Approx. Theory Spec. Funct. 2 (1) (2026) 72–94
https://doi.org/10.65135/atsf.2026.13
https://atsf.com.tr

APPROXIMATION THEORY and SPECIAL FUNCTIONS
[ISSN: 3108-5083 (Online)]

Research Article

Adversarial Neural Network Methods for Topology Optimization of Eigenvalue Problems

Xindi Hu, Jiaming Weng and Shengfeng Zhu (*)

School of Mathematical Sciences, East China Normal University, Shanghai 200241, China

Abstract: This paper presents a new method based on neural networks to optimize the first eigenvalues of second-order elliptic and fourth-order biharmonic operators subject to geometric constraints. These models can have applications in material design for drums and plates. Our approach tackles this problem by developing two neural networks for the eigenfunction and the density function, respectively. A strategic construction infuses empirical experience into the density function, enhancing the algorithm's robustness and efficiency. Automatic differentiation enables the algorithm to directly obtain the optimization direction instead of solving the eigenvalue problem as traditional optimization methods did. The algorithm, which is executable on a GPU, ensures efficiency aligned with GPU developments. Numerical examples substantiate the effectiveness of the algorithm.

Keywords: Neural network, deep learning, topology optimization, eigenvalue problem.

AMS Math. Subject Class. (2020): 49Q10, 65N21.

1 Introduction

As an optimization problem subject to constraints of geometry and physics, shape design aims to find a shape that optimizes some objective useful in engineering [1, 2]. Shape design for eigenvalue problems has extensive applications [3], including mechanical vibrations [4–9], acoustics [10], photonic crystals [11], population dynamics [12], etc.

In PDE-constrained optimization, traditional methods based on sensitivity analysis need to use a mesh-based numerical discretization method, such as finite element, in either the optimize-and-discretize [2] or the discretize-and-optimize [1] framework. After discretization, large-scale optimization problems need to be solved, in which high computational costs are required, especially in 3D for sufficiently accurate mesh resolutions.

This paper aims to find a shape that maximizes or minimizes the first, i.e., the smallest eigenvalue, using a neural network and without relying on the classical theoretical framework of shape sensitivity analysis. It is well-known that, with the explosion of available data and computational resources, recent advances in machine learning, especially deep learning, have produced breakthroughs in different disciplines, including image recognition [13], cognitive science [14], genomics [15], etc. Researchers

(*)Corresponding author.
Emails: 52205500022@stu.ecnu.edu.cn (X. Hu), 763212966@qq.com (J. Weng) and sfzhu@math.ecnu.edu.cn (S. Zhu)

Received: 24.02.2026; *Accepted*: 06.04.2026; *Published*: 08.04.2026

have begun to explore the application of neural networks in scientific computing and engineering problems. The meshless deep neural network has made many contributions to solving partial differential equations, e.g., the deep Ritz method [16, 17]. They have also been applied to handle high-dimensional problems [18]. Physics-informed neural networks (PINNs) [19] integrating many physical constraints and establishing continuous-time models and discrete-time models were proposed to solve forward problems and inverse problems. Recently, tensor neural networks were developed [20] for solving eigenvalue problems. It is worth noting that these methods for solving equations are not exactly the same as the ideas behind deep learning. Although they all use neural networks, they are essentially only solving an optimization problem, not a generalization problem in deep learning.

In addition, to better handle boundary conditions of physical constraints, a form of automatically satisfying the boundary condition was proposed [21] by multiplying a signed distance function with the output of the neural network. For the variational problems in which it is difficult to find the boundary signed distance function on irregular domains, an augmented Lagrangian deep learning method was proposed [22].

Inspired by the use of PINNs for solving inverse problems, Lu et al. [23] introduced hybrid Physics-Informed Neural Networks (hPINNs), which represent a network architecture that automatically satisfies boundary conditions and circumvents the difficulties associated with the penalty methods on boundaries. Furthermore, the density function, under a reasonable assumption, was constructed and demonstrated the efficiency of hPINNs for shape optimization in optical holography and Stokes flows. See also the deep learning toolbox for solving partial differential equations [24].

In the field of topology optimization, the use of deep learning methods has received considerable attention in recent years. Deng et al. [25] introduced a generative design approach that integrates topology optimization and generative models iteratively to explore new design schemes. Oh et al. [26] proposed a deep neural network-based level set method for topology optimization parameters, integrating deep neural networks into the traditional level set method to construct an effective structural topology optimization approach. Lei et al. [27] employed machine learning to directly establish a mapping between optimal structural/topological design parameters and external loads. Furthermore, Huang et al. [28] introduced a Problem-Independent Machine Learning (PIML) technique to reduce the computational time associated with finite element analysis. These studies represent the application of deep learning methods to topology optimization problems from various angles, highlighting the vast potential of deep learning in the realm of topology optimization.

Recently, Zang et al. [29] proposed a weak adversarial network (WAN) method to solve partial differential equations by transforming the variational form into an operator norm minimization problem. In the training stage, the weak solution and test function in the weak form are parameterized into the original network and the adversarial network, respectively. The parameters of the two neural networks are alternately updated. Moreover, the WAN method was used to solve the inverse problems [30]. For a typical shape design problem, the density function as a variable to be optimized is related to the physical state.

The present paper proposes a neural network method for eigenvalue optimization problems. Different from the traditional mesh-based discretization method, there is no need in the present meshless method to repeatedly solve the eigenvalue problem for sensitivity analysis, such as explicit gradient computation. It directly uses automatic differentiation in machine learning [31]. In addition, adversarial neural networks enable different neural networks to be trained independently, which improves training efficiency and achieves acceptable optimized results.

The rest of the paper is organized as follows. In Section 2, we introduce topology optimization of second-order elliptic and fourth-order biharmonic eigenvalues. Section 3 introduces the structure of the neural network and the training mode, how to handle constraints, and presents the loss function of the neural network. Furthermore, the adversarial neural network algorithm is summarized for eigenvalue

optimization problems. In Section 4, numerical examples are presented to validate the effectiveness of the algorithm. Conclusions follow in Section 5.

2 Model Problems

Let $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ be an open bounded domain with a Lipschitz boundary $\partial\Omega$. Let $\rho : \Omega \to \mathbb{R}$ be a discontinuous function:

$$\rho = \begin{cases} \rho_1 & \text{in } \Omega\backslash D, \\ \rho_2 & \text{in } D, \end{cases} \tag{2.1}$$

where $D \subset \Omega$ is an unknown domain and ρ_1 and ρ_2 $(\rho_1 \neq \rho_2)$ are prescribed positive constants. Introduce the characteristic function of D as

$$\chi_D = \begin{cases} 1 & \text{in } D, \\ 0 & \text{in } \Omega\backslash D. \end{cases} \tag{2.2}$$

Let a constant $c \in (\rho_1|\Omega|, \rho_2|\Omega|)$ be prescribed $(\rho_1 < \rho_2)$, where $|\Omega|$ denotes the Lebesgue measure of Ω. Define a set of admissible functions

$$\mathcal{A} = \left\{ \rho \middle| \rho = \rho_1\chi_{\Omega\backslash D} + \rho_2\chi_D, \int_\Omega \rho \mathrm{d}x = c \right\}. \tag{2.3}$$

Notice that a mass constraint, or equivalently a geometry constraint, on D has been imposed in (2.3). Let $L^2(\Omega)$ be the space of square-integrable functions defined on Ω. Introduce the Hilbert spaces

$$\begin{aligned} H^1(\Omega) &:= \left\{ v \in L^2(\Omega) \middle| \frac{\partial v}{\partial x_i} \in L^2(\Omega), i = 1, \cdots, d \right\}, \\ H_0^1(\Omega) &:= \left\{ v \in H^1(\Omega) \mid v = 0 \text{ on } \partial\Omega \right\}. \end{aligned}$$

2.1 Second-order elliptic eigenvalue optimization

A natural application in vibrating structure engineering, e.g., acoustics [10, 32–34], is to design the material distribution subject to a mass constraint so that the resonant frequency of a drum reaches a maximum or minimum. See similar applications for the eigenvalue optimization of the composite coefficient problem [4].

Let us introduce the model problem. Given a domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$, let λ_1 be the first, i.e., the smallest eigenvalue of the problem

$$\begin{cases} -\Delta u + \alpha\chi_D u = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{2.4}$$

where $\alpha > 0$ is a prescribed constant. Consider the following eigenvalue optimization problem:

$$\min_{\rho\in\mathcal{A}} \lambda_1 \quad \text{or} \quad \max_{\rho\in\mathcal{A}} \lambda_1, \tag{2.5}$$

where $c \in (0, |\Omega|)$ is prescribed, $\rho_1 = 0, \rho_2 = 1$, and thus $\rho = \chi_D$.

Next, we consider the weak formulation of (2.4): Find $(\lambda, u) \in \mathbb{R}^+ \times H_0^1(\Omega)$ such that

$$\int_\Omega (\nabla u \cdot \nabla v + \alpha\chi_D uv)\mathrm{d}x = \lambda \int_\Omega uv\mathrm{d}x \quad \forall v \in H_0^1(\Omega).$$

By the Rayleigh quotient, we obtain

$$\lambda_1 = \min_{0\neq v\in H_0^1(\Omega)} \frac{\int_\Omega (|\nabla v|^2 + \alpha\chi_D v^2)\mathrm{d}x}{\int_\Omega v^2\mathrm{d}x}. \tag{2.6}$$

Thus, the minimization and maximization of the first eigenvalue in (2.5) can be written as

$$\min_{\rho\in\mathcal{A}} \min_{0\neq v\in H_0^1(\Omega)} \frac{\int_\Omega (|\nabla v|^2 + \alpha\rho v^2)\mathrm{d}x}{\int_\Omega v^2\mathrm{d}x} \tag{2.7}$$

and

$$\max_{\rho\in\mathcal{A}} \min_{0\neq v\in H_0^1(\Omega)} \frac{\int_\Omega (|\nabla v|^2 + \alpha\rho v^2)\mathrm{d}x}{\int_\Omega v^2\mathrm{d}x}, \tag{2.8}$$

respectively.

2.2 Fourth-order biharmonic eigenvalue optimization

Consider fourth-order biharmonic eigenvalue optimization with applications in the control of plate frequencies. The difference between the two eigenvalue problems is due to different boundary conditions [5, 9]. We try to find the optimal density distribution in the open bounded domain $\Omega \subset \mathbb{R}^2$. Consider the following two eigenvalue problems, subject to inhomogeneous clamped and inhomogeneous simply supported conditions, respectively:

$$\begin{cases} \Delta^2 u = \lambda\rho u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \\ \frac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega, \end{cases} \tag{2.9}$$

and

$$\begin{cases} \Delta^2 u = \lambda\rho u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega \\ \Delta u - (1-\nu)\kappa\frac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega, \end{cases} \tag{2.10}$$

where ρ is defined in (2.1), n denotes the unit outward normal to the boundary, κ is the mean curvature, and $\nu \in (-1 \leq \nu \leq 0.5)$ is the Poisson's ratio. We consider finding the optimal D that allows the first eigenvalue λ_1 of (2.9) or (2.10) to reach a minimum or maximum [9, 35]:

$$\min_{\rho\in\mathcal{A}} \lambda_1 \quad \text{or} \quad \max_{\rho\in\mathcal{A}} \lambda_1. \tag{2.11}$$

Let us introduce the following Hilbert spaces:

$$H^2(\Omega) := \left\{ v \in H^1(\Omega) \middle| \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2} \in H^1(\Omega) \right\},$$
$$H_0^2(\Omega) := \left\{ v \in H^2(\Omega) \middle| v = 0, \frac{\partial v}{\partial n} = 0 \text{ on } \partial\Omega \right\}.$$

By the Rayleigh theorem, we obtain

$$\lambda_1 = \min_{0\neq v\in H_0^2(\Omega)} \frac{\int_\Omega (\Delta v)^2\mathrm{d}x}{\int_\Omega \rho v^2\mathrm{d}x} \tag{2.12}$$

and

$$\lambda_1 = \min_{0\neq v\in H_0^1(\Omega)\cap H^2(\Omega)} \frac{\int_\Omega (\Delta v)^2 \mathrm{d}x - \int_{\partial\Omega}(1-\nu)\kappa\left(\frac{\partial v}{\partial n}\right)^2 \mathrm{d}s}{\int_\Omega \rho v^2 \mathrm{d}x} \tag{2.13}$$

for (2.9) and (2.10), respectively. Then, the problem (2.11) can be written as

$$\left(\max_{\rho\in\mathcal{A}}\right) \min_{\rho\in\mathcal{A}} \min_{0\neq v\in H_0^2(\Omega)} \frac{\int_\Omega (\Delta v)^2 \mathrm{d}x}{\int_\Omega \rho v^2 \mathrm{d}x} \tag{2.14}$$

and

$$\left(\max_{\rho\in\mathcal{A}}\right) \min_{\rho\in\mathcal{A}} \min_{0\neq v\in H_0^1(\Omega)\cap H^2(\Omega)} \frac{\int_\Omega (\Delta v)^2 \mathrm{d}x - \int_{\partial\Omega}(1-\nu)\kappa\left(\frac{\partial v}{\partial n}\right)^2 \mathrm{d}s}{\int_\Omega \rho v^2 \mathrm{d}x}, \tag{2.15}$$

associated with (2.9) and (2.10), respectively.

3 Neural Networks

Inspired by biology and neuroscience, artificial neural networks are mathematical models with networks between input arrays and output results. An artificial neural network imitates the neuronal network of the human brain to construct artificial neurons and establish the topological connections between them. Such a network can be regarded as a mathematical mapping. We take the feedforward neural network as an example to introduce the components of the neural network and their mathematical formulations [36].

3.1 Network structure

3.1.1 Layer structure of fully connected feedforward neural networks

The layer structure is the basic component of the network structure. Each layer is composed of multiple neurons. In a fully connected feedforward neural network, all neurons in a layer are connected to neurons in the previous layer. Now we describe the transmission between two adjacent layers in a fully connected feed-forward neural network. Let the l-th and the $(l+1)$-th layers of the neural network contain d_l and d_{l+1} neurons, respectively. The output of the l-th layer denoted by z_l is also the input of the $(l+1)$-th layer. Let $\boldsymbol{W}^{(l+1)}$ be the $(l+1)$-th layer's weight matrix consisting of weight vectors for all neurons of that layer. Denote by $\boldsymbol{b}^{(l+1)}$ and ϕ_{l+1} the bias vector and the activation function of the $(l+1)$-th layer, respectively. The mathematical relationship of the output vectors between the two consecutive layers reads:

$$\boldsymbol{z}^{(l+1)} = \phi_{l+1}\left(\boldsymbol{W}^{(l+1)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l+1)}\right), \tag{3.1}$$

where $z^{(l)} \in \mathbb{R}^{d_l}$, $z^{(l+1)} \in \mathbb{R}^{d_{l+1}}$, $\boldsymbol{W}^{(l+1)} \in \mathbb{R}^{d_l \times d_{l+1}}$, and $\boldsymbol{b}^{(l+1)} \in \mathbb{R}^{d_{l+1}}$.

The relationship of the output vectors of the two adjacent layers can be summarized as

$$\boldsymbol{z}^{(l+1)} = f_{d_l,d_{l+1}}^{(l+1)}\left(\boldsymbol{z}^{(l)}\right),$$

where $f_{d_l,d_{l+1}}^{(l+1)} : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$ denotes the layer mapping from the l-th layer with d_l neurons to the $(l+1)$-th layer with d_{l+1} neurons.

3.1.2 Block structure

Unlike simple hierarchical stacking, ResNet [37] is a convolutional neural network characterized by residual blocks and skip connections. For a simple layered stacked feed-forward neural network, the training gradient will disappear when the number of layers increases to a certain extent. Fortunately, the jump structure solves this problem well by inserting an identity mapping every several layers, such that the gradient is not close to zero. ResNet-type networks can be regarded as a block structure if we combine the identity map with several layers. Then the whole network can be regarded as multiple learning block-wise connections.

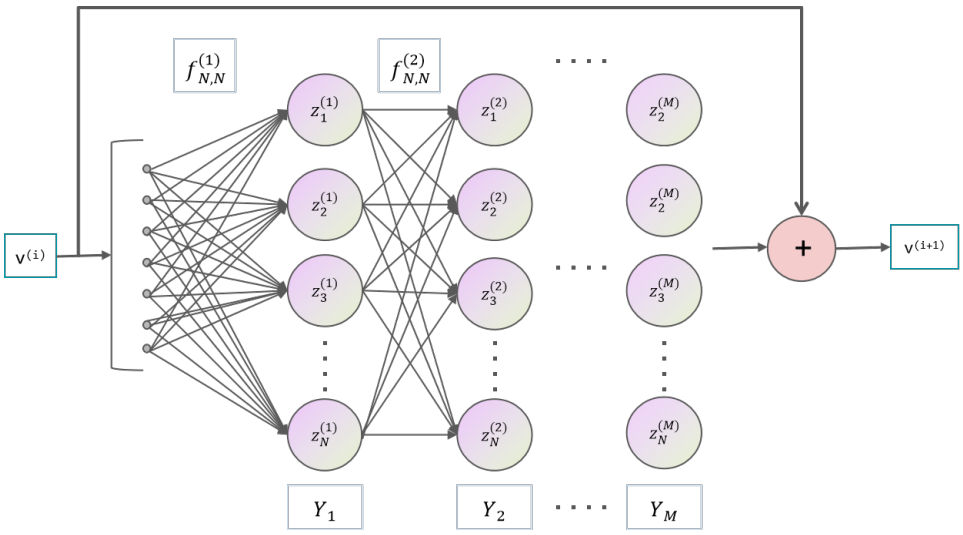


Figure 1: Block structure

Let us take the jumping hierarchy used in this paper as an example to illustrate the block structure (see Fig. 1). This is a block composed of M feed-forward fully-connected layers: $Y_1, Y_2, \cdots, Y_M$ and an identity map. Since the operational dimension of the identity mapping needs to be consistent with the output dimension of the M-th layer, we simply take the number of neurons in all fully connected layers as equal, which means the number of neurons d_l in a layer is degenerated to N. If we define $\boldsymbol{v}^{(i)}$ and $\boldsymbol{v}^{(i+1)}$ as the output of the i-th and $(i+1)$-th blocks, then we can express the relationship between $\boldsymbol{v}^{(i)}$ and $\boldsymbol{v}^{(i+1)}$:

$$\boldsymbol{v}^{(i+1)} = f_{N,N}^{(M)} \circ \ldots \circ f_{N,N}^{(2)} \circ f_{N,N}^{(1)}(\boldsymbol{v}^{(i)}) + \boldsymbol{v}^{(i)}, \tag{3.2}$$

where $f_{N,N}^{(m)}$ $(m = 1, 2, \cdots, M)$ is the layer mapping consisting of N neurons. From it, we can find that the relationship of the output of two adjacent blocks can be denoted as

$$\boldsymbol{v}^{(i+1)} = g_{M,N}^{(i)}(\boldsymbol{v}^{(i)}).$$

We define $g_{M,N}^{(i)}$ as the i-th block mapping, which consists of M layers with N neurons in every layer.

3.1.3 Complete structure

A complete network structure contains input layers, output layers, and hidden layers or hidden blocks (Fig. 2). For an input vector $\boldsymbol{x} \in \mathbb{R}^d$, it will first transform through the input layer so that it is consistent with the input dimension of the first hidden layer, then into the hidden block, and finally

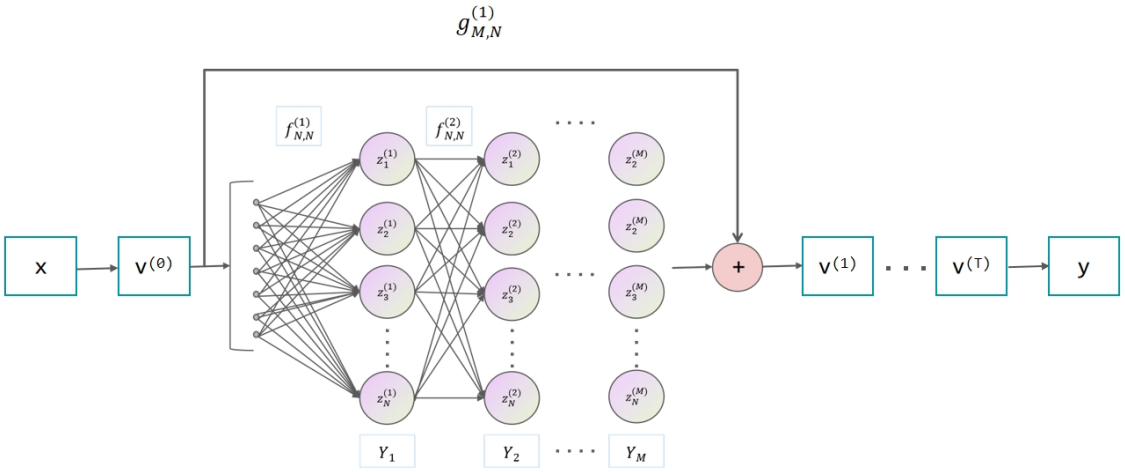


Figure 2: Network structure.

into the output layer, obtaining the output y. In this paper, the network is composed of T hidden blocks. Each block contains the M layers, and each layer contains N neurons. Then the transmission process of the whole network is as follows:

$$\begin{aligned} \boldsymbol{v}^{(0)} &= \boldsymbol{W}_{in} \cdot \boldsymbol{x}, \\ \boldsymbol{v}^{(T)} &= g^{(T)}_{M,N} \circ \ldots \circ g^{(1)}_{M,N}(\boldsymbol{v}^{(0)}), \\ y &= \phi\big(\boldsymbol{W}_{out} \cdot \boldsymbol{v}^{(T)} + b\big), \end{aligned} \tag{3.3}$$

where $\boldsymbol{W}_{in} \in \mathbb{R}^{d\times N}$ and $\boldsymbol{W}_{out} \in \mathbb{R}^{N\times 1}$ are weight matrices of the input layer and output layer, respectively, and T and N are the numbers of block structures and neurons in the layer structure, respectively.

Note that the activation function in the output layer can differ from that in the hidden block. Many functions with nonlinear properties can be used as activation functions, which are the key to the neural network's ability to deal with nonlinear problems. Considering the efficiency and stability of neural network training, the activation function generally needs to be continuously differentiable, and its derivative cannot be too large or too small. In this paper, we mainly use the **tanh function** and the **square function** when specified.

3.1.4 Neural network training

The previous subsection shows the process of forward transmission of the network. The neural network can be thought of as a mapping between an input $\boldsymbol{x}$ and an output y. After building the network, initial parameters are required to be set. The Xavier method [38] is used here to initialize the network. For a specific problem considered to be solved, a loss function $\mathcal{L}(H_\theta)$ with the neural network H_θ needs to be constructed. Therefore, the training of the neural network is equivalent to the following minimization problem:

$$\min_{\theta} \mathcal{L}(H_\theta),$$

where θ is the set of layer weight matrices and the biases. We try to find the optimal θ through training the network by backpropagation, which uses the automatic differential method to obtain the derivative [31]. In the backpropagation optimization method, in addition to the stochastic gradient

descent method, there are many optimization algorithms, such as the adaptive method, momentum method, Newton method, conjugate gradient method, etc. Here, we use the Adam method [39]. It is worth mentioning that using the Adam method can, to some extent, reduce the difficulty of parameter tuning, but it does not mean that we can obtain the global optimum through it. It is challenging to ensure global optimality with standard neural network methods, and often, solutions might not even be locally optimal. As supported by reference [40], the solutions may only represent points in the solution space where the gradient is relatively small. However, the characteristic of neural networks is that even so, the obtained solution may still be satisfactory. This is because neural networks search for solutions in a high-dimensional space, where many points can be very close to the global optimal solution.

3.2 Loss functions

3.2.1 Objectives

Generally, the loss function needs to consider internal state, boundary, and geometry constraints. In this case, the objective function (2.6) is equivalent to the first eigenvalue pair of equation (2.4). Therefore, the loss function only needs to consider the objective (2.6), boundary constraints, and geometry constraints.

Firstly, we discretize and approximate the objective based on the equivalence of the objective and (2.4). Instead of using a traditional numerical method such as the finite difference or finite element method, we use the deep Ritz method [17] to build neural networks that we mentioned in Section 2.

We build two adversarial neural networks $H\left(\boldsymbol{x};\boldsymbol{\theta}_u\right)$ and $H\left(\boldsymbol{x};\boldsymbol{\theta}_\rho\right)$ to approximate u and ρ, respectively, where $\boldsymbol{x}$ as points in Ω are inputs of the neural networks, $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_\rho$ are network parameters that will be updated after initialization. Let $\hat{u}$ and $\hat{\rho}$ be continuously differentiable approximations to u and ρ, respectively. Consider $\hat{u}$ and $\hat{\rho}$ as the outputs of the two neural networks:

$$\begin{aligned}\hat{u}\left(\boldsymbol{x};\boldsymbol{\theta}_u\right) &= H\left(\boldsymbol{x};\boldsymbol{\theta}_u\right),\\ \hat{\rho}\left(\boldsymbol{x};\boldsymbol{\theta}_\rho\right) &= H\left(\boldsymbol{x};\boldsymbol{\theta}_\rho\right).\end{aligned}$$

The integrals in the objective (2.6) can be approximated by numerical quadrature with points in Ω. Let $\left\{\boldsymbol{x}^{(j)}\right\}_{j=1}^{N_x}$ be the set of N_x points in Ω. Then the objective of (2.6) can be approximated as

$$\mathcal{L}_{in}\left(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\rho\right)=\frac{\sum_{j=1}^{N_x}\left[\left|\nabla\hat{u}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)\right|^2+\alpha\hat{\rho}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{\rho}}\right)\hat{u}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)\right]}{\sum_{j=1}^{N_x}\hat{u}^2\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)}.$$

Now we consider the discretization of the objective for biharmonic eigenvalue optimization. For the objective (2.12) of the clamped plate equation, we approximate it as

$$\mathcal{L}_{in}\left(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\rho\right)=\frac{\sum_{j=1}^{N_x}\left|\Delta\hat{u}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_u\right)\right|^2}{\sum_{j=1}^{N_x}\hat{\rho}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_\rho\right)\hat{u}^2\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_u\right)}. \tag{3.4}$$

As for the simply supported plate equation, note that its objective (2.13) includes both the volume integral and boundary integral. Introduce N_b points $\left\{\boldsymbol{x}_{\boldsymbol{b}}^{(j)}\right\}_{j=1}^{N_b}$ on the boundary such that the objective (2.13) can be approximated as

$$\mathcal{L}_{in}\left(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\rho\right)=\frac{\frac{A_1}{N_x}\sum_{j=1}^{N_x}\left|\Delta\hat{u}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)\right|^2-\frac{A_2}{N_b}\sum_{j=1}^{N_b}(1-\nu)\kappa\frac{\partial\hat{u}}{\partial n}\left(\boldsymbol{x}_{\boldsymbol{b}}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)^2}{\frac{A_1}{N_x}\sum_{j=1}^{N_x}\hat{\rho}\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{\rho}}\right)\hat{u}^2\left(\boldsymbol{x}^{(j)};\boldsymbol{\theta}_{\boldsymbol{u}}\right)}, \tag{3.5}$$

in which A_1 and A_2 represent the volume of Ω and the surface measure of $\partial\Omega$, respectively.

3.2.2 Boundary constraint

This section focuses on how to deal with boundary constraints. For neural network methods in solving partial differential equations (e.g., [17, 19]), a common method for dealing with boundary constraints is to add a penalty term to the objective, which is feasible for many types of boundary conditions: Dirichlet, Neumann, Robin, etc. However, the boundary penalty term probably causes difficulties in training neural networks. To satisfy boundary conditions and train the networks easily, we refer to [23], in which for the Dirichlet boundary condition, we can introduce a boundary function ℓ_u such that $\hat{u}$ directly satisfies the boundary condition automatically. More specifically, set

$$\hat{u}\left(\boldsymbol{x};\boldsymbol{\theta}_u\right) = g(\boldsymbol{x}) + \ell_u(\boldsymbol{x}) H\left(\boldsymbol{x};\boldsymbol{\theta}_u\right) \tag{3.6}$$

to make $\hat{u}$ directly satisfy the Dirichlet boundary condition

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega. \tag{3.7}$$

In our problems, $g(\boldsymbol{x}) = 0$ and ℓ_u satisfy

$$\begin{cases} \ell_u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \\ \ell_u(\boldsymbol{x}) > 0, & \boldsymbol{x} \in \Omega. \end{cases} \tag{3.8}$$

Specifically, if the $\partial\Omega$ has a simple geometric structure, an boundary function can be chosen. For example, we can choose $\ell_u = (x-a_1)(x-a_2)(y-a_3)(y-a_4)$ for $\Omega = [a_1, a_2] \times [a_3, a_4]$ with a_1, a_2, a_3, a_4 being constants given ($a_1 < a_2$ and $a_3 < a_4$).

For more complex domains, one can choose a signed distance function [41] denoted by $f : X \to \mathbb{R}$ for a larger domain $X \supset \Omega$:

$$f(\boldsymbol{x}) = \begin{cases} d(\boldsymbol{x}, \partial\Omega) & \text{if } \boldsymbol{x} \in \Omega, \\ -d(\boldsymbol{x}, \partial\Omega) & \text{if } \boldsymbol{x} \in \Omega^c, \end{cases} \tag{3.9}$$

where $\Omega^c = X \setminus \Omega$ and the distance function from $\boldsymbol{x}$ to $\partial\Omega$ is defined by

$$d(\boldsymbol{x}, \partial\Omega) := \inf_{\boldsymbol{y} \in \partial\Omega} d(\boldsymbol{x}, \boldsymbol{y}).$$

Obviously, the signed distance function satisfies (3.8) for ℓ_u. For example, if $\Omega = \left\{(x, y) : r_{\text{in}}^2 \le x^2 + y^2 \le r_{\text{out}}^2\right\}$, we can choose $\ell_u = \min(r_{\text{out}}^2 - x^2 - y^2, x^2 + y^2 - r_{\text{in}}^2)$, where r_{in} and r_{out} are the inner and outer radii, respectively. Notice that the choice of a boundary function ℓ_u is not unique.

It is worth mentioning that in (3.6), $\hat{u}$ is no longer the output of the neural network directly, but a continuously differentiable function (with the neural network as a kernel) that automatically satisfies the boundary condition (3.7).

Next, we deal with the boundary conditions of the fourth-order plate eigenvalue problems. Taking the clamped plate support plate equation as an example, we can find, similarly to above, a boundary condition function ℓ_u such that $\hat{u}$ directly meets both the Dirichlet and Neumann boundary conditions. As for $H_0^2(\Omega)$ in this case, we can construct

$$\hat{u}\left(\boldsymbol{x};\boldsymbol{\theta}_u\right) = \ell_u(\boldsymbol{x}) H\left(\boldsymbol{x};\boldsymbol{\theta}_u\right), \tag{3.10}$$

where ℓ_u satisfies

$$\begin{cases} \ell_u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \\ \frac{\partial}{\partial n}\ell_u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \\ \ell_u(\boldsymbol{x}) > 0, & \boldsymbol{x} \in \Omega, \end{cases}$$

with $\frac{\partial}{\partial n}\ell_u = \boldsymbol{n}\cdot\nabla\ell_u$. Let us call the above boundary function approach the **Full-satisfied boundary method**.

We may alternatively use another way to handle the boundary constraints. Taking the simply supported plate problem as an example, since the boundary conditions of the problem are relatively complex, we construct $\hat{u}$ to satisfy the Dirichlet boundary condition, while the other boundary condition, considered a constraint, is dealt with using a penalty method by adding a boundary loss

$$\mathcal{L}_b\left(\boldsymbol{\theta}_u\right) = M\frac{1}{N_b}\sum_{j=1}^{N_b}\left[\Delta\hat{u}^2\left(\boldsymbol{x}_{\boldsymbol{b}}^{(\boldsymbol{j})};\boldsymbol{\theta}_{\boldsymbol{u}}\right)^2 - (1-\nu)\kappa\frac{\partial\hat{u}}{\partial n}\left(\boldsymbol{x}_{\boldsymbol{b}}^{(\boldsymbol{j})};\boldsymbol{\theta}_{\boldsymbol{u}}\right)\right]^2, \tag{3.11}$$

to the total loss function $\mathcal{L}_{sum}$, where $M>0$ is a large penalty parameter. We call it **Half-satisfied boundary method**.

3.2.3 Geometry constraint

Finally, we deal with the geometry constraint using the penalty method. The original constrained optimization problem is transformed into an unconstrained optimization problem. We first consider numerical quadrature to approximate the geometry constraint. If there are enough, e.g., N_x, sampling points in domain Ω, $\int_\Omega \rho$ can be approximated as quadrature. Then, the geometry constraint $\int_\Omega \rho = c$ can be approximately replaced by $h\left(\boldsymbol{\theta}_\rho\right)=0$, where

$$h\left(\boldsymbol{\theta}_\rho\right) = \frac{\sum_{j=1}^{N_x}\hat{\rho}\left(\boldsymbol{x}_{\boldsymbol{j}};\boldsymbol{\theta}_\rho\right)}{N_x} - c.$$

If the penalty method is used, the loss function related to the geometry constraint is as follows:

$$\mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) = \mu h^2\left(\boldsymbol{\theta}_\rho\right),$$

where a penalty coefficient $\mu>0$ is introduced. In the general penalty method, μ is a fixed value. To satisfy the constraint better, the fixed value of μ usually should not be very large. However, a large μ leads to inefficient training due to suffering from a very small training rate, thus making it difficult to reach the global minimizer. One improvement is to allow μ to start from a small initial value and increase during iterations [42]. The loss function of this variable coefficient penalty method is

$$\mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) = \mu_k h^2\left(\boldsymbol{\theta}_\rho\right), \tag{3.12}$$

where $\mu_k>0$ is a penalty coefficient of the k-th iteration and $\mu_{k+1}=\beta\mu_k$ with $\beta>1$.

Another way compared with the variable coefficient penalty method is the augmented Lagrangian method [43] with

$$\mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) = \mu_k h^2\left(\boldsymbol{\theta}_\rho\right) + \lambda_k h\left(\boldsymbol{\theta}_\rho\right), \tag{3.13}$$

where the k-th Lagrange multiplier λ_k is updated during iterations:

$$\lambda_k = \lambda_{k-1} + 2\mu_{k-1}h\left(\boldsymbol{\theta}_\rho^{k-1}\right).$$

In addition, μ_k grows as

$$\mu_{k+1} = \min(\beta\mu_k, S),$$

where S is an upper bound for μ_k. A benefit of the augmented Lagrangian is the dynamic adjustment, which allows μ_k to reach convergence when μ_k does not need to go to infinity, avoiding slow training caused by too large a value of μ_k. In this paper, we address the geometry constraint with both the variable coefficient penalty method (3.12) and the augmented Lagrangian method (3.13).

3.2.4 Total loss function

Now we can write the total loss function $\mathcal{L}_{sum}$. For a minimization problem, we can use the **full-satisfied boundary method** for exact boundary condition satisfaction. Here, the total loss function is

$$\mathcal{L}_{sum}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) = \mathcal{L}_{in}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) + \mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right). \tag{3.14}$$

If the boundary conditions cannot be directly satisfied, we use the **half-satisfied boundary method**. Then the total loss $\mathcal{L}_{sum}$ consists of three parts:

$$\mathcal{L}_{sum}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) = \mathcal{L}_{in}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) + \mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) + \mathcal{L}_b\left(\boldsymbol{\theta}_u\right). \tag{3.15}$$

As for the maximization problem, since different loss functions can be defined in the inner loop, the adversarial neural network method successfully represents the objective of (2.8) in the neural network framework. This is an advantage that the general simultaneous neural network method does not have. For this objective of maximum-minimum formulation, when $\hat{u}$ is fixed in the inner loop, to find $\hat{\rho}$ that maximizes the first eigenvalue and satisfies the geometry constraint, we only need to add a minus sign before $\mathcal{L}_{in}$ in $\mathcal{L}_{sum}$, i.e.,

$$\mathcal{L}_{sum}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) = -\mathcal{L}_{in}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) + \mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) \tag{3.16}$$

for **full-satisfied boundary method** and

$$\mathcal{L}_{sum}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) = -\mathcal{L}_{in}\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right) + \mathcal{L}_w\left(\boldsymbol{\theta}_\rho\right) + \mathcal{L}_b\left(\boldsymbol{\theta}_u\right) \tag{3.17}$$

for **half-satisfied boundary method**.

3.3 Optimization training

In the last section, we have given the unconstrained loss functions under different methods. Now we summarize the complete training process in this section.

The first step is to establish two neural networks $H\left(\boldsymbol{x}; \boldsymbol{\theta}_u\right)$ and $H\left(\boldsymbol{x}; \boldsymbol{\theta}_\rho\right)$, which are the block structures introduced in Section 3.2.1. Next, we need to select appropriate auxiliary functions to construct $\hat{u}\left(\boldsymbol{x}; \boldsymbol{\theta}_u\right)$ and $\hat{\rho}\left(\boldsymbol{x}; \boldsymbol{\theta}_\rho\right)$, both of which contain neural networks as their main components. To address the boundary constraints mentioned above, we choose an appropriate boundary condition function ℓ_u attached to $H\left(\boldsymbol{x}; \boldsymbol{\theta}_u\right)$. Additionally, we can also make some appropriate deformations based on $H\left(\boldsymbol{x}; \boldsymbol{\theta}_\rho\right)$ so that $\hat{\rho}$ can satisfy some initial guess. After that, we uniformly sample points in Ω in preparation for the discretization. With the sample points, we can calculate the L_{sum} and start to train the neural networks. A common procedure is to update $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_\rho$ simultaneously by minimizing the loss function L_{sum} in one iteration. However, this method makes the network parameters of two different physical variables update with the same step size, which reduces the training efficiency. In this paper, we use the adversarial training to improve efficiency. More specifically, the alternating direction method is used to update $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_\rho$.

In one iteration of the inner loop, we first fix $\boldsymbol{\theta}_\rho$, update $\boldsymbol{\theta}_u$ by minimizing the loss function $\mathcal{L}_{sum}^k\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right)$ continuously T_u times. Then fix $\boldsymbol{\theta}_u$ and update $\boldsymbol{\theta}_\rho$ by minimizing the loss function $\mathcal{L}_{sum}^k\left(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho\right)$ continuously T_ρ times. In an inner loop, μ_k and λ_k do not change concerning k. Let R and K be the numbers of the inner loop and the outer loop, respectively. The total number of iterations $Epoch = K \times R$.

Algorithm 1 demonstrates a general algorithmic framework of the present adversarial neural network method for the eigenvalue optimization problem.

Remark 3.1. *It is worth noting that the effectiveness and convergence of the proposed adversarial neural network method exhibit some sensitivity to the initialization of the density function and the choice of hyperparameters.*

Initialization: *As demonstrated in the subsequent numerical examples, a purely random initialization may lead the neural network into suboptimal local minima or cause slower convergence. Therefore, incorporating physical intuition into the initial density construction (e.g., assigning specific density values on the boundaries versus the domain center) is necessary. This structured initialization provides a favorable starting point in the high-dimensional solution space, significantly enhancing the algorithm's robustness and accelerating the optimization process.*

Hyperparameters: *The adversarial training process also requires appropriate tuning of hyperparameters, particularly the learning rates (η_u and η_ρ) and the inner loop iteration counts (T_u and T_ρ). Through empirical numerical experiments, we found that maintaining an appropriate balance between the update frequencies and step sizes of the two networks is key to stable training. For instance, setting a slightly larger learning rate for the density network (η_ρ) compared to the eigenfunction network (η_u) generally yields better stability. Furthermore, for the geometric constraints, starting with a relatively small initial penalty coefficient (μ_0) and increasing it gradually (via the growth factor β) prevents the network from being overly restricted during its early exploratory phase.*

Remark 3.2. *Compared with classical mesh-based topology optimization methods, the proposed adversarial neural network framework improves computational efficiency by avoiding repeated large-scale eigenvalue solves inherent in finite element–based sensitivity analysis, which are particularly costly for fine meshes or three-dimensional problems. Instead, automatic differentiation is used to directly compute optimization directions in a meshless setting. The adversarial training strategy with alternating updates further enhances efficiency, and the method is well suited for GPU acceleration. Nevertheless, the overall performance still depends on training-related factors, such as hyperparameters and convergence behavior.*

4 Numerical Results

This section shows the use of an adversarial neural network algorithm on (2.7), (2.8), (2.14), and (2.15). All numerical examples were computed on the Google Colab platform.

4.1 Second-order problems

Example 4.1. *Consider an annular domain $\Omega = \left\{(x,y) : r_{\text{in}}^2 \le x^2+y^2 \le 1\right\}$. Let $\tau = 1/r_{in}$ (indicating the ratio of the outer radius to the inner radius), $\delta = c/|\Omega|$ (indicating the proportion of the area constraint to the total area). Set a uniform grid of 50×50 in the square $[-1,1]^2$ and then choose the grid points in the annular as the sampling points.*

The boundary condition function is constructed by $\ell_u = (1-x^2-y^2)(x^2+y^2-r_{\text{in}}^2)$. Set

$$\hat{u}\left(x,y;\boldsymbol{\theta}_u\right) = \ell_u(x,y) H\left(x,y;\boldsymbol{\theta}_u\right), \tag{4.1}$$

where $H\left(x,y;\boldsymbol{\theta}_u\right)$ is the neural network. For function $\rho \in [0,1]$, suitable estimation allows us to make the following construction of $\hat{\rho}$:

$$\hat{\rho}(x,y) = \max\left(0, \min\left(1, -g(x,y) H\left(x,y;\boldsymbol{\theta}_\rho\right)+1\right)\right), \tag{4.2}$$

with

$$g(x,y) = \min(1-\sqrt{x^2+y^2}, \sqrt{x^2+y^2}-r_{\text{in}}).$$

Algorithm 1: Adversarial neural network method for topology optimization of eigenvalue problems

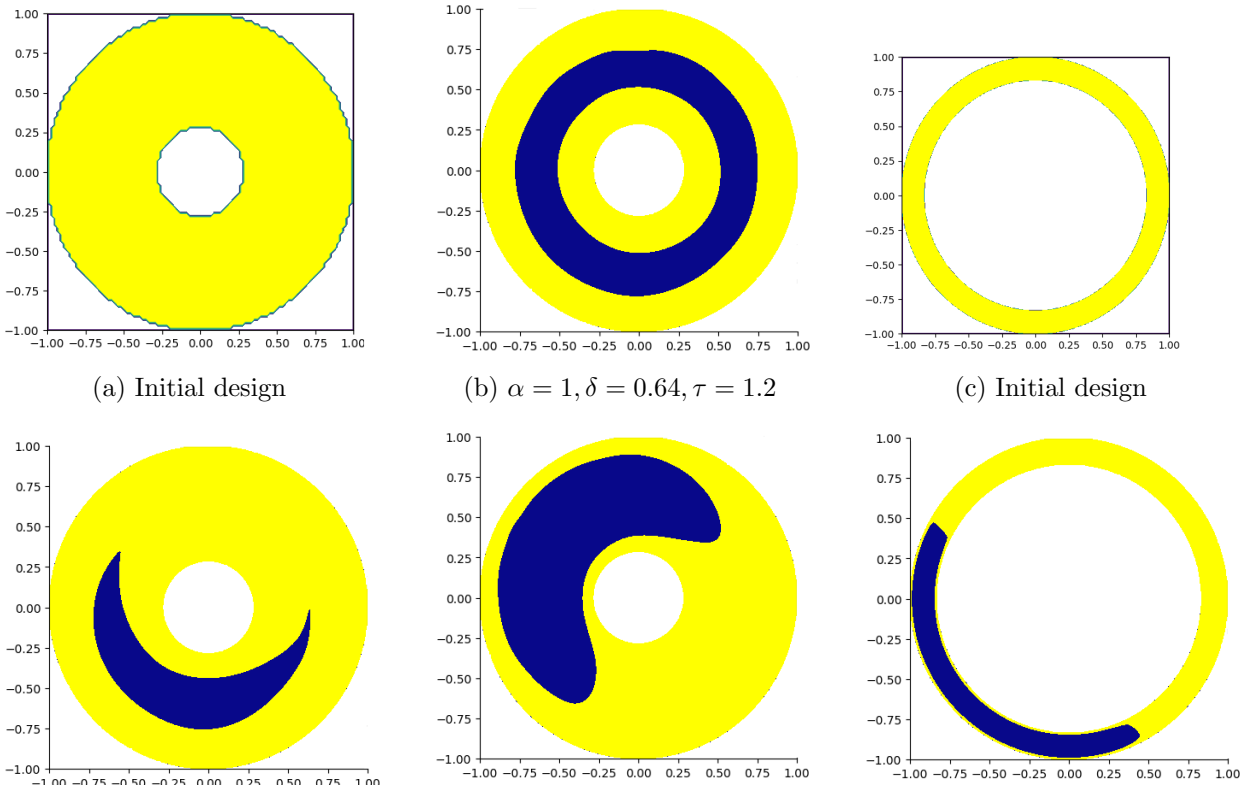
Input: Choose the proper loss function $\mathcal{L}_{sum}^{k}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho)$ according to the problem.
Data: K: number of outer loop iterations; R: number of inner loop iterations; T_u: number of inner iterations for training $\hat{u}$; T_ρ: number of inner iterations for training $\hat{\rho}$; η_u: learning rate of $\hat{u}$; η_ρ: learning rate of $\hat{\rho}$.
Data: $\lambda^0 = 0$, μ^0, β, S.
for $k = 1$ **to** K **do**
  for $i = 1$ **to** R **do**
    for $t = 1$ **to** T_u **do**
      Compute the loss function $\mathcal{L}_{sum}^{k}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho)$.
      Compute gradient about $\boldsymbol{\theta}_u$ in $\mathcal{L}_{sum}^{k}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho)$, and update $\boldsymbol{\theta}_u$ by Adam method.
    end
    for $t = 1$ **to** T_ρ **do**
      Compute the loss function $\mathcal{L}_{sum}^{k}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho)$.
      Compute gradient about $\boldsymbol{\theta}_\rho$ in $\mathcal{L}_{sum}^{k}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\rho)$, and update $\boldsymbol{\theta}_\rho$ by Adam method.
    end
  end
  $\mu_k = \min(\beta\mu_{k-1}, S)$.
  $\lambda_k = \lambda_{k-1} + 2\mu_{k-1}h(\boldsymbol{\theta}_\rho)$ for augmented Lagrangian method.
end

We apply the augmented Lagrangian method to address the geometry constraint. Set parameters: $T_u = 4, T_\rho = 1, R = 50, K = 15, \eta_u = 10^{-4}, \eta_\rho = 10^{-3}$, $\beta = 1.1, \mu_0 = 10$, *and* $S = 1000$. *Set* $\rho_1 = 0$ *and* $\rho_2 = 1$.

In Fig. 3, the blue and yellow regions represent $\Omega \setminus D$ *and* D, *respectively. If* α *or* δ *is not relatively large, i.e., (*$\alpha = 1$ *or* $\delta = 0.64$*), an optimized design with radial symmetry is obtained, as shown in Fig. 3* (b). *However, in the same annular* Ω, *Figs. 3* (d)-(e) *show radial symmetry breaking for final designs when increasing* α *from 1 to 10 or increasing the target area of* D. *Compared with Fig. 3* (b), *Fig. 3* (f) *also has radial symmetry breaks when reducing the width of the annulus. These observed phenomena are consistent with those described in* [4]. *Fig. 3* (a) *and Fig. 3* (c) *display the initial state of* $\hat{\rho}(x, y)$. *One can observe that shape and topological changes occur from an initial design (without blue density) to an optimized design (with blue density).*

Fig. 4 shows that the objectives converge. Note that the loss function in the second subfigure is an order of magnitude larger than the others. This is due to the fact that a reduction in the area of the design domain leads to an increase in the first eigenvalue [3], *which directly impacts the loss function values.*

Example 4.2. *Consider a dumbbell domain* $\Omega = B_1(-2, 0) \cup ((-2, 2) \times (-0.3, 0.3)) \cup B_1(2, 0)$, *where* $B_1(x, y)$ *denotes the unit disk centered at* (x, y). *We still use the uniform sampling method to establish a regular grid of* 60×60 *in the rectangle* $[-3, 3] \times [-1, 1]$ *that surrounds the dumbbell, and then take the square grid points in the dumbbell as the sampling points. There is a difference between the two adversarial neural networks* $H(x, y; \boldsymbol{\theta}_u)$ *and* $H(x, y; \boldsymbol{\theta}_\rho)$ *we established in the dumbbell domain and those in the annular domain: the output layer is no longer simply linear but is a square function.* $\hat{u}$ *and* $\hat{\rho}$ *are built according to* (4.1) *and* (4.2), *respectively, in which* ℓ_u *is replaced as follows with a*



(a) Initial design (b) $\alpha = 1, \delta = 0.64, \tau = 1.2$ (c) Initial design

(d) $\alpha = 1, \delta = 0.83, \tau = 3.5$ (e) $\alpha = 10, \delta = 0.64, \tau = 3.5$ (f) $\alpha = 1, \delta = 0.64, \tau = 3.5$

Figure 3: Optimized designs and initial designs for elliptic eigenvalue optimization for Example 4.1.

symbolic distance function and dumbbell features:

$$\ell_u = \max(0.3 - |y|, \ell_d)\ \chi_{|x|\leq 2.05} + \ell_d\ \chi_{|x|\geq 2.05},$$

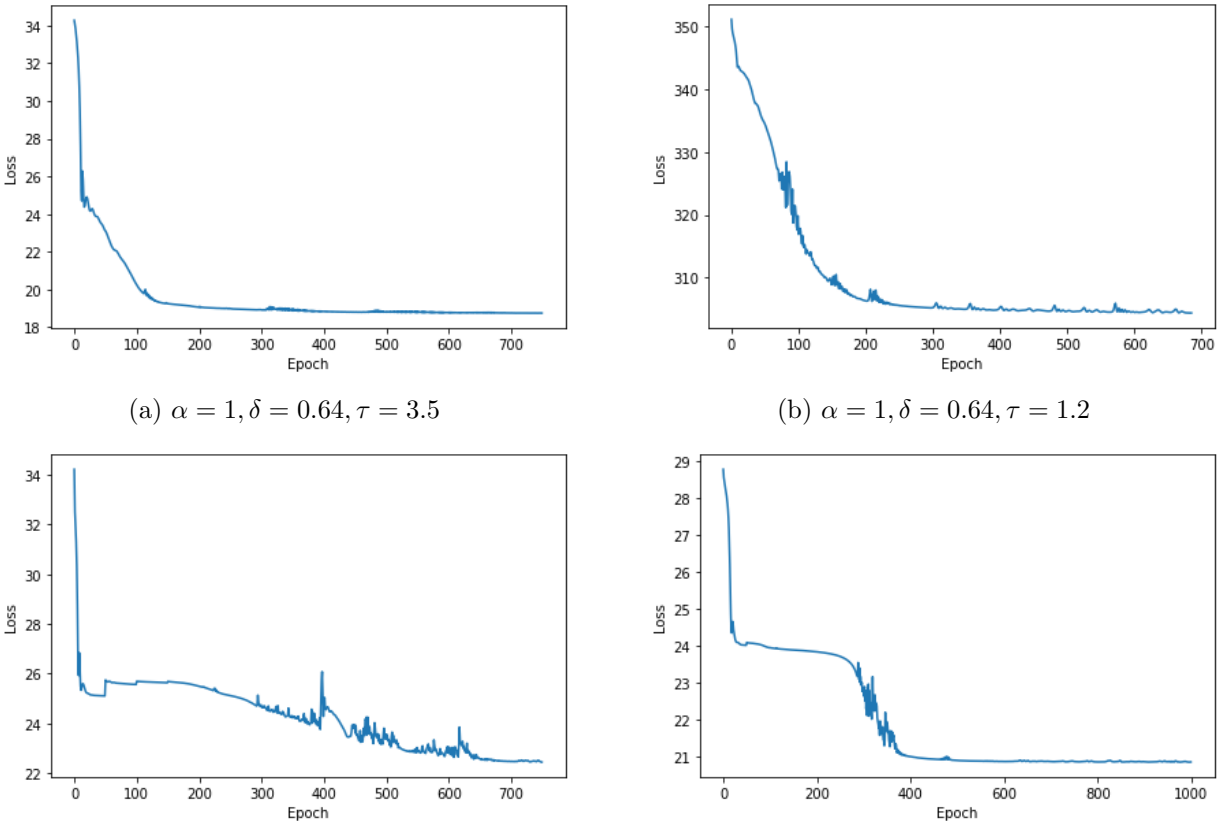
where $\ell_d = \max(1-(x-2)^2-y^2, 1-(x+2)^2-y^2)$. *Set* $\alpha = 0.1$ *and* $\delta = 0.3$ *in* (2.7), *we apply the varying coefficient penalty method and obtain the optimized shape of* D *in Fig.* 5, *which agrees well with that in* [4].

Example 4.3. *For* $3D$, *set* $\alpha = 10$ *and* $c = 0.5$. *Let* $\Omega = (0,1)^3$. *We establish a regular grid of* $40 \times 40 \times 40$ *in the cube through uniform sampling. The adversarial neural networks here are* $H(x,y,z;\boldsymbol{\theta}_u)$ *and* $H(x,y,z;\boldsymbol{\theta}_\rho)$. *We construct a boundary condition function* $\ell_u = 64\,(xyz(1-x)(1-y)(1-z))$ *to satisfy the boundary condition. Let* $\hat{u}(x,y,z;\boldsymbol{\theta}_u) = \ell_u(x,y,z)H(x,y,z;\boldsymbol{\theta}_u)$. *As to the minimization problem* (2.7), *we make the reasonable assumption that* $\rho = 1$ *on the boundary,* $\rho = 0$ *at the central point. The advantage of these initial guesses is that they not only enhance the neural network's convergence and robustness but also significantly facilitate the optimization process. Providing a structured initial condition helps the neural network navigate the solution space more efficiently, leading to improved optimization results. We choose*

$$\hat{\rho}(x,y,z) = \max\left(0, \min\left(1, \ell_u(x,y,z)\ell_c(x,y,z)H(x,y,z;\boldsymbol{\theta}_\rho) + g(x,y,z)\right)\right),$$

where

$$\ell_c(x,y,z) = (x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2, \quad g(x,y,z) = 1 - \ell_u'(x,y,z)$$



(a) $\alpha = 1, \delta = 0.64, \tau = 3.5$

(b) $\alpha = 1, \delta = 0.64, \tau = 1.2$

(c) $\alpha = 10, \delta = 0.64, \tau = 3.5$

(d) $\alpha = 1, \delta = 0.83, \tau = 3.5$

Figure 4: Convergence of loss functions on elliptic eigenvalue optimization for Example 4.1.

with $\ell_u'(x,y,z) = 2\min(x, 1-x, y, 1-y, z, 1-z)$. *Set* $T_u = 3$, $T_\rho = 1$, $R = 15$, $K = 15$, $\eta_u = 10^{-4}$, *and* $\eta_\rho = 5 \times 10^{-4}$. *We apply the augmented Lagrangian method for the geometry constraint and choose* $\beta = 1.5$, $\mu_0 = 1$, *and* $S = 1000$. *Finally, we obtain the optimized design* D *and the convergence process of the loss function, as shown in Fig. 6.*

As for the maximization problem (2.8), we reasonably set $\rho = 0$ *on the boundary and* $\rho = 1$ *at the central point. We choose*

$$\hat{\rho}(x,y,z) = \max\left(0, \min\left(1, \ell_u(x,y,z)\ell_c(x,y,z)H\left(x,y,z;\boldsymbol{\theta}_\rho\right) + g(x,y,z)\right)\right),$$

where

$$\ell_c(x,y,z) = (x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2, \quad g(x,y,z) = \ell_u'(x,y,z).$$

The final optimized design and the convergence process of the loss function are shown in Fig. 7.

In Fig. 6, the optimal design of the minimization problem is the complementary set of solid spheres. The first eigenvalue λ_1 *is optimized to a minimal value of* 30.07. *To assess the accuracy of the maximization of the first eigenvalue, we employ the finite element method and utilize Freefem++ [44] to solve the corresponding eigenvalue problem within this region. The minimum eigenvalue obtained is* 29.94, *which is close to the result of the neural network. In Fig. 7, the optimal region* D *of the maximization problem in* 3D *is the solid sphere in the center. We get a maximal eigenvalue of* 39.32, *which is also close to the result of* 39.20 *computed by the finite element method. Our model's results on optimization in* 3D *behave well, as expected.*

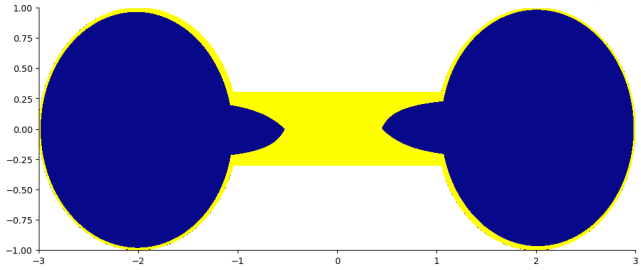


Figure 5: Optimized shape of D in dumbbell domain for Example 4.2.

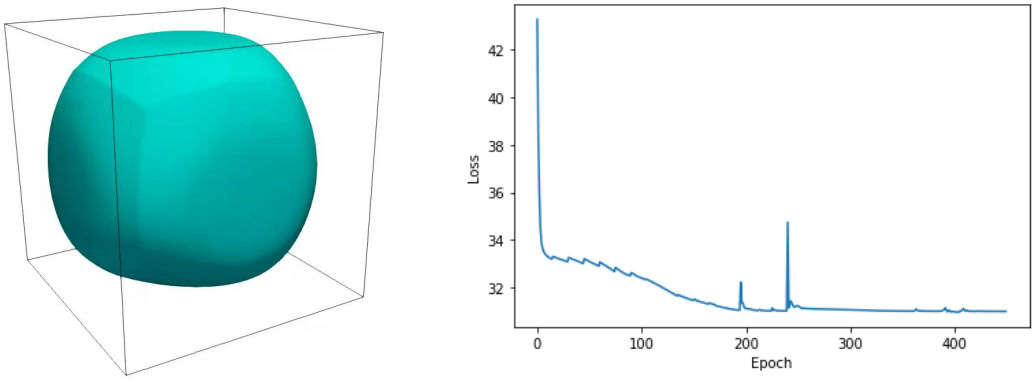


Figure 6: Elliptic eigenvalue minimization in 3D: optimized design (left) and convergence history of loss (right) for Example 4.3.

4.2 Fourth-order problems

4.2.1 Eigenvalue optimization in inhomogeneous clamped plate

For problem (2.14), choose $c = 1.5$, $\rho_1 = 1$, and $\rho_2 = 2$. Let Ω be a square, a circle, and an annulus. In particular, regarding the clamped plate problem, since it is easy to find a function satisfying the boundary conditions, we use the Full-satisfied boundary method. The neural networks that we build for $H(\boldsymbol{x};\boldsymbol{\theta}_u)$ and $H(\boldsymbol{x};\boldsymbol{\theta}_\rho)$ have 3 residual blocks. Every block has two fully connected layers, and every layer has 80 neurons. All the activation functions are tanh, except that we take the linear function in the output layer. Set $T_u = 3, T_\rho = 1, R = 15, K = 30, \eta_u = 10^{-4}, \eta_\rho = 10^{-3}$, $\beta = 2, \mu_0 = 1$, and $S = 1000$. In this fourth-order biharmonic eigenvalue optimization problem, the neural network structures and parameters are the same, while the constructions of $\hat{u}$ and $\hat{\rho}$ depend on the shape of Ω and the optimization target. We minimize (resp. maximize) the eigenvalue for Examples 4.4–4.6 (resp. Example 4.7).

Example 4.4. *Consider $\Omega = (0,1)\times(0,1)$ and choose $\ell_u = 256[x(x-1)y(y-1)]^2$ to satisfy boundary conditions. As to $\hat{\rho}$, we reasonably guess $\rho = 1$ on the boundary and $\rho = 2$ at the central point. Similar to the second-order problem, we construct*

$$\hat{\rho}(x,y) = \max\left(1, \min\left(2, \ell_u(x,y)\ell_c(x,y)H(x,y;\boldsymbol{\theta}_\rho) + g(x,y)\right)\right), \tag{4.3}$$

where

$$\ell_c(x,y) = (x-0.5)^2 + (y-0.5)^2, \quad g(x,y) = \ell_u(x,y) + 1.$$

The left subfigure of Fig. 8 shows the initial density function. It can be observed that the desired construction is achieved. After training, the optimal region (the middle figure of Fig. 8) is a circle in

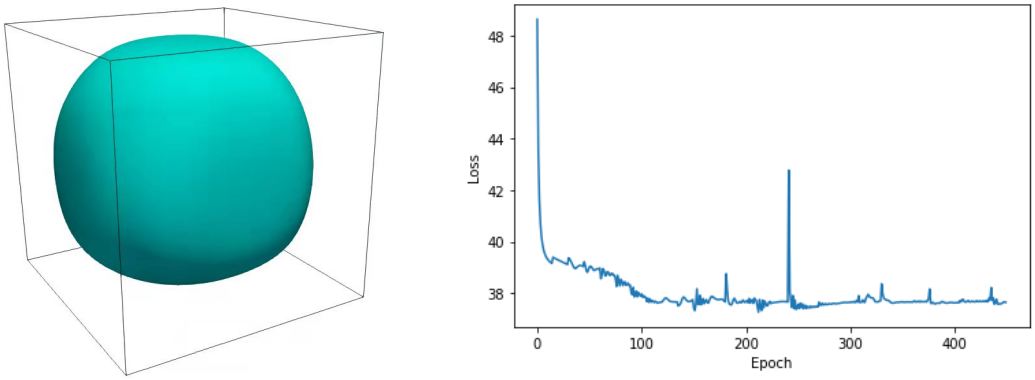


Figure 7: Elliptic eigenvalue maximization in 3D: optimized design (left) and convergence history of loss (right) for Example 4.3.

the center, while the blue region represents a density of 1 *and the yellow region represents a density of* 2. *When* Ω *is a square domain, our method has converged within 300 epochs and obtained the minimal eigenvalue of* 677.3, *while the result obtained by the rearrangement algorithm* [5] *is* 652.2, *resulting in a* 3.8% *difference. Each epoch takes* 1.5 *seconds. The rearrangement algorithms mentioned primarily employ traditional gradient-based optimization methods. These algorithms iterate on the material distribution by locally optimizing the gradient of the objective with respect to the material density.*

In contrast, our approach leverages a hybrid technique combining data-driven insights with optimization algorithms. This method accelerates the convergence process and enhances the adaptability and efficiency of the optimization under complex constraints and varying conditions.

Example 4.5. *Consider* Ω *as a unit disk domain:* $\Omega = \left\{(x, y) : x^2 + y^2 \leq 1\right\}$. *We choose* $\ell_u = \left(1 - x^2 - y^2\right)^2$ *to satisfy boundary conditions and then construct* $\hat{u}$ *by* (3.10). *As for* $\hat{\rho}$, *we reasonably guess that* $\rho = 1$ *on the boundary,* $\rho = 2$ *at the central point, and we construct*

$$\hat{\rho}(x, y) = \max\left(1, \min\left(2, \ell_u(x, y)\ell_c(x, y)H\left(x, y; \boldsymbol{\theta}_\rho\right) + g(x, y)\right)\right), \tag{4.4}$$

where

$$\ell_c(x, y) = x^2 + y^2, \quad g(x, y) = \ell_u(x, y) + 1.$$

The left subfigure in Fig. 9 *shows the initial density function. Our method has good performance (the middle subfigure of Fig.* 9*); the optimal region* D *is highly consistent with the shape optimized by the*

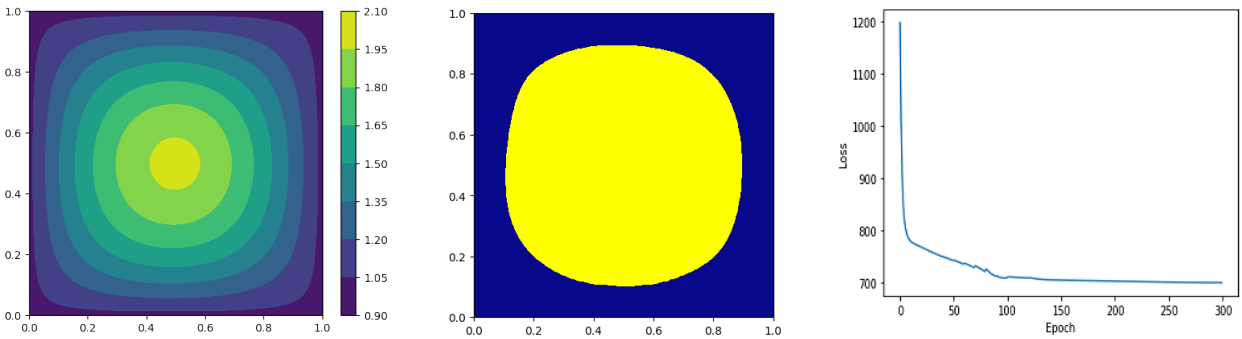


Figure 8: Eigenvalue minimization of the inhomogeneous clamped plate for Example 4.4: initial design (left), optimized design (middle) and convergence history of loss (right).

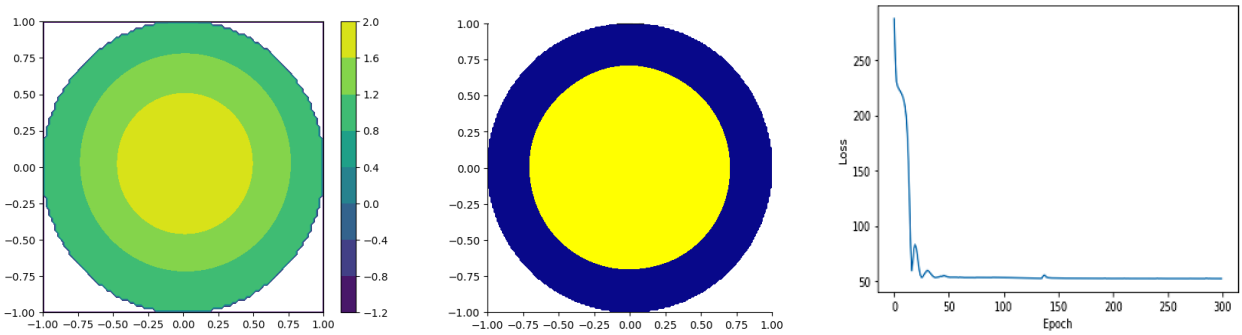


Figure 9: Eigenvalue minimization of the inhomogeneous clamped plate for Example 4.5: initial design (left), optimized design (middle) and convergence history of loss (right).

rearrangement algorithm [5]*, and the minimum has converged to* 52.3 *within* 50 *epochs, which is close to* 52.9 *obtained by the rearrangement algorithm.*

Example 4.6. *Consider an annulus* $\Omega = \left\{(x,y) : 0.4^2 \leq x^2+y^2 \leq 1\right\}$ *and choose*

$$\ell_u = \left(1-x^2-y^2\right)^2\left(x^2+y^2-0.16\right)^2.$$

We also guess that $\rho = 1$ *on the boundary and* $\rho = 2$ *at the central to construct*

$$\hat{\rho}(x,y) = \max\left(1, \min\left(2, \ell_u(x,y)H\left(x,y;\boldsymbol{\theta}_\rho\right)+1\right)\right),$$

The first eigenvalue is optimized to a minimum of 1998*, which differs from the result of* 1946 *optimized by the rearrangement algorithm by* 2.6%*. Fig.* 10 *shows the convergence of loss, and the optimized design agrees well with that in* [5].

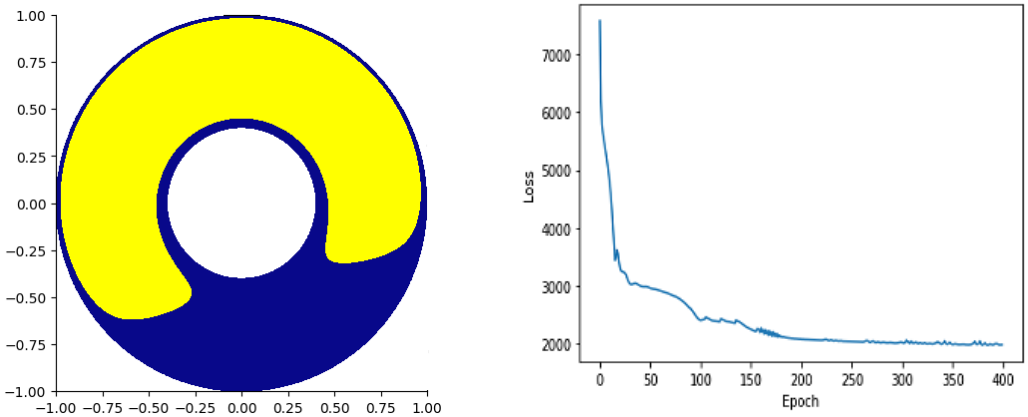


Figure 10: Eigenvalue minimization of the inhomogeneous clamped plate for Example 4.6: optimized design (left) and convergence history of loss (right).

Example 4.7. *Consider a unit disk domain:* $\Omega = \left\{(x,y) : x^2+y^2 \leq 1\right\}$*. We construct* $\hat{u}$ *the same as that in the minimization problem. As for* $\hat{\rho}$*, contrary to the situation in the minimization problem, we reasonably guess that* $\rho = 2$ *on the boundary,* $\rho = 1$ *at the central point, and we construct it as follows:*

$$\hat{\rho}(x,y) = \max\left(1, \min\left(2, \ell_u(x,y)\ell_c(x,y)H\left(x,y;\boldsymbol{\theta}_\rho\right)+g(x,y)\right)\right), \tag{4.5}$$

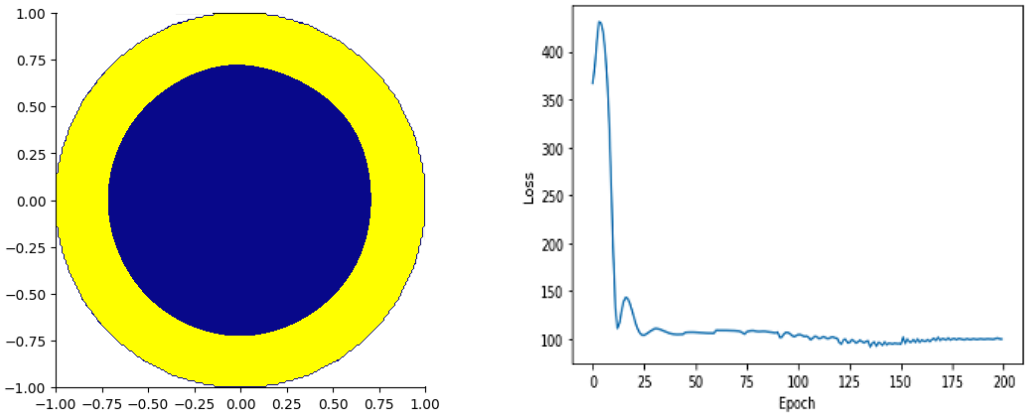


Figure 11: Eigenvalue maximization of the inhomogeneous clamped plate for Example 4.7: optimized design (left) and convergence history of loss (right).

where

$$\ell_c(x,y) = x^2 + y^2, \quad g(x,y) = \ell_c(x,y) + 1.$$

The maximized value of 101.8 *for the eigenvalue obtained agrees well with that* (*a maximum of* 101.7) *in* [45]*, and the optimized shape is similar to that in* [45] (*see Fig.* 11 *for design and history of loss*).

4.2.2 Eigenvalue optimization in inhomogeneous simply supported plate

In the simply supported plate problem (2.15), we set $\nu = 0.3$, $c = 1.5$, $\rho_1 = 1$, and $\rho_2 = 2$. Considering that the boundary condition is relatively complex, we apply the half-satisfied boundary method (3.15) and (3.17). The structures of $H(\boldsymbol{x};\boldsymbol{\theta}_u)$ and $H(\boldsymbol{x};\boldsymbol{\theta}_\rho)$ are the same as those in the clamped plate problem. We minimize (resp. maximize) the eigenvalue for Examples 4.8-4.9 (resp. Example 4.10).

The total loss function $\mathcal{L}_{sum}$ is (3.15) and $\mathcal{L}_w$ in the (3.15) is that of the augmented Lagrangian method, while $\mathcal{L}_{in}$ is given in (3.15).

Example 4.8. *Consider a square* $\Omega = (0,1)\times(0,1)$. *We choose* $\ell_u = 16x(1-x)y(1-y)$. *Then construct* $\hat{u}$ *and* $\hat{\rho}$ *according to* (3.10) *and* (4.3) *respectively. In addition,* $\kappa = 0$ *on the square boundary.*

Example 4.9. *Consider a disk* $\Omega = \{(x,y) : x^2+y^2 \le 1\}$ *and choose* $\ell_u = 1-x^2-y^2$. *Then construct* $\hat{u}$ *and* $\hat{\rho}$ *according to* (3.10) *and* (4.4)*, respectively. Set* $\kappa = 1$ *on the boundary In* $\mathcal{L}_{in}$ *and take* $A_1 = 1$ *and* $A_2 = 2$.

It can be seen from Fig. 12 *and Fig.* 13 *that the adversarial neural network method under the half-satisfied boundary method has good performance in both square and unit disk, since the shape of the optimal density distribution under the two regions is consistent with that in the rearrangement algorithm* [5]. *With our method, the first eigenvalue in the square domain is minimized to* 202.04, *the first eigenvalue in the unit disk domain is minimized to* 12.83, *while the result of the rearrangement algorithm is* 201.60 *and* 12.84. *so, our algorithm is still efficient and accurate.*

Example 4.10. *Consider a unit disk* $\Omega = \{(x,y) : x^2+y^2 \le 1\}$ *and choose* $\ell_u = 1-x^2-y^2$. *Then construct* $\hat{u}$ *and* $\hat{\rho}$ *according to* (3.10) *and* (4.5)*, respectively. The total loss function* $\mathcal{L}_{sum}$ *is* (3.17)*, and the loss* $\mathcal{L}_w$ *in* (3.17) *is that of the augmented Lagrangian method, while* $\mathcal{L}_{in}$ *is* (3.5). *After training, the first eigenvalue of* λ_1 *was finally optimized to a maximum of* 22.3, *and the optimal density distribution map is shown in Fig.* 14.

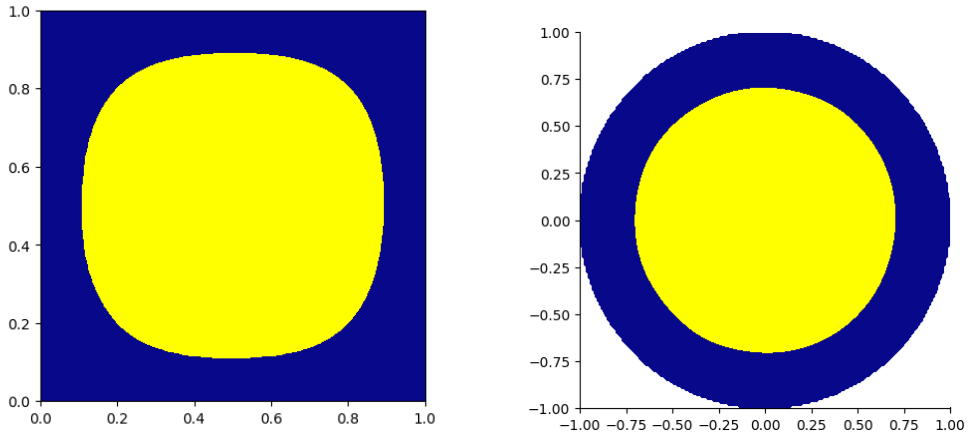


Figure 12: Eigenvalue minimization of the inhomogeneous simply supported plate problem: optimal region D when Ω is a square (left) for Example 4.8 and a unit disk (right) for Example 4.9.

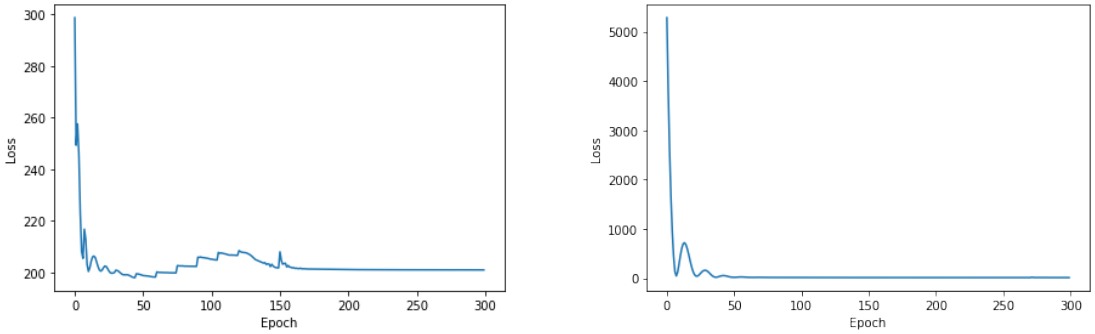


Figure 13: Convergence history of loss for biharmonic eigenvalue minimization of the inhomogeneous simply supported plate: square for Example 4.8 (left) and disk for Example 4.9 (right).

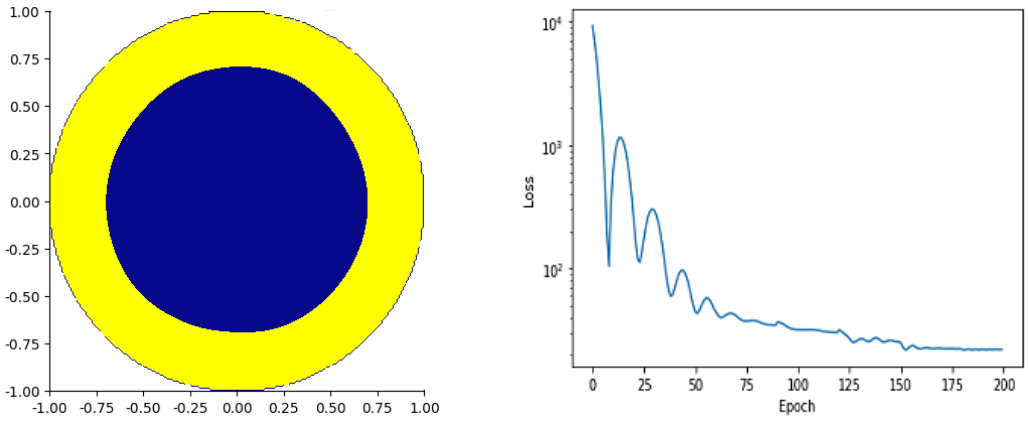


Figure 14: Eigenvalue maximization of the inhomogeneous simply supported plate for Example 4.10: optimized design (left) and convergence history of loss (right).

5 Conclusions

We have developed an adversarial neural network optimization method for optimizing the eigenvalues of the second-order elliptic operator and fourth-order plates. The advantages of the proposed method over traditional gradient-based algorithms include: (1) Traditional gradient-based algorithms require solving eigenvalue problems. In contrast, our method uses automatic differentiation to directly obtain the gradient direction for the entire optimization problem, which simplifies computations. (2) Our algorithm is designed to be fully executable on a GPU, ensuring increasing efficiency in line with advancements in GPU technology. (3) Our approach explores a broader solution space for eigenvalue optimization problems, encouraging innovative problem-solving methods and insights. (4) The algorithm has scalability and adaptability to higher-dimensional and more complex problems.

Numerical examples of the minimization and maximization of the first eigenvalue in typical domains are given, with well-constructed initial guesses that incorporate real-world experience into the optimization, enhancing the neural network's convergence and robustness during the optimization process. The numerical examples demonstrate the effectiveness and efficiency of the algorithm.

Acknowledgement

The work was supported in part by the National Key Basic Research Program (2022YFA1004402), the National Natural Science Foundation of China (12471377) and the Science and Technology Commission of Shanghai Municipality (22ZR1421900 and 22DZ2229014).

References

[1] M. P. Bendsoe and O. Sigmund, Topology Optimization: Theory, Methods and Applications. Springer Press, Berlin, 2003. DOI: 10.1007/978-3-662-05086-6

[2] J. Sokolowski and J. P. Zolesio, Introduction to the Shape Optimization: Shape Sensitivity Analysis. Springer, Heidelberg, 1992. DOI: 10.1007/978-3-642-58106-9

[3] A. Henrot, Extremum Problems for Eigenvalues of Elliptic Operators. Springer Science & Business Media, 2006. DOI: 10.1007/3-7643-7706-2

[4] S. Chanillo, D. Grieser, M. Imai, K. Kurata and I. Ohnishi, Symmetry breaking and other phenomena in the optimization of eigenvalues for composite membranes. Comm. Math. Phys. 214 (2000), no. 2, 315–337. DOI: 10.1007/PL00005534

[5] W. Chen, C. S. Chou and C.-Y. Kao, Minimizing eigenvalues for inhomogeneous rods and plates. J. Sci. Comput. 69 (2016), no. 3, 983–1013. DOI: 10.1007/s10915-016-0222-9

[6] C. Liu, X. Hu and S. Zhu, A two-grid binary level set method for structural topology optimization. Eng. Optim. 55 (2023), 1100–1117. DOI: 10.1080/0305215X.2022.2067991

[7] M. Qian, X. Hu and S. Zhu, A phase field method based on multi-level correction for eigenvalue topology optimization. Comput. Methods Appl. Mech. Engrg. 401 (2022), part B, Paper No. 115646, 42 pp. DOI: 10.1016/j.cma.2022.115646

[8] X. Hu, M. Qian and S. Zhu, Accelerating a phase field method by linearization for eigenfrequency topology optimization. Struct. Multidiscip. Optim. 66 (2023), no. 12, Paper No. 242, 17 pp. DOI: 10.1007/s00158-023-03692-9

[9] S. A. Mohammadi and F. Bahrami, Extremal principal eigenvalue of the bi-Laplacian operator. Appl. Math. Model. 40 (2016), no. 3, 2291–2300. DOI: 10.1016/j.apm.2015.09.058

[10] S. Zhu, C. Liu and Q. Wu, Binary level set methods for topology and shape optimization of a

two-density inhomogeneous drum. Comput. Methods Appl. Mech. Engrg. 199 (2010), no. 45-48, 2970–2986. 2010. DOI: 10.1016/j.cma.2010.06.007

[11] S. J. Cox and D. C. Dobson, Band structure optimization of two-dimensional photonic crystals in H-polarization. J. Comput. Phys. 158 (2000), no. 2, 214–224. DOI: 10.1006/jcph.1999.6415

[12] Y. Lou and E. Yanagida, Minimization of the principal eigenvalue for an elliptic boundary value problem with indefinite weight, and applications to population dynamics. Japan J. Indust. Appl. Math. 23 (2006), no. 3, 275–292. DOI: 10.1007/BF03167595

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60 (2012), no. 6, 84–90. DOI: 10.1145/3065386

[14] B. M. Lake, R. Salakhutdinov and J. B. Tenenbaum, Human-level concept learning through probabilistic program induction. Science 350 (2015), no. 6266, 1332–1338. DOI: 10.1126/science.aab3050

[15] B. Alipanahi, A. Delong, M. T. Weirauch and B. J. Frey, Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat. Biotechnol. 33 (2015), 831–838, 2015. DOI: 10.1038/nbt.3300

[16] E. Weinan, A proposal on machine learning via dynamical systems, Commun. Math. Stat. 5 (2017), no. 5, 1–11. DOI: 10.1007/s40304-017-0103-z

[17] E. Weinan and B. Yu, The Deep Ritz method: A deep learning based numerical algorithm for solving variational problems. Comm. Meth. Stat. 6 (2018), no. 6, 1–12. DOI: 10.1007/s40304-018-0127-z

[18] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375 (2018), 1339–1364. DOI: 10.1016/j.jcp.2018.08.029

[19] M. Raissi, P. Perdikaris and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378 (2019), 686–707. DOI: 10.1016/j.jcp.2018.10.045

[20] Y. Wang and H. Xie, Computing multi-eigenpairs of high-dimensional eigenvalue problems using tensor neural networks. J. Comput. Phys. 506 (2024), Paper No. 112928, 17 pp. DOI: 10.1016/j.jcp.2024.112928

[21] J. Berg and K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing 317 (2018), 28–41. DOI: 10.1016/j.neucom.2018.06.056

[22] J. Huang, H. Wang and T. Zhou, An augmented Lagrangian deep learning method for variational problems with essential boundary conditions, Commun. Comput. Phys. 31 (2022), no. 3, 966–986. DOI: 10.4208/cicp.OA-2021-0176

[23] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo and S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design. SIAM J. Sci. Comput. 43 (2021), no. 6, B1105–B1132. DOI: 10.1137/21M1397908

[24] L. Lu, X. Meng, and Z. Mao, DeepXDE: A deep learning library for solving differential equations. SIAM Rev. 63 (2021), no. 1, 208–228. DOI: 10.1137/19M1274067

[25] H. Deng and A. C. To, A parametric level set method for topology optimization based on deep neural network. ASME J. Mech. Des. 143 (2021), no. 9, 9 pp. DOI: 10.1115/1.4050105

[26] S. Oh, Y. Jung, S. Kim, I. Lee and N. Kang, Deep generative design: Integration of topology optimization and generative models. ASME J. Mech. Des. 141 (2019), no. 11, 111405, 13 pp. DOI: 10.1115/1.4044229

[27] X. Lei, C. Liu, Z. Du, W. Zhang and X. Guo, Machine learning-driven real-time topology optimization under moving morphable component-based framework. ASME J. Appl. Mech.86 (2019), no. 1, 011004, 9 pp. DOI: 10.1115/1.4041319

[28] M. Huang, Z. Du, C. Liu, Y. Zheng, T. Cui, Y. Mei, X. Li, X. Zhang and X. Guo, A problem-

independent machine learning (PIML) enhanced substructure-based approach for large-scale structural analysis and topology optimization of linear elastic structures. Extreme Mech. Lett. 63 (2023), 102041, 14 pp. DOI: 10.1016/j.eml.2023.102041

[29] Y. Zang, G. Bao, X. Ye and H. Zhou, Weak adversarial networks for high-dimensional partial differential equations. J. Comput. Phys. 411 (2020), 109409, 14 pp. DOI: 10.1016/j.jcp.2020.109409

[30] G. Bao, X. Ye, Y. Zang and H. Zhou, Numerical solution of inverse problems by weak adversarial networks. Inverse Problems 36 (2020), no. 11, 115003, 31 pp. DOI: 10.1088/1361-6420/abb447

[31] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind, Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. 18 (2018), 1–43.

[32] K. Liang, X. Lu and J. Z. Yang, Finite element approximation to the extremal eigenvalue problem for inhomogeneous materials. Numer. Math. 130 (2015), no. 4, 741–762. DOI: 10.1007/s00211-014-0678-1

[33] S. Osher and F. Santosa, Level set methods for optimization problems involving geometry and constraints I. Frequencies of a two-density inhomogeneous drum. J. Comput. Phys. 171 (2001), 272–288. DOI: 10.1006/jcph.2001.6789

[34] J. Zhang, S. Zhu, C. Liu and X. Shen, A two-grid binary level set method for eigenvalue optimization. J. Sci. Comput. 89 (2021), no. 3, Paper No. 57, 21 pp. DOI: 10.1007/s10915-021-01662-1

[35] D. Kang and C.-Y. Kao, Minimization of inhomogeneous biharmonic eigenvalue problems. Appl. Math. Model. 51 (2017), 587–604. DOI: 10.1016/j.apm.2017.07.015

[36] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning. MIT Press, Cambridge, 2016.

[37] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition. In: IEEE Conf. Comput. Vis. Pattern Recognit. pp. 770–778, 2026. DOI: 10.1109/CVPR.2016.90

[38] X. Glorot, A. Bordes, and Y. Bengio, Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323, 2011.

[39] D. Kingma and J. Ba, Adam: A method for stochastic optimization. In: International Conference on Learning Representations, 2014. DOI: 10.48550/arXiv.1412.6980

[40] L. Sagun, L. Bottou and Y. LeCun, Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond. *arXiv*, 2016. DOI: 10.48550/arXiv.1611.07476

[41] S. Osher and R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag, New York, 2002. DOI: 10.1007/b98879

[42] D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods. Academic Press, 1982. DOI: 10.1016/C2013-0-10366-2

[43] T. F. Chan and X. C. Tai, Level set and total variation regularization for elliptic inverse problems with discontinuous coefficients. J. Comput. Phys. 193 (2003), 40–66. DOI: 10.1016/j.jcp.2003.08.003

[44] F. Hecht, New development in FreeFEM++. J. Numer. Math. 20 (2012), no. 3-4, 251–265. DOI: 10.1515/jnum-2012-0013

[45] P. R. S. Antunes, Optimal bilaplacian eigenvalues. SIAM J. Control Optim. 52 (2014), no. 4, 2250–2260. DOI: 10.1137/130948860