SIAM J. SCI. COMPUT.
Vol. 43, No. 6, pp. B1105–B1132

PHYSICS-INFORMED NEURAL NETWORKS WITH HARD CONSTRAINTS FOR INVERSE DESIGN*

LU LU†, RAPHAËL PESTOURIE‡, WENJIE YAO‡, ZHICHENG WANG§, FRANCESC VERDUGO¶, AND STEVEN G. JOHNSON‡

Abstract. Inverse design arises in a variety of areas in engineering such as acoustic, mechanics, thermal/electronic transport, electromagnetism, and optics. Topology optimization is an important form of inverse design, where one optimizes a designed geometry to achieve targeted properties parameterized by the materials at every point in a design region. This optimization is challenging, because it has a very high dimensionality and is usually constrained by partial differential equations (PDEs) and additional inequalities. Here, we propose a new deep learning method—physics-informed neural networks with hard constraints (hPINNs)—for solving topology optimization. hPINN leverages the recent development of PINNs for solving PDEs, and thus does not require a large dataset (generated by numerical PDE solvers) for training. However, all the constraints in PINNs are soft constraints, and hence we impose hard constraints by using the penalty method and the augmented Lagrangian method. We demonstrate the effectiveness of hPINN for a holography problem in optics and a fluid problem of Stokes flow. We achieve the same objective as conventional PDE-constrained optimization methods based on adjoint methods and numerical PDE solvers, but find that the design obtained from hPINN is often smoother for problems whose solution is not unique. Moreover, the implementation of inverse design with hPINN can be easier than that of conventional methods because it exploits the extensive deep-learning software infrastructure.

Key words. inverse design, topology optimization, partial differential equations, physics-informed neural networks, penalty method, augmented Lagrangian method

AMS subject classifications. 35R30, 65K10, 68T20

DOI. 10.1137/21M1397908

1. Introduction.

Metamaterials are artificial materials that achieve targeted properties through designed fine-scale geometry rather than via material properties. They arise in a variety of areas in engineering, e.g., acoustics, mechanics, thermal/electronic transport, and electromagnetism/optics [25]. Designing a metamaterial's geometry for a particular functionality is a challenging task, because the design space has a very high dimensionality (millions to billions of parameters). In contrast to intuition-based approaches using heuristics and a handful of hand-tweaked geometric parameters, *inverse design* starts with the targeted functionality and sets it as an objective function to be optimized via partial-differential-equation-constrained (PDE-constrained) large-scale optimization [38]. When the parameterization of the geometry is via the parameters of discrete geometric components, inverse design is

*Submitted to the journal's Computational Methods in Science and Engineering section February 9, 2021; accepted for publication (in revised form) August 11, 2021; published electronically November 11, 2021.

https://doi.org/10.1137/21M1397908

Funding: This work was supported by the MIT-IBM Watson AI Laboratory (challenge 2415). The work of the fifth author was supported by the Spanish Ministry of Economy and Competitiveness through the "Severo Ochoa Programme for Centers of Excellence in R&D (CEX2018-000797-S)."

†Department of Chemical and Biomolecular Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (lulu1@seas.upenn.edu).

‡Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (raphael.pestourie@gmail.com, jayyao@mit.edu, stevenj@math.mit.edu).

§Laboratory of Ocean Energy Utilization of Ministry of Education, Dalian University of Technology, Dalian, 116024, China (zhicheng_wang@brown.edu).

¶Centre Internacional de Mètodes Numèrics en Enginyeria, Esteve Terradas 5, 08860 Castelldefels, Barcelona, Spain (fverdugo@cimne.upc.edu).

called *shape optimization* [4,45]. When the parameterization of the geometry is via a density function or level set so that the connectivity/topology is arbitrary, inverse design is called *topology optimization* [38]; this is the variant of inverse design considered in this study.

PDE-constrained inverse design, especially PDE-constrained topology optimization, is important for various subjects in computational science and engineering, e.g., optics and photonics [4,38,45], fluid dynamics [7,14,16,52], and solid mechanics [5,37]. Different numerical methods have been developed to solve PDE-constrained inverse design, and these traditional methods are commonly based on a numerical PDE solver (either approximate [45] or brute force [38]). The large-scale optimization is performed via gradient-based optimization algorithms, where the gradient is obtained using an adjoint method, which computes the gradient of the objective function with respect to all parameters at the cost of at most a single additional simulation.

Recently, deep learning in the form of deep neural networks has been used in many areas of inverse design, including nanophotonics [21, 57], mechanical materials [17], aerodynamics [53], and electromagnetism [51]. As the dominant approach nowadays for data-driven problems, deep neural networks are usually employed in the supervised-learning paradigm to learn the nonlinear mapping from arbitrary designs to their associated functional properties [17, 30, 33, 43, 44, 46, 51, 58, 63] or vice versa [33, 53, 58], i.e., neural networks are used as surrogate models of PDE solvers to accelerate the optimization. However, it may require a vast quantity of data to train a neural network for complex problems, and the generation of such datasets via "brute-force" PDE solvers could be very expensive (although many strategies are being investigated to reduce the amount of training data, e.g., active learning [44]). Alternatively, in unsupervised learning and reinforcement learning, neural networks are employed as generators to generate candidate designs [21, 22, 34, 57], which are then evaluated by numerical solvers.

Physics-informed neural networks (PINNs) have been developed as an alternative method to traditional numerical PDE solvers [35, 48]. PINNs solve PDEs by minimizing a loss function constructed from the PDEs, and the PDEs are solved when the loss is close to zero. Compared to traditional numerical solvers, a PINN is meshfree and thus can easily handle irregular-domain problems. Moreover, PINNs have been successfully employed to solve diverse *inverse problems*, including in optics [11], fluid mechanics [49, 59], systems biology [64], biomedicine [29, 50], and even inverse problems of stochastic PDEs [66] and fractional PDEs [42]. Here, the inverse problems were to infer unknown data (such as PDE coefficients/geometries) from partial observations of the PDE solutions in some regions, and a major concern is often regularizing the problem in order to ensure a unique "correct" solution. In contrast, for inverse design, one may not have any data of the PDE solution, and any realizable design that approximately maximizes the objective functional is acceptable (even if it is not unique).

For conventional numerical methods, inverse problems and inverse design might be solved by similar methods, because the PDE is directly solved via a numerical solver, and we need only find the best PDE solution to minimize the objective function or match the observed data. However, in PINN, inverse design poses new challenges. The inverse problems are solved with PINNs by using the mismatch between the PDE solution and the data as a loss function (data-based loss), so that the network is trained to minimize the sum of the data-based loss and the PDE-based loss [11,35]. Such an optimization problem can be solved relatively easily, because these two losses are consistent and can be minimized to zero simultaneously. In contrast, for inverse

design, the PDE-based loss and the objective function are usually not consistent—it is not generally possible to both exactly solve the PDE and make the design objective arbitrarily good with the same solution—and so they compete with each other during optimization. Hence, a PINN that merely optimizes the sum of the objective and the PDE loss will usually end up in an optimum that is not a solution to the PDE. Moreover, inverse design problems often have additional inequality constraints, such as from manufacturing constraints, that must be satisfied for the design to be acceptable.

To overcome these difficulties, we develop a new PINN method with hard constraints (hPINN) to solve PDE-constrained inverse design. We also consider inequality constraints in hPINN. We propose two approaches to impose the equality and inequality constraints as hard constraints, including the penalty method and the augmented Lagrangian method. We also use the approach of soft constraints for comparison. Imposing hard constraints to neural networks has been considered very recently [12, 40] for data-driven problems in the supervised learning paradigm.

This paper is organized as follows. In section 2, after introducing the setup of inverse design and the algorithm of PINN, we present the method to exactly impose Dirichlet and periodic boundary conditions by directly modifying the neural network architecture. We then propose a soft-constraint approach and two hard-constraint approaches to impose PDEs and inequality constraints in hPINN, including the penalty method and the augmented Lagrangian method. In section 3, we demonstrate the effectiveness and convergence of hPINN for a holography problem in optics and a fluid problem of Stokes flow. By comparing with traditional PDE-constrained optimization methods based on adjoint methods with the finite-difference frequency-domain (FDFD) method and finite element method (FEM) as the numerical PDE solvers, we find that hPINN achieves the same objective-function performance but often seems to obtain a smoother design (without imposing additional constraints on the design lengthscales or smoothness, as is typically done in topology optimization). Finally, we conclude the paper in section 4.

2. Methods. We first introduce the problem setup of inverse design considered in this paper and then present the method of physics-informed neural networks (PINNs) with hard constraints (hPINNs) for solving inverse design.

2.1. Inverse design. We consider a physical system governed by partial differential equations (PDEs) defined on a domain $\Omega \subset \mathbb{R}^d$:

$$\mathcal{F}\left[\mathbf{u}(\mathbf{x}); \gamma(\mathbf{x})\right] = \mathbf{0}, \quad \mathbf{x} = (x_1, x_2, \ldots, x_d) \in \Omega \tag{2.1}$$

with suitable boundary conditions (BCs):

$$\mathcal{B}\left[\mathbf{u}(\mathbf{x})\right] = 0, \quad \mathbf{x} \in \partial\Omega, \tag{2.2}$$

where $\mathcal{F}$ includes N PDE operators $\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_N\}$, $\mathcal{B}$ is a general form of a boundary-condition operator, and $\partial\Omega$ is the boundary of the domain Ω. $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}), \ldots, u_n(\mathbf{x})) \in \mathbb{R}^n$ is the solution of the PDEs and is determined by the parameter $\gamma(\mathbf{x})$, which is our quantity of interest (QoI) for the inverse design problem. (γ generally describes some structure to be manufactured in order to realize an optimized device.)

In an inverse-design problem, we search for the best γ by minimizing an objective function $\mathcal{J}$ that depends on $\mathbf{u}$ and γ. The pair $(\mathbf{u}, \gamma)$ must satisfy the equality constraints enforced by the PDEs in (2.1) and the BCs in (2.2); in certain situations, we may also have additional equality or inequality constraints for $\mathbf{u}$ and γ (e.g.,

stemming from manufacturing constraints or multiobjective problems). In this paper, we only consider inequality constraints, as equality constraints can be tackled in the same way as the PDE and BC constraints. Then the inverse design problem is formulated as a constrained optimization problem:

$$\min_{\mathbf{u},\gamma} \mathcal{J}(\mathbf{u};\gamma)$$

subject to

$$\begin{cases} \mathcal{F}[\mathbf{u};\gamma] = \mathbf{0}, \\ \mathcal{B}[\mathbf{u}] = 0, \\ h(\mathbf{u},\gamma) \le 0, \end{cases} \tag{2.3}$$

where the last equation is the inequality constraint(s). We note that the optimal solution γ of this optimization problem may not be unique, and moreover, there may be many acceptable local optima with similar performance. (An inverse problem could be viewed a special case of inverse design, where the objective function $\mathcal{J}$ is the error between the PDE solution $\mathbf{u}$ and the observed measurements, but in this context there is typically a unique "ground-truth" solution that is desired, and hence special attention must be paid to conditioning and regularization.)

2.2. Physics-informed neural networks. One difficulty of the constrained optimization problem is that $\mathbf{u}$ and γ must satisfy the PDEs, and a common strategy is for $\mathbf{u}$ to be obtained from solving the PDEs for a γ by using numerical methods such as finite differences or finite elements. In this paper, we will instead use PINNs.

In a PINN, we employ n fully connected deep neural networks $\hat{\mathbf{u}}(\mathbf{x};\boldsymbol{\theta}_u)$ to approximate the solution $\mathbf{u}(\mathbf{x})$ (Figure 1A), where $\boldsymbol{\theta}_u$ is the set of trainable parameters in the network. The network takes the coordinates $\mathbf{x}$ as the input and outputs the approximate solution $\hat{\mathbf{u}}(\mathbf{x})$. Similarly, we also employ another, independent, fully connected network $\hat{\gamma}(\mathbf{x};\boldsymbol{\theta}_\gamma)$ for the unknown parameters γ (Figure 1A). We then restrict the two networks of $\hat{\mathbf{u}}$ and $\hat{\gamma}$ to satisfy the PDEs by using a PDE-informed loss function (Figure 1A):

$$\mathcal{L}_{\mathcal{F}}(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma) = \frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N} \left|\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right]\right|^2, \tag{2.4}$$

where $\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_M\}$ are a set of M residual points in the domain Ω, and $\left|\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right]\right|$ measures the discrepancy of the ith PDE $\mathcal{F}_i[\mathbf{u};\gamma] = 0$ at the residual point $\mathbf{x}_j$.

There are multiple ways to sample the residual points, e.g., uniformly distributed random points or grid points, and in this study, we use a Sobol sequence to sample the residual points [42]. $\mathcal{F}_i$ requires the derivatives of the network output $\hat{\mathbf{u}}$ with respect to the input $\mathbf{x}$ (e.g., $\nabla\hat{\mathbf{u}}$), which are evaluated exactly and efficiently via automatic differentiation (AD; also called "backpropagation" in deep learning) without generating a mesh. In order to compute arbitrary-order derivatives, we need to use a smooth activation function; in this study, we choose the hyperbolic tangent (tanh). For more details of PINNs, we refer the reader to the review article [35].

2.3. Hard-constraint boundary conditions. We can also enforce the BCs in (2.2) via loss functions in the same way as the PDE loss in (2.4). This approach can be used for all types of BCs, including Dirichlet, Neumann, Robin, or periodic BCs.

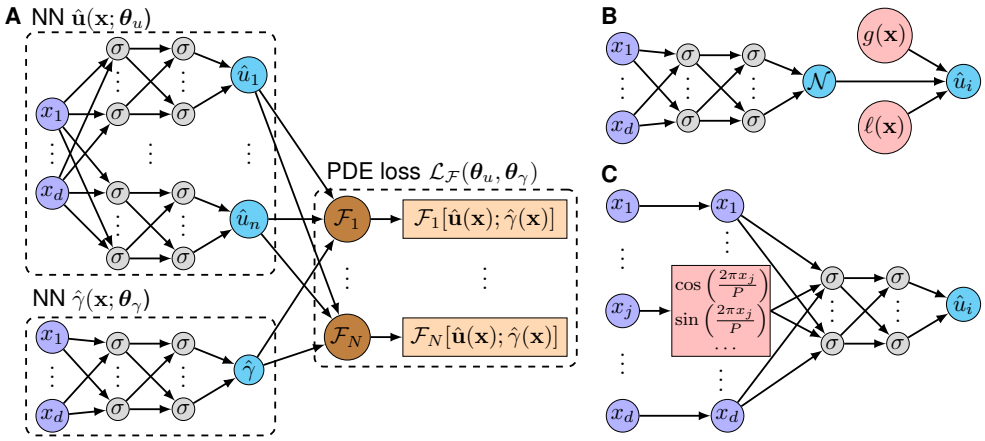

FIG. 1. Physics-informed neural networks with hard-constraint Dirichlet and periodic boundary conditions. (A) *Two independent neural networks* $\hat{\mathbf{u}}(\mathbf{x};\boldsymbol{\theta}_u)$ *and* $\hat{\gamma}(\mathbf{x};\boldsymbol{\theta}_\gamma)$ *are constructed to approximate* $\mathbf{u}(\mathbf{x})$ *and* $\gamma(\mathbf{x})$. *The gradients in the PDE-informed loss is computed via AD.* (B) *Dirichlet BCs are strictly imposed into the network architecture by modifying the network output.* (C) *Periodic BCs are strictly imposed into the network architecture by modifying the network input.*

In the examples of this study, we mainly consider Dirichlet and periodic BCs, and here we introduce another way to strictly impose the Dirichlet BCs and periodic BCs by modifying the network architecture. Compared to the approach of loss functions, this approach satisfies the BCs exactly and thus reduces the computational cost, and is also easier to implement.

Dirichlet BCs. Let us consider a Dirichlet BC for the solution u_i $(1 \le i \le n)$:

$$u_i(\mathbf{x}) = g_0(\mathbf{x}), \quad \mathbf{x} \in \Gamma_D,$$

where $\Gamma_D \subset \partial\Omega$ is a subset of the boundary. To make the approximate solution $\hat{u}_i(\mathbf{x};\boldsymbol{\theta}_u)$ satisfy this BC, we first construct a function $g(\mathbf{x})$ as a continuous extension of $g_0(\mathbf{x})$ from Γ_D to Ω. If the expression of g_0 has a simple analytic form, then it is straightforward to construct g, as shown in the numerical examples; otherwise, we can approximate g by spline functions or even train a network to represent g. Next, we construct the solution in Ω as (Figure 1B)

$$\hat{u}_i(\mathbf{x};\boldsymbol{\theta}_u) = g(\mathbf{x}) + \ell(\mathbf{x})\mathcal{N}(\mathbf{x};\boldsymbol{\theta}_u),$$

where $\mathcal{N}(\mathbf{x};\boldsymbol{\theta}_u)$ is the network output, and ℓ is a function satisfying the following two conditions:

$$\begin{cases} \ell(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_D, \\ \ell(\mathbf{x}) > 0, & \mathbf{x} \in \Omega - \Gamma_D. \end{cases}$$

If Γ_D is a simple geometry, then it is possible to choose $\ell(\mathbf{x})$ analytically [31, 32, 42]. For example, when Γ_D is the boundary of an interval $\Omega = [a, b]$, i.e., $\Gamma_D = \{a, b\}$, we can choose $\ell(x)$ as $(x-a)(b-x)$ or $(1-e^{a-x})(1-e^{x-b})$. For complex domains, it is difficult to obtain an analytical formula of $\ell(\mathbf{x})$, and we can use spline functions to approximate $\ell(\mathbf{x})$ [54].

Periodic BCs. If $u_i(\mathbf{x})$ is a periodic function with respect to x_j of the period P, then in the x_j direction, $u_i(\mathbf{x})$ can be decomposed into a weighted summation of the basis functions of the Fourier series $\{1, \cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P}), \cos(\frac{4\pi x_j}{P}), \sin(\frac{4\pi x_j}{P}), \dots\}$. Hence, we can replace the network input x_j with the Fourier basis functions to impose the periodicity in the x_j direction (Figure 1C):

$$u_i(\mathbf{x}) = \mathcal{N}\Big(x_1, \dots, x_{j-1}, \Big[\cos\Big(\frac{2\pi x_j}{P}\Big), \sin\Big(\frac{2\pi x_j}{P}\Big), \cos\Big(\frac{4\pi x_j}{P}\Big), \sin\Big(\frac{4\pi x_j}{P}\Big), \dots\Big], x_{j+1}, \dots, x_d\Big).$$

In classical Fourier analysis, many basis functions may be required to approximate an arbitrary periodic function with a good accuracy, but as demonstrated in [65], we can use as few as two terms $\{\cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P})\}$ without loss of accuracy, because all the other basis functions $\{\cos(\frac{4\pi x_j}{P}), \sin(\frac{4\pi x_j}{P}), \dots\}$ can be written as a nonlinear continuous function of $\cos(\frac{2\pi x_j}{P})$ and $\sin(\frac{2\pi x_j}{P})$ and neural networks are universal approximators of nonlinear continuous functions. Here, we consider the case where u_i and all its derivatives are periodic; this approach can also be extended to the case where its derivatives up to a finite order is periodic [13].

2.4. Soft constraints. While the BCs in (2.3) can be imposed directly during constrained optimization, it is difficult to satisfy the PDEs and inequality constraint exactly. The simplest way to deal with these constraints is to consider them as soft constraints via loss functions. Specifically, using the PDE loss in (2.4), we convert the original constrained optimization to an unconstrained optimization problem:

$$\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma) = \mathcal{J} + \mu_{\mathcal{F}} \mathcal{L}_{\mathcal{F}} + \mu_h \mathcal{L}_h, \tag{2.5}$$

where $\mathcal{L}_h$ is a quadratic penalty to measure the violation of the hard constraint $h(\mathbf{u}, \gamma) \le 0$:

$$\mathcal{L}_h(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma) = \mathbb{1}_{\{h(\hat{\mathbf{u}}, \hat{\gamma}) > 0\}} h^2(\hat{\mathbf{u}}, \hat{\gamma}), \tag{2.6}$$

and $\mu_{\mathcal{F}}$ and μ_h are the fixed penalty coefficients of the soft constraints. Then the final solution is obtained by minimizing the total loss via gradient-based optimizers:

$$\boldsymbol{\theta}_u^*, \boldsymbol{\theta}_\gamma^* = \arg\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma).$$

If $\mu_{\mathcal{F}}$ and μ_h are larger, we penalize the constraint violations more severely, thereby forcing the solutions to better satisfy the constraints. However, when the penalty coefficients are too large, the optimization problem becomes ill-conditioned and hence makes it difficult to converge to a minimum [6, 41]. On the other hand, if the penalty coefficients are too small, then the obtained solution will not satisfy the constraints and thus is not a valid solution. Therefore, although this approach is simple, it cannot be used in general. In contrast, the soft-constraint approach has worked well for inverse problems to match observed measurements [11, 49, 64], as we discussed in section 1.

2.5. Penalty method. To overcome the optimization difficulty of the soft constraints with large penalty coefficients, we consider the penalty method. Unlike the approach of soft constraints, which converts a constrained optimization problem to an

unconstrained optimization problem with fixed coefficients, a penalty method replaces the constrained optimization problem by a *sequence* of unconstrained problems with varying coefficients. The unconstrained problem in the kth "outer" iteration is

$$\min_{\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma} \mathcal{L}^k(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma) = \mathcal{J} + \mu_{\mathcal{F}}^k \mathcal{L}_{\mathcal{F}} + \mu_h^k \mathcal{L}_h,$$

where $\mu_{\mathcal{F}}^k$ and μ_h^k are the penalty coefficients in the kth iteration. In each iteration, we increase the penalty coefficients by constant factors $\beta_{\mathcal{F}} > 1$ and $\beta_h > 1$:

$$\mu_{\mathcal{F}}^{k+1} = \beta_{\mathcal{F}} \mu_{\mathcal{F}}^k, \quad \mu_h^{k+1} = \beta_h \mu_h^k.$$

Here, we need to choose the initial coefficients $\mu_{\mathcal{F}}^0$ and μ_h^0 and the factors $\beta_{\mathcal{F}}$ and β_h. The choice of these hyperparameters is problem dependent [6]. Similar to the soft-constraint approach, we would have an ill-conditioned optimization problem when their values are large, leading to slow convergence. However, if their values are small, we may need many outer iterations, and gradient-descent optimization may get stuck at poor local minima, as we will show in our numerical experiments. Algorithm 2.1 presents the pseudocode for the penalty method.

Algorithm 2.1 hPINNs via the penalty method.

Hyperparameters: initial penalty coefficients $\mu_{\mathcal{F}}^0$ and μ_h^0, factors $\beta_{\mathcal{F}}$ and β_h
$k \longleftarrow 0$
$\boldsymbol{\theta}_u^0, \boldsymbol{\theta}_\gamma^0 \longleftarrow \arg\min_{\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma} \mathcal{L}^0(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x};\boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x};\boldsymbol{\theta}_\gamma)$ from random initialization, until the training loss is converged
repeat
  $k \longleftarrow k+1$
  $\mu_{\mathcal{F}}^k \longleftarrow \beta_{\mathcal{F}} \mu_{\mathcal{F}}^{k-1}$
  $\mu_h^k \longleftarrow \beta_h \mu_h^{k-1}$
  $\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k \longleftarrow \arg\min_{\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma} \mathcal{L}^k(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x};\boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x};\boldsymbol{\theta}_\gamma)$ from the initialization of $\boldsymbol{\theta}_u^{k-1}$ and $\boldsymbol{\theta}_\gamma^{k-1}$, until the training loss is converged
until $\mathcal{L}_{\mathcal{F}}(\boldsymbol{\theta}_u^k,\boldsymbol{\theta}_\gamma^k)$ and $\mathcal{L}_h(\boldsymbol{\theta}_u^k,\boldsymbol{\theta}_\gamma^k)$ are smaller than a tolerance

As $k \to \infty$, given that the networks are well trained, the solutions of the successive unconstrained optimization problems will converge to the solution of the original constrained optimization problem [6, 41]. In the first iteration ($k = 0$), the neural networks are trained from a random initialization, while the neural networks in the $(k+1)$th iteration are trained by using the solution of the kth iteration as the initialization—this good starting point helps to counteract the slow convergence that would otherwise arise when $\mu_{\mathcal{F}}^k$ and μ_h^k are large.

2.6. Augmented Lagrangian method. The third method we considered in this study is the method of multipliers or the augmented Lagrangian method [60]. Similar to penalty methods, the augmented Lagrangian method also uses penalty terms, but it adds new terms designed to mimic Lagrange multipliers. The uncon-

strained problem in the kth iteration is

$$\begin{aligned}\min_{\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma} \mathcal{L}^k(\boldsymbol{\theta}_u,\boldsymbol{\theta}_\gamma) = \quad & \mathcal{J} \\ & + \mu_{\mathcal{F}}^k \mathcal{L}_{\mathcal{F}} \\ & + \mu_h^k \mathbb{1}_{\{h>0 \vee \lambda_h^k>0\}} h^2 \\ & + \tfrac{1}{MN}\textstyle\sum_{j=1}^{M}\sum_{i=1}^{N} \lambda_{i,j}^k \mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right] \\ & + \lambda_h^k h,\end{aligned} \tag{2.7}$$

where the symbol "$\vee$" in the third term is the logical OR, and $\lambda_{i,j}^k$ and λ_h^k are multipliers. We note that the second penalty term $\mathcal{L}_{\mathcal{F}}$ is the same as the term in (2.4) in the soft constraints and penalty method, but the third penalty term of h depends on the multiplier λ_h^k, and is slightly different from the penalty term $\mathcal{L}_h$ in (2.6).

The last two terms in (2.7) are Lagrangian terms, and we take the term $\mathcal{F}_i\big[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\big]$ as an example to demonstrate its effect and how to choose $\lambda_{i,j}^k$. It is clear that the gradient $\nabla\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right]$ is always orthogonal to the constraint of $\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right]$. In the kth iteration, we choose $\lambda_{i,j}^k$ to generate exactly the gradient that was previously generated in the $(k-1)$th iteration by the penalty term $\left|\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j);\hat{\gamma}(\mathbf{x}_j)\right]\right|^2$ in $\mathcal{L}_{\mathcal{F}}$ [6, 41, 60], i.e., we require that

$$\begin{aligned}&\lambda_{i,j}^k \nabla\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j;\boldsymbol{\theta}_u^{k-1});\hat{\gamma}(\mathbf{x}_j;\boldsymbol{\theta}_\gamma^{k-1})\right] \\ &= \mu_{\mathcal{F}}^{k-1}\nabla\left|\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j;\boldsymbol{\theta}_u^{k-1});\hat{\gamma}(\mathbf{x}_j;\boldsymbol{\theta}_\gamma^{k-1})\right]\right|^2 + \lambda_{i,j}^{k-1}\nabla\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j;\boldsymbol{\theta}_u^{k-1});\hat{\gamma}(\mathbf{x}_j;\boldsymbol{\theta}_\gamma^{k-1})\right],\end{aligned}$$

and thus we have

$$\lambda_{i,j}^k = \lambda_{i,j}^{k-1} + 2\mu_{\mathcal{F}}^{k-1}\mathcal{F}_i\left[\hat{\mathbf{u}}(\mathbf{x}_j;\boldsymbol{\theta}_u^{k-1});\hat{\gamma}(\mathbf{x}_j;\boldsymbol{\theta}_\gamma^{k-1})\right].$$

Similarly, for λ_h^k, we have [60]

$$\lambda_h^k = \max\left(\lambda_h^{k-1} + 2\mu_h^{k-1} h\left(\hat{\mathbf{u}}(\mathbf{x};\boldsymbol{\theta}_u^{k-1}),\hat{\gamma}(\mathbf{x};\boldsymbol{\theta}_\gamma^{k-1})\right),0\right).$$

The initial values of all the multipliers are chosen as 0. We will show in our numerical examples that $\lambda_{i,j}^k/\mu_{\mathcal{F}}^k$ and λ_h^k/μ_h^k converge after several iterations. The pseudocode is presented in Algorithm 2.2. Compared to the penalty method, the augmented Lagrangian method has two main advantages: (1) it is not necessary to increase $\mu_{\mathcal{F}}$ and μ_h to infinity in order to induce convergence to a feasible solution, which further avoids the ill-conditioning; (2) the convergence rate is considerably better than that of the penalty method.

We note that the proposed hPINNs use an idea similar to full-space methods for PDE-constrained optimization [19], where the objective is optimized and the PDE is solved simultaneously. In this sense, hPINN is a type of full-space method, but in hPINN we compute the PDE residual using neural networks instead of traditional numerical methods.

3. Results. We will apply our proposed hPINNs to solve two different problems of inverse design in optics and fluids. We compare hPINNs with a few traditional PDE-constrained optimization methods based on the adjoint method and a numerical PDE solver, and demonstrate the capacity and effectiveness of hPINNs. All the hPINN codes in this study are implemented by using the library DeepXDE [35], and will be deposited in GitHub at https://github.com/lululxvi/hpinn.

Algorithm 2.2 hPINNs via the augmented Lagrangian method.

Hyperparameters: initial penalty coefficients $\mu_{\mathcal{F}}^0$ and μ_h^0, factors $\beta_{\mathcal{F}}$ and β_h
$k \longleftarrow 0$
$\lambda_{i,j}^0 \longleftarrow 0$ for $1 \le i \le N$, $1 \le j \le M$
$\lambda_h^0 \longleftarrow 0$
$\boldsymbol{\theta}_u^0, \boldsymbol{\theta}_\gamma^0 \longleftarrow \arg\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}^0(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma)$ from random initialization, until the training loss is converged
repeat
  $k \longleftarrow k + 1$
  $\mu_{\mathcal{F}}^k \longleftarrow \beta_{\mathcal{F}} \mu_{\mathcal{F}}^{k-1}$
  $\mu_h^k \longleftarrow \beta_h \mu_h^{k-1}$
  $\lambda_{i,j}^k \longleftarrow \lambda_{i,j}^{k-1} + 2\mu_{\mathcal{F}}^{k-1} \mathcal{F}_i \left[\hat{\mathbf{u}}(\mathbf{x}_j; \boldsymbol{\theta}_u^{k-1}); \hat{\gamma}(\mathbf{x}_j; \boldsymbol{\theta}_\gamma^{k-1})\right]$ for $1 \le i \le N$, $1 \le j \le M$
  $\lambda_h^k \longleftarrow \max\left(\lambda_h^{k-1} + 2\mu_h^{k-1} h\left(\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u^{k-1}), \hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma^{k-1})\right), 0\right)$
  $\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k \longleftarrow \arg\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}^k(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma)$ from the initialization of $\boldsymbol{\theta}_u^{k-1}$ and $\boldsymbol{\theta}_\gamma^{k-1}$, until the training loss is converged
until $\mathcal{L}_{\mathcal{F}}(\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k)$ and $\mathcal{L}_h(\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k)$ are smaller than a tolerance

3.1. Holography. We first consider the challenging problem of holography for an image in depth (in the direction of propagation). In contrast to holograms of images that are parallel to the scattering surface, and which can be designed via algorithms relying on Fourier optics [15], in-depth holograms require us to use inverse design on the full Maxwell"s equations. We propose designing the permittivity map of a scattering slab that scatters light so that the transmitted intensity has a targeted shape.

3.1.1. Problem setup. We consider a holography problem defined on a rectangular domain $\Omega = [-2, 2] \times [-2, 3]$ (Figure 2A). In the lower part of Ω (i.e., Ω_1), we have a time-harmonic current J to generate an electromagnetic wave $E(x, y) = \Re[E] + i\Im[E]$ in the whole space. There is a lens in the region Ω_2 (the blue region), whose permittivity function $\varepsilon(x, y)$ (the square of the refractive index) is to be designed to produce our target transmitted-wave pattern $f(x, y)$ in the top region $\Omega_3 = [-2, 2] \times [0, 3]$, while the permittivity in other region is one. Specifically, our objective function in this problem is

$$\mathcal{J}(E) = \frac{1}{\text{Area}(\Omega_3)} \| |E(x,y)|^2 - f(x,y) \|_{2,\Omega_3}^2 = \frac{1}{\text{Area}(\Omega_3)} \int_{\Omega_3} \left(|E(x,y)|^2 - f(x,y)\right)^2 dxdy, \tag{3.1}$$

where $|E|^2 = (\Re[E])^2 + (\Im[E])^2$ is the square of the magnitude of the electric field. In this study, we choose the target function as

$$f(x,y) = \begin{cases} 1, & (x,y) \in [-0.5, 0.5] \times [1, 2], \\ 0 & \text{otherwise}, \end{cases}$$

i.e., f is equal to 1 inside the black square in Figure 2A and 0 otherwise. We can think of the target as an input defined by a user who does not need to know Helmholtz's equation. Still in that case, our inverse design tool should find the geometry that scatters the transmitted intensity in a shape as close to the user input as possible. In

the case of our example, the target function is not a solution of Helmholtz's equation (equivalent to Maxwell's equations in two dimensions). So, we expect the neural network to have to make a trade-off between reaching the target shape and making sure that Helmholtz's equations are accurately solved.

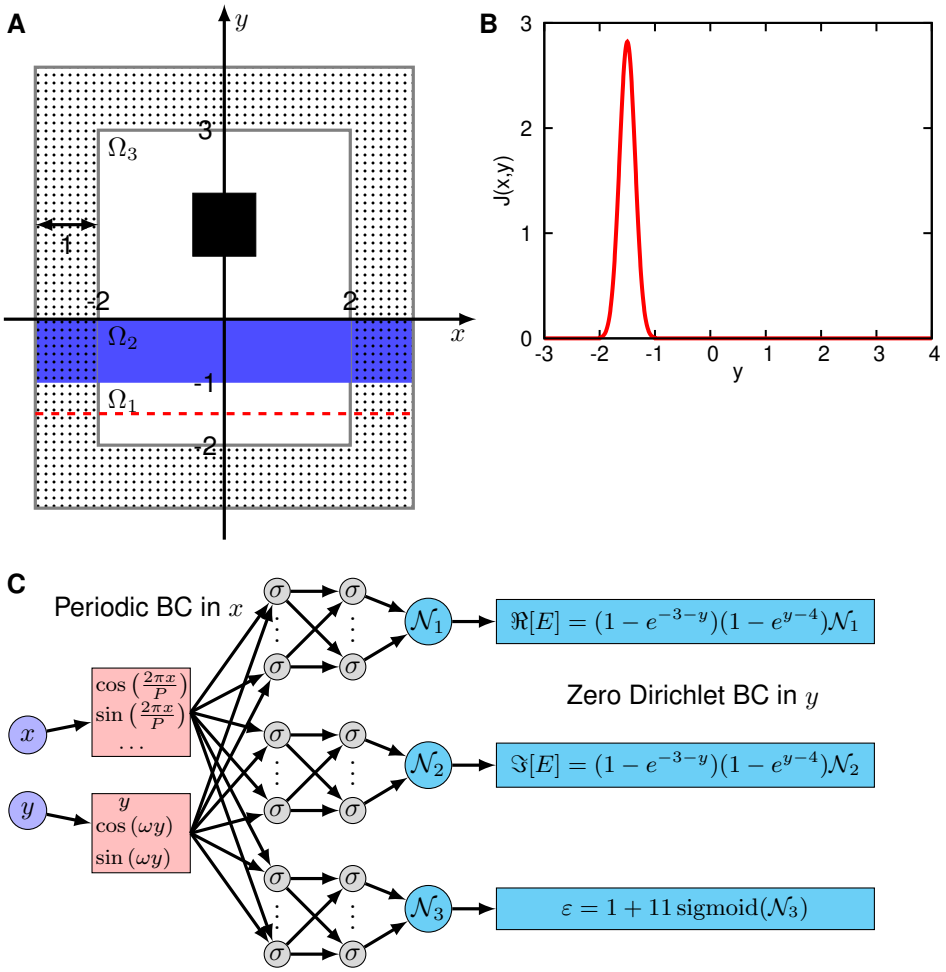

FIG. 2. Holography problem setup and the neural network architecture. (A) *The whole computational domain includes the main domain* $\Omega = [-2, 2] \times [-2, 3]$ *and a PML of depth one in the shaded region. The design region for the permittivity* ε *is in blue, and the center of the current* J *is the dashed red line in the domain* Ω_1*. The target electrical field is defined in* Ω_3*.* (B) *The time-harmonic current* J*.* (C) *The architecture of the hPINN with the Dirichlet and periodic BCs embedded directly in the network. The network inputs are* x *and* y*, and the outputs are* $\Re[E]$*,* $\Im[E]$*, and* ε*.*

The holography problem can be described by the following PDEs:

$$\nabla^2 E + \varepsilon\omega^2 E = -i\omega J, \tag{3.2}$$

where the frequency ω is chosen as 2π (corresponding to a wavelength 1 in the $\varepsilon = 1$ region), and the electric current source J is chosen as a Gaussian profile in y (Figure 2B) and a constant in x, which generates an incident planewave propagating in

the y direction:

$$J(x,y) = \frac{1}{h\sqrt{\pi}} e^{-(\frac{y+1.5}{h})^2} \mathbb{1}_{[-1,-2]}(y),$$

where $h = 0.2$ and $\mathbb{1}_{[-1,-2]}$ is the indicator function that truncates J to be supported in a finite-width strip $y \in [-1,-2]$ (centered on the dashed-red line in Figure 2A). This problem is formally defined in an infinitely large domain with outgoing (radiation) boundary conditions. To reduce the problem to a finite domain for computation, we apply the technique of perfectly matched layers (PMLs) [24], which works as an artificial absorbing layer (the black-dotted region in Figure 2A) to truncate the computational domain. After applying the PML, the PDE in (3.2) becomes

(3.3)
$$\frac{1}{1+i\frac{\sigma_x(x)}{\omega}}\frac{\partial}{\partial x}\left(\frac{1}{1+i\frac{\sigma_x(x)}{\omega}}\frac{\partial E}{\partial x}\right) + \frac{1}{1+i\frac{\sigma_y(y)}{\omega}}\frac{\partial}{\partial y}\left(\frac{1}{1+i\frac{\sigma_y(y)}{\omega}}\frac{\partial E}{\partial y}\right) + \varepsilon\omega^2 E = -i\omega J,$$

where

$$\sigma_x(x) = \sigma_0(-2-x)^2\mathbb{1}_{(-\infty,-2)}(x) + \sigma_0(x-2)^2\mathbb{1}_{(2,\infty)}(x),$$
$$\sigma_y(y) = \sigma_0(-2-y)^2\mathbb{1}_{(-\infty,-2)}(y) + \sigma_0(y-3)^2\mathbb{1}_{(3,\infty)}(y),$$

with $\sigma_0 = -\ln 10^{-20}/(4d^3/3) \gg 1$ where $d = 1$ is the depth of the PML layer. For BCs of the PML, we use a periodic BC for the x direction ($x = -3$ and $x = 3$) and a zero Dirichlet BC for the y direction ($y = -3$ and $y = 4$). Corresponding to semiconductor dielectric materials commonly used in infrared optics, we require $\varepsilon \in [1, 12]$ in the design region.

3.1.2. hPINN. We will apply hPINN to solve this inverse design problem of holography. The objective function is defined in (3.1) and is approximated by Monte-Carlo integration. The PDEs are the real and imaginary parts of (3.3) (see section SM1 of the supplementary material), i.e., $N = 2$, and the PDE-informed loss function is

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{2M}\sum_{j=1}^{M}\left(\Re[\mathcal{F}[\mathbf{x}_j]]\right)^2 + \left(\Im[\mathcal{F}[\mathbf{x}_j]]\right)^2, \tag{3.4}$$

where

$$\mathcal{F}[\mathbf{x}_j] = \frac{1}{\omega + i\sigma_x(x)}\frac{\partial}{\partial x}\left(\frac{1}{1+i\frac{\sigma_x(x)}{\omega}}\frac{\partial E}{\partial x}\right) + \frac{1}{\omega + i\sigma_y(y)}\frac{\partial}{\partial y}\left(\frac{1}{1+i\frac{\sigma_y(y)}{\omega}}\frac{\partial E}{\partial y}\right)\varepsilon\omega E + iJ\Big|_{\mathbf{x}_j}.$$

Here, we scaled the PDE in (3.3) by $\frac{1}{\omega}$ to make $\mathcal{F}$ of order one. In this problem, we do not have any inequality constraint.

We construct three networks to approximate $\Re[E]$, $\Im[E]$, and ε (Figure 2C), respectively. The periodic and Dirichlet BCs are imposed directly into the network. The restriction of $\varepsilon \in [1, 12]$ is satisfied by using the transformation $\varepsilon(x,y) = 1 + 11\,\mathrm{sigmoid}(\mathcal{N}_3(x,y))$, where $\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$ and $\mathcal{N}_3(x,y)$ is a network output. In addition, as demonstrated in [64], it is usually beneficial for the network training to add extra features, which may have a similar pattern of the solution (even if they

are not accurate), to the network input. In our study, we use the two extra features $\cos(\omega y)$ and $\sin(\omega y)$, because we expect the current J to generate an incident plane wave with the frequency ω in the y direction (in addition to scattered waves from $\varepsilon \neq 1$ regions).

3.1.3. Hyperparameters and verification on a forward problem. To verify that hPINN is capable of solving the holography problem, we first solve a forward problem, where $\varepsilon = 1$ is given and we only optimize the network to minimize the loss $\mathcal{L}_{\mathcal{F}}$ in (3.4). We choose $M = 17000$ such that the average spacing between two adjacent random points is ~ 0.05. Each network in Figure 2C has 5 layers with 32 neurons per layer, and we use 4 Fourier basis functions $\{\cos\left(\frac{2\pi x}{P}\right), \sin\left(\frac{2\pi x}{P}\right), \cos\left(\frac{4\pi x}{P}\right), \sin\left(\frac{4\pi x}{P}\right)\}$. To train the networks, we first use the Adam optimizer [28] with the learning rate 1×10^{-3} for 2×10^4 steps, and then switch to L-BFGS [8] until the loss is converged.

After the network training, the solution of hPINN (Figures 3A and B) is consistent with the reference solution obtained from the finite-difference frequency-domain (FDFD) method [9] with the spatial resolutions $\Delta x = 0.01$ (Figures 3C and D). We also show that the pointwise PDE-informed loss $\mathcal{F}[\mathbf{x}]$ is very small (between -5×10^{-3} and 5×10^{-3}; Figures 3E and F). During the training process, the loss function decreases (Figure 3G), while the L^2 relative error computed using the reference solution also decreases (Figure 3H). There is a clear correlation between the training loss and the L^2 relative error (Figure 3I), and when the training loss is $\lesssim 10^{-4}$, the L^2 relative error is $< 1\%$. This criterion is useful for the following reverse design problem, because we can easily check the accuracy of the hPINN solution during the training process by monitoring the loss $\mathcal{L}_{\mathcal{F}}$ directly without comparing to the FDFD reference.

3.1.4. Soft constraints. We first use the approach of soft constraints to solve the holography inverse-design problem by minimizing

$$\mathcal{L} = \mathcal{J} + \mu_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}.$$

The permittivity function ε is randomly initialized, and three random examples are shown in Figure 4A. The objective value for a random permittivity function is ~ 0.1. As we discussed in section 2.4, the choice of $\mu_{\mathcal{F}}$ plays an important role in the final design, and we will compare the performance of different values of $\mu_{\mathcal{F}}$ using the first case in Figure 4A as the initial permittivity.

We use the same hyperparameters and training procedure as we used in the forward problem. After the network training, the third network is the permittivity function ε. However, we will show that the electric field E of the first and second networks is not accurate enough, and thus we do not use the electric field of the network to compute the objective $\mathcal{J}$. Instead, we use FDFD to simulate the correct electric field for the obtained permittivity, and then compute the objective. We show that when $\mu_{\mathcal{F}}$ is too large or too small, the final objective is relatively large, and the smallest objective is obtained when $\mu_{\mathcal{F}} \approx 2$ (Figure 4B).

When $\mu_{\mathcal{F}}$ is small, e.g., 0.1, the PDE loss cannot be optimized well ($\mathcal{L}_{\mathcal{F}} \sim 10^{-1}$; Figure 4C, bottom left), and thus the PDEs are not satisfied well. If we use the obtained permittivity function (Figure 4C, top left) to simulate the corresponding electric field $|E|^2$ via FDFD (Figure 4C, right), this is very different from the $|E|^2$ obtained from hPINN (Figure 4C, center). On the other hand, when $\mu_{\mathcal{F}}$ is large, e.g., 10, the final PDE loss is $\sim 10^{-5}$, and the fields of $|E|^2$ from hPINN and FDFD are almost identical (Figure 4D), i.e., the PDE constraints are satisfied very well.

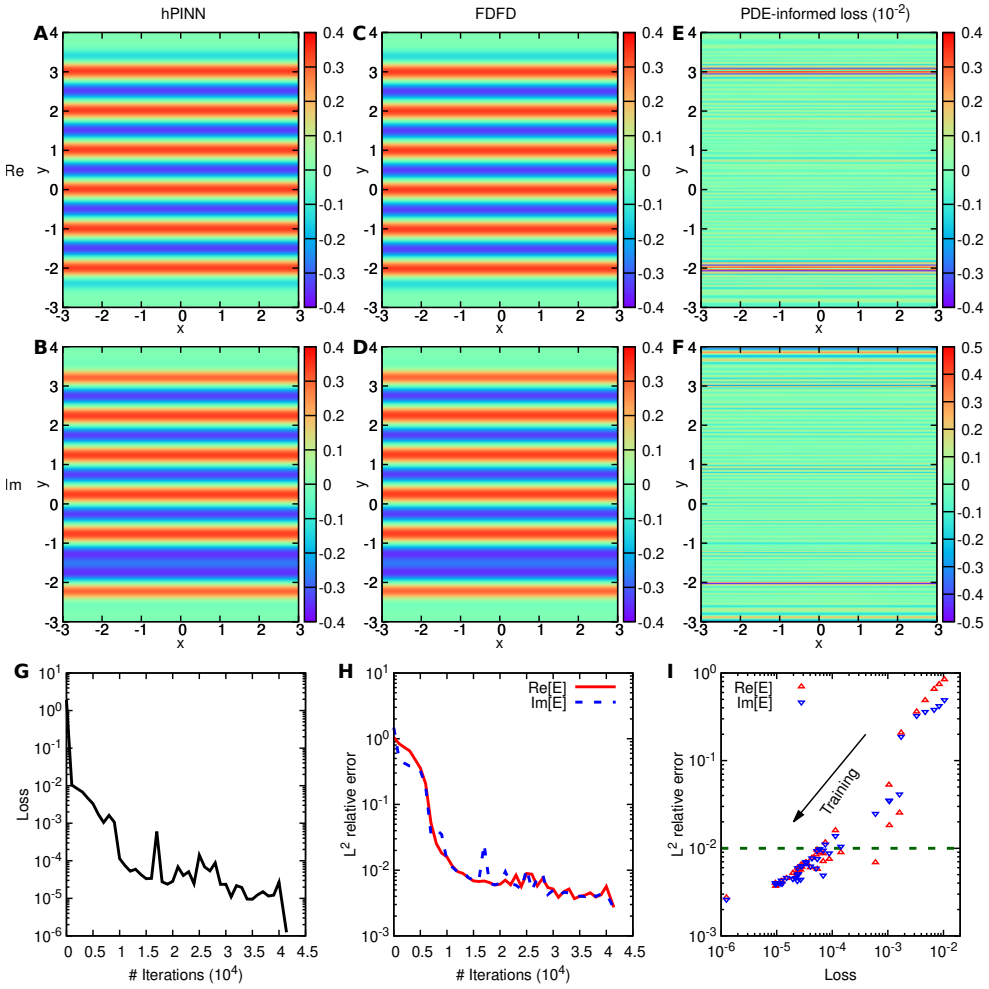

FIG. 3. hPINN for solving a forward problem. (A) *and* (B) *The* (A) *real part* $\Re[E]$ *and* (B) *imaginary part* $\Im[E]$ *of the hPINN solution.* (C) *and* (D) *The* (C) *real part* $\Re[E]$ *and* (D) *imaginary part* $\Im[E]$ *of the FDFD solution.* (E) *and* (F) *The* (E) *real part* $\Re[\mathcal{F}[\mathbf{x}]]$ *and* (F) *imaginary part* $\Im[\mathcal{F}[\mathbf{x}]]$ *of the PDE-informed loss in* (3.4). (G) *and* (H) *The* (G) *training loss* $\mathcal{L}_{\mathcal{F}}$ *and* (H) L^2 *relative error versus the number of optimization iterations.* (I) *The correlation between the training loss and the* L^2 *relative error during the training process.*

However, the permittivity function is not meaningful and the objective is very large (Figure 4D), because of the ill-conditioning of the optimization. The case of the smallest objective (≈ 0.0547) with $\mu_{\mathcal{F}} = 2$ is shown in Figure 4E, but the prediction of $|E|^2$ is still of low accuracy. Therefore, the soft constraint approach cannot satisfy the PDE constraints for small $\mu_{\mathcal{F}}$, and thus a large $\mu_{\mathcal{F}}$ is required. However, for large $\mu_{\mathcal{F}}$, the optimization failed to make progress on the design objective due to the ill-conditioning.

3.1.5. Penalty method. To address the issue in the soft constraint approach, we gradually increase the value of $\mu_{\mathcal{F}}$ by using the penalty method in Algorithm 2.1. The initial value of $\mu_{\mathcal{F}}$ is $\mu_{\mathcal{F}}^0$ and the increasing factor is $\beta_{\mathcal{F}}$, i.e., $\mu_{\mathcal{F}}^k = (\beta_{\mathcal{F}})^k \mu_{\mathcal{F}}^0$.

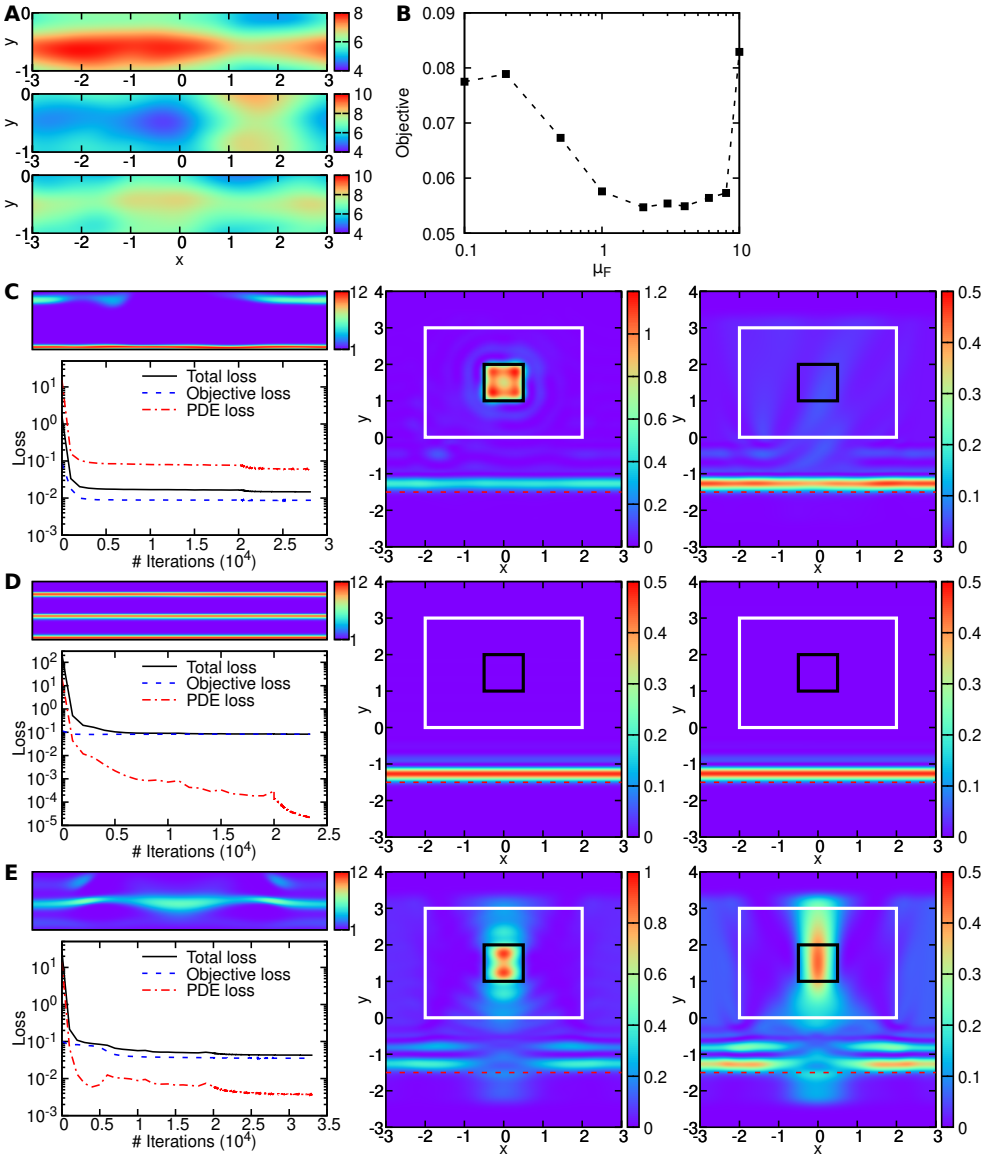

FIG. 4. hPINN for the inverse design of holography via the approach of soft constraints. (A) *Three examples of random initialization of* ε. (B) *The objective* $\mathcal{J}$ *of different designs obtained from hPINNs with different* $\mu_{\mathcal{F}}$. (C) *Results of* $\mu_{\mathcal{F}} = 0.1$*: (top left) the final permittivity* ε*, (bottom left) training trajectory, (center) hPINN solution* $|E|^2$*, and (right) the reference* $|E|^2$ *solved by FDFD for the final permittivity.* (D) *Results of* $\mu_{\mathcal{F}} = 10$. (E) *Results of* $\mu_{\mathcal{F}} = 2$.

We need to tune $\mu_{\mathcal{F}}^0$ and $\beta_{\mathcal{F}}$ in the penalty method. As an example, we first show the results of the case $\mu_{\mathcal{F}}^0 = 2$ and $\beta_{\mathcal{F}} = 2$ (Figures 5A–D). When we increase the value of $\mu_{\mathcal{F}}$, the PDE loss decreases (Figure 5A), and we find that in order to make $\mathcal{L}_{\mathcal{F}} < 10^{-4}$, we need to increase $\mu_{\mathcal{F}}$ until $\mu_{\mathcal{F}} > 100$. After we train the network with $\mu_{\mathcal{F}}^k$, the objective first decreases and then increases (Figure 5B). In the soft-constraint

approach, when $\mu_{\mathcal{F}} = 10$, the network would get stuck at a poor local minimum due to the ill-conditioning, but here, even if $\mu_{\mathcal{F}} > 100$, we still obtain a meaningful result and objective. However, the objective still becomes worse when $\mu_{\mathcal{F}}^k > 10$, and thus the ill-conditioning problem still exists, albeit weaker than before. The smallest objective ~ 0.0537 is obtained at $k = 1$, which is $\sim 2\%$ better than the approach of soft constraints. The design for $k = 1$ is in Figure 5C, and the corresponding $|E|^2$ obtained from FDFD is in Figure 5D. In terms of the computational cost, although the penalty method requires seven rounds of training, the total number of optimization iterations is $\sim 5.6 \times 10^4$, which is only 70% more than that of the soft-constraint approach with $\mu_{\mathcal{F}} = 2$ ($\sim 3.3 \times 10^4$), because as shown in Algorithm 2.1, for the kth training, we use the $(k-1)$th solution as the initial guess and thus the network can be trained much faster than the first round of training which is initialized randomly.

Next, we investigate the effects of $\mu_{\mathcal{F}}^0$ and $\beta_{\mathcal{F}}$. We choose $\mu_{\mathcal{F}}^0 = 1$ and compare different values of β_F. For different β_F, we increase $\mu_{\mathcal{F}}$ until $\mu_{\mathcal{F}} > 100$, and the objective has a similar behavior, i.e., decreasing first and then increasing (Figure 5E). Among these β_F, the smallest objective is obtained from $\beta_{\mathcal{F}} = 2$. When $\beta_{\mathcal{F}}$ increases, we need fewer rounds of training and thus the computational cost decreases, but if $\beta_{\mathcal{F}}$ is very large, the computational cost increases again because in each round of training we need more iterations (Figure 5F). We also investigate the effects of $\mu_{\mathcal{F}}^0$ by fixing $\beta_{\mathcal{F}} = 2$, and the smallest objective is obtained by using $\mu_{\mathcal{F}}^0 = 2$. Therefore, the combination of $\mu_{\mathcal{F}}^0 = 2$ and $\beta_{\mathcal{F}} = 2$ we presented is almost the best case in the problem.

3.1.6. Augmented Lagrangian method. In the penalty method, the objective eventually worsens when $\mu_{\mathcal{F}}^k$ is large due to the ill-conditioning. To overcome this difficulty of convergence, we employ the augmented Lagrangian method in Algorithm 2.2. The (2.7) in this case becomes

$$\mathcal{L}^k = \mathcal{J} + \mu_{\mathcal{F}}^k \mathcal{L}_{\mathcal{F}} + \frac{1}{2M} \sum_{j=1}^{M} \left(\lambda_{\Re,j}^k \Re\left[\mathcal{F}[\mathbf{x}_j]\right] + \lambda_{\Im,j}^k \Im\left[\mathcal{F}[\mathbf{x}_j]\right] \right)$$

and

$$\lambda_{\Re,j}^k = \lambda_{\Re,j}^{k-1} + 2\mu_{\mathcal{F}}^{k-1} \Re\left[\mathcal{F}[\mathbf{x}_j]\right], \quad \lambda_{\Im,j}^k = \lambda_{\Im,j}^{k-1} + 2\mu_{\mathcal{F}}^{k-1} \Im\left[\mathcal{F}[\mathbf{x}_j]\right]. \tag{3.5}$$

We use the same hyperparameters as the penalty method, i.e., $\mu_{\mathcal{F}}^0 = 2$ and $\beta_{\mathcal{F}} = 2$. After the training, the PDE loss is below 10^{-4} (Figure 6A), and the L^2 relative error of $|E|^2$ between hPINN and FDFD for the final ε is 1.2%, so the solution of hPINN satisfies the PDEs very well. The final value of the objective function from hPINN also converges. The objective does not worsen even when $\mu_{\mathcal{F}}^k > 10^3$ (Figure 6B), and thus the augmented Lagrangian method solves the convergence issue in the penalty method.

We next investigate the distribution and evolution of the multipliers. Because we choose $\lambda_{\Re,j}^0 = \lambda_{\Im,j}^0 = 0$ in (3.5), then $\lambda_{\Re,j}^1$ and $\lambda_{\Im,j}^1$ are proportional to the pointwise PDE loss $\mathcal{F}[\mathbf{x}_j]$. After the first round of training, the points around the region $[-0.5, 0.5] \times [1, 2]$ (i.e., the black square in Figure 2A) have the largest values of $\lambda_{\Re,j}^1$ and $\lambda_{\Im,j}^1$ (Figure 6C, left) and thus the largest PDE error. Hence, the points with larger PDE errors would have larger weights for the next round of training, i.e., the multipliers automatically tune the relative weights of different training points. The distributions of $\lambda_{\Re,j}^k$ and $\lambda_{\Im,j}^k$ become more uniform for a larger k (Figure 6C). Moreover, for each single point, $\lambda_{\Re,j}^k$ and $\lambda_{\Im,j}^k$ increase with k, but the normalized

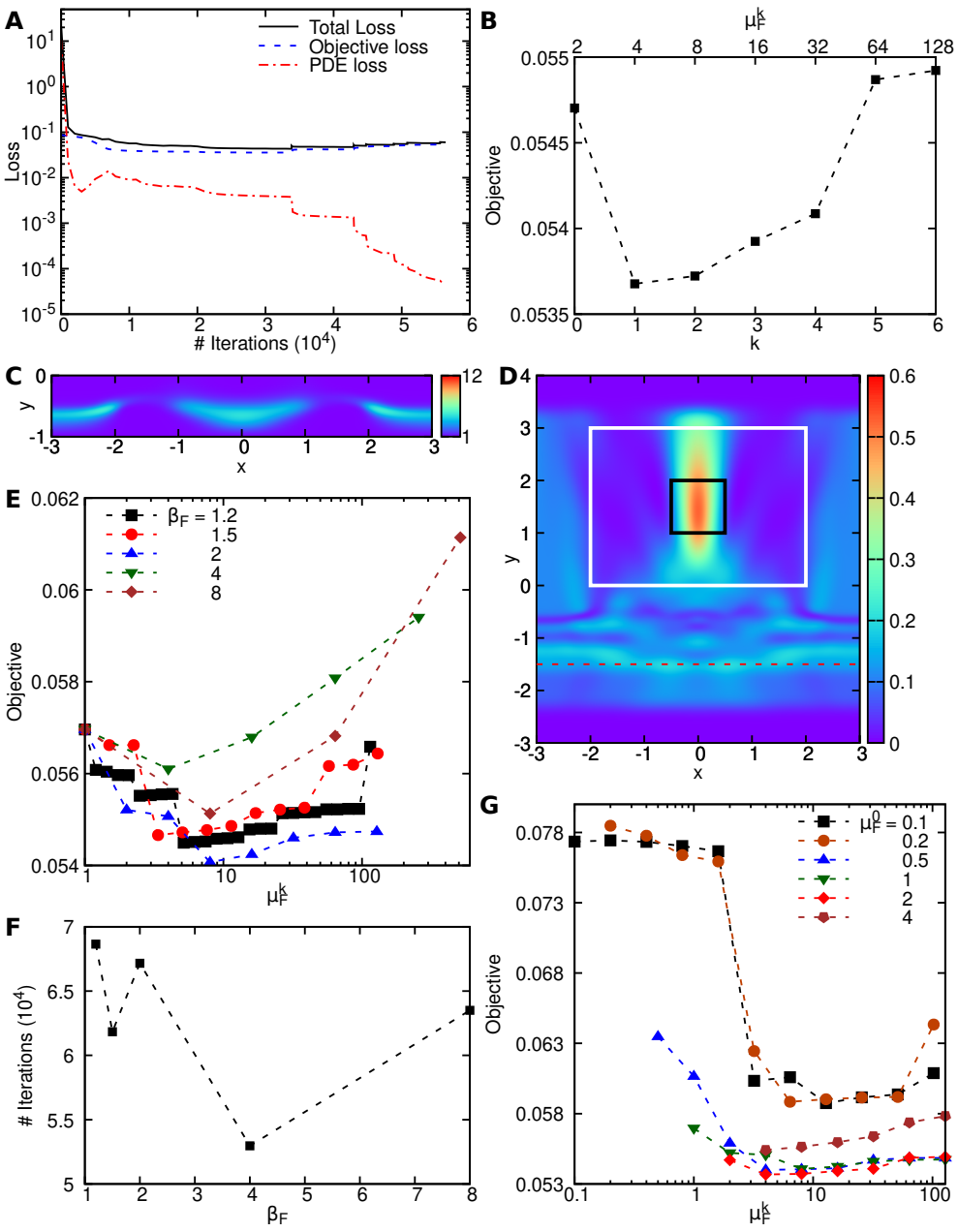

FIG. 5. hPINN for the inverse design of holography via the penalty method. (A) *to* (D) *The results of* $\mu_{\mathcal{F}}^0 = 2$ *and* $\beta_{\mathcal{F}} = 2$. (A) *The losses versus the number of optimization iterations.* (B) *the objective value after the network is trained with* $\mu_{\mathcal{F}}^k$, (C) *the optimized permittivity function* ε *at* $k = 1$, *and* (D) *the corresponding electric field* $|E|^2$. (E) *The objective value versus* $\mu_{\mathcal{F}}^k$ *when using different values of* $\beta_{\mathcal{F}}$. (F) *The total number of optimization iterations (which is proportional to the computational cost) for different* $\beta_{\mathcal{F}}$. (G) *The objective value versus* $\mu_{\mathcal{F}}^k$ *for different values of* $\mu_{\mathcal{F}}^0$.

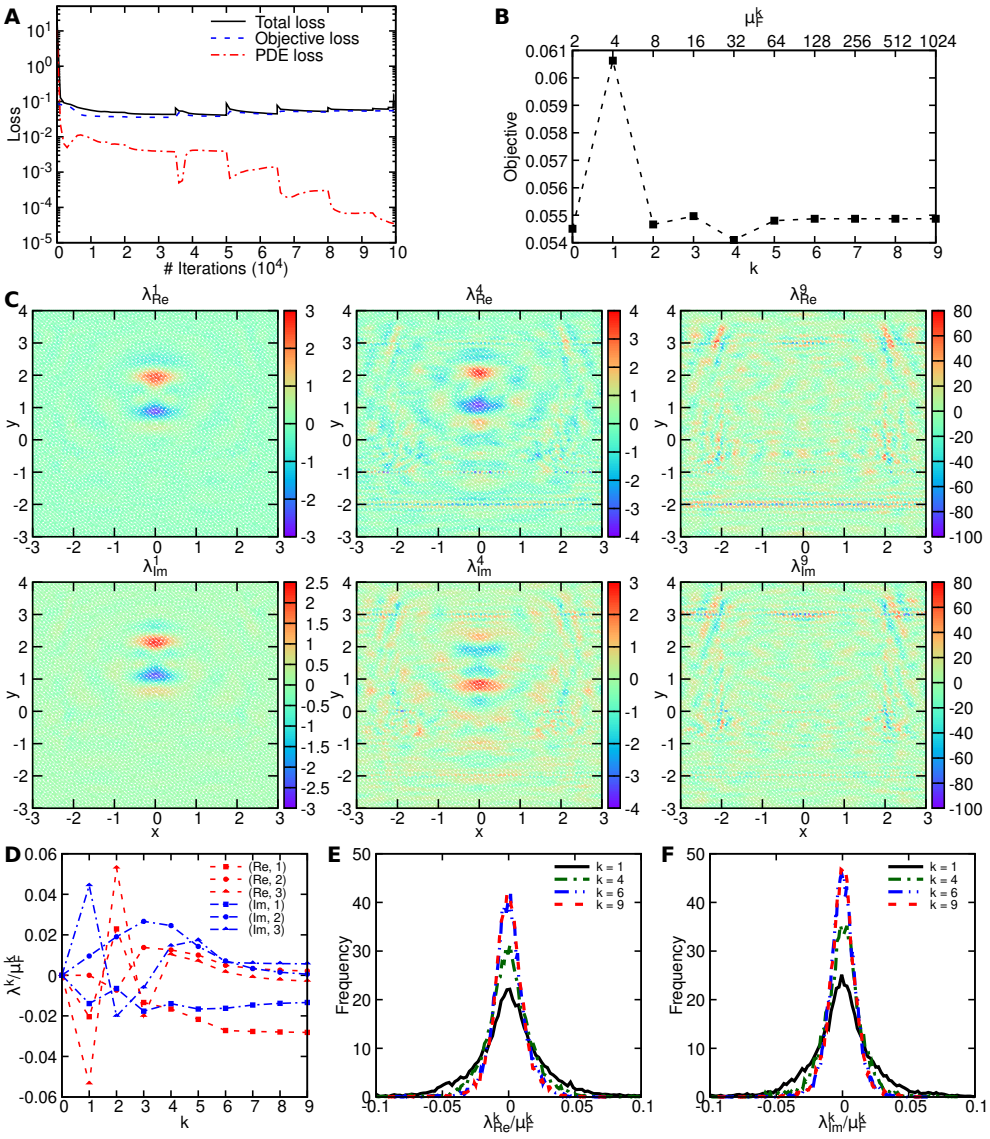

FIG. 6. hPINN for the inverse design of holography via the augmented Lagrangian method. (A) *and* (B) *The results of* $\mu_{\mathcal{F}}^0 = 2$ *and* $\beta_{\mathcal{F}} = 2$. (A) *The losses versus the number of optimization iterations and* (B) *the objective value after the network is trained with* $\mu_{\mathcal{F}}^k$. (C) *The values of* $\lambda_{\Re,j}^k$ *and* $\lambda_{\Im,j}^k$ *at all* $\mathbf{x}_j$ *locations for* $k = 1$, 4, *and* 9. (D) *The convergence of* $\lambda_{\Re,j}^k/\mu_{\mathcal{F}}^k$ *and* $\lambda_{\Im,j}^k/\mu_{\mathcal{F}}^k$ *in three examples.* (E) *and* (F) *The distributions of* (E) $\lambda_{\Re,j}^k/\mu_{\mathcal{F}}^k$ *and* (F) $\lambda_{\Im,j}^k/\mu_{\mathcal{F}}^k$ *converge.*

multipliers $\lambda_{\Re,j}^k/\mu_{\mathcal{F}}^k$ and $\lambda_{\Im,j}^k/\mu_{\mathcal{F}}^k$ converge with respect to k; see the three examples in Figure 6D. Also, the histogram of $\lambda_{\Re,j}^1/\mu_{\mathcal{F}}^1$ for all $\mathbf{x}_j$ is shown in the black curve in (Figure 6E), which is symmetric and concentrates around zero. The distributions of $\lambda_{\Re,j}^k/\mu_{\mathcal{F}}^k$ converge with respect to k, and become almost converged when $k \geq 6$ (Figure 6E). Similarly, $\lambda_{\Im,j}^k/\mu_{\mathcal{F}}^k$ also converge in a similar way (Figure 6F).

We have demonstrated the effectiveness of the augmented Lagrangian method, and to further improve our design, we tune two hyperparameters: the network width and the number of Fourier basis terms used in the periodic BC. We need to choose an optimal network size to avoid the problems of underfitting and overfitting, and the optimal network width is around 60 when we choose the depth to be 5 (Figure 7A). As we discussed in section 2.3, two Fourier basis terms are sufficient in the network to implement the periodic BC, but by using more basis terms, we can achieve a smaller objective (Figure 7B). Hence, we choose the width as 48 and use 12 Fourier basis terms. This problem has many local optima with similar performance, and the exact solution found depends on the initialization of ε: when we trained the network from three random initializations, the objective values were 0.0517, 0.0525, and 0.0528. Here, the final PDE loss is $\sim 3 \times 10^{-5}$, which is small enough for this problem. To reach an even smaller PDE loss, we can use a larger network with more representation power. For example, when we used a wider network of 128 neurons per layer, we can achieve the PDE loss of 3×10^{-6}. We also note that, instead of the single-precision floating point commonly used in deep learning, double precision is required to achieve the PDE loss of 10^{-6}.

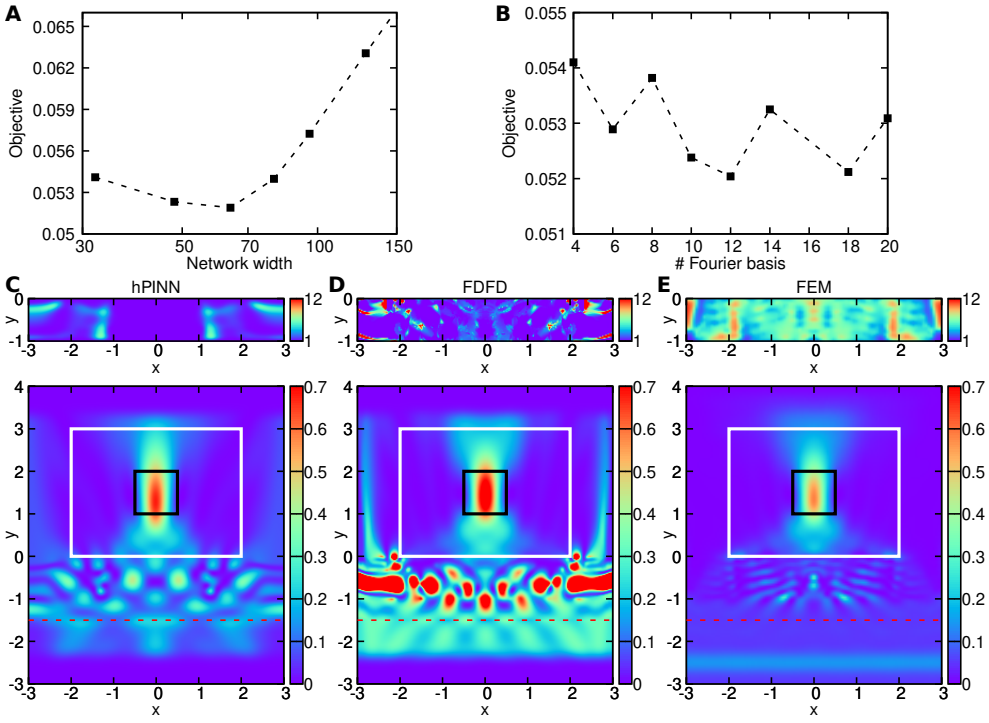

FIG. 7. Effect of hyperparameters and comparison between hPINN, FDFD, and FEM. (A) *Effect of the network width on the objective.* (B) *Effect of the number of Fourier basis terms used in the periodic BC on the objective.* (C)–(E) *The design ε and the field of $|E|^2$ of* (C) *hPINN,* (D) *FDFD, and* (E) *FEM.*

As a comparison, we also solve this problem by using the method for PDE-constrained inverse design discussed in [38], and here we have considered two numerical PDE solvers: FDFD [9] and FEM [2, 5, 20] (see the details in supplementary

sections SM2 and SM3, respectively). The objective values obtained from FDFD and FEM are 0.0523 and 0.0528, respectively, which is almost the same as that of hPINN, even though the designs are very different (because of the many local optima discussed above). In addition to the good performance, we observe that hPINN designs (Figure 7C, top) tend to be smoother than the designs of FDFD (Figure 7D, top) and FEM (Figure 7E, top), although smoothness can also be explicitly imposed by filtering if desired [37]. This could be explained by the analysis that neural networks trained with gradient descent have an implicit regularization and tend to converge to smooth solutions [23, 39, 47]. The electrical fields $|E|^2$ in the target domain Ω_3 for these three methods are similar (Figures 7C, D, and E, bottom).

3.2. Fluids in Stokes flow. Optimal design has wide and valuable applications in many fluid mechanics problems. Next, we use the proposed hPINN to solve the problem of topology optimization of fluids in Stokes flow, which was introduced by [7] and has been considered as a benchmark example in many works since then [14,16,52].

3.2.1. Problem setup. In this problem, we consider a design domain $\Omega = [0,1]\times[0,1]$ composed of solid material and fluids (Figure 8A). The goal is to determine at what places of Ω there should be fluid and where there should be solid, in order to minimize an objective function of dissipated power. We use $\rho = 0$ to represent the solid places, and $\rho = 1$ as the fluid. Then this problem is a discrete topology optimization for ρ, and solving a large-scale discrete topology optimization is generally computationally prohibitive [16]. To circumvent this issue, the common procedure is to allow the intermediate media values of ρ between 0 and 1.

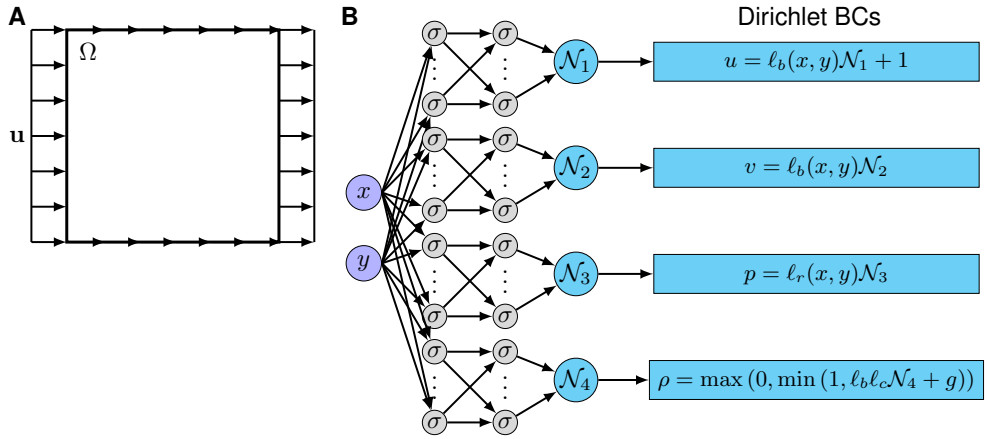

FIG. 8. Problem setup of fluids in Stokes flow and the neural network architecture. (A) *The design domain with the boundary condition.* (B) *The architecture of the hPINN with the Dirichlet BCs embedded directly in the network. The network inputs are x and y, and the outputs are u, v, p, and ρ.*

We consider the flow is a Stokes flow, and the fluid and the solid satisfy a generalized Stokes equation by treating the solid phase as a porous medium with flow governed by the Darcy's law [7, 14, 16]:

$$-\nu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \tag{3.6}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3.7}$$

where $\mathbf{u} = (u, v)$ is the velocity, and p is the pressure. $\nu = 1$ is the viscosity, and $\mathbf{f} = \alpha\mathbf{u}$ is the Brinkman term from the Darcy's law. α is the inverted permeability depending on ρ, and we use the following interpolation function proposed in [7]:

$$\alpha(\rho) = \overline{\alpha} + (\underline{\alpha} - \overline{\alpha})\rho\frac{1+q}{\rho+q},$$

where $\overline{\alpha} = \frac{2.5\nu}{0.01^2}$ and $\underline{\alpha} = 0$ are the inverted permeability of solid and fluid phases, respectively. The parameter $q > 0$ is used to control the transition between solid and fluid phases. When q is large, the interpolation has a sharper transition, and the optimization becomes more ill-conditioned. Here, we choose $q = 0.1$ as suggested in [7]. At the boundary, the velocity is constant $\mathbf{u} = (1, 0)$, and the pressure at the right boundary is zero (Figure 8A).

The objective function of dissipated power is defined as

$$\mathcal{J} = \int_\Omega \left(\frac{1}{2}\nabla\mathbf{u} : \nabla\mathbf{u} + \frac{1}{2}\alpha\mathbf{u}^2\right) dxdy. \tag{3.8}$$

To make the problem meaningful, we also need to consider a fluid volume constraint:

$$\int_\Omega \rho\, dxdy \le \gamma,$$

and the volume fraction γ is chosen as 0.9. Without this volume constraint, the optimal solution is $\rho = 1$ everywhere.

3.2.2. hPINN. The PDE-informed loss function for this problem is

$$\mathcal{L}_\mathcal{F} = \frac{1}{3M}\sum_{j=1}^{M}(\mathcal{F}_1[\mathbf{x}_j])^2 + (\mathcal{F}_2[\mathbf{x}_j])^2 + (\mathcal{F}_3[\mathbf{x}_j])^2,$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ correspond to the u and v components of (3.6), respectively, and $\mathcal{F}_3$ corresponds to (3.7). Similar to the holography problem, we scale $\mathcal{F}_1$ and $\mathcal{F}_2$ by 0.01 and $\mathcal{F}_3$ by 100. In this problem, we also have an inequality constraint for the fluid volume:

$$h(\rho) = \int_\Omega \rho\, dxdy - \gamma \le 0,$$

which is used to compute the loss in (2.6) and (2.7).

We construct four networks to approximate u, v, p, and ρ (Figure 8B), and each network has 5 layers with 64 neurons per layer. To impose the Dirichlet BCs of u and v into the network, we choose

$$\ell_b(x, y) = 16xy(1-x)(1-y),$$

which is equal to zero at the boundary, and the coefficient "16" is used to scale the function to be of order one inside Ω. Similarly, for the BC of p at the right boundary, we use

$$\ell_r(x, y) = 1 - x.$$

To restrict ρ between 0 and 1, we use the function $\max(0, \min(1, \cdot))$. We do not use the sigmoid function as in the holography problem, because the sigmoid function

cannot be equal to 0 and 1 exactly. To prevent the optimization getting stuck at a local minimum [7, 14], a small portion of Ω close to the boundary is prescribed as fluid, and the initial guess of ρ should be chosen properly. Here, we also add similar restrictions: the boundary is prescribed as fluid, and the center is prescribed as solid, i.e., $\rho(x, y) = 1$ when (x, y) is at the boundary, and $\rho(x, y) = 0$ when $x = y = 0.5$. This is imposed by constructing ρ as

$$\rho(x,y) = \max\left(0, \min\left(1, \ell_b(x,y)\ell_c(x,y)\mathcal{N}_4(x,y) + g(x,y)\right)\right), \tag{3.9}$$

where

$$\ell_c(x,y) = (x-0.5)^2 + (y-0.5)^2,$$
$$g(x,y) = \left(1 + \frac{\epsilon}{\min_{(x,y)} \ell_c(x,y)}\right)(1 - \ell_b(x,y))\,\frac{\ell_c(x,y)}{\ell_c(x,y)+\epsilon}.$$

By choosing a small number $\epsilon > 0$ (e.g., $\epsilon = 10^{-6}$), it is easy to check that $\rho(0.5, 0.5) = 0$ and $\rho(x, y) = 1$ when (x, y) is at the boundary.

3.2.3. Soft constraints. We first solve this problem using the approach of soft constraints by minimizing (2.5) with $\mu_{\mathcal{F}} = 0.1$ and $\mu_h = 10^4$. We use 10000 points for training, i.e., $M = 10000$. The same training procedure in the holography problem is used, but with a smaller learning rate 1×10^{-4}.

One random initialization of ρ using (3.9) is shown in Figure 9A, which satisfies our requirements for ρ at the boundary and center. Using this initial guess, the loss trajectory during the training is shown in Figure 9B. Initially, ρ satisfies the volume constraint, and after about 2000 iterations, the fluid volume $\int_\Omega \rho\, dxdy$ is always larger than γ, because more fluid makes the objective smaller. At the end of the training, the fluid volume is about 0.906, which is 0.7% larger than γ. The final fields of ρ (Figure 9C) and α (Figure 9D) have rugby ball-like shapes, which is consistent with the designs optimized by other methods [7, 14, 16, 52]. We use the smoothed profile method (SPM) [36, 62] in the context of the spectral element method [26] to compute the velocity from the design (see the details in section SM4 of the supplementary material), and the velocity magnitude $|\mathbf{u}|$ and the streamline are in Figures 9E and F. We note that although the PDE loss only decreases by two orders of magnitude from $\sim 1.5 \times 10^3$ to ~ 7 (Figure 9B), the L^2 relative error of $|\mathbf{u}|$ between hPINN and SPM is 4.3%, i.e., the PDEs are satisfied well.

The objective value in (3.8) of our design is 13.73. For comparison, in [7], the same setup and parameters are used as ours, and their objective value obtained from the method of moving asymptotes with FEM is 14.07, which is 2.4% worse than ours. However, this does not mean that the soft-constraints approach is a better method: our objective is smaller only because the design slightly violates the volume constraint, permitting a reduced objective value.

3.2.4. Augmented Lagrangian method. To make the design satisfy the constraints better, we use the augmented Lagrangian method with $\mu_{\mathcal{F}}^0 = 0.1$, $\mu_h^0 = 10^4$, $\beta_{\mathcal{F}} = \beta_h = 2$, and $k \le 9$. As we discussed in the previous section, in the soft constraint approach, the network solution has already satisfied the PDE and volume constraints well, and thus when using the augmented Lagrangian method, we only need ~ 400 more iterations (Figures 9B and 10A), i.e., $\sim 2\%$ more computational cost.

After training with the augmented Lagrangian method, compared to the soft constraint approach, the loss of the fluid volume decreases from 10^{-5} to 10^{-8} (Figure 10A). The final design (Figure 10B) has the fluid volume of 0.901, which is almost

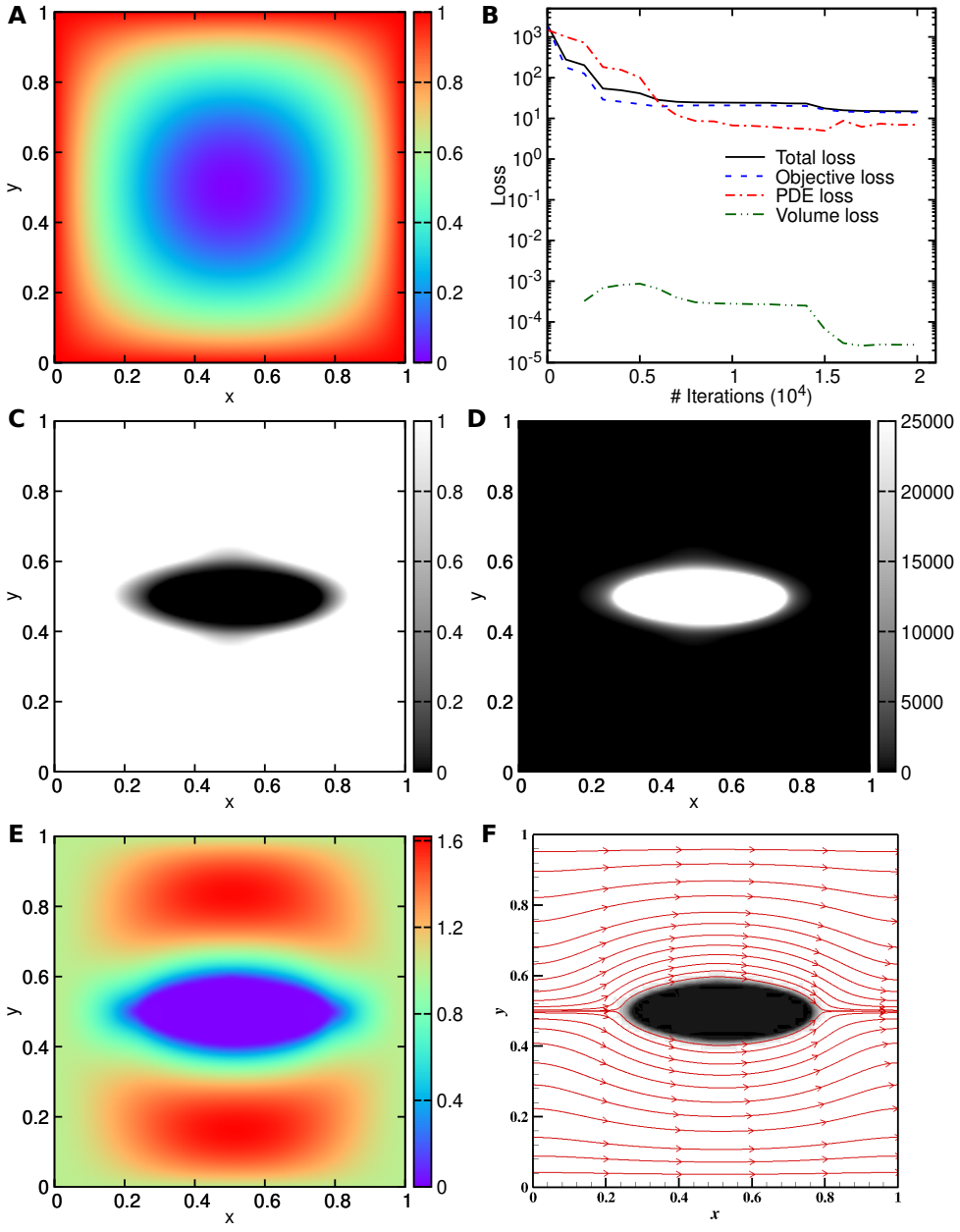

FIG. 9. hPIN for the topology optimization of fluids in Stokes flow via the soft constraint approach. (A) *The initialization of* ρ. (B) *The training losses versus the number of optimization iterations.* (C)–(F) *The optimization result:* (C) ρ, (D) *inverted permeability* α, (E) *velocity magnitude* $|\mathbf{u}|$, *and* (F) *the velocity streamline.*

identical to our restriction γ. On the other hand, the PDE loss only becomes a little bit smaller, and the L^2 relative error of $|\mathbf{u}|$ between hPINN and SPM is improved to 2.0%. The corresponding objective value is 14.12, which is consistent with the result

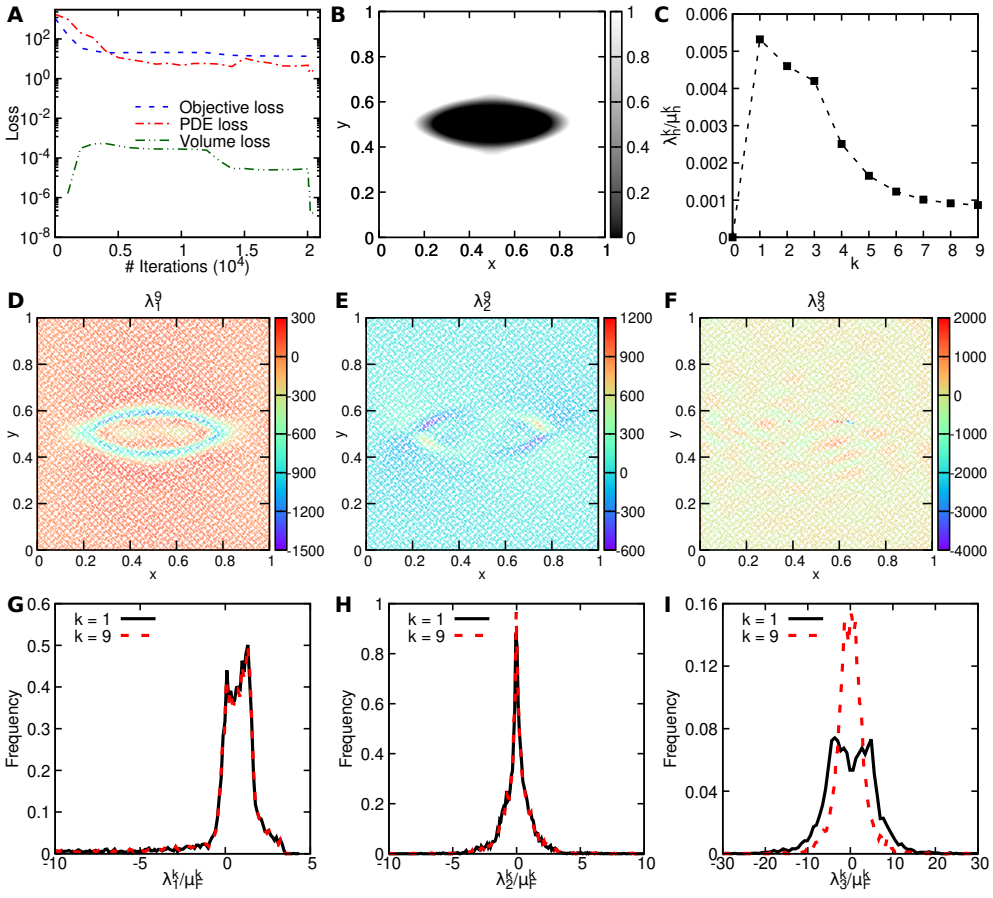

FIG. 10. hPIN for the topology optimization of fluids in Stokes flow via the augmented Lagrangian method. (A) *The training losses versus the number of optimization iterations.* (B) *The design of* ρ. (C) λ_h^k/μ_h^k *converges with respect to* k. (D)–(F) *The values of* (D) $\lambda_{1,j}^9$, (E) $\lambda_{2,j}^9$, *and* (F) $\lambda_{3,j}^9$ *at all* $\mathbf{x}_j$. (G)–(I) *The distributions of* (G) $\lambda_{1,j}^k/\mu_{\mathcal{F}}^k$, (H) $\lambda_{2,j}^k/\mu_{\mathcal{F}}^k$, *and* (I) $\lambda_{3,j}^k/\mu_{\mathcal{F}}^k$ *for* $k = 1$ *and* $k = 9$.

in [7].

Similar to the holography problem, here the normalized multipliers also converge. The normalized multiplier for the volume constraint λ_h^k/μ_h^k converges to 0.001 (Figure 10C). The values of $\lambda_{i,j}^k$ for $i = 1, 2$, and 3 at $k = 9$ are shown in Figures 10D, E, and F, respectively. $\lambda_{3,j}^9$ is almost uniform, but $\lambda_{1,j}^9$ and $\lambda_{2,j}^9$ have the largest values near the interface between the solid and fluid phases. The distributions of normalized $\lambda_{1,j}^k$ and $\lambda_{2,j}^k$ do not change too much from $k = 1$ to $k = 9$ (Figures 10G and H). However, the distribution of normalized $\lambda_{3,j}^k$ changes a lot and converges to a Gaussian distribution (Figure 10I), because $\lambda_{3,j}^k$ corresponds to (3.7), i.e., the mass continuity equation for incompressible fluid. This is consistent with our analysis that during the training, the fluid volume is the main variable to be improved.

4. Conclusion. In an inverse design problem, we aim to find the best design by minimizing an objective function, which is subject to various constraints, including

partial differential equations (PDEs), boundary conditions (BCs), and inequalities. In this study we have developed a new method of physics-informed neural networks (PINNs) with hard constraints (hPINNs) for solving inverse design. In hPINN, we enforce the PDE and inequality constraints by using loss functions, while we impose exactly Dirichlet and periodic BCs into the neural network architecture. We proposed the two approaches of the penalty method and the augmented Lagrangian method to enforce the loss function as hard constraints. We note that hPINN belongs to unsupervised learning and does not need a training dataset generated by numerical solvers.

We demonstrated the effectiveness of hPINN for two examples: the holography problem in optics and fluids of Stokes flow. Our numerical results of both examples show that although the approach of soft constraints cannot satisfy the PDE or inequality constraints to a good accuracy during the network training, we may still obtain a relatively good design if the penalty coefficients are chosen properly. The penalty method is able to impose hard constraints, but it has the issue of convergence when the penalty coefficients are too large. By using the augmented Lagrangian method, we can impose hard constraints and also achieve a better design. In addition, the objective function and the multipliers converge quickly after only several rounds of training. We also show that the computational cost of the augmented Lagrangian method is comparable to the soft constraint approach, because we use the network solution of the previous round as the network initialization for the next round of training.

Compared to traditional PDE-constrained optimization methods based on adjoint methods and numerical PDE solvers, the results obtained from hPINN have the same objective value, but the design is smoother for the case of nonunique solutions, which could also be achieved in traditional methods by using explicit filtering or other regularizations. The code implementation of hPINN is quite similar for different problems of inverse design because it leverages extensive deep-learning frameworks such as TensorFlow [1], so hPINN potentially reduces the human effort of programming. However, the suitable hyperparameters of hPINN (e.g., network size) are problem dependent, and thus hyperparameter tuning is required. This tuning could be performed by trial-and-error or even automatically, such as by a brute-force grid search, Bayesian approaches [10], or other advanced search algorithms [18]. hPINN saves human effort at the cost of code runtime: for the holography problem, hPINN ($\sim$1.7 hours using an NVIDIA TITAN Xp GPU) is slower than the traditional methods ($\sim$1 hour using CPU); but hPINN can be sped up at least one order of magnitude by parallel training, as demonstrated recently in [56]. Traditional PDE-constrained optimization methods are very mature and often provide strong convergence guarantees, while PINNs, as a complementary approach to solving PDEs, are still in their infancy. Our paper shows that hPINN can solve inverse-design problems where the PDE is a hard constraint, unlike previous PINN work—this is a baseline requirement for further development in the emerging fields of scientific machine learning [3] and physics-informed machine learning [27].

In the current hPINN, because we used one multiplier for each residual location $\mathbf{x}$, the residual locations have to be fixed. Hence, the optimization of hPINN is deterministic, and other algorithms for constrained optimization could also be used, such as sequential quadratic programming or interior-point methods. In addition, the number of multipliers is equal to the number of residual locations, so that many multipliers are required, which could be expensive and inefficient for large-scale problems. In fact, we observed that the profile of multipliers is continuous with respect to $\mathbf{x}$, and

thus it is possible to use one single multiplier function $\lambda(\mathbf{x})$ (e.g., represented by a neural network) for the entire computational domain. This approach also leverages the continuity of the multiplier, and may induce a faster convergence. Moreover, by using one single multiplier function, we can apply stochastic sampling of residual locations in hPINN, which is extremely widespread in scaling NNs to very large problems. Hence, it is important to have a constrained-optimization method that generalizes to stochastic constrained-optimization problems. The augmented Lagrangian method can work with stochastic optimization [61], whereas other constrained-optimization methods largely do not.

In our numerical results, we showed that hPINN with an augmented Lagrangian converges to a good solution, but there are currently few theoretical guarantees due to the nonlinear, nonconvex nature of this formulation. However, the convergence of stochastic optimization with the augmented Lagrangian method was analyzed in [61], and the convergence of PINN for certain linear PDEs was proved in [55]. In future work, we hope to theoretically analyze the convergence of hPINN based on these results.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: A system for large-scale machine learning*, in Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 2016, pp. 265–283.

[2] S. Badia and F. Verdugo, *Gridap: An extensible finite element toolbox in Julia*, J. Open Source Softw., 5 (2020), 2520.

[3] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Wilcox, and S. Lee, *Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence*, Tech. report, U.S. DOE Office of Science, Washington, DC, 2019.

[4] E. Bayati, R. Pestourie, S. Colburn, Z. Lin, S. G. Johnson, and A. Majumdar, *Inverse designed metalenses with extended depth of focus*, ACS Photonics, 7 (2020), pp. 873–878.

[5] M. P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*, Springer-Verlag, Berlin, Heidelberg, 2013.

[6] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, London, 2014.

[7] T. Borrvall and J. Petersson, *Topology optimization of fluids in Stokes flow*, Internat. J. Numer. Methods Fluids, 41 (2003), pp. 77–107.

[8] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208, https://doi.org/10.1137/0916069.

[9] N. J. Champagne II, J. G. Berryman, and H. M. Buettner, *FDFD: A 3D finite-difference frequency-domain code for electromagnetic induction tomography*, J. Comput. Phys., 170 (2001), pp. 830–848.

[10] X. Chen, J. Duan, and G. E. Karniadakis, *Learning and meta-learning of stochastic advection–diffusion–reaction systems from sparse measurements*, Eur. J. Appl. Math., 32 (2021), pp. 397–420.

[11] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro, *Physics-informed neural networks for inverse problems in nano-optics and metamaterials*, Opt. Express, 28 (2020), pp. 11618–11633.

[12] A. Dener, M. A. Miller, R. M. Churchill, T. Munson, and C.-S. Chang, *Training Neural Networks under Physical Constraints Using a Stochastic Augmented Lagrangian Approach*, preprint, https://arxiv.org/abs/2009.07330, 2020.

[13] S. Dong and N. Ni, *A Method for Representing Periodic Functions and Enforcing Exactly Periodic Boundary Conditions with Deep Neural Networks*, preprint, https://arxiv.org/abs/2007.07442, 2020.
[14] X. Duan, X. Qin, and F. Li, *Topology optimization of Stokes flow using an implicit coupled level set method*, Appl. Math. Model., 40 (2016), pp. 5431–5441.
[15] J. W. Goodman, *Introduction to Fourier Optics*, Roberts and Company Publishers, Greenwood Village, CO, 2005.
[16] J. K. Guest and J. H. Prévost, *Topology optimization of creeping fluid flows using a Darcy–Stokes finite element*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 461–484.
[17] K. Guo, Z. Yang, C. Yu, and M. J. Buehler, *Artificial intelligence and machine learning in design of mechanical materials*, Mater. Horiz., 8 (2021), pp. 1153–1172.
[18] X. He, K. Zhao, and X. Chu, *AutoML: A survey of the state-of-the-art*, Knowl. Based Syst., 212 (2021), 106622.
[19] J. Hicken and J. Alonso, *Comparison of reduced-and full-space algorithms for PDE-constrained optimization*, in Proceedings of the 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, 2013, 1043.
[20] J. S. Jensen and O. Sigmund, *Topology optimization for nano-photonics*, Laser Photonics Rev., 5 (2011), pp. 308–321.
[21] J. Jiang, M. Chen, and J. A. Fan, *Deep neural networks for the evaluation and design of photonic devices*, Nat. Rev. Mater., 6 (2021), pp. 679–700.
[22] J. Jiang and J. A. Fan, *Simulator-based training of generative neural networks for the inverse design of metasurfaces*, Nanophotonics, 1 (2019), pp. 1059–1069.
[23] P. Jin, L. Lu, Y. Tang, and G. E. Karniadakis, *Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness*, Neural Netw., 130 (2020), pp. 85–99.
[24] S. G. Johnson, *Notes on Perfectly Matched Layers (PMLs)*, preprint, https://arxiv.org/abs/2108.05348, 2021.
[25] M. Kadic, G. W. Milton, M. van Hecke, and M. Wegener, *3D metamaterials*, Nat. Rev. Phys., 1 (2019), pp. 198–210.
[26] G. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*, 2nd ed., Numer. Math. Sci. Comput., Oxford University Press, New York, 2005.
[27] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nat. Rev. Phys., 3 (2021), pp. 422–440.
[28] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, preprint, https://arxiv.org/abs/1412.6980, 2014.
[29] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, *Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks*, Comput. Methods Appl. Mech. Eng., 358 (2020), 112623.
[30] K. Kojima, B. Wang, U. Kamilov, T. Koike-Akino, and K. Parsons, *Acceleration of FDTD-based inverse design using a neural network approach*, in Integrated Photonics Research, Silicon and Nanophotonics, Optical Society of America, 2017, ITu1A.4.
[31] P. L. Lagari, L. H. Tsoukalas, S. Safarkhani, and I. E. Lagaris, *Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions*, Int. J. Artif. Intell., 29 (2020), 2050009.
[32] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. Neural Netw. Learn. Syst., 9 (1998), pp. 987–1000.
[33] D. Liu, Y. Tan, E. Khoram, and Z. Yu, *Training deep neural networks for the inverse design of nanophotonic structures*, ACS Photonics, 5 (2018), pp. 1365–1369.
[34] Z. Liu, D. Zhu, S. P. Rodrigues, K. Lee, and W. Cai, *Generative model for the inverse design of metasurfaces*, Nano Lett., 18 (2018), pp. 6570–6576.
[35] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Rev., 63 (2021), pp. 208–228, https://doi.org/10.1137/19M1274067.
[36] X. Luo, M. R. Maxey, and G. E. Karniadakis, *Smoothed profile method for particulate flows: Error analysis and simulations*, J. Comput. Phys., 228 (2009), pp. 1750–1769.
[37] K. Maute and O. Sigmund, *Topology optimization approaches: A comparative review*, Struct. Multidiscip. Optim., 48 (2013), pp. 1031–1055.
[38] S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez, *Inverse design in nanophotonics*, Nat. Photonics, 12 (2018), pp. 659–670.
[39] V. Nagarajan and J. Z. Kolter, *Generalization in Deep Networks: The Role of Distance*

from Initialization, preprint, https://arxiv.org/abs/1901.01672, 2019.
[40] Y. Nandwani, A. Pathak, and P. Singla, *A primal dual formulation for deep learning with constraints*, in Advances in Neural Information Processing Systems, 2019, pp. 12157–12168.
[41] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, 2006.
[42] G. Pang, L. Lu, and G. E. Karniadakis, *fPINNs: Fractional physics-informed neural networks*, SIAM J. Sci. Comput., 41 (2019), pp. A2603–A2626, https://doi.org/10.1137/18M1229845.
[43] R. Pestourie, *Assume Your Neighbor is Your Equal: Inverse Design in Nanophotonics*, Ph.D. thesis, Harvard University, Cambridge, MA, 2020.
[44] R. Pestourie, Y. Mroueh, T. V. Nguyen, P. Das, and S. G. Johnson, *Active learning of deep surrogates for PDEs: Application to metasurface design*, npj Comput. Mater., 6 (2020), 164.
[45] R. Pestourie, C. Pérez-Arancibia, Z. Lin, W. Shin, F. Capasso, and S. G. Johnson, *Inverse design of large-area metasurfaces*, Opt. Express, 26 (2018), pp. 33732–33747.
[46] J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, and M. Soljačić, *Nanophotonic particle simulation and inverse design using artificial neural networks*, Sci. Adv., 4 (2018), eaar4206.
[47] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar, *Theory of Deep Learning* III*: Explaining the Non-overfitting Puzzle*, preprint, https://arxiv.org/abs/1801.00173, 2017.
[48] 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), pp. 686–707.
[49] M. Raissi, A. Yazdani, and G. E. Karniadakis, *Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations*, Science, 367 (2020), pp. 1026–1030.
[50] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, *Physics-informed neural networks for cardiac activation mapping*, Front. Phys., 8 (2020), 42.
[51] H. Sasaki and H. Igarashi, *Topology optimization accelerated by deep learning*, IEEE Trans. Magn., 55 (2019), pp. 1–5.
[52] V. Schulz and M. Siebenborn, *Computational comparison of surface metrics for PDE constrained shape optimization*, Comput. Methods Appl. Math., 16 (2016), pp. 485–496.
[53] V. Sekar, M. Zhang, C. Shu, and B. C. Khoo, *Inverse design of airfoil using a deep convolutional neural network*, AIAA J., 57 (2019), pp. 993–1003.
[54] H. Sheng and C. Yang, *PFNN: A Penalty-Free Neural Network Method for Solving a Class of Second-Order Boundary-Value Problems on Complex Geometries*, preprint, https://arxiv.org/abs/2004.06490, 2020.
[55] Y. Shin, J. Darbon, and G. E. Karniadakis, *On the Convergence of Physics Informed Neural Networks for Linear Second-order Elliptic and Parabolic Type PDEs*, preprint, https://arxiv.org/abs/2004.01806, 2020.
[56] K. Shukla, A. D. Jagtap, and G. E. Karniadakis, *Parallel Physics-Informed Neural Networks via Domain Decomposition*, preprint, https://arxiv.org/abs/2104.10013, 2021.
[57] S. So, T. Badloe, J. Noh, J. Rho, and J. Bravo-Abad, *Deep learning enabled inverse design in nanophotonics*, Nanophotonics, 9 (2020), pp. 1041–1057.
[58] M. H. Tahersima, K. Kojima, T. Koike-Akino, D. Jha, B. Wang, C. Lin, and K. Parsons, *Deep neural network inverse design of integrated photonic power splitters*, Sci. Rep., 9 (2019), pp. 1–9.
[59] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, *Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems*, Water Resour. Res., 56 (2020), e2019WR026731.
[60] M. Toussaint, *Introduction to Optimization: Constrained Optimization*, teaching lecture, 2014.
[61] I. Wang and J. C. Spall, *Stochastic optimisation with inequality constraints using simultaneous perturbations and penalty functions*, Int. J. Control, 81 (2008), pp. 1232–1238.
[62] Z. Wang, M. S. Triantafyllou, Y. Constantinides, and G. E. Karniadakis, *A spectral-element/Fourier smoothed profile method for large-eddy simulations of complex VIV problems*, Comput. Fluids, 172 (2018), pp. 84–96.
[63] D. A. White, W. J. Arrighi, J. Kudo, and S. E. Watts, *Multiscale topology optimization using neural network surrogate models*, Comput. Methods Appl. Mech. Engrg., 346 (2019), pp. 1118–1135.
[64] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis, *Systems biology informed deep learning for inferring parameters and hidden dynamics*, PLoS Comput. Biol., 16 (2020), e1007575.

[65] D. Zhang, L. Guo, and G. E. Karniadakis, *Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks*, SIAM J. Sci. Comput., 42 (2020), pp. A639–A665, https://doi.org/10.1137/19M1260141.

[66] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, *Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems*, J. Comput. Phys., 397 (2019), 108850.