Inverse Problems

PAPER

Numerical solution of inverse problems by weak adversarial networks

To cite this article: Gang Bao *et al* 2020 *Inverse Problems* **36** 115003

IOP Publishing Inverse Problems
Inverse Problems **36** (2020) 115003 (31pp) https://doi.org/10.1088/1361-6420/abb447

Numerical solution of inverse problems by weak adversarial networks

Gang Bao[1], Xiaojing Ye[2], Yaohua Zang[1,*] and Haomin Zhou[3]

[1] School of Mathematical Sciences, Zhejiang University, Hangzhou, Zhejiang, People's Republic of China
[2] Department of Mathematics and Statistics, Georgia State University, Atlanta, GA, 30303, United States of America
[3] School of Mathematics, Georgia Institute of Technology, Atlanta, GA, 30332, United States of America

E-mail: baog@zju.edu.cn, xye@gsu.edu, yhchuang@zju.edu.cn and hmzhou@math.gatech.edu

Received 26 February 2020, revised 22 August 2020
Accepted for publication 1 September 2020
Published 23 October 2020

Abstract

In this paper, a weak adversarial network approach is developed to numerically solve a class of inverse problems, including electrical impedance tomography and dynamic electrical impedance tomography problems. The weak formulation of the partial differential equation for the given inverse problem is leveraged, where the solution and the test function are parameterized as deep neural networks. Then, the weak formulation and the boundary conditions induce a minimax problem of a saddle function of the network parameters. As the parameters are alternatively updated, the network gradually approximates the solution of the inverse problem. Theoretical justifications are provided on the convergence of the proposed algorithm. The proposed method is completely mesh-free without any spatial discretization, and is particularly suitable for problems with high dimensionality and low regularity on solutions. Numerical experiments on a variety of test inverse problems demonstrate the promising accuracy and efficiency of this approach.

Keywords: inverse problem, deep learning, weak formulation, adversarial network, stochastic gradient

(Some figures may appear in colour only in the online journal)

*Author to whom any correspondence should be addressed.

1. Introduction

Inverse problems (IP) are ubiquitous in a vast number of scientific disciplines, including geophysics [50], signal processing and imaging [7], computer vision [41], remote sensing and control [57], statistics [35], and machine learning [23]. Let Ω be an open and bounded set in $\mathbb{R}^d$, then an IP defined on Ω can be presented in a general form as:

$$\mathcal{A}[u,\gamma]=0, \quad \text{in } \Omega \tag{1a}$$

$$\mathcal{B}[u,\gamma]=0, \quad \text{on } \partial\Omega \tag{1b}$$

where $\mathcal{A}[u,\gamma]$ specifies a differential equation, in which u is the solution and γ the coefficient in the inverse medium problem or the source function in the inverse source problem. Equation $\mathcal{A}$ can be an ordinary differential equation, or a partial differential equation (PDE), or an integro-differential equation, that (u,γ) needs to satisfy (almost) everywhere inside the region Ω. The boundary value (and initial value if applicable) is given by $\mathcal{B}[u,\gamma]$ on $\partial\Omega$. Depending on specific applications, partial information of u and/or γ may be available in the interior of Ω. Then IP (1) is to find (u,γ) that satisfies both (1a) and (1b).

To instantiate our approach, we mostly use the classical inverse conductivity problem in electrical impedance tomography (EIT) [11, 37] as an example to present our main idea and the derivations in this paper. However, our methodology can be readily applied to other classes of IPs with modifications. An example of dynamic EIT problem will be shown in section 4. The goal of EIT is to determine the electrical conductivity distribution $\gamma(x)$ of an unknown medium defined on Ω based on the potential u, the current $-\gamma\partial_{\vec{n}}u$ measurements, and the knowledge of γ (and hence $\partial_{\vec{n}}u$) on/near the boundary $\partial\Omega$ of the domain Ω:

$$-\nabla\cdot(\gamma\nabla u)-f=0, \quad \text{in } \Omega \tag{2a}$$

$$u-u_b=0, \qquad \gamma-\gamma_b=0, \qquad \partial_{\vec{n}}u-u_n=0, \quad \text{on } \partial\Omega \tag{2b}$$

where u_b is the measured voltage, γ_b is the conductivity near the surface of the object and $u_n \triangleq \nabla u\cdot\vec{n}$ with $\vec{n}$ being the outer normal of $\partial\Omega$. Note that our approach is not to estimate the Dirichlet-to-Neumann (DtN) map associated with the conductivity function as in classical methods specific to the EIT problem [12, 21, 40]. Instead, our goal is to directly solve a general class of IPs (1) numerically using the given data, with the EIT problem (2) as a prototype example without exploiting its special structure (e.g., the DtN map). To make our presentation concise and focused, we only consider IPs with $\mathcal{A}[u,\gamma]$ characterized by PDEs in (1a), and assume that the given IP is well-defined and admits at least one (weak) solution.

Our approach is to train deep neural networks that can represent the solution (u,γ) of a given IP, with substantial improvement over classical numerical methods especially for problems with high dimensionality. More specifically, we leverage the weak formulation of the PDE (1a) and convert the IP into an operator norm minimization problem of u and γ. Then we parameterize both u, the unknown coefficient γ, and the test function φ as deep neural networks u_θ, γ_θ, and φ_η respectively, with network parameters (θ,η), and form a minimax problem of a saddle function of the parameters (θ,η). Finally, we apply the stochastic gradient descent (SGD) method to alternately update the network parameters so that (u_θ,γ_θ) gradually approximates the solution of the IP. The parameterization of (u,γ) using deep neural networks requires no discretization of the spatial and temporal domain, and hence is completely mesh free. This is a promising alternative compared to the classical finite difference method (FDM) and finite element methods which suffer the issue of the so-called curse of

dimensionality, a term first used in [6]. Moreover, our approach combines the training of the weak solution (primal network) (u, γ) and the test function (adversarial network) φ governed by the weak formulation of the PDE, which requires less regularity of the solution (u, γ) and can be more advantageous in many real-world applications when the solution has singularities.

The remainder of this paper is organized as follows. We first review the recent work on deep learning based solutions to forward and IP in section 2. In section 3, we provide the detailed derivation of our method and a series of theoretical results to support the validity of the proposed approach. We discuss several implementation techniques that can improve practical performance and conduct a series of numerical experiments to demonstrate the effectiveness of the proposed approach in section 4. Section 5 concludes this paper with some general remarks.

2. Related work

The past few years have witnessed an emerging trend of using deep learning based methods to solve forward and IP. These methods can be roughly classified into two categories. The first category includes methods that approximate the solution of a given problem based on supervised learning approaches. These methods require a large number of input–output pairs through numerical simulation and experiments to train the desired networks. In this category, deep neural networks are used to generate approximate intermediate results from measurement data for further refinement [20, 45, 48, 53, 54, 58], applied to improve the solution of classical numerical methods in the post-processing phase [5, 26, 27, 29, 33, 38, 46, 56], or approximate the mapping from given parameters of an IP to its solution but require spatial discretization and cannot be applied to high-dimensional problems [2, 32, 43].

The second category features unsupervised learning methods that directly solve the forward or IP based on the problem formulation rather than additional training data, which can be more advantageous than those in the first category in practice. For example, feed-forward neural networks are used to parameterize the coefficient functions and trained by minimizing the performance function in [13]. In [39], a neural network architecture called SwitchNet is proposed to solve the inverse scattering problem through the mapping between the scatterers and the scattered field. In [18], a deep learning approach specific to 2D and 3D EIT problems is developed to represent the DtN map by a compact neural network architecture. The backward stochastic differential equation (BSDE) corresponding to the PDE in a forward problem is parameterized in part by neural networks, such that the solution of the PDE can be obtained by integrating the BSDE for a target point in the domain [9, 16, 28]. In [17], the solution of a forward problem is parameterized as a deep neural network, which is trained by minimizing the loss function composed of the energy functional associated with the PDE and a penalty term on the boundary value condition. Another mesh-free framework, called physics-informed neural networks (PINN), for solving both the forward and IP using deep neural networks based on the strong formulation of PDEs is proposed in [51], where a constant coefficient function is considered for the IP part. Specifically, PINN parameterizes the unknowns of a given PDE using deep neural networks, which are trained by minimizing the loss function formed as the least squares of the violation of the PDE at sampled points in the domain and boundary conditions. Some empirical study of PINN is also conducted in [14]. Solutions to IPs based on PINN with data given in problem domain are also considered in [34], and refinement of solutions using adaptively sampled collocation points is proposed in [4]. In [59], the weak formulation of the PDE is leveraged as the objective function, where the solution of the PDE and the test function are both parameterized as deep neural networks trying to

minimize and maximize the objective function, respectively. In [36], a similar variational form is used where the test function is fixed basis instead of neural networks to be learned. In [47], three neural networks, one for low-fidelity data and the other two for the linear and nonlinear functions for high-fidelity data, are used by following the PINN approach. The PINN with a multi-fidelity network structure is also proposed for stochastic PDE cases, where polynomial chaotic expansions are used to express the solutions, i.e., as a linear combination of random basis with coefficient functions to be learned [10]. In [8], the solution of an IP is parameterized by deep neural network and learned by minimizing a cost function that enforces the conditions of IP and additional regularization, where solutions to the PDE are required during the training.

Recently, meta-learning based approaches for forward problems are also considered [10, 19, 44]. In [19], the mapping from the coefficient of a differential operator to the pseudo-differential operator (e.g., the Green function) is learned by leveraging the compressed form of the wavelet transform. In [44], a deep operator network consisting of a branch network and a trunk network is introduced. The network encodes the input function evaluated at a finite number of locations (branch-net) and the locations for the output function (trunk-net) and, the output function is given by the inner product of the two plus a bias. Learning network width and depth parameters are also considered using Bayesian optimization in [10].

Our approach to the IP follows our earlier work [59] for forward problems, which differs from the aforementioned existing methods in the use of the weak formulation of PDEs. The weak formulation is a powerful approach for solving PDEs as it requires less regularity and allows for necessary singularities of the solutions, which is an important feature appreciated in many real-world applications such as imaging and abnormality detections. From the theoretical point of view, our method employs neural network parameterizations of both the solution (as the primal network) and the test function (as the adversarial network), and performs an adversarial training in a way that the test function critics on the solution network where the PDE is violated, and the solution network corrects itself at those spots until the PDE is satisfied (almost) everywhere in the domain. However, as IP are often ill-posed and more difficult to solve than forward problems in general, we mostly focus on the IP (2) in EIT in this work. Some experimental results on similar problems are also presented in section 4.

The adversarial training in the present work has a similar flavor as the one used in generative adversarial network [24], where a generator network is aimed at mapping generic random samples (such as those from a given multivariate Gaussian) to ones following the same distribution as the training samples, and a discriminator network is to distinguish these samples produced by the generator network from the true samples. The generator and adversarial networks act as the two players in a zero-sum game, and are alternately updated by gradient descent and ascent on the objective function respectively to reach an equilibrium. In particular, a notable variant of GAN, called Wasserstein GAN [3], also has a min–max structure of a primal network (generator) and adversarial network (dual function of optimal transport due to the Wasserstein distance between generated and sample distributions) as our formulation. However, WGAN requires its dual function in the max problem to be 1-Lipschitz, which is very difficult to realize numerically and has generated a series of followup work to overcome the issue [25, 49], spectral normalization for generative adversarial networks. In contrast, the structure of weak solution versus test function in our work arises naturally from the weak formulation in the PDE theory, which enjoys numerous theoretical justifications and computational benefits for solving IPs for PDEs without imposing restrictive constraint on the adversarial network (test function), as we show in the present work.

In contrast to many existing deep learning methods that require a large amount of demonstration data (e.g., coefficient/boundary value and solution pairs) for training, our method follows

an unsupervised learning strategy and only needs the formulation of the PDE and boundary conditions in the given IP. In [55], an unsupervised learning study reveals that generic convolutional neural networks automatically bias toward smooth signals and can produce results similar to some sophisticated reconstructions in image denoising without any training data. This phenomenon, known as deep image prior (DIP), is further exploited in [15, 30]. The most notable difference between DIP and the present work is that, our method is completely mesh-free and does not require any spatial discretization, which is suitable for high-dimensional problems. In DIP and its followup works, on the other hand, the reconstruction network is applied to discretized 2D or 3D images. Moreover, our goal is to use the representation power of deep networks to parameterize the solution of an IP in continuous space, whereas the main interests in DIP are on its intriguing automatic regularization properties.

3. Weak adversarial network for IP

The proposed weak adversarial network approach for IPs is inspired by the weak formulation of PDEs. To obtain the weak formulation of the PDE in (1a), we multiply both sides of (1a) by an arbitrary test function $\varphi \in H_0^1(\Omega)$ (the Hilbert space of functions with bounded first-order weak derivatives and compactly supported in Ω) and integrate over Ω:

$$\langle \mathcal{A}[u,\gamma],\varphi\rangle := \int_\Omega \mathcal{A}[u,\gamma](x)\varphi(x)\mathrm{d}x = 0. \tag{3}$$

One of the main advantages of weak formulation (3) is that we can subsequently apply integration by parts to transfer certain gradient operator(s) in $\mathcal{A}[u,\gamma]$ to φ, such that the requirement on the regularity of u (and γ if applicable) can be reduced. For example, in the case of inverse conductivity problem (2), the integration by parts and the fact that $\varphi = 0$ on $\partial\Omega$ together yield

$$\langle \mathcal{A}[u,\gamma],\varphi\rangle = \int_\Omega (\gamma\nabla u\cdot\nabla\varphi - f\varphi)\,\mathrm{d}x = 0, \tag{4}$$

where $\gamma\nabla u$ is not necessarily differentiable as in (2) in the classical sense anymore (we use ∇ to denote the gradient operator with respect to x, and ∇_θ as the gradient with respect to θ and so on in this paper). We call $(u,\gamma)\in H^1(\Omega)\times L^2(\Omega)$ a *weak solution* (or *generalized solution*) of the IP (1) if (u,γ) satisfies the boundary condition (1b) and (3) for all $\varphi\in H_0^1(\Omega)$. Here $L^2(\Omega)$ is the Lebesgue space of square integrable functions on Ω, and $H^1(\Omega)\subset L^2(\Omega)$ is the Hilbert space of functions with bounded first-order weak derivatives. Note that any classical (strong) solution of (1) is also a weak solution. In this work, we seek for weak solutions of IP (1) so that we may be able to provide an answer to the problem even if it does not admit a solution in the classical sense.

Following the work [59], we consider the weak formulation of the PDE $\mathcal{A}[u,\gamma]=0$ in (1). To cope with the unknown solution u and parameter γ of the PDE in an IP, we parameterize both u and γ as deep neural networks, and consider $\mathcal{A}[u,\gamma]: H_0^1(\Omega)\to\mathbb{R}$ as a linear functional such that $\mathcal{A}[u,\gamma](\varphi) := \langle\mathcal{A}[u,\gamma],\varphi\rangle$ as defined in (3). We define the norm of $\mathcal{A}[u,\gamma]$ induced by the H_1 norm as

$$\|\mathcal{A}[u,\gamma]\|_{\mathrm{op}} := \sup_{\varphi\in H_0^1,\varphi\neq 0}\frac{\langle\mathcal{A}[u,\gamma],\varphi\rangle}{\|\varphi\|_{H^1}}, \tag{5}$$

where the H^1-norm of φ is given by $\|\varphi\|^2_{H^1(\Omega)} = \int_\Omega(|\varphi(x)|^2+|\nabla\varphi(x)|^2)\mathrm{d}x$. Therefore, (u,γ) is a weak solution of (1) if and only if $\|\mathcal{A}[u,\gamma]\|_{\mathrm{op}}=0$ and $\mathcal{B}[u,\gamma]=0$ on $\partial\Omega$. As $\|\mathcal{A}[u,\gamma]\|_{\mathrm{op}}\geqslant 0$,

we know that a weak solution (u, γ) to (1) thus solves the following problem in observation of (5):

$$\underset{u,\gamma}{\text{minimize}} \ \|\mathcal{A}[u,\gamma]\|_{\text{op}}^2 = \underset{u,\gamma}{\text{minimize}} \sup_{\varphi \in H_0^1, \varphi \neq 0} \frac{|\langle \mathcal{A}[u,\gamma], \varphi \rangle|^2}{\|\varphi\|_{H^1}^2}, \tag{6}$$

among all $(u, \gamma) \in H^1(\Omega) \times L^2(\Omega)$, and attains minimal value 0. This result is summarized in the following theorem, and the proof is provided in appendix A.1.

Theorem 1. *Suppose that* (u^*, γ^*) *satisfies the boundary condition* $\mathcal{B}[u^*, \gamma^*] = 0$*, then* (u^*, γ^*) *is a weak solution of* (1) *if and only if* $\|\mathcal{A}[u^*, \gamma^*]\|_{\text{op}} = 0$.

Theorem 1 implies that, to find the weak solution of (1), we can instead seek for the optimal solution (u^*, γ^*) that satisfies $\mathcal{B}[u^*, \gamma^*] = 0$ and meanwhile minimizes (6) by achieving minimum operator norm value $\|\mathcal{A}[u^*, \gamma^*]\|_{\text{op}} = 0$ due to the nonnegativity of the operator norm. In other words, (u^*, γ^*) is a weak solution of the problem (1) if and only if both $\|\mathcal{A}[u^*, \gamma^*]\|_{\text{op}}$ and $\|\mathcal{B}[u^*, \gamma^*]\|_{L^2(\partial\Omega)}$ vanish. Therefore, we can solve (u^*, γ^*) from the following minimization problem which is equivalent to (1):

$$\underset{u,\gamma}{\text{minimize}} \ \ I(u,\gamma) = \|\mathcal{A}[u,\gamma]\|_{\text{op}}^2 + \beta \|\mathcal{B}[u,\gamma]\|_{L^2(\partial\Omega)}^2, \tag{7}$$

and $\beta > 0$ is a weight parameter that balances the two terms in the objective function $I(u, \gamma)$. Note that both terms of the objective function in (7) are nonnegative and vanish simultaneously only at a weak solution (u^*, γ^*) of (1).

A promising alternative to classical numerical methods for high-dimensional PDEs is the use of deep neural networks since they do not require domain discretization and are completely mesh free. Deep neural networks are compositions of multiple simple functions (called layers) so that they can approximate rather complicated functions. Consider a simple multi-layer neural network u_θ as follows:

$$u_\theta(x) = w_K^\top \, l_{K-1} \circ \cdots \circ l_0(x) + b_K, \tag{8}$$

where the kth layer $l_k : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ is given by $l_k(z) = \sigma_k(W_k z + b_k)$ with weight $W_k \in \mathbb{R}^{d_{k+1} \times d_k}$ and bias $b_k \in \mathbb{R}^{d_{k+1}}$ for $k = 0, 1, \ldots, K-1$, and the network parameters of all layers are collectively denoted by θ as follows,

$$\theta := (w_K, b_K, W_{K-1}, b_{K-1}, \ldots, W_0, b_0). \tag{9}$$

Throughout, all vectors in this paper are column vectors by default. In (8), $x \in \Omega$ is the input of the network, $d_0 = d$ is the problem dimension of (1) (also known as the size of input layer), $w_K \in \mathbb{R}^{d_K}$ and $b_K \in \mathbb{R}$ are parameters in the last Kth layer (also called the output layer). Typical choices of the nonlinear activation function σ_k include sigmoid function $\sigma(z) = (1 + \mathrm{e}^{-z})^{-1}$, hyperbolic tangent (tanh) function $\sigma(z) = (\mathrm{e}^z - \mathrm{e}^{-z})/(\mathrm{e}^z + \mathrm{e}^{-z})$, and rectified linear unit function $\sigma(z) = \max(0, z)$, which are applied componentwisely. The training of deep neural networks refers to the process of optimizing θ using available data or constraints such that the function u_θ can approximate the (unknown) target function. More details about deep neural networks can be found in [23].

Despite of the simple structures like (8), deep neural networks are capable to approximate rather complicated continuous function (and its derivatives if needed) uniformly on a compact support $\bar{\Omega}$. This significant result is known as the universal approximation theorem [31]. The expressive power of neural networks ensured by the universal approximation theorem suggests a promising mesh-free parameterization of the weak solution (u, γ) of (1). In what follows,

we select sufficiently deep neural network structures of form (8) for both u and γ. Specific structures, i.e., layer number K and sizes $\{d_1, \ldots, d_{K-1}\}$, used in our numerical experiments will be provided in section 4. Note that u and γ are two separate networks, but we use a single letter θ to denote their network parameters rather than θ_u and θ_γ to simplify notations. That is, we parameterize (u, γ) as deep neural networks $(u_\theta, \gamma_\theta)$, and attempt to find the parameter θ such that $(u_\theta, \gamma_\theta)$ solves (7). To this end, the test function φ in the weak formulation (3) is also parameterized as a deep neural network φ_η in a similar form of (8) and (9) with parameter denoted by η. With the parameterized $(u_\theta, \gamma_\theta)$ and φ_η, we follow the inner product notation in (3) and define

$$E(\theta, \eta) := |\langle \mathcal{A}[u_\theta, \gamma_\theta], \varphi_\eta \rangle|^2. \tag{10}$$

Instead of normalizing $E(\theta, \eta)$ by $\|\varphi_\eta\|_{H_1}^2$ as in the original definition of (squared) operator norm (5), we approximate (up to a constant scaling of) the squared operator norm in (5) by the following max-type function of θ:

$$L_{\text{int}}(\theta) := \max_{|\eta|^2 \leqslant 2B} E(\theta, \eta) \tag{11}$$

where $B > 0$ is a prescribed bound to constrain the magnitude of network parameter η. Here $|\eta|^2 = \sum_k (\sum_{ij} [W_k]_{ij}^2 + \sum_i [b_k]_i^2)$, and $[M]_{ij} \in \mathbb{R}$ stands for the (i,j)th entry of a matrix M, and $[v]_i \in \mathbb{R}$ the ith component of a vector v. It is worth noting that the bound constraint on the ℓ_2-norm of η in (11) is similar to the weight clipping (equivalent to bound on ℓ_∞-norm) method used in WGAN [3]. However, they serve different purposes: the constraint in (11) is introduced so that the integrals, such as (4), are bounded (the actual value of this bound can be arbitrary). In this case, the stochastic gradients obtained by the Monte-Carlo (MC) approximations in our numerical implementation have bounded variance, which is needed in the proof of theorem 4 below. On the other hand, the weight clipping in WGAN is to ensure the dual function realized by the neural network is in the class of 1-Lipschitz functions $\mathcal{F} := \{f : \Omega \to \mathbb{R} : |f(x) - f(y)| \leqslant |x - y|, \ \forall x, y \in \Omega\}$. As noted in [3], weight clipping is a simple but not appropriate way to implement the 1-Lipschitz constraint, and hence there is a series of followup work to tackle this issue, such as [25, 49].

Furthermore, we define the loss function associated with the boundary condition (1b) by

$$L_{\text{bdry}}(\theta) := \|\mathcal{B}[u_\theta, \gamma_\theta]\|_{L^2(\partial\Omega)}^2 = \int_{\partial\Omega} |\mathcal{B}[u_\theta, \gamma_\theta](x)|^2 \, \mathrm{d}S(x). \tag{12}$$

For instance, if the boundary condition of (u, γ) is given in (2b) with known boundary value (u_b, γ_b, u_n), then $L_{\text{bdry}}(\theta) = \int_{\partial\Omega} |u_\theta(x) - u_b(x)|^2 + |\gamma_\theta(x) - \gamma_b(x)|^2 + |\partial_{\vec{n}(x)} u(x) - u_n(x)|^2 \, \mathrm{d}S(x)$. Finally, we define the total loss function $L(\theta)$, and solve the following minimization problem of its optimal θ^*:

$$\underset{\theta}{\text{minimize}} \ L(\theta), \quad \text{where } L(\theta) := L_{\text{int}}(\theta) + \beta L_{\text{bdry}}(\theta), \tag{13}$$

where we also constrain on the magnitude of the parameter θ such that $|\theta|^2 \leqslant 2B$ for the same B to simplify notation. Note that here both θ and η are finite dimensional vectors, and $L_{\text{int}}(\theta), L_{\text{bdry}}(\theta), E(\theta, \eta) \in \mathbb{R}_+$, hence it is possible to apply numerical optimization algorithms to find the minimizer of $L(\theta)$.

A standard approach to solving a minimization problem like (13) is the projected gradient descent method which performs the following iteration:

$$\theta \leftarrow \Pi(\theta - \tau \nabla_\theta L(\theta)), \tag{14}$$

where $\Pi(\theta) = \min(\sqrt{2B}, |\theta|) \cdot (\theta/|\theta|)$ is the projection of θ to the ball centered at origin with radius $\sqrt{2B}$, and $\tau > 0$ is the step size. As we can see, the main computation of (14) is on the gradient $\nabla_\theta L(\theta) = \nabla_\theta L_{\text{int}}(\theta) + \beta \nabla_\theta L_{\text{bdry}}(\theta)$. The computation of $\nabla_\theta L_{\text{bdry}}(\theta)$ is straightforward as shown later. The loss $L_{\text{int}}(\theta)$, however, is defined as a maximization problem (11), and we need to write its gradient as a function of θ first. To this end, we have the following lemma to compute the gradient $\nabla_\theta L_{\text{int}}(\theta)$, and the proof is provided in appendix A.2.

Lemma 2. *Suppose $L_{\text{int}}(\theta)$ is defined in (11). Then the gradient $\nabla_\theta L_{\text{int}}(\theta)$ at any θ is given by $\nabla_\theta L_{\text{int}}(\theta) = \partial_\theta E(\theta, \eta(\theta))$, where $\eta(\theta)$ is a solution of $\max_{|\eta|^2 \leqslant 2B} E(\theta, \eta)$ for the specified θ.*

Remark 3.1. Lemma 2 suggests that, to obtain $\nabla_\theta L_{\text{int}}(\theta)$ at any given θ, we can first take the partial derivative of E with respect to θ with η untouched, and then evaluate the partial derivative using θ and any solution $\eta(\theta)$ of the maximization problem (11).

The exact gradients of $L_{\text{int}}(\theta)$ and $L_{\text{bdry}}(\theta)$ require integrations of functions parameterized by deep neural networks over Ω and $\partial\Omega$ in continuous space, which are computationally intractable in practice. Therefore, we use MC approximations of these integrals. To this end, we need the following result on the approximation of integrals using samples, and the proof is provided in appendix A.3.

Lemma 3. *Suppose $\Omega \subset \mathbb{R}^d$ is bounded, and ρ is a probability density defined on Ω such that $\rho(x) > 0$ for all $x \in \Omega$. Given a function $\psi \in L^2(\Omega)$, denote $\Psi = \int_\Omega \psi(x)\mathrm{d}x$. Let $x^{(1)}, \ldots, x^{(N)}$ be N independent samples drawn from ρ. Consider the following estimator $\hat{\Psi}$ of Ψ:*

$$\hat{\Psi} = \frac{1}{N} \sum_{i=1}^{N} \frac{\psi(x^{(i)})}{\rho(x^{(i)})}. \tag{15}$$

Then the first and second moments of $\hat{\Psi}$ are given by

$$\mathbb{E}[\hat{\Psi}] = \Psi \quad \textit{and} \quad \mathbb{E}[\hat{\Psi}^2] = \frac{N-1}{N}\Psi^2 + \frac{1}{N}\int_\Omega \frac{\psi(x)^2}{\rho(x)}\,\mathrm{d}x. \tag{16}$$

Hence the variance of $\hat{\Psi}$ is $N^{-1} \cdot (\int_\Omega (\psi^2/\rho)\mathrm{d}x - (\int_\Omega \psi\ \mathrm{d}x)^2)$. In particular, with the uniform distribution $\rho(x) = 1/|\Omega|$, the variance of $\hat{\Psi} = (|\Omega|/N) \cdot \sum_i \psi(x^{(i)})$ is $N^{-1} \cdot (|\Omega| \int_\Omega \psi^2\ \mathrm{d}x - (\int_\Omega \psi\ \mathrm{d}x)^2)$.

Remark 3.2. We have several remarks regarding lemma 3:

- The estimator $\hat{\Psi}$ of the integral Ψ is unbiased.
- The variance of $\hat{\Psi}$ shown above decreases at the rate of $O(1/N)$ in the number N of sample collocation points. By Hölder's inequality and that ρ is a probability density, we know

$$\left|\int_\Omega \psi\ \mathrm{d}x\right| \leqslant \int_\Omega |\psi|\mathrm{d}x = \int_\Omega \frac{|\psi|}{\sqrt{\rho}}\sqrt{\rho}\ \mathrm{d}x \leqslant \left(\int_\Omega \frac{|\psi|^2}{\rho}\ \mathrm{d}x\right)^{1/2}\left(\int_\Omega \rho\ \mathrm{d}x\right)^{1/2}$$
$$= \left(\int_\Omega \frac{|\psi|^2}{\rho}\ \mathrm{d}x\right)^{1/2},$$

  which also verifies that $\mathrm{V}(\hat{\Psi}) \geqslant 0$. More importantly, the equalities hold if ψ does not change sign and $\rho \propto |\psi|$. Therefore, we can set ρ as close to $|\psi|$ (up to a normalizing constant) as possible to reduce the variance, but meanwhile ensure ρ is easy to sample

from and evaluate as required in (15). This is closely related to the concept of importance sampling.

- The result (15) and (16) in lemma 3 can be easily extend to the case with unbounded domain Ω, provided that $\psi/\sqrt{\rho} \in L^2(\Omega)$.

Lemma 3 provides a feasible way to approximate the gradient of $L(\theta)$ for (14). For instance, to compute $\nabla_\theta L_{\text{bdry}}(\theta)$, we can take gradient of (12) with respect to θ, sample N_b collocation points $\{x_b^{(i)} : 1 \leqslant i \leqslant N_b\}$ on the boundary $\partial\Omega$ and approximate $\nabla_\theta L_{\text{bdry}}(\theta)$ by summation of function evaluations at the sample points. If we take $\mathcal{B}[u,\gamma] = (u - u_b, \gamma - \gamma_b, \partial_{\vec{n}} u - u_n)$ and uniformly sample $x_b^{(i)}$, the estimate becomes

$$\begin{aligned}
\nabla_\theta L_{\text{bdry}}(\theta) = 2\int_{\partial\Omega} &((u_\theta - u_b)\nabla_\theta u_\theta + (\gamma_\theta - \gamma_b)\nabla_\theta \gamma_\theta \\
&+ (\partial_{\vec{n}} u_\theta - u_n)\nabla_\theta \nabla u \cdot \vec{n})\, \mathrm{d}S(x) \\
\approx \frac{2|\partial\Omega|}{N_b}\sum_{i=1}^{N_b} &\Big((u_\theta(x_b^{(i)}) - u_b(x_b^{(i)}))\nabla_\theta u_\theta(x_b^{(i)}) \\
&+ (\gamma_\theta(x_b^{(i)}) - \gamma_b(x_b^{(i)}))\nabla_\theta \gamma_\theta(x_b^{(i)}) \\
&+ (\partial_{\vec{n}} u_\theta(x_b^{(i)}) - u_b(x_b^{(i)}))\nabla_\theta \nabla u_\theta x_b^{(i)} \cdot \vec{n} x_b^{(i)}\Big).
\end{aligned} \tag{17}$$

Similarly, we can compute the stochastic gradient of $\nabla_\theta L_{\text{int}}(\theta)$. In the case of taking $\mathcal{A}[u,\gamma] = \nabla\cdot(\gamma\nabla u) - f$ in Ω with f given, and uniformly sampling N_r collocation points $\{x_r^{(i)} : 1 \leqslant i \leqslant N_r\}$ inside the region Ω, $\nabla_\theta L_{\text{int}}(\theta)$ can be estimated by

$$\begin{aligned}
\nabla_\theta L_{\text{int}}(\theta) &= 2I(\theta)\int_\Omega \left(\nabla_\theta \gamma_\theta(\nabla u_\theta \cdot \nabla \varphi_{\eta(\theta)}) + \gamma_\theta(\nabla_\theta \nabla u_\theta \cdot \nabla \varphi_{\eta(\theta)})\right) \mathrm{d}S(x) \\
&\approx \frac{2|\Omega|\hat{I}(\theta)}{N_r}\sum_{i=1}^{N_r} \left(\nabla_\theta \gamma_\theta(x_r^{(i)})(\nabla u_\theta(x_r^{(i)})\nabla \varphi_{\eta(\theta)}(x_r^{(i)}))\right. \\
&\quad \left. + \gamma_\theta(x_r^{(i)})(\nabla_\theta \nabla u_\theta(x_r^{(i)})\nabla \varphi_{\eta(\theta)}(x_r^{(i)}))\right)
\end{aligned} \tag{18}$$

where $I(\theta)$ and its estimator $\hat{I}(\theta)$ are given by

$$I(\theta) = \int_\Omega \gamma_\theta(\nabla u_\theta \cdot \nabla \varphi_{\eta(\theta)})\mathrm{d}x, \qquad \hat{I}(\theta) = \frac{|\Omega|}{2}\sum_{i=1}^{N_r} \gamma_\theta(x_r^{(i)})(\nabla u_\theta(x_r^{(i)}) \cdot \nabla \varphi_{\eta(\theta)}(x_r^{(i)})),$$

and $\eta(\theta)$ is a solution of the maximization problem (11) according to lemma 2. All integrals in the gradients can be approximated in a similar way. These approximated gradients are in fact stochastic gradients, which are unbiased and have bounded variances due to the boundedness of the network parameters. With these approximations, (14) reduces to the stochastic projected gradient descent method, which ensures convergence to a local stationary point of (13) with proper choice of step sizes. Since (13) is constrained, the *gradient mapping*, defined by $\mathcal{G}(\theta) := \tau^{-1}[\theta - \Pi(\theta - \tau\nabla_\theta L(\theta))]$, is used as the convergence criterion of θ [22, 42, 52]. Note that the definition of gradient mapping takes the normalization of step size τ into consideration. Moreover, without the projection Π, the gradient mapping reduces to $\mathcal{G}(\theta) = \nabla_\theta L(\theta)$, whose magnitude is an evaluation criterion for local stationary points (i.e., $|\nabla_\theta L(\theta)| = 0$) for

Algorithm 1. IP solver by weak adversarial network (IWAN).

Input: the domain Ω and data for the IP (1).
Initialize: $(u_\theta, \gamma_\theta)$, φ_η.
for $j = 1, \ldots, J$: **do**
  Sample $X_r = \{x_r^{(i)} : 1 \leqslant i \leqslant N_r\} \subset \Omega$ and $X_b = \{x_b^{(i)} : 1 \leqslant i \leqslant N_b\} \subset \partial\Omega$.
  $\eta \leftarrow \mathrm{SGD}(-\nabla_\eta E(\theta, \eta), X_r, \eta, \tau_\eta, J_\eta)$.
  $\theta \leftarrow \mathrm{SGD}(\partial_\theta E(\theta, \eta) + \beta \nabla_\theta L_{\mathrm{bdry}}(\theta), (X_r, X_b), \theta, \tau_\theta, 1)$.
end for
Output: $(u_\theta, \gamma_\theta)$.

unconstrained case. This result is stated in the following theorem, and the proof is given in appendix A.4.

Theorem 4. *For any $\varepsilon > 0$, let $\{\theta_j\}$ be a sequence of the network parameter in $(u_\theta, \gamma_\theta)$ generated by the gradient descent algorithm (14) with integrals in $\nabla_\theta L(\theta)$ approximated by sample averages as in (15) with sample complexities $N_r, N_b = O(\varepsilon^{-1})$ in each iteration, then $\min_{1 \leqslant j \leqslant J} \mathbb{E}[|\mathcal{G}(\theta_j)|^2] \leqslant \varepsilon$ after $J = O(\varepsilon^{-1})$ iterations.*

Remark 3.3. Theorem 4 establishes the convergence and iteration complexity of (14) to the so-called ε-solution of the problem. The result is based on the expected magnitude of the gradient mapping, which is a standard convergence criterion in nonconvex constrained stochastic optimization. However, this only ensures approximation to a stationary point (not necessarily a local or global minimizer) on expectation. In theory, one can apply additional global optimization techniques to (7) in order to find a global minimizer (possibly only with high probability at best) with substantially higher computational cost. However, we will not exploit this issue further in this work.

Now we summarize the steps of our algorithm for solving IPs using weak adversarial networks. To simplify the presentation, we introduce the following notation to indicate the SGD procedure for finding a minimizer of a loss function $L(\theta)$:

$$\theta^* \leftarrow \mathrm{SGD}(G(\theta), X, \theta_0, \tau, J), \tag{19}$$

which means the output θ^* is the result θ_J after we execute the (projected) SGD scheme with step size τ below for $j = 0, \ldots, J-1$ with initial θ_0:

$$\theta_{j+1} \leftarrow \Pi(\theta_j - \tau \hat{G}(\theta_j; X)). \tag{20}$$

Here $X = \{x^{(i)} : 1 \leqslant i \leqslant N\}$ is the set of N sampled collocation points, $G(\theta) := \nabla_\theta L(\theta)$ is the gradient of the loss function $L(\theta)$ to be minimized, and $\hat{G}(\theta; X)$ stands for the stochastic approximation of $G(\theta)$ at any given θ, where the integrals are estimated as in (15) using the sampled collocation points X. Therefore, each iteration of our algorithm consists of two steps. In step 1, we fix θ and solve the maximization problem with objective function $E(\theta, \eta)$ defined in (11) by applying stochastic gradient ascent for J_η steps to obtain an approximate maximizer η; in step 2, we fix this η, and update θ by one SGD step using gradient $\nabla_\theta L(\theta) = \partial_\theta E(\theta, \eta) + \beta \nabla_\theta L_{\mathrm{bdry}}(\theta)$. Then we go to step 1 to start the next iteration. Hence, our objective function is $E(\theta, \eta) + \beta L_{\mathrm{bdry}}(\theta)$, for which we seek for the optimal point (θ^*, η^*) via a min–max optimization $\min_\theta \max_\eta E(\theta, \eta) + \beta L_{\mathrm{bdry}}(\theta)$. This procedure is referred to IP solver using weak adversarial network (IWAN) and summarized in algorithm 1. The parameter values in our numerical implementations are presented in section 4.

4. Numerical experiments

4.1. Implementation details

In this subsection, we discuss several implementation details and modifications regarding algorithm 1. First, to avoid spending excessive time in solving the inner maximization problem $\max_\eta E(\theta, \eta)$ in (11) with a fixed θ, we only apply a few iterations J_η to compute η. Then we switch to update θ for one iteration. See the two SGD steps in algorithm 1. This can improve overall efficiency and avoid spending excessive time on the inner maximization problem of η, especially when θ is still far from optimal yet. In fact, we can employ two separate test functions φ_η and $\bar{\varphi}_\eta$ (we again use the same η for notation simplicity). In each iteration j, we alternately update $(u_\theta, \varphi_\eta, \gamma_\theta, \bar{\varphi}_\eta)$ in order, each with one or a few SGD steps (20). We will specify the numbers of steps for these networks for our experiments below.

During the derivations in section 3, we require bounded network parameters θ and η, where the bound B can be arbitrarily large, to ensure finite variances of the integral estimators using samples so that the SGD is guaranteed to converge. An alternative way to handle the boundedness constraints is to add $|\theta|^2$ and $|\eta|^2$ as regularization terms to the objective function in (7). One can also use the operator norm (5) with denominator replaced by $\|\varphi\|_2^2 := \int_\Omega |\varphi|^2 \, \mathrm{d}x$ (approximated by MC similarly as in (15)), which is also adopted in our implementation. This replacement does not cause issue in numerical implementation since the test function φ_η is realized by a network with fixed width/depth and bounded parameters, and hence is guaranteed to be in H^1.

A test function φ_η is required to vanish on $\partial\Omega$ in the weak formulation (3). One simple technique to ensure this is to precompute a function $\varphi_0 \in C(\Omega)$ such that $\varphi_0(x) = 0$ if $x \in \partial\Omega$ and $\varphi_0(x) > 0$ if $x \in \Omega$ (e.g., a distance function to $\partial\Omega$ would work). Then we seek for a parameterized network φ'_η with no constraint on its arbitrary boundary, and set the test function φ_η to $\varphi_0 \varphi'_\eta$ which still takes zero value on $\partial\Omega$.

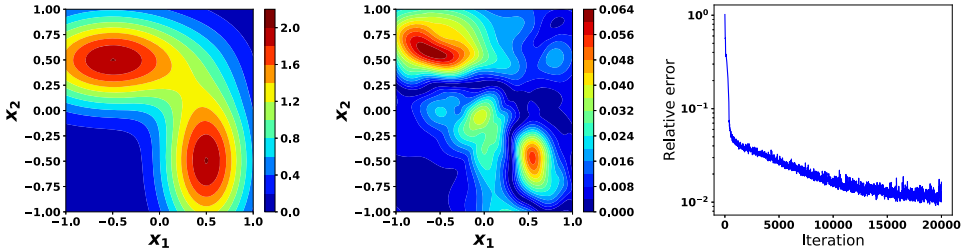
We implemented our algorithm using TensorFlow [1] (Python version 3.7), a state-of-the-art deep learning package that can efficiently employ GPUs for parallel computing. The gradients with respect to network parameters (θ and η) and input (x) are computed by the TensorFlow builtin auto-differentiation module. During training, we can also substitute the standard SGD optimizer by many of its variants, such as AdaGrad, RMSProp, Adam, Nadam etc. In our experiments, we use AdaGrad supplied by the TensorFlow package, which appears to provide better performance than other optimizers in most of our tests. All other parameters, such as the network structures (numbers of layers and neurons), step sizes (also known as the learning rates), number of iterations, will be specified in section 4.

4.2. Experiment setup

In this section, we conduct a set of numerical experiments to show the practical performance of algorithm 1 in solving IP. To quantitatively evaluate the accuracy of an approximate solution γ, we use the *relative error* (in the L^2 sense) of γ to the ground truth γ^*, defined by $\|\gamma - \gamma^*\|_2 / \|\gamma^*\|_2$, where $\|\gamma\|_2^2 := \int_\Omega |\gamma(x)|^2 \, \mathrm{d}x$. In practice, we compute $\|\gamma\|_2^2 = (|\Omega|/N) \cdot \sum_{i=1}^N |\gamma(x^{(i)})|^2$ by evaluating γ on a fixed set of N mesh grid points $\{x^{(i)} \in \Omega : 1 \leqslant i \leqslant N\}$ in Ω. More specifically, we used a regular mesh grid of size 100×100 for (x_1, x_2), and sampled one point x for each of these grid points, i.e., for each grid point (x_1, x_2), randomly draw values of the other coordinates within the domain Ω such that $N = 10^4$. These points were sampled in advance and then used for all comparison algorithms to compute their test relative error. Note that these points are different from those sampled for training in these methods.

Table 1. Problem settings for tests 1–4 in subsection 4.3, where γ^* denotes the true conductivity, u^* denotes the true potential, and f is the source function.

Problem	γ^*	u^*	f
Test 1	$2(\exp(-\lvert x-c_1\rvert^2_{\Sigma_1})+\exp(-\lvert x-c_2\rvert^2_{\Sigma_2}))$ where $\Sigma_1=\mathrm{diag}(1.25,5,0,0,0)$, $\Sigma_2=\mathrm{diag}(5,1.8,0,0,0)$, $c_1=(-0.5,0.5,0,0,0)$, $c_2=(0.5,-0.5,0,0,0)$, and $\lvert x\rvert^2_\Sigma \coloneqq x^\top\Sigma x$	$\cos(\lvert x\rvert^2)$	$8\sum_{i,j=1}^2[\Sigma_j]_{ii}(x_i-[c_j]_i)x_i\ \sin(\lvert x\rvert^2)\exp(-\lvert x-c_j\rvert^2_{\Sigma_j})$ $+\gamma^*(2d\sin(\lvert x\rvert^2)+4\lvert x\rvert^2\cos(\lvert x\rvert^2))$ where $[\Sigma]_{ij}$ and $[c]_j$ stand for the (i,j)th entry of the matrix Σ and jth component of the vector c, respectively
Test 2 & Test 3	$0.5+1.5/(1+\delta(x))$ where $\delta(x)=\exp((\lvert x-c\rvert^2_\Sigma-0.6^2)/0.02)$ with $\Sigma=\mathrm{diag}(0.81,2,0.09,\ldots,0.09)$ and $c=(0.1,0.3,0,\ldots,0)$	$\lvert x\rvert^2$	$\frac{6(x-c)^\top\Sigma x}{\lambda(1/\delta(x)+2+\delta(x))}-2d\gamma^*$
Test 4(1)	$0.5+3.5/(1+\delta_1(x))+1.5/(1+\delta_2(x))$ where $\delta_1(x)=\exp((\lvert x-c_1\rvert^2_{\Sigma_1}-0.4^2)/0.02)$ and $\delta_2(x)=\exp((\lvert x-c_2\rvert^2_{\Sigma_2}-0.4^2)/0.02)$ with $\Sigma_1=\mathrm{diag}(0.81,2,0.09,0.09,0.09)$, $\Sigma_2=\mathrm{diag}(2,0.81,0.09,0.09,0.09)$, and $c_1=(-0.5,-0.5,0,0,0)$, $c_2=(0.5,0.5,0,0,0)$	$\lvert x\rvert^2$	$\frac{14(x-c_1)^\top\Sigma_1 x}{\lambda(1/\delta_1(x)+2+\delta_1(x))}+\frac{6(x-c_2)^\top\Sigma_2 x}{\lambda(1/\delta_2(x)+2+\delta_2(x))}$ $-2d\gamma^*$
Test 4(2)	$0.5+1.5/(1+\delta_1(x))+1.5/(1+\delta_2(x)+\delta_3(x))$ where $\delta_1(x)=\exp((\lvert x-c\rvert^2_{\Sigma_2}-0.4^2)/0.02)$ $\delta_2(x)=\exp((\lvert x_1+0.5\rvert-0.15)/0.02)$ and $\delta_3(x)=\exp((\lvert x_2\rvert-0.6)/0.02)$ with $c=(0.55,0,0,0,0)$ and $\Sigma=\mathrm{diag}(1,4,0,0,0)$	$\lvert x\rvert^2$	$\frac{6(x-c)^T\Sigma x}{\lambda(1/\delta_1(x)+2+\delta_1(x))}+\frac{3(\lvert x_1+0.5\rvert\delta_2(x)+\lvert x_2\rvert\delta_3(x))}{\lambda(1+\delta_2(x)+\delta_3(x))^2}$ $-2d\gamma^*$
Test 4(3)	$0.5+\sum_{j=1}^3 1.5/(1+\delta_{j1}(x)+\delta_{j2}(x))$ where $\delta_{j1}(x)=\exp((\lvert x_1-c_j(1)\rvert-r_j(1))/0.02)$, $\delta_{j2}(x)=\exp((\lvert x_2-c_j(2)\rvert-r_j(2))/0.02)$, for $j=1,2,3$ with $c_1=(-0.5,0)$, $c_2=(-0.1,0.6)$, $c_3=(-0.1,-0.6)$ and $r_1=(0.15,0.8)$, $r_2=r_3=(0.55,0.2)$	$\lvert x\rvert^2$	$3\sum_{j=1}^3\frac{\lvert x_1-c_j(1)\rvert\delta_{j1}(x)+\lvert x_2-c_j(2)\rvert\delta_{j2}(x)}{\lambda(1+\delta_{j1}(x)+\delta_{j2}(x))^2}$ $-2d\gamma^*$



(a) Ground truth γ^*. (b) Pointwise error $|\gamma^* - \gamma_\theta|$ (c) Relative error of γ_θ.

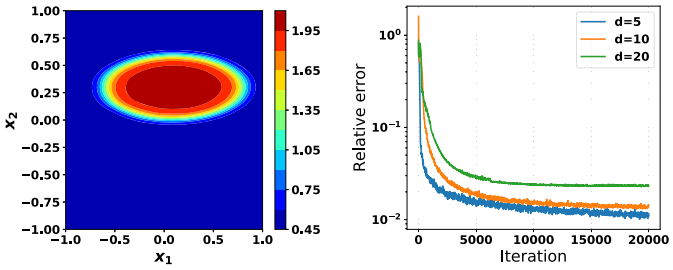
Figure 1. Test 1 result on (2) with smooth γ^* and problem dimension $d = 5$.

In all of our experiments, we parameterize each of $(u_\theta, \varphi_\eta, \gamma_\theta, \bar{\varphi}_\eta)$ as a nine-layer fully connected neural network with 20 neurons per layer as in (8) unless otherwise noted. We set σ_k to `tanh` for $k = 1, 2$, `softplus` for $k = 4, 6, 8$, `sinc` for $k = 3, 5, 7$ in u_θ, and `tanh` for $k = 1, 2, 4, 6$, `elu` for $k = 3, 5$, and `sigmoid` for $k = 7, 8$ in γ_θ. We use `elu` in the output layer of γ_θ. In parallel, we set σ_k to `tanh` for $k = 1, 2$ and `sinc` for $k \geqslant 3$ in φ_η and $\bar{\varphi}_\eta$. Unless otherwise noted, we apply one SGD update with step size $\tau_\theta = 0.01$ to both of u_θ and γ_θ (each of them performs the θ update in algorithm 1), and two ($J_\eta = 2$) SGD updates with step size $\tau_\eta = 0.008$ to both of φ_η and $\bar{\varphi}_\eta$ (each of them performs the η update in algorithm 1), following the order of $u_\theta, \varphi_\eta, \gamma_\theta, \bar{\varphi}_\eta$ in every iteration j of algorithm 1. We set the weight $\beta = 10\,000$ for the boundary loss function $L_{\text{bdry}}(\theta)$ in (7), but also set a weight β' to $L_{\text{int}}(\theta)$ and specify its value in the experiment. Other parameters will also be specified below. All the experiments are implemented, trained, and tested in the TensorFlow framework [1] on a machine equipped with Intel 2.3 GHz CPU and an Nvidia Tesla P100 GPU and 16 GB of graphics card memory.

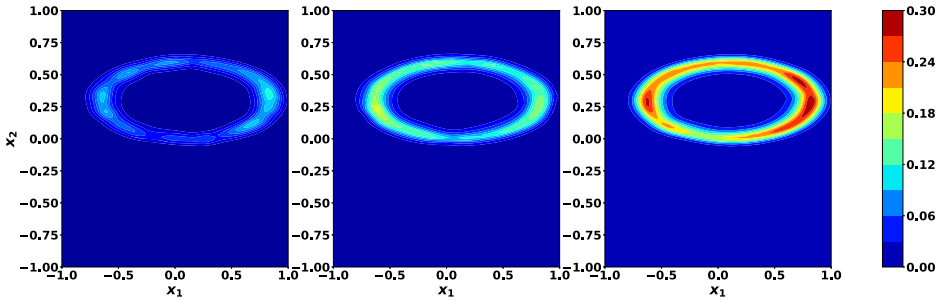
4.3. Experimental results on inverse conductivity problems

Test 1: inverse conductivity problem with smooth γ. We first test our method on the inverse conductivity problem (2) with a smooth conductivity distribution γ. In this test, we set $\Omega = (-1, 1)^d \subset \mathbb{R}^d$ with problem dimension $d = 5$. The setup for the ground truth conductivity distribution γ^*, the ground truth potential u^* and the source term f are provided in the table 1 which is in the appendix B. We set $N_r = 100\,000$ and $N_b = 100d$, $\beta' = 10$, and run algorithm 1 for 20 000 iterations. The true γ^* and the point-wise error $|\gamma^* - \gamma_\theta|$ (with relative error 2.54%) are shown in figures 1(a) and (b) respectively. The progress of the relative error of γ_θ versus iteration number is shown in figure 1(c) (all plots of relative error versus iteration number in this section are shown using moving average with a window size 7). For the demonstration purpose, only the (x_1, x_2) cross sections that have main spatial variations are shown (same for the other test results below).

Test 2: inverse conductivity problem with nearly piecewise constant γ. We consider (2) with a less smooth, nearly piecewise constant conductivity γ. In this test, we set $\Omega = (-1, 1)^d$, define $\Omega_0 = \{x \in \Omega : |x - c|_\Sigma^2 \leqslant 0.6^2\}$ where $\Sigma = \text{diag}(0.81, 2, 0.09, \ldots, 0.09)$ and $c = (0.1, 0.3, 0, \ldots, 0)$, and set γ^* to 2 in Ω_0 and 0.5 in Ω_0^c. For ease of implementation, we slightly smooth the ground truth conductivity. One can find the setup of the smoothed conductivity γ^*, the ground truth potential u^*, and the source term f in the table 1. Then we solve the IP (2) with dimensionality $d = 5, 10, 20$. We set $N_r = 20\,000d$ and $N_b = 100d$, and $\beta' = 10, 1, 0.005$ for $d = 5, 10, 20$ respectively. In each case, we run algorithm 1 for 20 000



(a) Ground truth conductivity γ^* (left) and relative error versus iteration (right).



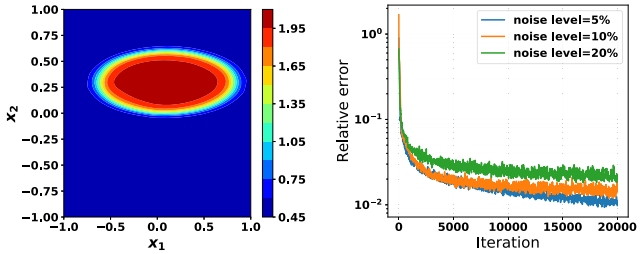
(b) From left to right: absolute error $|\gamma_\theta - \gamma^*|$ for $d = 5, 10, 20$.

Figure 2. Test 2 result on (2) with nearly piecewise constant γ^* and problem dimension $d = 5, 10, 20$ without measurement noise.

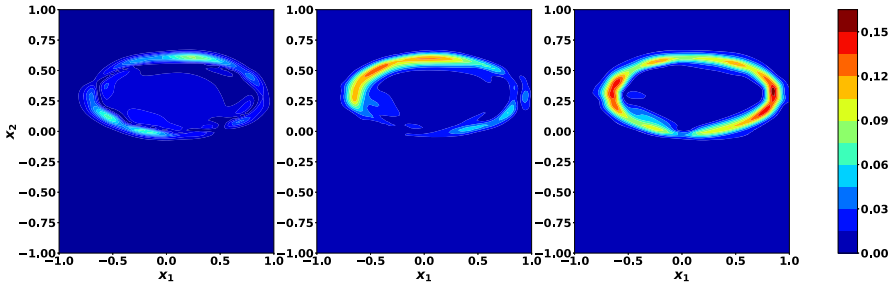
iterations, and obtain relative errors $1.16\%, 1.43\%, 2.29\%$ for $d = 5, 10, 20$, respectively. The recovery results are shown in figure 2. Figure 2(a) shows the ground truth γ^* (left) and the progress of relative errors versus iteration number for different d (right). The pointwise absolute errors $|\gamma_\theta - \gamma^*|$ for these dimensions are shown in figure 2(a).

Test 3: inverse conductivity problem with noisy measurements. Under the same experiment setting as test 2, we solve the IP (2) where the measurement data are perturbed by random noise for the $d = 5$ case. Specifically, we scale every measurement data value by $1 + 5\%e, 1 + 10\%e$, $1 + 20\%e$ where e is drawn independently from the standard normal distribution every time, followed by a truncation into interval $[-100, 100]$. We do not perturb f. The results are given in figure 3 in parallel to the noiseless case above, where figure 3(a) shows the ground truth conductivity γ^* (left) and the progress of relative error of γ_θ versus iteration number (right). The pointwise absolute error after 20 000 iterations $|\gamma_\theta - \gamma^*|$ with noise levels $5\%, 10\%, 20\%$ are shown in figure 3(b). We observe that the progress becomes more oscillatory due to the random measurement noise in figure 3(a), and the final reconstruction error is larger for higher noise level in figure 3(b) as expected.

Test 4: inverse conductivity problem with different features in γ. We consider several cases with more challenging ground truth conductivity γ^*. The first case has γ^* with two disjoint modes. We define $\Omega = (-1, 1)^5$, and set $\Omega_1 = \{x : |x - c_1|^2_{\Sigma_1} \leqslant 0.4^2\}$ and $\Omega_2 = \{x : |x - c_2|^2_{\Sigma_2} \leqslant 0.4^2\}$, where $c_1 = (-0.5, -0.5, 0, 0, 0)$, $c_2 = (0.5, 0.5, 0, 0, 0)$, $\Sigma_1 = \mathrm{diag}(0.81, 2, 009, 0.09, 0.09)$, and $\Sigma_2 = \mathrm{diag}(2, 0.81, 009, 0.09, 0.09)$. We set the conductivity γ^* to 4 in Ω_1, 2 in Ω_2, and 0.5 in $(\Omega_1 \cup \Omega_2)^c$. We also smooth γ^* using a Gaussian kernel. The setup of the ground truth conductivity γ^*, the ground truth potential u^*, and



(a) Ground truth conductivity γ^* (left) and relative error of γ_θ versus iteration (right).

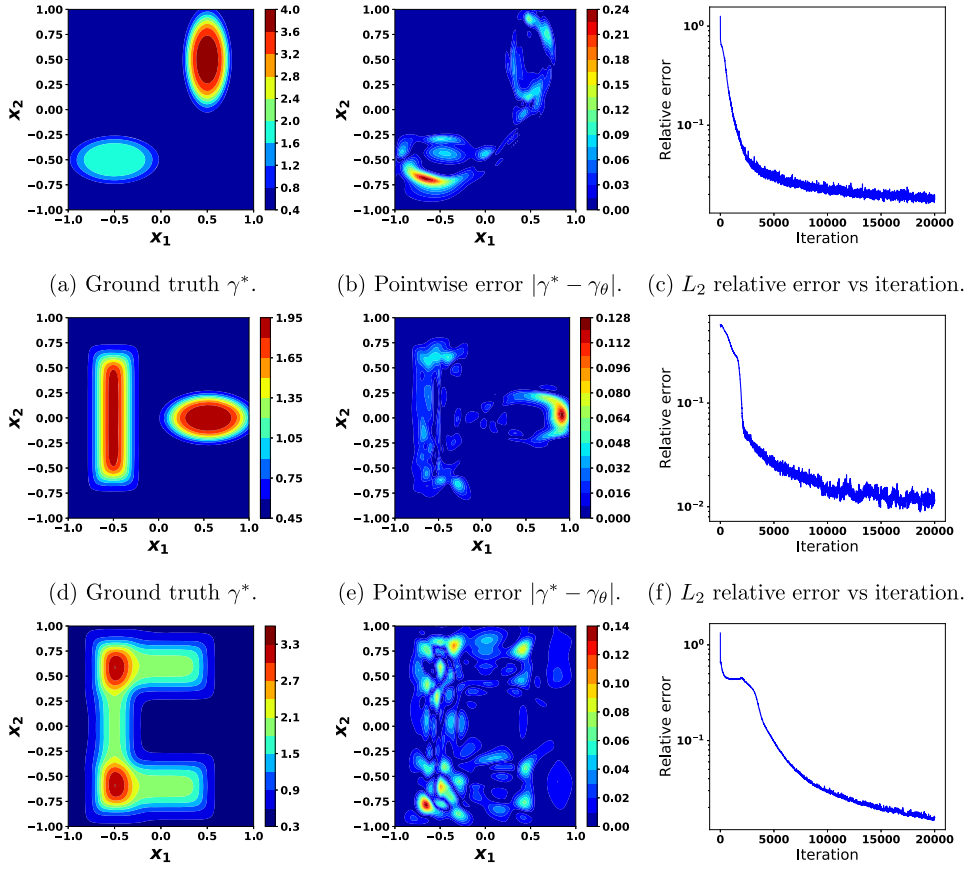


(b) From left to right: absolute error $|\gamma_\theta - \gamma^*|$ with measurement noise level $5\%, 10\%, 20\%$.

Figure 3. Test 3 result on (2) with nearly piecewise constant γ^* and noisy data.

the source function $f(x)$ are provided in the table 1 (see test 4(1)). We set $N_r = 200,000$, $N_b = 100d$, $\beta' = 10$, and run algorithm 1 for 20 000 iterations. Figure 4(a) shows the ground truth γ^* (left) and the recovered conductivity γ_θ with relative error 1.77% (right). Figure 4(c) plots the progress of relative error of γ_θ versus iteration number. In the second case, we follow the same setting but define $\Omega_1 = \{x : |x_1 + 0.5| \leqslant 0.15, |x_2| \leqslant 0.6\}$ (which has sharp corner) and $\Omega_2 = \{x : |x - c|^2_\Sigma \leqslant 0.4^2\}$ where $c = (0.55, 0, 0, 0, 0)$, $\Sigma = \mathrm{diag}(1, 4, 0, 0, 0)$, and set $\gamma^* = 2$ in $\Omega_1 \cup \Omega_2$ and 0.5 in $(\Omega_1 \cup \Omega_2)^c$. We again smooth γ^* and provide the ground truth conductivity γ^*, the ground truth potential u^*, and the source function f in the table 1 (see test 4(2)). We set $\beta' = 1$ and again run algorithm 1 for 20 000 iterations. The recovered γ_θ (with relative error 1.15%) and the progress of relative error are shown in figures 4(d) and (f) respectively. Lastly, we consider a non-convex shaped γ, and show the recovered γ_θ (with relative error 1.57%) and the progress of relative error in figures 4(g) and (i) respectively. We set the domain $\Omega = (-1, 1)^5$ and define $\Omega_j = \{x \in \Omega : \sum_{i=1}^2 |x_i - c_j(i)| \leqslant r_j(i)\}$, $j = 1, 2, 3$, where $c_1 = (-0.5, 0)$, $c_2 = (-0.1, 0.6)$, $c_3 = (-0.1, -0.6)$ and $r_1 = (0.15, 0.8)$, $r_2 = r_3 = (0.55, 0.2)$. We set $\gamma^* = 4$ in $\Omega_0 = (\Omega_1 \cap \Omega_2) \cup (\Omega_1 \cap \Omega_3)$ and 2 in $\Omega_1 \cup \Omega_2 \cup \Omega_3 / \Omega_0$ and 0.5 in $(\Omega_1 \cup \Omega_2 \cup \Omega_3)^c$. Once again, we sightly smooth the conductivity γ^* and provide the setup of the ground truth conductivity γ^*, the ground truth potential u^* and the source function f in the table 1 (see test 4(3)).

Test 5: EIT problem. We consider an artificial 5D EIT problem (2) on $\Omega = (0, 1)^5$ but replace (2b) with a different boundary condition given by $\gamma \nabla u \cdot \vec{n}$ on $\partial\Omega$, where $\vec{n}$ is the outer normal vector at the boundary point. We define $\Gamma_1 = \{x \in \partial\Omega : x_1 = 0, 1\}$ and $\Gamma_2 = \partial\Omega \backslash \Gamma_1$. We set ground truth conductivity $\gamma^*(x) = \pi^{-1} \exp\{(d-1)\pi^2(x_1 - x_1^2)/2\}$ and potential $u^*(x) = \exp\{(d-1)\pi^2(x_1^2 - x_1)/2\} \cdot \Pi_{i=2}^d \sin(x_i)$, and compute the corresponding boundary



(a) Ground truth γ^*. (b) Pointwise error $|\gamma^* - \gamma_\theta|$. (c) L_2 relative error vs iteration.

(d) Ground truth γ^*. (e) Pointwise error $|\gamma^* - \gamma_\theta|$. (f) L_2 relative error vs iteration.

(g) Ground truth γ^*. (h) Pointwise error $|\gamma^* - \gamma_\theta|$. (i) L_2 relative error vs iteration.

Figure 4. Test 4 results on (2) with problem dimension $d = 5$. (a)–(c) Two separate modes in γ^*; (d)–(f) two separate modes (one has sharp corner) in γ^*; (g)–(i) nonconvex shaped γ^*.

value as our input data. We set the source term $f = 0$ in (2a), $N_r = 100\,000$, $N_b = 100d$, $\beta' = 10$, and run algorithm 1 for 20 000 iterations. The x_1 cross section of the recovered γ_θ (with relative error 0.56%) and the progress of relative error versus iteration number are shown in figures 5(a) and (b), respectively.

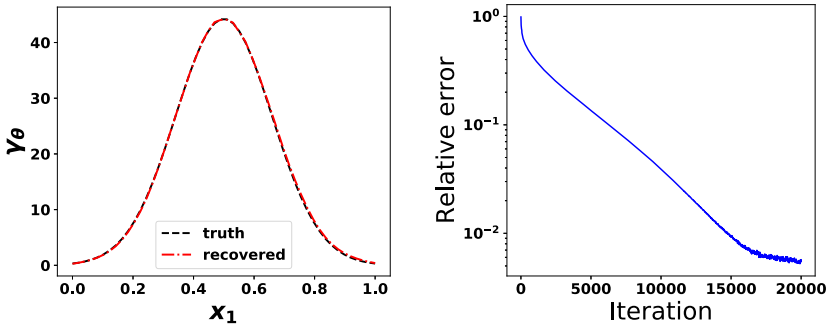
Test 6: inverse thermal conductivity problem involving time. We consider an inverse thermal conductivity problem of (1) with temporally varying $u(x, t)$ as follows,

$$\partial_t u - \nabla \cdot (\gamma \nabla u) - f = 0, \quad \text{in } \Omega_T = \Omega \times [0, T] \tag{21a}$$

$$u - u_i = 0, \quad \text{in } \Omega \times \{0\} \tag{21b}$$

$$\nabla u \cdot \vec{n} - u_n = 0, \qquad u - u_b = 0, \qquad \gamma - \gamma_b = 0, \quad \text{on } \partial\Omega \times [0, T] \tag{21c}$$

where $\Omega = (0, 1)^5 \subset \mathbb{R}^5$ and the final time $T = 1$, γ is the thermal conductivity, u indicates the temperature and $f(x, t)$ is the source function which indicates the rate of heat generation per unit volume, where $\vec{n}$ is the unit outer normal vector of $\partial\Omega$, $u_i(x)$ for $x \in \Omega$ is the given initial



(a) True γ^* vs recovered γ_θ. (b) L_2 relative error vs iteration.

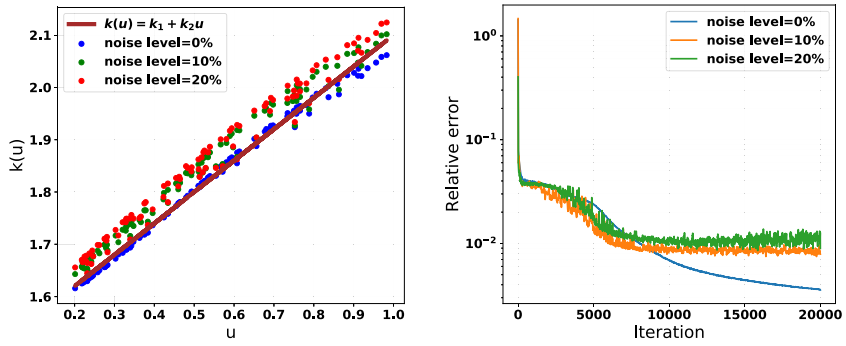
Figure 5. Test 5 result on artificial 5D EIT problem (2) with boundary condition on $\gamma\nabla u \cdot \vec{n}$.

value, and $u_n(x,t), u_b(x,t), \gamma_b(x,t)$ for $(x,t) \in \partial\Omega \times [0,T]$ are given boundary values. In this problem, we would like to recover the thermal conductivity $\gamma(u)$ which is a function of the temperature u in the standard setting, but we simply treat $\gamma(x,t) := \gamma(u(x,t))$ as a function of (x,t) in our experiment here. We set the source function $f(x,t)$ as follows,

$$f(x,t) = \pi^2 \left(k_1 + k_2 s(t) \sum_{i=1}^{d} \sin(\pi x_i) - \lambda_2 \right) s(t) \sum_{i=1}^{d} \sin(\pi x_i) - k_2 \pi^2 s^2(t) \sum_{i=1}^{d} \cos^2(\pi x_i)$$

where $s(t) := \exp(-3t/2)/5$, the initial value $u_i = \lambda_1 \sum_{i=1}^{d} \sin(\pi x_i)$, the Neumann boundary value of u as $u_n(x,t) = \pi s(t)(\cos(\pi x_1), \ldots, \cos(\pi x_d)) \cdot \vec{n}$. We set the ground truth $\gamma^*(x,t) = k_1 + k_2 u^*(x,t)$ where $k_1 = 1.5$, $k_2 = 0.6$, and $u^*(x,t) = s(t) \sum_{i=1}^{d} \sin(\pi x_i)$, and use noisy boundary value measurements $u_b = (1 + \sigma e)u^*$ and $\gamma_b = (1 + \sigma e)\gamma^*$, where e is independently drawn for u_b and γ_b and all $x \in \partial\Omega$ from the standard normal distribution followed by a truncation to $[-100, 100]$, and the noise levels are set to $\sigma = 0\%, 10\%, 20\%$. We consider the case with problem dimension $d = 5$, and set $N_r = 100\,000$, $N_b = 100d$, $\beta' = 1$ and $\beta = 1\,000$. The results are shown in figure 6, where figure 6(a) plots the sampled values of recovered $(u_\theta, \gamma_\theta)$ in comparison with the ground truth relation $\gamma^* = k_1 + k_2 u^*$, and figure 6(b) shows the progress of relative error of γ_θ versus iteration number for the three different noise levels. The reconstructions of γ_θ are all faithful, while higher noise levels decreases the accuracy and make convergence to true solution more challenging.

Test 7: comparison with PINN. We compare the proposed method with a state-of-the-art method called PINNs [51]. PINN is also a deep-learning based method designed for solving forward problem as well as IP for PDEs. PINN is based on the strong form of the PDEs, where the loss function in the minimization problem of PINN consists of the sum of squared errors in the violation of the PDE and the boundary condition at points sampled inside Ω and on $\partial\Omega$, respectively. In contrast, our method is based on the weak form of PDE which employs a test function and yields a min–max problem to better tackle singularities of the problem. We first compare the proposed method and the PINN method in the problem in test 1. Note that PINN was only applied to inverse conductivity problem with constant conductivity in [51], it is straightforward to extend this method to non-constant conductivity by also parameterizing the conductivity γ as an additional deep neural network. In this problem, we take $N_r = 10\,000, N_b = 100 * d$ with $d = 5$. For the PINN method, we also



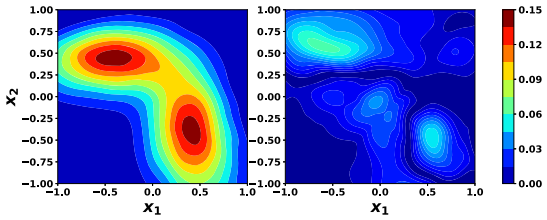
(a) Recovered $(u_\theta, \gamma_\theta)$. (b) relative error of γ_θ.

Figure 6. Test 6 result on inverse thermal conductivity problem with dimension $d = 5$ and noise levels $0\%, 10\%$ and 20%. Ground truth relation between u^* and γ^* is $\gamma^* = k_1 + k_2 u^*$ where $k_1 = 1.5$ and $k_2 = 0.6$.

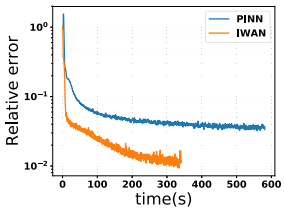
parameterize $u(x)$ as a nine-layer fully-connected network with 20 neurons in each hidden layer and tanh as activation functions in all hidden layers. For γ, we parameterized it by using the same network structure as that used for the proposed method in test 1. We let the weight of the boundary term in the loss function is 1.0 and use the builtin Adam optimizer of TensorFlow with learning rate 0.001 to update the network parameters in PINN. For fair comparison, we also use the Adam optimizer with learning rate 0.001 for updating θ and η in the proposed method. The results after 20 000 iterations of both methods are given in figure 7. In figure 7(a), we can see the error of γ obtained by IWAN is much lower than that by PINN. This can also be seen from figure 7(b), where the error decays very fast for the proposed IWAN. We tried a variety of network structures and parameter settings of PINN and obtain similar results.

We also compared the proposed method and PINN on the inverse conductivity problem in test 2, where the ground truth conductivity γ^* is less smooth and nearly piecewise constant. We use the same parameter settings for both methods as above except for the network structure of γ, which follows the one in test 2. The conductivity γ recovered by PINN and IWAN and the progresses of their relative error versus computation time (in seconds) are given in figures 7(c) and (d), respectively. From figure 7(d), it appears that PINN cannot get close to the ground truth γ^* within 20 000 iterations. Therefore, we rerun PINN for 100 000 iterations, and plot the relative error versus computation time in figure 7(f), from which it seems that PINN still cannot converge to the desired solution. However, the result obtained by PINN does satisfy the PDE closely, as shown in figure 7(e): the difference between the two sides of PDE (left), i.e., $|-\nabla(\gamma\nabla u) - f|$, is much smaller than $|f|$ (right), but PINN cannot capture the irregularities and singularities of the solution since it is based on the strong form of the PDE. In contrast, IWAN can overcome this issue and recover the weak solution properly. We also show the objective function value of PINN and IWAN in figures 7(g) and (h), respectively. (Note that the objective function $L(\theta, \eta)$ in IWAN is defined as $E(\theta, \eta) + \beta L_{\text{bdry}}(\theta)$ for min–max optimization, and the objective function of PINN is for minimization only and hence different from IWAN).

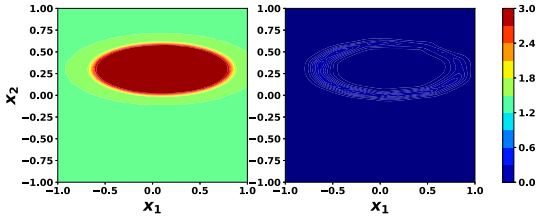
Test 8: efficiency improvement using important sampling. As shown in lemma 3, adaptive sampling may reduce the variance of the sample-based approximation of integrals, which in turn can improve the convergence of SGD. To demonstrate this, we



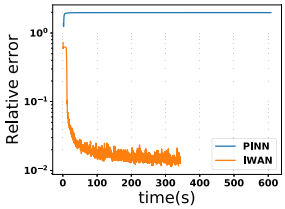
(a) $|\gamma - \gamma^*|$ of PINN (left) and IWAN (right) on smooth γ^*



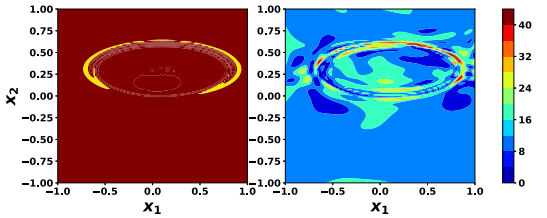
(b) Relative error vs time (s).



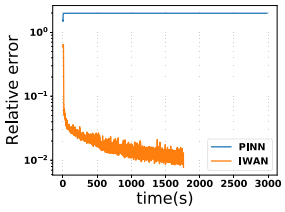
(c) $|\gamma - \gamma^*|$ of PINN (left) and IWAN (right) on less smooth γ^*



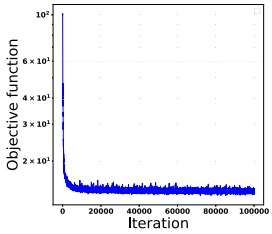
(d) Relative error vs time (s).



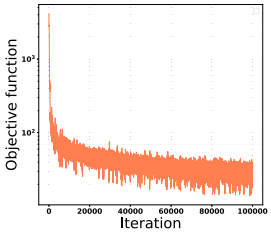
(e) $|f|$ (left) and $|-\nabla(\gamma\nabla u) - f|$ by the PINN (right)



(f) Relative error vs time (s).



(g) Objective vs. iter by PINN



(h) Objective vs. iter by IWAN

Figure 7. Test 7 on the comparison of IWAN and PINN on the recovery smooth ((a) and (b)) and less smooth ((c)–(h)) conductivity γ^*. (a) Pointwise absolute error $|\gamma - \gamma^*|$ with γ obtained by PINN (left) and the proposed method IWAN (right) for smooth γ^*. (b) Relative error versus time in seconds for smooth γ^*. (c) Pointwise absolute error $|\gamma - \gamma^*|$ with γ obtained by PINN (left) and the proposed method IWAN (right) for less smooth, nearly piecewise constant γ^*. (d) Relative error versus time in seconds for 20 000 iterations for less smooth, nearly piecewise constant γ^*. (e) $|f|$ (left) and the map of $|-\nabla(\gamma\nabla u) - f|$ by the PINN (right) for the less smooth γ^*. (f) Relative error versus time in seconds for 100 000 iterations for less smooth, nearly piecewise constant γ^*. (g) Objective function value versus iteration number by PINN for nearly piecewise constant γ^*. (h) Objective function value versus iteration number by the proposed IWAN for nearly piecewise constant γ^*.

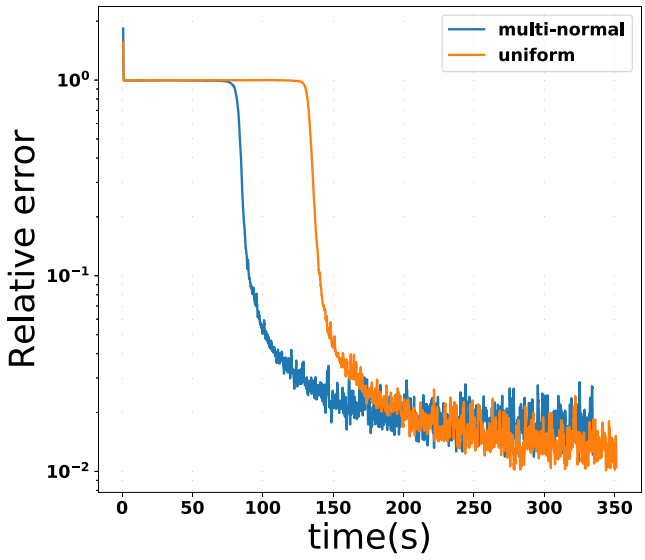


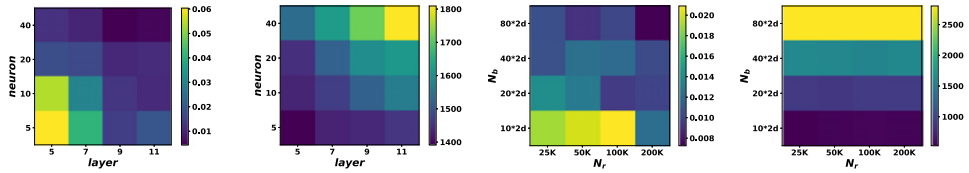
Figure 8. Test 8 result on the difference of relative errors versus computation time (s) using collocations points $\{x_r^{(i)} \in \Omega : i \in [N_r]\}$ sampled from uniform distribution (orange) and adaptive multivariate normal distribution (blue).

consider the inverse conductivity problem on $\Omega = (-1,1)^d \subset \mathbb{R}^d$ with problem dimension $d = 5$, ground truth conductivity distribution $\gamma^*(x) = 2\ \exp(-|x-c|^2_\Sigma/2)$, where $\Sigma = \mathrm{diag}(4.0, 100.0, 0, 0, 0)$, $c_1 = (-0.2, 0.2, 0, 0, 0)$. We set $u_n(x) = -2\ \sin(|x|^2)(x_1, x_2, \ldots, x_d) \cdot \vec{n}$, $u_b = \cos(|x|^2)$ on $\partial\Omega$ and $f(x) = 8\sum_{i=1}^2 \Sigma_{ii}(x_i - c_i)x_i\ \sin(|x|^2)\exp(-|x-c|^2_\Sigma/2) + 2\ \exp(-|x-c|^2_\Sigma/2) * (2d\ \sin(|x|^2) + 4|x|^2\ \cos(|x|^2))$. For the parameter setup, we use the same setup as that in test 7. Then we solve this IP using the proposed method IWAN with points in the domain Ω sampled from the uniform distribution as above and also a multivariate normal distribution respectively. Specifically, to obtain multivariate normal samples, we first sample N_r points of (x_1, x_2) from the multivariate normal distribution with mean value $\mu = (-0.2, 0.2)$ and inverse covariance matrix $\Sigma = \mathrm{diag}(1.0, 25.0)$ (points outside of Ω is discarded), and then draw each of the remaining coordinates randomly from interval $(-1, 1)$ independently. The result was shown in the figure 8. The progress of relative error versus computation time (in second) for 20 000 iterations is shown in figure 8, which shows that the convergence using adaptive multivariate normal distribution is faster than that with uniform distribution.

4.4. Empirical robustness analysis

We conduct a series of experiments to evaluate the robustness of algorithm 1 in terms of network structure (number of layers and neurons) and the number of sampled collocation points.

Test 9: network structure. In this experiment, we test the performance of algorithm 1 with different network structures, i.e., the layer number (network depth) K and the per-layer neuron number (network width) d_k. We test different combinations of K and d' (in each combination we set $d_k = d'$ for all $k = 1, \ldots, K-1$). More specifically, we apply algorithm 1 to the inverse conductivity problem (2) in test 2 above with problem dimension $d = 5$ and



(a) Relative error (b) Time(s) (c) Relative error (d) Time(s)

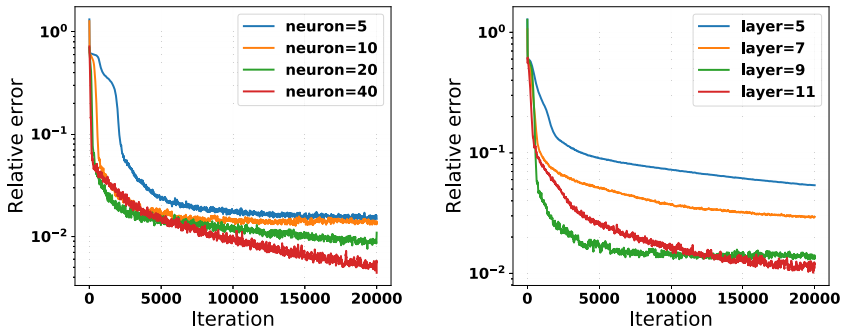
Figure 9. (a) and (b) Test 9 result on relative error of recovered conductivity γ_θ and the corresponding running time using various combinations of (K, d'), where K is the layer number and d' is the per-layer neuron number. (c) and (d) Test 10 result on relative error of recovered conductivity γ_θ and the corresponding running time using various combination of (N_r, N_b), where N_r is the number of sampled collocation points inside the region Ω and N_b is the number of those on the boundary $\partial\Omega$.

Table 2. Test 9 result on relative error of recovered conductivity γ_θ (top) and running time (bottom) using various combinations of (K, d') for problem dimension $d = 5$, where K is the layer number and d' is the per-layer neuron number.

d'	$K = 5$	$K = 7$	$K = 9$	$K = 11$
5	0.060 347	0.040 950	0.014 905	0.019 489
10	0.053 842	0.029 382	0.013 325	0.011 165
20	0.017 955	0.016 862	0.010 916	0.011 490
40	0.012 390	0.010 213	0.004 422	0.005 309
5	1391.76 (s)	1432.90 (s)	1438.87 (s)	1458.12 (s)
10	1435.78 (s)	1468.95 (s)	1520.67 (s)	1568.49 (s)
20	1447.66 (s)	1522.84 (s)	1596.83 (s)	1614.04 (s)
40	1539.93 (s)	1623.20 (s)	1717.53 (s)	1809.30 (s)

a total of 16 combinations (K, d') with $K = 5, 7, 9, 11$ and $d' = 5, 10, 20, 40$. For each combination (K, d'), we run algorithm 1 for 20 000 iterations and plot the relative error of γ_θ in figure 9(a) and the corresponding running time (in seconds) in figure 9(b). The exact values of errors and running times are present in table 2 in appendix C. Based on figure 9(a), it seems that deeper (larger K) and/or wider (larger d') neural networks yield lower reconstruction error (but at the expense of higher per-iteration computational cost). For fixed layer number $K = 9$, we show the progress of relative error versus iteration number with varying per-layer neuron number d' in figure 10(a). Similarly, for fixed per-layer neuron number $d' = 10$, we also show the progress of relative error versus iteration number with varying layer number K in figure 10(b). These figures also suggest that larger K and d' yield better accuracy, although the per-iteration computational cost also increases and it may take more iterations to converge.

Test 10: number of sampled collocation points. In section 3, we showed that the number N of sampled collocation points affects the variance of the integral estimator, so that the variance reduces at the order of $O(1/N)$. In this experiment, we test the empirical effect of the collocation point numbers N_r in the region Ω and N_b on the boundary $\partial\Omega$ in algorithm 1 for the same inverse conductivity problem (2) of dimension $d = 5$ in test 2 above. We choose different combinations of (N_r, N_b) for $N_r = 25K, 50K, 100K, 200K$ $(K = 1000)$ and $N_b = (10 \times 2d, 20 \times 2d, 40 \times 2d, 80 \times 2d)$, and keep all other parameters in test 2 unchanged. We run algorithm 1 for 20 000 iterations, and plot the final relative error of γ_θ in figure 9(a)



(a) Relative error for $K = 9$. (b) Relative error for $d' = 10$.

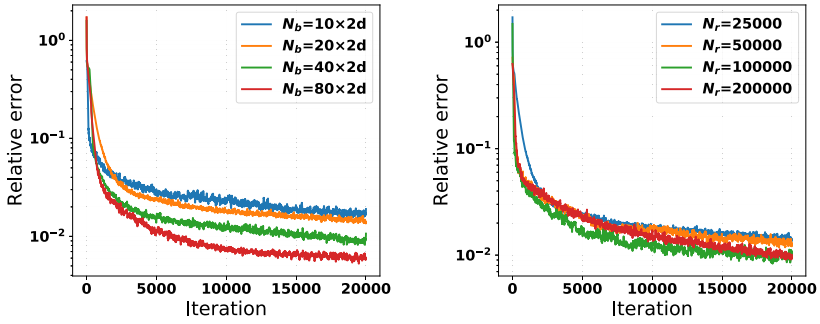
Figure 10. Test 9 result on the effect of network structure. (a) Relative error of γ_θ versus iteration number with fixed layer number $K = 9$ and varying per-layer neuron number $d' = 5, 10, 20, 40$; (b) relative error of γ_θ versus iteration number with fixed per-layer neuron number $d' = 10$ and varying layer number $K = 5, 7, 9, 11$.

Table 3. Test 10 result on relative error of recovered conductivity γ_θ (top) and running time (bottom) with various combination of (N_r, N_b) for problem dimension $d = 5$, where N_r is the number of sampled collocation points inside the region Ω and N_b is the number of those on the boundary $\partial\Omega$.

N_b	$N_r = 25K$	$N_r = 50K$	$N_r = 100K$	$N_r = 200K$
$10 \times 2d$	0.019 023	0.020 007	0.020 898	0.012 208
$20 \times 2d$	0.013 999	0.012 920	0.009 681	0.010 050
$40 \times 2d$	0.010 668	0.012 292	0.012 053	0.010 385
$80 \times 2d$	0.010 61	0.009 207	0.010 200	0.007 267
$10 \times 2d$	528.88 (s)	554.64 (s)	556.08 (s)	552.86 (s)
$20 \times 2d$	915.66 (s)	898.32 (s)	932.98 (s)	928.78 (s)
$40 \times 2d$	1597.66 (s)	1577.03 (s)	1556.92 (s)	1590.87 (s)
$80 \times 2d$	2798.17 (s)	2802.79 (s)	2795.28 (s)	2801.45 (s)

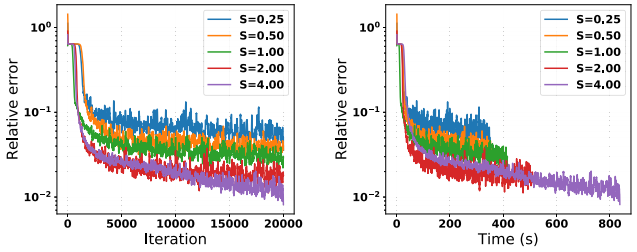
and the corresponding running time (in seconds) in figure 9(b). The exact values of errors and running times are present in table 3 in appendix C. We also plot the progress of relative error of γ_θ versus iteration number for fixed $N_r = 25K$ and varying N_b in figure 11(a), and that for fixed $N_b = 20 \times 2d = 200$ and varying N_r in figure 11(b). The results in figures 9(c) and 11 show that larger amounts of collocation points can generally improve accuracy of the reconstruction.

Test 11: relation between relative error, gradient value and sample points. We conduct an experiment to show the relation between the relative error, the expected value of gradient mapping, and the numbers of collocation points N_r and N_b. In this experiment, we use the Adam optimizer for u_θ, γ_θ with learning rate $\tau_\theta = 0.001$ and use the Adagrad optimizer for φ_η with learning rate $\tau_\eta = 0.008$ and $J_\eta = 1$. We keep other settings unchanged as that used for case $N_r = 25K$ and $N_b = 20 \times 2d$ in test 10. Then, we solve problem (2) of dimension $d = 5$ using the proposed algorithm with $N_r = S \times 25K$

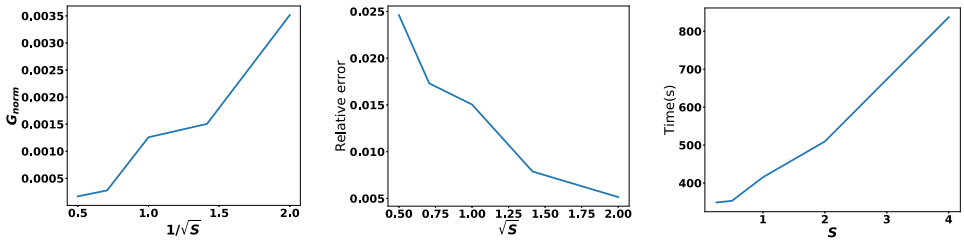


(a) Relative error for $N_r = 25\text{K}$. (b) Relative error for $N_b = 200$.

Figure 11. Test 10 result on the different numbers of collocation points N_r and N_b. (a) Relative error of γ_θ versus iteration number with fixed $N_r = 25K$ and varying N_b; (b) relative error of γ_θ versus iteration number with fixed $N_b = 200$ and varying N_r.



(a) Relatives vs. iteration (left) and time (right)



(b) G_{norm} vs. $1/\sqrt{S}$ (c) Relative error vs. $\sqrt{S}$ (d) Time vs. S

Figure 12. Test 11 result on the effect of collocation points. (a) Relative error of γ_θ versus iteration number (left) and running time (right) with collocation point numbers $N_r = S \times 25K$ and $N_b = S \times 200$ with varying S; (b) the value of G_{norm} versus $1/\sqrt{S}$; (c) relative error versus $\sqrt{S}$; (d) running time (in seconds) versus S.

and $N_b = S \times 200$, where S takes value in $\{0.25, 0.5, 1.0, 2.0, 4.0\}$. After $J = 20\,000$ iterations, we evaluate $G_{\text{norm}} := \sqrt{\min_{1 \leqslant j \leqslant J} \mathbb{E}[|\mathcal{G}(\theta_j)|^2]}$ (expectation is approximated by empirical average of 5 runs), the relative error, and the running time for each value of S, and plot their values versus S in figures 12(b)–(d), respectively. From figure 12(b), we can see that G_{norm} is approximately proportional to $1/\sqrt{S}$. Figures 12(c) and (d) show that the solution error generally decreases in S at the expense of longer computational time. For

reference, the relative error versus iteration and running time with varying S are shown in figure 12(a).

5. Concluding remarks

We have presented a weak adversarial network approach to solve a class of IP numerically. We leverage the weak formulation of PDEs in the IP, and parameterize the unknown solution as primal neural network and the test function as adversarial network. The weak formulation and the boundary conditions yield a saddle function in the parameters of the primal network and adversarial network, which only rely on the IP itself but not any other training data. These parameters are alternately updated until convergence. We provide a series of theoretical justifications on the convergence of our proposed algorithm. Our method does not require any spatial discretization, and can be applied to a large class of IP, especially those with high dimensionality and less regularity on solutions. Numerical experiments have been conducted by applying the proposed method to a variety of challenging IP. The results suggest promising accuracy and efficiency of our approach.

Acknowledgments

GB and YZ are supported in part by the NSFC Innovative Group Fund (No. 11621101). XY is supported in part by the NSF under Grants DMS-1620342, CMMI-1745382, DMS-1818886 and DMS-1925263. HZ is supported in part by the NSF under Grants DMS-1620345 and DMS-1830225, and ONR N00014-18-1-2852.

Appendix A. Proofs

A.1. Proof of theorem 1

For ease of presentation, our proof of theorem 1 here is based on the problem formulation (2). However, it can be easily modified for the PDEs in many other IP.

Proof. For any fixed $u \in H^1(\Omega) \cap C(\bar{\Omega})$ and $\gamma \in C(\bar{\Omega})$, the maximum of $\langle \mathcal{A}[u,\gamma], \varphi\rangle$ is achievable over $Y := \{\varphi \in H_0^1(\Omega) : \|\varphi\|_{H^1} = 1\}$ since $\langle \mathcal{A}[u,\gamma], \cdot\rangle$ is continuous and Y is closed in $H_0^1(\Omega)$. Define $h(u,\gamma) = \max_{\varphi\in Y}\langle \mathcal{A}[u,\gamma], \varphi\rangle$, then $h(u,\gamma) = \|\mathcal{A}[u,\gamma]\|_{\text{op}}$ due to the definition of operator norm in (5). On the other hand, let $X = \{(u,\gamma) \in H^1(\Omega) \times C(\Omega) : \mathcal{B}[u,\gamma] = 0\}$, then it is clear that the minimum value 0 of $h(u,\gamma)$ over X can be attained at any of the weak solutions. Hence the minimax problem (6) is well-defined.

Now we show that (u^*,γ^*) satisfying $\mathcal{B}[u^*,\gamma^*] = 0$ is the solution of the minimax problem (6) if and only if it is a weak solution of the problem (1). Suppose (u^*,γ^*) is a weak solution of the problem (1), namely (u^*,γ^*) satisfies (3) for all $\varphi \in Y$, then $\langle \mathcal{A}[u^*,\gamma^*], \varphi\rangle \equiv 0$ for all $\varphi \in Y$. Therefore, $\|\mathcal{A}[u^*,\gamma^*]\|_{\text{op}} = 0$, and (u^*,γ^*) is the solution of the minimax problem (6). On the other hand, suppose a weak solution $(\hat{u},\hat{\gamma})$ of (1) exists. Assume that (u^*,γ^*) is a minimizer of the problem (6), i.e., $(u^*,\gamma^*) = \text{argmin}_{(u,\gamma)\in H^1\times C} h(u,\gamma)$, but not a weak solution of the problem (1), then there exists $\varphi^* \in Y$ such that $\langle \mathcal{A}[u^*,\gamma^*], \varphi^*\rangle > 0$. Therefore $h(u^*,\gamma^*) = \max_{\varphi\in Y}|\langle \mathcal{A}[u^*,\gamma^*], \varphi\rangle| > 0$. However, as we showed above, $h(\hat{u},\hat{\gamma}) = 0$ since $(\hat{u},\hat{\gamma})$ is a weak solution of (1), which contradicts to the assumption that (u^*,γ^*) is the minimizer of (6). Hence (u^*,γ^*) must also be a weak solution of (1), i.e., (u^*,γ^*) satisfies (3). □

A.2. Proof of lemma 2

Proof. Due to the definition of $L_{\text{int}}(\theta)$ in (11) and the optimality of $\eta(\theta)$, we know that $L_{\text{int}}(\theta) = E(\theta, \eta(\theta))$. Therefore, we have

$$\nabla_\theta L_{\text{int}}(\theta) = \partial_\theta E(\theta, \eta(\theta)) + \partial_\eta E(\theta, \eta(\theta))\nabla_\theta \eta(\theta). \tag{22}$$

Now we form the Lagrange function $\mathcal{L}(\theta, \eta, \mu) = E(\theta, \eta) + \mu(\frac{1}{2}|\eta|^2 - B)$ for the maximization problem $\max_{|\eta|^2 \leqslant 2B} E(\theta, \eta)$. Then the Karush–Kuhn–Tucker condition of $\eta(\theta)$ is given by

$$\partial_\eta \mathcal{L}(\theta, \eta(\theta), \mu) = \partial_\eta E(\theta, \eta(\theta)) + \mu(\theta)\eta(\theta) = 0, \tag{23a}$$

$$\mu(\theta)\left((1/2)\cdot|\eta(\theta)|^2 - B\right) = 0, \tag{23b}$$

$$\mu(\theta) \geqslant 0, \qquad |\eta(\theta)|^2 \leqslant 2B. \tag{23c}$$

The complementary slackness condition (23b) implies that

$$\nabla_\theta \mu(\theta)\left((1/2)\cdot|\eta(\theta)|^2 - B\right) + \mu(\theta)\eta(\theta)\nabla_\theta \eta(\theta) = 0 \tag{24}$$

If $\mu(\theta) = 0$, then we know $\partial_\eta E(\theta, \eta(\theta)) = 0$ due to (23a) and hence (22) reduces to $\nabla_\theta L_{\text{int}}(\theta) = \partial_\theta E(\theta, \eta(\theta))$. If $\mu(\theta) > 0$, then $|\eta(\theta)|^2 = 2B$ due to (23b), and hence (24) implies $\mu(\theta)\eta(\theta)\nabla_\theta \eta(\theta) = 0$. Thus multiplying (23a) by $\nabla_\theta \eta(\theta)$ yields $\partial_\eta E(\theta, \eta(\theta))\nabla_\theta \eta(\theta) = 0$, from which we can see (22) also reduces to $\nabla_\theta L_{\text{int}}(\theta) = \partial_\theta E(\theta, \eta(\theta))$. □

A.3. Proof of lemma 3

Proof. The first moment, i.e., expectation of $\hat{\Psi}$, can be computed as follows:

$$\mathbb{E}[\hat{\Psi}] = \mathbb{E}[\psi/\rho] = \int_\Omega \rho \frac{\psi}{\rho}\, \mathrm{d}x = \int_\Omega \psi\, \mathrm{d}x = \Psi. \tag{25}$$

To compute the second moment of $\hat{\Psi}$, we first observe that the variance of $\hat{\Psi}$ is

$$\mathrm{V}(\hat{\Psi}) = \mathrm{V}\left(\frac{1}{N}\sum_{i=1}^{N} \frac{\psi(x^{(i)})}{\rho(x^{(i)})}\right) = \frac{1}{N}\mathrm{V}\left(\frac{\psi}{\rho}\right).$$

Note that the variance of ψ/ρ is

$$\mathrm{V}\left(\frac{\psi}{\rho}\right) = \mathbb{E}\left[\left(\frac{\psi}{\rho}\right)^2\right] - \left(\mathbb{E}\left[\frac{\psi}{\rho}\right]\right)^2 = \int_\Omega \frac{\psi^2}{\rho}\, \mathrm{d}x - \left(\int_\Omega \psi\, \mathrm{d}x\right)^2 = \int_\Omega \frac{\psi^2}{\rho}\, \mathrm{d}x - \Psi^2$$

Hence the second moment of $\hat{\Psi}$ is

$$\mathbb{E}[\hat{\Psi}^2] = \mathrm{V}(\hat{\Psi}) + \mathbb{E}[\hat{\Psi}]^2 = \frac{1}{N}\left(\int_\Omega \frac{\psi^2}{\rho}\, \mathrm{d}x - \Psi^2\right) + \Psi^2 = \frac{N-1}{N}\Psi^2 + \frac{1}{N}\int_\Omega \frac{\psi(x)^2}{\rho(x)}\, \mathrm{d}x,$$

which completes the proof. □

A.4. Proof of theorem 4

Proof. Due to the parameterization of $(u_\theta, \gamma_\theta)$ using finite-depth neural network (8) and the compactness of $\Theta := \{\theta : |\theta| \leqslant \sqrt{2B}\}$, we know $\partial^\alpha u_\theta$ and γ_θ have Lipschitz continuous gradient with respect to θ for all $|\alpha| \leqslant 1$. As Ω is bounded and $\partial^\alpha u_\theta, \gamma_\theta \in C(\bar{\Omega})$, there exists $M > 0$ such that $L(\theta)$ has M-Lipschitz continuous gradient $\nabla_\theta L(\theta)$, since $L(\theta)$ is composed of integrals of $\partial^\alpha u_\theta$ and γ_θ over Ω.

Recall that the projected SGD step (14), started from initial θ_1, generates the sequence $\{\theta_j\}$ as follows:

$$\theta_{j+1} = \Pi(\theta_j - \tau G_j) = \operatorname*{argmin}_{\theta\in\Theta} \left(G_j^\top \theta + \frac{1}{2\tau}|\theta - \theta_j|^2 \right) \tag{26}$$

where G_j denotes the stochastic gradient of $L(\theta)$ at θ_j using N_r (N_b resp.) sample collocation points in Ω (on $\partial\Omega$ resp.) with $N_r, N_b = O(N)$. We let $g_j := \nabla_\theta L(\theta_j)$ denote the true (but unknown) gradient of L at θ_j, and define a companion sequence $\{\bar{\theta}_j\}$ using g_j as

$$\bar{\theta}_{j+1} = \Pi(\theta_j - \tau g_j) = \operatorname*{argmin}_{\theta\in\Theta} \left(g_j^\top \theta + \frac{1}{2\tau}|\theta - \theta_j|^2 \right). \tag{27}$$

Note that $\{\bar{\theta}_j\}$ is not computed in practice (computation of $\bar{\theta}_j$ is not possible as g_j is unknown), but only defined for convergence analysis here. Also note that Θ and Ω are bounded, and hence all integrals of $\partial^\alpha u_\theta$ and γ_θ are bounded, we know G_j is an unbiased estimate of g_j with bounded variance, denoted by $\sigma^2 > 0$, according to lemma 3. Moreover, lemma 3 implies that there exists $E > 0$ dependent on B, Ω, $\mathcal{A}$, and $\mathcal{B}$ only (E is the bound of $\|\mathcal{A}[u_\theta, \gamma_\theta]\|_{\text{op}}^2$ and $\|\mathcal{B}[u_\theta, \gamma_\theta]\|_{L^2(\partial\Omega)}^2$ due to the boundedness of Θ and Ω) the integral such that $\mathbb{E}[|G_j - g_j|^2] \leqslant \sigma^2 \leqslant E/N$ as $N_r, N_b = O(N)$.

Now we are ready to verify the convergence of the projected SGD iterations (26). First, the M-Lipschitz continuity of $\nabla_\theta L$ implies that

$$L(\theta_{j+1}) \leqslant L(\theta_j) + g_j^\top e_j + \frac{M}{2}|e_j|^2, \tag{28}$$

where we denote $e_j := \theta_{j+1} - \theta_j$ for all j. Also due to the M-Lipschitz continuity of $\nabla_\theta L$, we have

$$-L(\bar{\theta}_{j+1}) \leqslant -L(\theta_j) - g_j^\top \bar{e}_j + \frac{M}{2}|\bar{e}_j|^2, \tag{29}$$

where we denote $\bar{e}_j := \bar{\theta}_{j+1} - \theta_j$. Note that $\mathcal{G}(\theta_j) = \tau^{-1}[\theta_j - \Pi(\theta_j - \tau g_j)] = \tau^{-1}(\theta_j - \bar{\theta}_{j+1}) = -\tau^{-1}\bar{e}_j$, whose magnitude is what we want to bound eventually. Furthermore, due to the optimality of θ_{j+1} in (26) (which is convex in θ), we know that

$$0 \leqslant \left(G_j + \frac{\theta_{j+1} - \theta_j}{\tau} \right)^\top (\bar{\theta}_{j+1} - \theta_{j+1}) = \left(G_j + \frac{e_j}{\tau} \right)^\top (\bar{e}_j - e_j). \tag{30}$$

Adding (28)–(30) yields

$$L(\theta_{j+1}) - L(\bar{\theta}_{j+1}) \leqslant (g_j - G_j)^\top (e_j - \bar{e}_j) + \frac{e_j^\top (\bar{e}_j - e_j)}{\tau} + \frac{M}{2}|e_j|^2 + \frac{M}{2}|\bar{e}_j|^2. \tag{31}$$

Repeating (28)–(31) with θ_{j+1} and $\bar{\theta}_{j+1}$ replaced by $\bar{\theta}_{j+1}$ and θ_j respectively, and using the optimality of $\bar{\theta}_{j+1}$ in (27) with g_j, we obtain

$$L(\bar{\theta}_{j+1}) - L(\theta_j) \leqslant -\left(\frac{1}{\tau} - \frac{M}{2}\right)|\bar{e}_j|^2. \tag{32}$$

Adding (31) and (32) yields

$$L(\theta_{j+1}) - L(\theta_j) \leqslant (g_j - G_j)^\top (e_j - \bar{e}_j) + \frac{e_j^\top(\bar{e}_j - e_j)}{\tau} + \frac{M}{2}|e_j|^2 - \left(\frac{1}{\tau} - M\right)|\bar{e}_j|^2. \tag{33}$$

Now due to Cauchy–Schwarz inequality, the definitions of θ_{j+1} and $\bar{\theta}_{j+1}$ in (26) and (27), and that the projection Π onto the convex set Θ is a non-expansive operator (i.e., $|\Pi(\theta) - \Pi(\hat{\theta})| \leqslant |\theta - \hat{\theta}|$ for any $\theta, \hat{\theta}$), we can show that

$$\begin{aligned}(g_j - G_j)^\top (e_j - \bar{e}_j) &= (g_j - G_j)^\top (\theta_{j+1} - \bar{\theta}_{j+1}) = (g_j - G_j)^\top \left(\Pi(\theta_j - \tau G_j) - \Pi(\theta_j - \tau g_j)\right) \\ &\leqslant |g_j - G_j|\ |\Pi(\theta_j - \tau G_j) - \Pi(\theta_j - \tau g_j)| \leqslant \tau |g_j - G_j|^2. \end{aligned} \tag{34}$$

Moreover, we have that

$$\frac{e_j^\top(\bar{e}_j - e_j)}{\tau} = \frac{1}{2\tau}\left(|\bar{e}_j|^2 - |e_j|^2 - |e_j - \bar{e}_j|^2\right). \tag{35}$$

Substituting (34) and (35) into (33), we obtain

$$L(\theta_{j+1}) - L(\theta_j) \leqslant \tau |g_j - G_j|^2 - \left(\frac{1}{2\tau} - M\right)|\bar{e}_j|^2 - \left(\frac{1}{2\tau} - \frac{M}{2}\right)|e_j|^2 - \frac{1}{2\tau}|e_j - \bar{e}_j|^2. \tag{36}$$

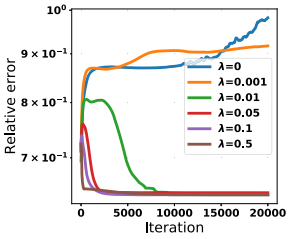
Taking expectation on both sides of (36) and discarding the last negative term, we obtain

$$\begin{aligned}\left(\frac{1}{2} - \tau M\right)\tau \mathbb{E}[|\mathcal{G}(\theta_j)|^2] = \left(\frac{1}{2\tau} - M\right)\mathbb{E}[|\bar{e}_j|^2] &\leqslant L_j - L_{j+1} + \tau\sigma^2 \\ &\quad - \left(\frac{1}{2\tau} - \frac{M}{2}\right)\mathbb{E}[|e_j|^2] \end{aligned} \tag{37}$$

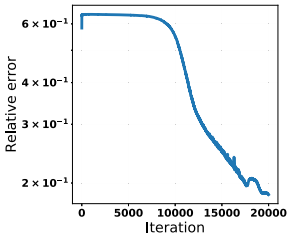
where we used the fact $\mathbb{E}[|G_j - g_j|^2] \leqslant \sigma^2$ and the notation $L_j := \mathbb{E}[L(\theta_j)]$. Now taking sum of (37) for $j = 1, \ldots, J$, dividing both sides by $(\frac{1}{2} - \tau M)\tau J$, and setting $\tau = \frac{1}{4M}$ (hence $\frac{1}{2} - \tau M = \frac{1}{4}$ and $\frac{1}{\tau} - \frac{M}{2} = \frac{7M}{2} > 0$), we know that

$$\begin{aligned}\min_{1 \leqslant j \leqslant J} \mathbb{E}[|\mathcal{G}(\theta_j)|^2] \leqslant \frac{1}{J}\sum_{j=1}^{J}\mathbb{E}[|\mathcal{G}(\theta_j)|^2] &\leqslant \frac{16M(L_1 - L_{J+1})}{J} + 4\sigma^2 \\ &\leqslant \frac{16M(L_1 - L^*)}{J} + \frac{4E}{N} \leqslant \varepsilon \end{aligned} \tag{38}$$

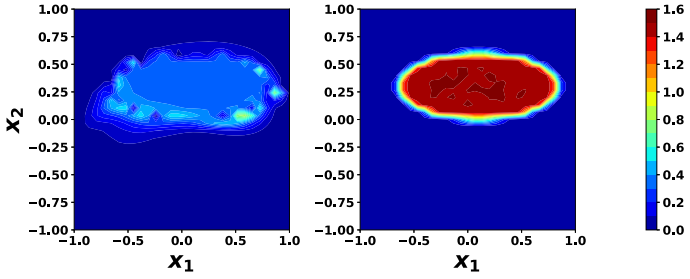
by choosing per-iteration sample complexity N and iteration number J as $N = J = [16M(L_1 - L^*) + 4E]\varepsilon^{-1} = O(\varepsilon^{-1})$ where $L^* := \min_{\theta \in \Theta} L(\theta) \geqslant 0$ (and hence $L_{J+1} = \mathbb{E}[L(\theta_{J+1})] \geqslant J^*$). This completes the proof. □



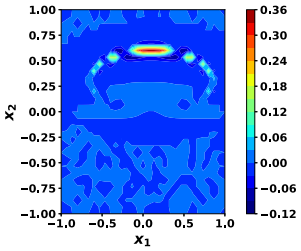
(a) Error vs. iter by FDM



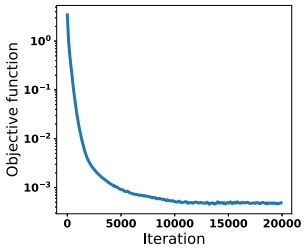
(b) Error vs. iter by IWAN



(c) $|\gamma_\theta - \gamma^*|$ by IWAN (left) and FDM (right)



(d) $|-\nabla(\gamma\nabla u) - f|$ by FDM with $\lambda = 0.1$



(e) Objective vs. iter by FDM with $\lambda = 0.1$

Figure 13. Test 2 on the comparison of IWAN and FDM to recover the conductivity γ^* with in 2D case. (a) Relative error versus iteration number obtained by FDM with different λ. (b) Relative error versus iteration obtained by IWAN. (c) Pointwise absolute error $|\gamma - \gamma^*|$ with γ obtained by IWAN (left) and FDM (right). (d) $|-\nabla(\gamma\nabla u) - f|$ by FDM. (e) Objective function value versus iteration number by FDM.

Appendix B. Problem setting

The functions and parameters used in our experiments are summarized in table 1.

Appendix C. Recorded errors and running times in tests 9 and 10

The recorded errors and running times for test 9 and Test 10 were present in tables 2 and 3, respectively.

Appendix D. Comparison with classical numerical methods

To further evaluate the proposed method, we provide an example that compares IWAN and a classical FDM on a 2D problem in test 2. We would like to acknowledge an anonymous

reviewer for suggesting this valuable comparison. It is worth pointing out that the majority of existing numerical methods, such as FDM, require the knowledge of DtN map for the EIT problem. However, to be consistent with the settings used in the present work, we conduct the comparison with FDM under the same setting of test 2, where a DtN map is not available but only the boundary conditions of u and γ in (2) are given.

In FDM, we discretize the domain into 31×31 mesh grids (about 900 unknowns for each of u and γ). We also experiment with higher resolution but it does not improve solution quality; see later for more explanations. We approximate the partial derivatives in the PDE by finite differences. As the problem is underdetermined, we use regularization and formulate as a minimization problem of (u, γ), where the objective function is the sum of two terms: the mean square error of the PDE, and the TV regularization on γ. For comparison, we choose 4 hidden layers with 15 neurons per-layer to parameterize u_θ and γ_θ (each with < 800 unknowns). We set the number of collocation points to $N_r = 1\,000$ and $N_b = 120$ (similar to the discretization resolution of FDM). For FDM, we test different regularization hyperparameter λ (the weight of the TV term), and show the result in figure 13(a). We observe that FDM achieved the best result when $\lambda = 0.1$, which is used to generate the other images in figure 13. The relative error obtained by IWAN is shown in figure 13(b). Figure 13(c) shows the absolute error $|\gamma_\theta - \gamma|$ obtained by IWAN and FDM (γ_θ is the recovered conductivity by IWAN or FDM, and γ is the ground truth), respectively, which demonstrates that IWAN can faithfully recover γ_θ but FDM cannot in the setting of test 2. Figure 13(e) shows the objective function value versus iteration by FDM, which suggests that FDM has converged. We also show $|-\nabla(\gamma\nabla u) - f|$ obtained by FDM in figure 13(d), which shows that the (u, γ) obtained by FDM indeed satisfies the PDE approximately. In addition, we increase the domain discretization resolution up to 101×101 for FDM, but did not observe any noticeable improvement, and hence we omitted the results here.

ORCID iDs

Gang Bao https://orcid.org/0000-0003-1591-8095
Xiaojing Ye https://orcid.org/0000-0002-0516-4974
Yaohua Zang https://orcid.org/0000-0003-2858-643X
Haomin Zhou https://orcid.org/0000-0001-7647-2600

References

[1] Abadi M *et al* 2016 Tensorflow: a system for large-scale machine learning *12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16)* pp 265–83
[2] Adler J and Öktem O 2017 Solving ill-posed inverse problems using iterative deep neural networks *Inverse Problems* **33** 124007
[3] Arjovsky M, Chintala S and Bottou L 2017 Wasserstein generative adversarial networks *Int. Conf. on Machine Learning* pp 214–23
[4] Anitescu C, Atroshchenko E, Alajlan N and Rabczuk T 2019 Artificial neural network methods for the solution of second order boundary value problems *Comput. Mater. Continua* **59** 345–59
[5] Antholzer S, Haltmeier M and Schwab J 2019 Deep learning for photoacoustic tomography from sparse data *Inverse Problems Sci. Eng.* **27** 987–1005
[6] Bellman R 1966 Dynamic programming *Science* **153** 34–7
[7] Bertero M and Boccacci P 1998 *Introduction to Inverse Problems in Imaging* (Boca Raton, FL: CRC Press)

[8] Bar L and Sochen N 2019 Unsupervised deep learning algorithm for pde-based forward and inverse problems (arXiv:1904.05417)
[9] Beck C, W E and Jentzen A 2019 Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations *J. Nonlinear Sci.* **29** 1563–619
[10] Chen X, Duan J and Karniadakis G E 2019 Learning and meta-learning of stochastic advection-diffusion-reaction systems from sparse measurements (arXiv:1910.09098)
[11] Cheney M, Isaacson D and Newell J C 1999 Electrical impedance tomography *SIAM Rev.* **41** 85–101
[12] Curtis E B and Morrow J A 1991 The Dirichlet to Neumann map for a resistor network *SIAM J. Appl. Math.* **51** 1011–29
[13] Dadvand P, Lopez R and Onate E 2006 Artificial neural networks for the solution of inverse problems *Proc. of the Int. Conf. on Design Optimisation Methods and Applications ERCOFTAC* vol 2006
[14] Dockhorn T 2019 A discussion on solving partial differential equations using neural networks (arXiv:1904.07200)
[15] Dittmer S, Kluth T, Maass P and Baguer D O 2019 Regularization by architecture: a deep prior approach for inverse problems *J. Math. Imaging Vis.* **62** 456
[16] E W, Han J and Jentzen A 2017 Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations *Commun. Math. Stat.* **5** 349–80
[17] E W and Yu B 2018 The deep ritz method: a deep learning-based numerical algorithm for solving variational problems *Commun. Math. Stat.* **6** 1–12
[18] Fan Y and Ying L 2019 Solving electrical impedance tomography with deep learning (arXiv:1906.03944)
[19] Feliu-Faba J, Fan Y and Ying L 2019 Meta-learning pseudo-differential operators with deep neural networks (arXiv:1906.06782)
[20] Fernández-Fuentes X, Mera D, Gómez A and Vidal-Franco I 2018 Towards a fast and accurate eit inverse problem solver: a machine learning approach *Electronics* **7** 422
[21] Francini E 2000 Recovering a complex coefficient in a planar domain from the dirichlet-to-neumann map *Inverse Problems* **16** 107
[22] Ghadimi S, Lan G and Zhang H 2016 Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization *Math. Program.* **155** 267–305
[23] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[24] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems* pp 2672–80
[25] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V and Courville A C 2017 Improved training of wasserstein gans *Advances in Neural Information Processing Systems* pp 5767–77
[26] Hamilton S J, Hänninen A, Hauptmann A and Kolehmainen V 2019 Beltrami-net: domain independent deep d-bar learning for absolute imaging with electrical impedance tomography (a-eit) *Physiol. Meas.* **40** 074002
[27] Hamilton S J and Hauptmann A 2018 Deep d-bar: real-time electrical impedance tomography imaging with deep neural networks *IEEE Trans. Med. Imaging* **37** 2367–77
[28] Han J, Jentzen A and E W 2018 Solving high-dimensional partial differential equations using deep learning *Proc. Natl Acad. Sci. USA* **115** 8505–10
[29] Hauptmann A, Lucka F, Betcke M, Huynh N, Cox B, Beard P, Ourselin S and Arridge S 2017 Model based learning for accelerated, limited-view 3d photoacoustic tomography *IEEE Trans. Med. Imaging* **37** 1382
[30] Heckel R and Soltanolkotabi M 2019 Denoising and regularization via exploiting the structural bias of convolutional generators (arXiv:1910.14634)
[31] Hornik K 1991 Approximation capabilities of multilayer feedforward networks *Neural Netw.* **4** 251–7
[32] Joshi A, Shah V, Ghosal S, Pokuri B, Sarkar S, Ganapathysubramanian B and Hegde C 2019 Generative models for solving nonlinear partial differential equations *Workshop on Machine Learning and the Physical Sciences*
[33] Jin K H, Mccann M T, Froustey E and Unser M 2017 Deep convolutional neural network for inverse problems in imaging *IEEE Trans. Image Process.* **26** 4509–22

[34] Jo H, Son H, Hwang H J and Kim E 2019 Deep neural network approach to forward-inverse problems (arXiv:1907.12925)
[35] Kaipio J and Somersalo E 2006 *Statistical and Computational Inverse Problems* vol 160 (Berlin: Springer)
[36] Khodayi-Mehr R and Varnet M M Z 2019 Variational neural networks for the solution of partial differential equations (arXiv:1912.07443)
[37] Khan T A and Ling S H 2019 Review on electrical impedance tomography: artificial intelligence methods and its applications *Algorithms* **12** 88
[38] Kang E, Min J and Ye J C 2017 A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction *Med. Phys.* **44** e360
[39] Khoo Y and Ying L 2018 Switchnet: a neural network model for forward and inverse scattering problems (arXiv:1810.09675)
[40] Klibanov M V, Li J and Zhang W 2019 Convexification of electrical impedance tomography with restricted dirichlet-to-neumann map data *Inverse Problems* **35** 035005
[41] Paragios N, Chen Y and Faugeras O D 2006 *Handbook of Mathematical Models in Computer Vision* (Berlin: Springer)
[42] Li Z and Li J 2018 A simple proximal stochastic gradient method for nonsmooth nonconvex optimization *Advances in Neural Information Processing Systems* pp 5564–74
[43] Li H, Schwab J, Antholzer S and Haltmeier M 2020 Solving inverse problems with deep neural networks NETT *Inverse Problems* **36** 065005
[44] Lu L, Jin P and Karniadakis G E 2019 Deeponet: learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators (arXiv:1910.03193)
[45] Martin S and Choi C T 2015 Nonlinear electrical impedance tomography reconstruction using artificial neural networks and particle swarm optimization *IEEE Trans. Magn.* **52** 7203904
[46] Martin S and Choi C T 2017 A post-processing method for three-dimensional electrical impedance tomography *Sci. Rep.* **7** 7212
[47] Meng X and Karniadakis G E 2019 A composite neural network that learns from multi-fidelity data: application to function approximation and inverse pde problems (arXiv:1903.00104)
[48] Michalikova M, Abed R, Prauzek M and Koziorek J 2014 Image reconstruction in electrical impedance tomography using neural network *Proc. Cairo Int. Biomedical Engineering Conf., CIBEC* (IEEE) pp 39–42
[49] Miyato T, Kataoka T, Koyama M and Yoshida Y 2018 Spectral normalization for generative adversarial networks (arXiv:1802.05957)
[50] Portal A, Fargier Y and Labazuy P 2016 Contribution of 3d inversion of electrical resistivity tomography data applied to volcanic structures *Egu General Assembly*
[51] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[52] Reddi S J, Sra S, Poczos B and Smola A J 2016 Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization *Advances in Neural Information Processing Systems* pp 1145–53
[53] Rymarczyk T, Kłosowski G, Kozłowski E and Tchórzewski P 2019 Comparison of selected machine learning algorithms for industrial electrical tomography *Sensors* **19** 1521
[54] Tan C, Lv S, Dong F and Takei M 2018 Image reconstruction based on convolutional neural network for electrical resistance tomography *IEEE Sens. J.* **19** 196–204
[55] Ulyanov D, Vedaldi A and Lempitsky V 2018 Deep image prior *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* pp 9446–54
[56] Wei Z, Liu D and Chen X 2019 Dominant-current deep learning scheme for electrical impedance tomography *IEEE Trans. Biomed. Eng.* **66** 2546
[57] Yamamoto M 1995 Stability, reconstruction formula and regularization for an inverse source hyperbolic problem by a control method *Inverse Problems* **11** 481
[58] Yao H, Wei E and Jiang L 2019 Two-step enhanced deep learning approach for electromagnetic inverse scattering problems *IEEE Antennas Wirel. Propag. Lett.* **18** 2254
[59] Zang Y, Bao G, Ye X and Zhou H 2020 Weak adversarial networks for high-dimensional partial differential equations *J. Comput. Phys.* **411** 109409