. https://doi.org/10.1016/j.cma.2012.07.008.
[30] Kai L, Gu C, Wang X, Liu J, Du Y, Zhuo C, Saeed N. A novel minimum weight formulation of topology optimization implemented with reanalysis approach. Int J Numer Methods Eng. 2019;120:567–79. https://doi.org/10.1002/nme.6148.
[31] Yin J, Li S, Guo D, Wang H. A multigrid assisted reanalysis method for accelerated heat transfer topology optimization. Appl Math Modell. 2024;125:402–23. https://doi.org/10.1016/j.apm.2023.08.048.
[32] Amir O, Aage N, Lazarov B. On multigrid-CG for efficient topology optimization. Struct Multidiscip Optim. 2014;48:815–29. https://doi.org/10.1007/s00158-013-1015-5.
[33] Wang Y, Liao Z, Ye M, Zhang Y, Li W, Xia Z. An efficient isogeometric topology optimization using multilevel mesh, MGCG and local-update strategy. Adv Eng Softw 2020;139:102733. https://doi.org/10.1016/j.advengsoft.2019.102733.
[34] Notay Y, Vassilevski PS. Recursive Krylov-based multigrid cycles. Numer Lin Alg Appl. 2008;15:473–87. https://doi.org/10.1002/nla.542.

[35] Briggs W, Henson V, McCormick S. A multigrid tutorial. USA: Society for Industrial and Applied Mathematics; 2000. second ed.
[36] Huang W-Z. Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance. Appl Math Comput. 1991;46:145–64. https://doi.org/10.1016/0096-3003(91)90022-F.
[37] Liu H-T, Zhang J, Ben-Chen M, Jacobson A. Surface multigrid via intrinsic prolongation. ACM Trans Graph 2021;40:1–13. https://doi.org/10.1145/3450626.3459768.
[38] Xie X, Wang S, Xu M, Wang Y. A new isogeometric topology optimization using moving morphable components based on R-functions and collocation schemes. Comput Methods Appl Mech Eng. 2018;339:61–90. https://doi.org/10.1016/j.cma.2018.04.048.
[39] Piegl L, Tiller W. The nurbs book. Berlin, Heidelberg: Springer-Verlag; 1995.
[40] Braess D. On the combination of the multigrid method and conjugate gradients. In: Hackbusch W, Trottenberg U, editors. Multigrid methods ii. Berlin, Heidelberg: Springer Berlin Heidelberg; 1986. p. 52–64.
[41] Wang B, You H, Ma X, Shi Y, Hao P, Zhang J. Multigrid reduced-order topology optimization scheme for structures subjected to stationary random excitations. Struct Multidiscip Optim. 2023;66:102. https://doi.org/10.1007/s00158-023-03541-9.
[42] Adams M, Brezina M, Hu J, Tuminaro R. Parallel multigrid smoothing: polynomial versus Gauss–Seidel. J Comput Phys. 2003;188:593–610. https://doi.org/10.1016/S0021-9991(03)00194-3.
[43] Fehn N, Munch P, Wall WA, Kronbichler M. Hybrid multigrid methods for high-order discontinuous Galerkin discretizations. J Comput Phys. 2020;415:109538. https://doi.org/10.1016/j.jcp.2020.109538.
[44] McAdams A, Sifakis E, Teran J. A parallel multigrid poisson solver for fluids simulation on large grids. Madrid: Eurographics Association; 2010.
[45] Nocedal J, Wright SJ. Numerical optimization. New York: Springer; 2006. second ed.
[46] Andrilli S, Hecker D. Numerical Techniques. In: Andrilli S, Hecker D, editors. Elementary linear algebra. 5th Edition. Boston: Academic Press; 2016. p. 607–66.
[47] Parlett B, Simon H, Stringer L. On Estimating the largest eigenvalue with the Lanczos algorithm. Mathem Comput - Math Comput. 1982;38:153. -153.
[48] Wang Y, Xiao M, Xia Z, Li P, Gao L. From Computer-Aided Design (CAD) toward Human-Aided Design (HAD): an isogeometric topology optimization approach. Engineering 2023;22:94–105. https://doi.org/10.1016/j.eng.2022.07.013.
[49] Gao J, Xiao M, Zhang Y, Gao L. A comprehensive review of isogeometric topology optimization: methods, applications and prospects. Chin J Mech Eng. 2020;33:87. https://doi.org/10.1186/s10033-020-00503-w.
[50] M. Phillips, P.F. Fischer, Optimal chebyshev smoothers and one-sided V-cycles, ArXiv abs/2210.03179 (2022).
[51] Baker AH, Falgout RD, Kolev TV, Yang UM. Multigrid smoothers for ultraparallel computing. SIAM J Sci Comput 2011;33:2864–87. https://doi.org/10.1137/100798806.