Journal of Computational Physics 463 (2022) 111301

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion

Shamsulhaq Basir, Inanc Senocak *

Department of Mechanical Engineering and Materials Science, University of Pittsburgh, 3700 O'Hara St., Pittsburgh, PA 15261, USA

ARTICLE INFO

Article history:
Received 30 September 2021
Received in revised form 10 February 2022
Accepted 11 May 2022
Available online 16 May 2022

Keywords:
Constrained optimization
Augmented Lagrangian method
Residual neural networks
Partial differential equations
Forward and Inverse problems
Multi-fidelity learning

ABSTRACT

Physics-informed neural networks (PINNs) have been proposed to learn the solution of partial differential equations (PDE). In PINNs, the residual form of the PDE of interest and its boundary conditions are lumped into a composite objective function as soft penalties. Here, we show that this specific way of formulating the objective function is the source of severe limitations in the PINN approach when applied to different kinds of PDEs. To address these limitations, we propose a versatile framework based on a constrained optimization problem formulation, where we use the augmented Lagrangian method (ALM) to constrain the solution of a PDE with its boundary conditions and any high-fidelity data that may be available. Our approach is adept at forward and inverse problems with multi-fidelity data fusion. We demonstrate the efficacy and versatility of our physics- and equality-constrained deep-learning framework by applying it to several forward and inverse problems involving multi-dimensional PDEs. Our framework achieves orders of magnitude improvements in accuracy levels in comparison with state-of-the-art physics-informed neural networks.

1. Introduction

Deep learning has been highly impactful in a plethora of fields such as pattern recognition [1,2], speech recognition [3], natural language processing [4–6] and in the solution of partial differential equations (PDE) for forward and inverse problems. The success of these models owes to the rapid development of available information, the advancement of computing power, and the advent of efficient learning algorithms for training neural networks [7]. With the emergence of universal approximation theorem [8,9], new studies have focused on using neural networks to solve ODEs and PDEs. One of the motivations for using neural networks in solving differential equations is their potential to break the curse of dimensionality [10–12] and its ability to fuse data in the learned solution. Neural network-based methods with their meshless nature can reduce the tedious effort of mesh generation, which is common with finite-difference, element, or volume methods. Moreover, in contrast to conventional numerical methods, once the neural network is trained, it can produce results at any point in the domain.

Dissanayake and Phan-Thien pioneered using neural networks to solve PDEs. They combined the residual form of a given PDE and its boundary conditions as soft constraints for training their neural network model. van Milligen et al. presented a similar approach and demonstrated its potential on a magnetohydrodynamics plasma equilibrium problem. This general

* Corresponding author.
E-mail address: senocak@pitt.edu (I. Senocak).

https://doi.org/10.1016/j.jcp.2022.111301
0021-9991/

neural network-based technique was applied with satisfactory results to non-linear Schrodinger equations in [15], to a non-steady fixed bed non-catalytic solid-gas reactor problems in [16], and to the one-dimensional Burgers equation in [17]. A neural network-based approach to solving PDEs and ODEs on orthogonal box domains was also proposed by Lagaris et al. by constructing trial functions that satisfy boundary conditions by construction. Unlike the approach in [13,14], the approach in [18] is limited to regular geometries as it is not trivial to create trial functions for irregular domains. Also, creating trial functions imposes inductive bias toward a certain class of functions that might not be optimal. These early works did not receive broader acceptance and appreciation by other researchers likely because of a lack of computing resources and a limited understanding of neural networks at the time of their introduction.

Machine learning frameworks with automatic differentiation capabilities [19,20] have revived the use of neural networks to solve ODEs and PDEs. The overall technical approach for using neural networks to solve PDEs and ODEs that was adopted in the aforementioned works, particularly the method described in [13,14], has found a resurgent interest in recent years [21–24]. Raissi et al. [25] dubbed the term physics-informed neural networks (PINNs), which has been growing fast in popularity and applied to several unique forward and inverse problems [26–33]. Even though neural networks offer a powerful framework to faithfully integrate data and physical laws in solving forward and ill-posed inverse problems, training these models is not trivial for challenging problems [34–36]. Extensive reviews of the current state in physics-informed machine learning are available in literature [37,38], but we will also elaborate on the challenges faced by the PINN approach in later sections.

Our paper is structured as follows. In §2 we present a technical overview of the physics-based neural networks following the original formulation of [13,14]. Subsequently, we describe a recently proposed empirical algorithm for improving the predictive capability of these models as well as its limitations. Next, we describe the augmented Lagrange method, which forms the backbone of our approach. In §3 we propose the physics and equality constrained artificial neural networks (PECANN) framework and provide a training algorithm for it. In §4 we conduct a comparative analysis of our method on several benchmark problems. In §5 we demonstrate the performance of the PECANN approach on three different inverse problems with multi-fidelity data fusion. Finally, in §6 we summarize our results and provide several directions for future research. All the codes and data accompanying this paper are publicly available at https://github.com/HiPerSimLab/PECANN.

2. Technical background

Consider a scalar function $u(\boldsymbol{x},t):\mathbb{R}^{d+1}\rightarrow\mathbb{R}$ on the domain $\Omega\subset\mathbb{R}^d$ with its boundary $\partial\Omega$ satisfying the following partial differential equation

$$\mathcal{F}(\boldsymbol{x},t;\frac{\partial u}{\partial t},\frac{\partial^2 u}{\partial t^2},\cdots,\frac{\partial u}{\partial \boldsymbol{x}},\frac{\partial^2 u}{\partial \boldsymbol{x}^2},\cdots,\boldsymbol{\nu})=0,\quad \forall(\boldsymbol{x},t)\in\mathcal{U}, \tag{1}$$

$$\mathcal{B}(\boldsymbol{x},t,g;u,\frac{\partial u}{\partial \boldsymbol{x}},\cdots)=0,\quad \forall(\boldsymbol{x},t)\in\partial\mathcal{U}, \tag{2}$$

$$\mathcal{I}(\boldsymbol{x},t,h;u,\frac{\partial u}{\partial t},\cdots)=0,\quad \forall(\boldsymbol{x},t)\in\Gamma, \tag{3}$$

where $\mathcal{F}$ is the residual form of the PDE containing differential operators, $\boldsymbol{\nu}$ is a vector PDE parameters, $\mathcal{B}$ is the residual form of the boundary condition containing a source function $g(\boldsymbol{x},t)$ and $\mathcal{I}$ is the residual form of the initial condition containing a source function $h(\boldsymbol{x},t)$. $\mathcal{U}=\{(\boldsymbol{x},t)\mid \boldsymbol{x}\in\Omega,t=[0,T]\}$, $\partial\mathcal{U}=\{(\boldsymbol{x},t)\mid \boldsymbol{x}\in\partial\Omega,t=[0,T]\}$ and $\Gamma=\{(\boldsymbol{x},t)\mid \boldsymbol{x}\in\partial\Omega,t=0\}$.

2.1. Physics-informed neural networks

Here, we present the common elements of the physics-informed learning framework that was presented in the works of Dissanayake and Phan-Thien and van Milligen et al., and, in the work of Raissi et al. as part of contemporary developments in physics based deep learning methods. Suppose we seek a solution $u_\theta(\boldsymbol{x})$ represented by a neural network parameterized by θ for Eq. (1) with its boundary condition Eq. (2) and its initial condition Eq. (3). We can write the following loss functional $\mathcal{L}(\theta)$ to train a physics-informed neural network.

$$\mathcal{L}(\theta)=\lambda_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta)+\lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}}(\theta)+\lambda_{\mathcal{I}}\mathcal{L}_{\mathcal{I}}(\theta), \tag{4}$$

$$\mathcal{L}_{\mathcal{F}}(\theta)=\frac{1}{N_{\mathcal{F}}}\sum_{i=1}^{N_{\mathcal{F}}}\|\mathcal{F}(\boldsymbol{x}^{(i)},t^{(i)})\|_2^2, \tag{5}$$

$$\mathcal{L}_{\mathcal{B}}(\theta)=\frac{1}{N_{\mathcal{B}}}\sum_{i=1}^{N_{\mathcal{B}}}\|\mathcal{B}(\boldsymbol{x}^{(i)},t^{(i)},g^{(i)})\|_2^2, \tag{6}$$

$$\mathcal{L}_{\mathcal{I}}(\theta)=\frac{1}{N_{\mathcal{I}}}\sum_{i=1}^{N_{\mathcal{I}}}\|\mathcal{I}(\boldsymbol{x}^{(i)},t^{(i)},h^{(i)})\|_2^2, \tag{7}$$

where $\{\boldsymbol{x}^{(i)}, t^{(i)}\}_{i=1}^{N_{\mathcal{F}}}$ is the set of residual points in $\mathcal{U}$ for approximating the physics loss $\mathcal{L}_{\mathcal{F}}(\theta)$, $(\{\boldsymbol{x}^{(i)}, t^{(i)}), g^{(i)}\}_{i=1}^{N_{\mathcal{B}}}$ is the set boundary points on $\partial\mathcal{U}$ for approximating the boundary loss $\mathcal{L}_{\mathcal{B}}(\theta)$ and $\{(\boldsymbol{x}^{(i)}, t^{(i)}), h^{(i)}\}_{i=1}^{N_{\mathcal{I}}}$ is the set of initial data on Γ for approximating the loss on initial condition $\mathcal{L}_{\mathcal{I}}(\theta)$. $\lambda_{\mathcal{F}}$, $\lambda_{\mathcal{B}}$ and $\lambda_{\mathcal{I}}$ are hyperparameters to balance the interplay between the loss terms and $\mathcal{L}(\theta)$ is the sum of all the objective functions used for training a neural network model. It is worth noting that in conventional PINNs $\lambda_{\mathcal{F}} = \lambda_{\mathcal{B}} = \lambda_{\mathcal{I}} = 1$.

Since training PINNs minimizes a weighted sum of several objective functions as in Eq. (4), the prediction of the network highly depends on the choice of these weights. Manual setting of these weights by trial and error tuning is extremely tedious and time-demanding. Based on our own experience, we find that manual tuning of these weights is not ideal, because it creates a ripple effect as we then need to tune other hyperparameters, such as the number of collocations points, the learning rate, and the architecture. Also, the optimal choice of these weights for a problem under a certain training setting might not transfer across different problems and may not even produce acceptable results if the training setting is changed. Proper choice of these free parameters is still an active area of research [36,35,34]. Next, we discuss an empirical algorithm proposed by Wang et al. [36] for choosing these hyperparameters.

2.2. Learning rate annealing for physics-informed neural networks

Consider a physics-informed neural network with parameters θ and a loss function as follows

$$\mathcal{L}(\theta) = \lambda_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta) + \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\theta), \tag{8}$$

where $\mathcal{L}_{\mathcal{F}}(\theta)$ is the PDE residual loss as in Eq. (5), $\mathcal{L}_i(\theta)$ correspond to data-fit terms (e.g., measurements, initial or boundary conditions), $\lambda_{\mathcal{F}}$ and $\lambda_i, i = 1, \cdots, M$ are free parameters used to balance the interplay between different loss terms. The necessary optimality condition for Eq. (8) is

$$\nabla_\theta \mathcal{L}(\theta) = \lambda_{\mathcal{F}} \nabla_\theta \mathcal{L}_{\mathcal{F}}(\theta) + \sum_{i=1}^{M} \lambda_i \nabla_\theta \mathcal{L}_i(\theta) = 0, \tag{9}$$

where λs are learned such that the optimality condition is satisfied. Wang et al. recently proposed an empirical algorithm for setting these weights based on matching the magnitude of the back-propagated gradients as follows

$$\lambda_{\mathcal{F}} = 1, \tag{10a}$$

$$\hat{\lambda}_i = \frac{\max_{\theta_n}\{|\nabla_\theta \mathcal{L}_{\mathcal{F}}(\theta_n)|\}}{\overline{|\nabla_{\theta_n} \lambda_i \mathcal{L}_i(\theta_n)|}}, \; i = 1, \cdots, M, \tag{10b}$$

$$\lambda_i = (1-\alpha)\lambda_i + \alpha\hat{\lambda}_i, \tag{10c}$$

where θ_n denotes the values of the network parameters at nth iteration, $|\cdot|$ denotes the elementwise absolute value, and the overbar signifies the algebraic mean of the gradient vector. Although this method improves on the original PINN approach ($\lambda_{\mathcal{F}} = \lambda_i = 1, i = 1, \cdots, M$), there are fundamental issues with this approach. First, approximating $\hat{\lambda}_i$ in Eq. (10b) does not necessarily meet the optimality condition as in Eq. (9). Therefore, the optimizer may settle to a point in the space of parameters that may not be an actual local minimum for the objective function as in Eq. (8). Second, the values of the network parameters can oscillate back and forth around a minimum, which requires slowing down the parameter update by decreasing the learning rate [39]. However, $\hat{\lambda}_i$ grows unbounded when the denominator in Eq. (10b) approaches zero which makes the effective learning rate extremely high and causes the optimizer to diverge. Also, in the case of noisy measurement data, this algorithm tries to fit the noise in the objective function as it is agnostic to the quality of data, and because of the noise in the objective function, its approximated free parameter will oscillate which could hinder convergence. Finally, the method is computationally expensive as it requires $M + 1$ number of backward passes through the computational graph to evaluate the gradients of the network parameter with respect to each term in the objective function.

2.3. Augmented Lagrangian method for constrained optimization

Consider the following nonlinear, equality-constrained optimization problem with n decision variables, and m equality constraints

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \quad \mathcal{J}(\theta), \\ &\text{subject to} \quad \mathcal{C}_i(\theta) = 0, \; \forall i = 1, \cdots, m \end{aligned} \tag{11}$$

where $\mathcal{J}$ is a nonlinear function of $\mathbb{R}^n$ in $\mathbb{R}$, $\mathcal{C}_i$ is a nonlinear function of $\mathbb{R}^n$ in $\mathbb{R}$ and θ is a given subset of $\mathbb{R}^n$, n-dimensional Euclidean space. Augmented Lagrangian method (ALM) [40,41] which is also the method of choice in the

present work can be used to convert the constrained optimization problem of Eq. (11) into an unconstrained optimization problem as follows

$$\min_{\theta\in\Theta} \quad \mathcal{L}(\theta;\lambda,\mu) = \mathcal{J}(\theta) + \sum_{i=1}^{m}\lambda_i\mathcal{C}_i(\theta) + \frac{\mu}{2}\sum_{i=1}^{m}|\mathcal{C}_i(\theta)|^2, \tag{12}$$

where $\lambda \in \mathbb{R}^m$ is a vector of Lagrange multipliers and μ is a positive penalty parameter, and the semicolon denotes that λ and μ are fixed. We update the vector of Lagrange multipliers based on the current estimate of the Lagrange multipliers and constraint values using the following rule

$$\lambda_i \leftarrow \lambda_i + \mu\mathcal{C}_i(\theta). \tag{13}$$

In ALM, the objective function is minimized possibly by violating the constraints. Subsequently, the feasibility is restored progressively as the iterations proceed [42]. If λ vanish, the penalty method is recovered, whereas when μ vanishes we get the method of Lagrange multipliers. As discussed in Martins and Ning [43], ALM avoids the ill-conditioning issue of the penalty method while having a better convergence rate than the Lagrange multiplier method [44]. Therefore, we could say that ALM combines the merit of both methods. Convergence in ALM may occur with finite μ, and optimization problem does not even have to possess a locally convex structure [45,46,44,42,43]. These aspects of the ALM make it a suitable choice for neural networks as their objective functions are typically non-convex with respect to the parameters of the network.

ALM has been used in scientific machine learning in the context of PDE-constrained optimization [47,48]. In Dener et al. [47], authors train a physics-constrained encoder-decoder neural network using ALM in a supervised learning fashion. In Lu et al. [48], the authors use ALM to train a PDE-constrained neural network model that satisfies the boundary conditions by construction, following an approach similar to the one proposed in [18].

3. Proposed method: physics & equality constrained artificial neural networks

Here, we propose a novel approach in using neural networks for the solution of forward problems and inverse problems with multi-fidelity data. This framework is noise-aware, physics-informed and equality constrained. We start by presenting a constrained optimization problem aimed at minimizing the sum of physics loss and noisy data (low-fidelity) loss such that any high fidelity data (boundary conditions, known equality constraints) are strictly satisfied. Considering Eq. (1) with its boundary condition (2) and initial condition (3), we write the following constrained optimization problem:

$$\min_{\theta} \mathcal{J}_{\mathcal{F}}(\theta) + \mathcal{J}_{\mathcal{M}}(\theta), \tag{14}$$

subject to

$$\phi(\mathcal{B}(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)})) = 0,\ \forall(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)}) \in \partial\mathcal{U},\ i = 1, \cdots, N_{\mathcal{B}} \tag{15}$$

$$\phi(\mathcal{I}(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)})) = 0,\ \forall(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)}) \in \Gamma,\ i = 1, \cdots, N_{\mathcal{I}}, \tag{16}$$

where $\mathcal{J}_{\mathcal{F}}(\theta)$ is the loss function for the given PDE, ϕ is a distance function and $\mathcal{J}_{\mathcal{M}}(\theta)$ is the objective function for noisy (low-fidelity) measurement data given

$$\tilde{u}(\boldsymbol{x}^{(i)}, t^{(i)}) = u_\theta(\boldsymbol{x}^{(i)}, t^{(i)}) + \epsilon^{(i)}, \forall i = 1, \cdots, N_{\mathcal{M}} \tag{17}$$

where $N_{\mathcal{M}}$ is the number of observations, $\tilde{u}(\boldsymbol{x}^{(i)}, t^{(i)})$ is the ith measurement at $(\boldsymbol{x}^{(i)}, t^{(i)})$, $u_\theta(\boldsymbol{x}^{(i)}, t^{(i)})$ is ith prediction from our neural network model at $(\boldsymbol{x}^{(i)}, t^{(i)})$ and $\epsilon^{(i)}$ captures the error associated with the ith data point. Assuming that the errors are normally distributed with mean zero and a standard deviation of σ, we can minimize the *log likelihood* of the predictions $u_\theta(\boldsymbol{x}, t)$ conditioned on the observed data $\tilde{u}_\theta(\boldsymbol{x}, t)$ to obtain $\mathcal{J}_{\mathcal{M}}(\theta)$ as follows [43]

$$\mathcal{J}_{\mathcal{M}}(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{N_{\mathcal{M}}} \|u_\theta(\boldsymbol{x}^{(i)}, t^{(i)}) - \tilde{u}(\boldsymbol{x}^{(i)}, t^{(i)})\|_2^2. \tag{18}$$

In this work, we set $\sigma = 1/\sqrt{2} \approx 0.7$ which results in a sum-of-squared errors for the noisy data, however, the user can assign any value to σ depending on the quality of the measurement data. It is worth noting, that a smaller value of σ which corresponds to less noisy data will put more weight on $\mathcal{J}_{\mathcal{M}}$ and vice versa. Using the augmented Lagrange method, we can write the resulting objective function as follows

$$\mathcal{L}(\theta;\lambda,\mu) = \mathcal{J}_{\mathcal{F}}(\theta) + \mathcal{J}_{\mathcal{M}}(\theta) + \sum_{i=1}^{N_{\mathcal{B}}}\lambda_{\mathcal{B}}^{(i)}\phi(\mathcal{B}(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)})) + \sum_{i=1}^{N_{\mathcal{I}}}\lambda_{\mathcal{I}}^{(i)}\phi(\mathcal{I}(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)})) + \frac{\mu}{2}\pi(\theta), \tag{19}$$

$$\pi(\theta) = \sum_{i=1}^{N_{\mathcal{B}}} |\phi(\mathcal{B}(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)}))|^2 + \sum_{i=1}^{N_{\mathcal{I}}} |\phi(\mathcal{I}(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)}))|^2, \tag{20}$$

$$\mathcal{J}_{\mathcal{F}}(\theta) = \sum_{i=1}^{N_{\mathcal{F}}} \|\mathcal{F}(\boldsymbol{x}^{(i)}, t^{(i)})\|_2^2, \tag{21}$$

where $N_{\mathcal{F}}$, $N_{\mathcal{B}}$, $N_{\mathcal{I}}$ are the number of data points in $\mathcal{U}$, $\partial\mathcal{U}$ and Γ respectively. We note that any equality constraints can be incorporated as Eq. (15) and (16) should they arise. $\lambda_{\mathcal{B}} \in \mathbb{R}^{N_{\mathcal{B}}}$ is an $N_{\mathcal{B}}$-dimensional vector of Lagrange multipliers for the constraints on $\partial\mathcal{U}$, $\lambda_{\mathcal{I}} \in \mathbb{R}^{N_{\mathcal{I}}}$ is an $N_{\mathcal{I}}$-dimensional vector of Lagrange multipliers for the constraints on Γ, and μ is a positive penalty parameter. We update the vector of Lagrange multipliers using the following rule

$$\lambda_{\mathcal{B}}^{(i)} \leftarrow \lambda_{\mathcal{B}}^{(i)} + \mu\phi(\mathcal{B}(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)})),\ \forall(\boldsymbol{x}^{(i)}, t^{(i)}, g^{(i)}) \in \partial\mathcal{U}, i = 1, \cdots N_{\mathcal{B}}, \tag{22}$$

$$\lambda_{\mathcal{I}}^{(i)} \leftarrow \lambda_{\mathcal{I}}^{(i)} + \mu\phi(\mathcal{I}(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)})),\ \forall(\boldsymbol{x}^{(i)}, t^{(i)}, h^{(i)}) \in \Gamma, i = 1, \cdots N_{\mathcal{I}} \tag{23}$$

Algorithm 1: Training algorithm for the PECANN framework.

Input: $\theta^0, \mu_{max}, E, S$
$\lambda_{\mathcal{B}}, \lambda_{\mathcal{I}} \leftarrow 0$ /* Initializing the multipliers */
$\epsilon \leftarrow 10^{-8}$ /* Assigning the tolerance for constraints violation */
$\mu_0 \leftarrow 1.0$ /* Initializing the penalty term */
$\eta \leftarrow 0$/* Placeholder for violation of constraint */
Output: θ^*
for *epoch* $\leftarrow 1$ **to** E **do**
  /* Iterate over all training batches */
  for *batch* $\leftarrow 1$ **to** S **do**
    $\theta^* \leftarrow \underset{\theta}{\text{argmin}}\ \mathcal{L}(\theta; \lambda, \mu)$/* Optimizing the network's parameters */
    if $(\sqrt{\pi(\theta)} \geq 0.25\eta)$ & $(\sqrt{\pi(\theta)} > \epsilon)$ **then**
      $\mu \leftarrow \min(2\mu, \mu_{max})$ /* Updating the penalty parameter */
      $\lambda_{\mathcal{B}} \leftarrow \lambda_{\mathcal{B}} + \phi(\mathcal{B}(\boldsymbol{x}, t, g))$/* Updating the Lagrange multiplier for the boundary condition */
      $\lambda_{\mathcal{I}} \leftarrow \lambda_{\mathcal{I}} + \phi(\mathcal{I}(\boldsymbol{x}, t, h))$ /* Updating the Lagrange multipliers for the initial condition */
    $\eta = \sqrt{\pi(\theta)}$/* Recording the current penalty loss */
  end
end

In Algorithm 1, we present a training algorithm using the objective function presented in (19). The input to the algorithm is an initialized set of parameters for the network, a maximum value μ_{max} for safeguarding the penalty term, the number of epochs E, and the training set S. We should note that over-focusing on the constraints might result in a trivial prediction, where the constraints are satisfied, but the solution has not been found. Therefore, we tackle this issue by updating the multipliers when two conditions are met simultaneously: First, the ratio of the penalty loss term from successive iterations has not decreased. Second, the maximum allowable violation on the constraints has not been met. The first condition helps prevent aggressive updating of multipliers that might cause the aforementioned issue. In the second condition, we relax updating the multipliers if a satisfactory precision set by the user ϵ has been achieved. This, in return, enables the network to freely choose to optimize any loss terms in the objective function to not sacrifice any loss term.

Next, we discuss a "lean" residual neural network that we employ for some of our numerical experiments. Conventional feed-forward neural networks are prone to the notorious problem of vanishing-gradients, which makes learning significantly stiff. He et al. [2] proposed residual learning to alleviate this issue by introducing skip connections. Fig. 1(a) shows a schematic representation of a residual block that has two weight layers and a nonlinear activation function σ [2]. However, to preclude the problem of vanishing gradient, the non-linearity after the summation junction $\oplus$ and the shortcut connection should be identity as proposed by He et al. [49] as well as in [50]. We further observe that the weight layer before the junction becomes redundant because the output of the current residual layer will be fed to another residual layer that processes its input through a weight layer. In other words, linearly stacking two weight layers can be collapsed into a single weight layer. Therefore, we eliminate this extra weight layer and obtain a leaner residual layer. A schematic representation of our proposed modified residual layer is shown in Fig. 1(b) with $\mathcal{S}(x)$ shortcut mappings, which are identity mappings except for the input layer to project the input dimension to the correct dimension of the hidden layers.

3.1. Performance metrics

We assess the accuracy of our models by providing the L_∞ and the relative L_2 errors. Given an n-dimensional vector of predictions $\hat{\boldsymbol{u}} \in \mathbf{R}^n$ and an n-dimensional vector of exact values $\boldsymbol{u} \in \mathbf{R}^n$, we define the relative L_2 norm and L_∞ norm as follows:

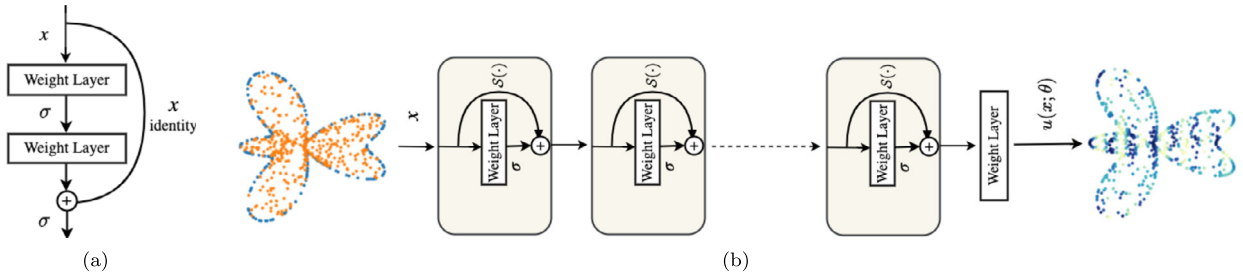

Fig. 1. (a) a schematic representation of the original residual block with a set of parameters θ and nonlinear activation functions σ, (b) a schematic representation of our proposed residual neural network architecture with a set of parameters θ and nonlinear activation functions σ with $\mathcal{S}(\cdot)$ skip connections.

$$\text{Relative } L_2 = \frac{\|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_2}, \quad L_\infty = \|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_\infty \tag{24}$$

where $\|\cdot\|_2$ indicates the Euclidean norm.

4. Application to forward problems

We apply our framework to learn the solution of several prototypical partial differential equations (PDE) that appear in computational physics. We also compare our results with existing methods to highlight the marked improvements in accuracy levels.

4.1. Two-dimensional Poisson's equation

Elliptic PDEs lack any characteristic path, which makes the solution at every point in the domain influenced by all other points. Therefore, learning the solution to elliptic PDEs with neural network based approaches that do not properly constrain the boundary conditions becomes challenging as we will show in this section. Here, we solve a two-dimensional Poisson's equation on a complex domain to not only highlight the applicability of our approach to irregular domains, but also show that our framework properly imposes the boundary conditions and produces physically feasible solutions. We also conduct a study to show the impact of distance functions ϕ and the maximum penalty parameter $\mu_{\max}$ that appear in Eq. (19) on the prediction of our neural network model. Let us consider the following PDE:

$$\nabla^2 u(x, y) = f(x, y), \qquad (x, y) \in \Omega, \tag{25a}$$

$$u(x, y) = h(x, y) \qquad (x, y) \in \partial\Omega, \tag{25b}$$

where $f(x, y)$ and $h(x, y)$ are source functions, $\Omega = \{(x, y) \mid x = 0.55\rho(\theta)\cos(\theta),\ \ y = 0.75\rho(\theta)\sin(\theta)\}$ and $\rho(\theta) = 1 + \cos(\theta)\sin(4\theta)$ for $0 \le \theta \le 2\pi$. We manufacture a complex oscillatory solution for Eq. (25a) and its boundary conditions Eq. (25b) as follows:

$$u(x, y) = \cos(\pi x)\cos(3\pi y), \qquad (x, y) \in \Omega. \tag{26}$$

The corresponding source functions $f(x, y)$ and $g(x, y)$ can be calculated exactly using Eq. (26). We use our "lean" residual neural network architecture with 3-layer hidden layers and 50 neurons per layer. We generate $N_\Omega = 512$ residual points uniformly from the interior part of the domain at each optimization step and $N_{\partial\Omega} = 512$ from the boundaries only once before training. Our optimizer is Adam with its default parameters and an initial learning rate of 10^{-2}. We train our network for 25000 epochs. We reduce our learning rate by a factor of 0.95 after 100 epochs with no improvement using *ReduceLROnPlateau* learning scheduler that is built in PyTorch framework [20]. For the present case, the prediction of our PECANN model for the entire domain is juxtaposed in Fig. 2.

From Fig. 2 we observe that our neural network model trained with our proposed approach has successfully learned the underlying solution. Since our physics-informed neural network model diverged, we do not portray its prediction for the entire domain. However, we present a summary of our error norms averaged over five independent trials with random Xavier initialization scheme [51] for both approaches in Table 1. The results indicate that our method achieves a relative $L_2 = 5.90 \times 10^{-4}$, which is three orders of magnitude lower than the one obtained from conventional physics-informed neural networks.

Next, we conduct an ablation study to investigate the impact of the distance function ϕ on the prediction of our model. A schematic representation of two different distance functions is presented in Fig. 3(a). Our analysis reveals that quadratic distance functions are not only insensitive to the choice of the maximum penalty parameter $\mu_{\max}$ but also significantly outperform the absolute distance function as shown in Fig. 3(b)-(c). Therefore, we adopt the quadratic distance function in our proposed method. To complement our analysis of elliptic PDEs, we present the applications of the PECANNs for a one-dimensional and a three-dimensional Poisson equation in the appendix.

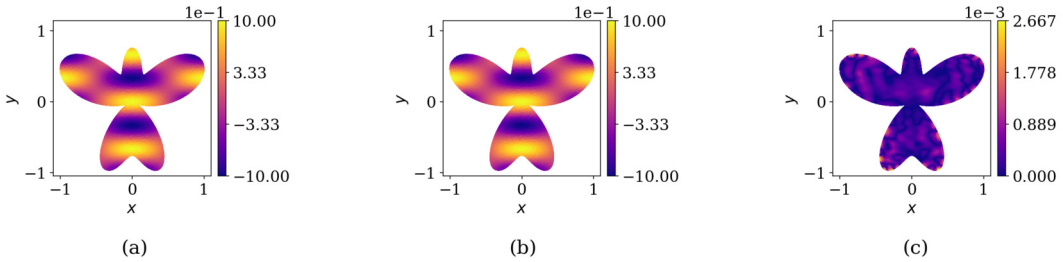

Fig. 2. Poisson's equation: (a) exact solution, (b) predicted solution by PECANN with quadratic distance function, (c) absolute point-wise error. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

Table 1
2D Poisson's equation. Summary of the average and the standard deviation of the relative L_2 and L_∞ errors over 5 independent trials along with the number of generated collocation points for training a fixed neural network architecture with different methods.

Models	Relative L_2	L_∞	N_Ω	$N_{\partial\Omega}$
PINN	$1.29 \times 10^{-1} \pm 2.28 \times 10^{-2}$	$4.67 \times 10^{-1} \pm 8.68 \times 10^{-2}$	512×25000	512
PECANN	$\mathbf{5.90 \times 10^{-4} \pm 7.69 \times 10^{-5}}$	$\mathbf{4.12 \times 10^{-3} \pm 1.47 \times 10^{-3}}$	512×25000	512

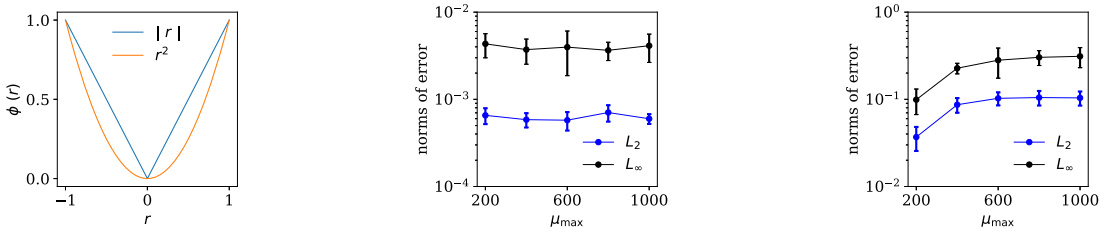

Fig. 3. (a) quadratic and absolute distance functions, (b) relative L_2 error bars versus $\mu_{\max}$ for quadratic distance function averaged over 5 independent trials, (c) relative L_2 error bars versus $\mu_{\max}$ for absolute distance function averaged over 5 independent trials.

4.2. Two-dimensional Helmholtz equation

Helmholtz equation arises in the study of electromagnetic radiation [52,53], seismology [54], acoustics [55] and many areas of engineering science. In this section, we study the following benchmark problem that was presented in [36]

$$\nabla^2 u(x, y) + k^2 u(x, y) = q(x, y), \ \forall (x, y) \in \Omega, \tag{27a}$$

$$u(x, y) = 0, \ \forall (x, y) \in \partial\Omega, \tag{27b}$$

where $k = 1$, $\Omega = \{(x, y) \mid -1 \leq x \leq 1, -1 \leq y \leq 1\}$ and $\partial\Omega$ is its boundary. Following the equation presented above, we manufacture an oscillatory solution that satisfy Eq. (27b) with its boundary conditions as follows:

$$u(x, y) = \sin(\pi x) \sin(4\pi y), \forall (x, y) \in \Omega. \tag{28}$$

We use the same fully connected neural network architecture as in [36], which consists of three hidden layers with 30 neurons per layer and the tangent hyperbolic activation function. We use a Sobol sequence to sample $N_\Omega = 512$ residual points from the interior part of the domain and $N_{\partial\Omega} = 256$ from the boundaries only once before training. We note that [36] is generating their data at every epoch, which amounts to $N_\Omega = 5.12 \times 10^6$ and $N_{\partial\Omega} = 20.48 \times 10^6$. Our optimizer is L-BFGS [56] with its default parameters and *strong Wolfe* line search function that is built in PyTorch framework [20]. We train our network for 5000 epochs with our safeguarding penalty parameter $\mu_{\max} = 10^4$.

As illustrated in Fig. 4(b), our PECANN model produces an accurate prediction to the underlying solution with uniform error distribution across the domain as shown in Fig. 4(c). We also present a summary of the error norms from our approach and state-of-the-art results presented in [36] averaged over ten independent trials with random Xavier initialization scheme [51] in Table 2. We observe that results obtained from our method achieve a relative $L_2 = 4.23 \times 10^{-4}$, which is two orders of magnitude lower than 4.31×10^{-2} obtained from the method presented in Wang et al. [36] with only a fraction of their generated data.

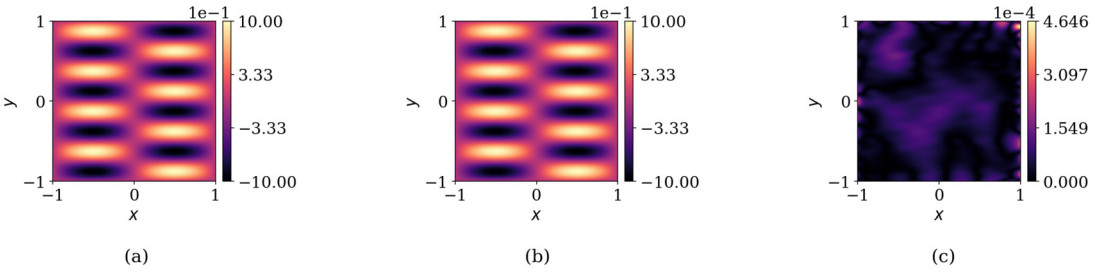

Fig. 4. Helmholtz equation: (a) exact solution, (b) predicted solution from PECANN model, (c) absolute point-wise error.

Table 2
Helmholtz equation: summary of the average and the standard deviations of the relative L_2 and L_∞ errors over 10 independent trials along with the number of generated collocation points for training a fixed neural network architecture with different methods along.

Models	Relative L_2	L_∞	N_Ω	$N_{\partial\Omega}$
Ref. [36]	$4.31 \times 10^{-2} \pm 1.68 \times 10^{-2}$	–	128×40000	$4 \times 128 \times 40000$
PECANN	$\mathbf{4.23 \times 10^{-4} \pm 3.09 \times 10^{-4}}$	$\mathbf{1.53 \times 10^{-3} \pm 7.66 \times 10^{-4}}$	512	4×64

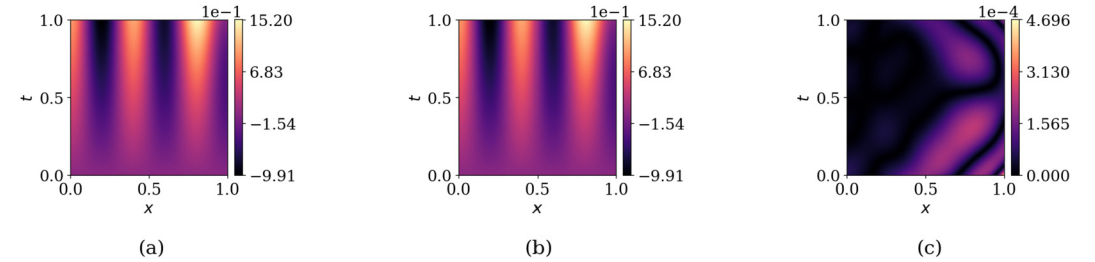

Fig. 5. Klein Gordon equation: (a) exact solution, (b) predicted solution by PECANN, (c) point-wise absolute error.

4.3. Klein-Gordon equation

We consider a nonlinear time-dependent benchmark problem known as the Klein-Gordon equation, which plays a significant role in many scientific applications such as particle physics, astrophysics, cosmology, and classical mechanics. This problem was considered in the work of Wang et al. [36] as well. Consider the following partial differential equation

$$\frac{\partial^2 u}{\partial t^2} + \alpha \frac{\partial^2 u}{\partial x^2} + \beta u + \gamma u^k = f(x,t),\ \forall (x,t) \in \Omega \times [0,T], \tag{29a}$$

$$u(x,0) = g_1(x),\ \forall x \in \Omega, \tag{29b}$$

$$\frac{\partial u(x,0)}{\partial x} = g_2(x),\ \forall x \in \Omega, \tag{29c}$$

$$u(x,t) = h(x,t)\ \forall (x,t) \in \partial\Omega \times [0,T], \tag{29d}$$

where $\alpha = -1$, $\beta = 0$, $\gamma = 1$ and $k = 3$ are known constants. $\Omega = [0,1] \times [0,1]$ with $T = 1$. The manufactured solution presented in [36] is as follows

$$u(x,t) = x\cos(5\pi t) + (xt)^3. \tag{30}$$

The corresponding forcing function $f(x,t)$, boundary condition $h(x,t)$ and initial conditions $g_1(x)$ and $g_2(x)$ can be calculated exactly using Eq. (30). We use the same neural network architecture as in [36] which is a deep fully connected neural network with 5 hidden layers each with 50 neurons that we train for 1500 epochs total. We use Sobol sequence to generate $N_\Omega = 512$ residual points from the interior part of the domain, $N_{\partial\Omega} = 512$ points from the boundaries and $N_I = 256$ points for each of the initial conditions as in Eq. (29b) and Eq. (29c) only once before training. Our optimizer is LBFGS with its default parameters and *strong Wolfe* line search function that is built in PyTorch framework [20]. Our safeguarding penalty parameter $\mu_{\max} = 10^4$ as in the previous problem.

As illustrated in Fig. 5(b), our PECANN model produces an accurate prediction to the underlying solution with uniform error distribution across the domain as shown in Fig. 5(c). In addition, we present a summary of the error norms averaged over ten independent trials with random Xavier initialization scheme [51] in Table 3. We observe that the best relative L_2

Table 3
Klein–Gordon equation: summary of the average and the standard deviations of the relative L_2 and L_∞ errors over 10 independent trials along with the number of generated collocation points for training a fixed neural network architecture with different methods along.

Models	Best Relative L_2	Relative L_2	L_∞	N_Ω	$N_{\partial\Omega}$	N_I
Ref. [36]	1.062×10^{-2}	–	–	128×40000	$2 \times 128 \times 40000$	$2 \times 128 \times 40000$
PECANN	$\mathbf{2.158 \times 10^{-4}}$	$\mathbf{6.139 \times 10^{-4} \pm 3.337 \times 10^{-4}}$	$\mathbf{1.043 \times 10^{-3} \pm 5.908 \times 10^{-4}}$	512	2×256	2×256

error obtained from our PECANN model is two orders of magnitude lower than the best relative L_2 norm error reported in [36] with only a fraction of their generated data. This highlights the predictive power of our method over state-of-the-art physics-informed neural networks for the solution of a non-linear time-dependent Klein-Gordon equation.

5. Application to inverse problems

In this section, we use our PECANN framework for the solution of inverse problems with multi-fidelity data. By multi-fidelity, we mean that we have both clean (high-fidelity) data and noisy (low-fidelity) data. We tackle three inverse problems involving PDEs. It is worth reiterating that we only impose equality constraints and use noisy data (e.g., noisy boundary conditions, noisy measurement data) as a soft-regularizer $\mathcal{J}_{\mathcal{M}}(\theta)$ in Eq. (19).

5.1. Learning hydraulic conductivity of nonlinear unsaturated flows from multi-fidelity data

Our PECANN framework is also suitable for the solution of inverse-PDE problems using multi-fidelity data. With multi-fidelity, we mean that the observed data may include both data with low accuracy and data with very high accuracy. As part of our objective function formulation, we can constrain the high-fidelity data in a principled fashion and take advantage of the low-fidelity data to regularize our hypothesis space. To demonstrate our framework, we study one of the difficult multi-fidelity example problems that were tackled in Meng and Karniadakis [32] with composite neural networks. This particular inverse-PDE problem arises in unsaturated flows as they are central in characterizing contaminant transport [57], soil-atmosphere interaction [58], soil-plant-water interaction [59], ground-subsurface water interaction zone [60] to name a few. Describing processes involving soil-water interactions at a microscopic level is very complex due to the existence of tortuous, irregular, and interconnected pores [61]. Therefore, these flows are generally characterized in terms of their macroscopic characteristics. An important quantity that is essential in describing flows through unsaturated soil is hydraulic conductivity, which is a nonlinear parameter that is highly dependent on the geometry of the porous media [61]. Let us consider the following nonlinear differential equation representing an unsaturated one-dimensional (1D) soil column with variable water content:

$$\frac{d}{dx}(K(h)\frac{dh(x)}{dx}) = 0, \qquad x \in \Omega, \tag{31}$$

subject to the following boundary conditions,

$$h(0) = -3, \tag{32a}$$

$$h(200) = -10, \tag{32b}$$

where $\Omega = \{x \mid 0 \leq x \leq 200 \text{ cm}\}$, $h(x)$ is the pressure head (cm) and $K(h)$ is the hydraulic conductivity (cm h^{-1}) which is described as follows:

$$K(h) = K_s S_e^{1/2}\Big[1 - (1 - S_e^{1/m})^m\Big]^2, \tag{33}$$

where K_s is the saturated hydraulic conductivity (cm h^{-1}), and S_e is the effective saturation expressed as follows [62]:

$$S_e = \frac{1}{(1 + |\alpha h|^n)^m}, m = 1 - 1/n, \tag{34}$$

where α is an empirical parameter that is inversely related to the air-entry pressure value (cm^{-1}) and m is an empirical parameter related to the pore-size distribution that is hard to measure due to the complex geometry of the porous media. We aim to infer the unknown empirical parameters α, and m from sparse measurements of pressure head h. To generate multi-fidelity synthetic measurements or experimental data, we select the soil type *loam* for which the empirical parameters are as follows: $\alpha = 0.036$ and $m = 0.36$.

We generate high-fidelity pressure data using the exact empirical parameters and low-fidelity data with $\alpha = 0.015$ and $m = 0.31$. Using the built-in `bvp5c` MATLAB function, we solve the governing PDE as given in Eqs. (31) through Eq. (34) using the selected empirical parameters to generate multi fidelity training data as shown in Fig. 6(a). In Fig. 6(b) we also depict the corresponding hydraulic conductivity $k(h)$ values for the pressure head data, which shows that low-fidelity hydraulic conductivity has a significant deviation from the exact hydraulic conductivity distribution. To highlight the

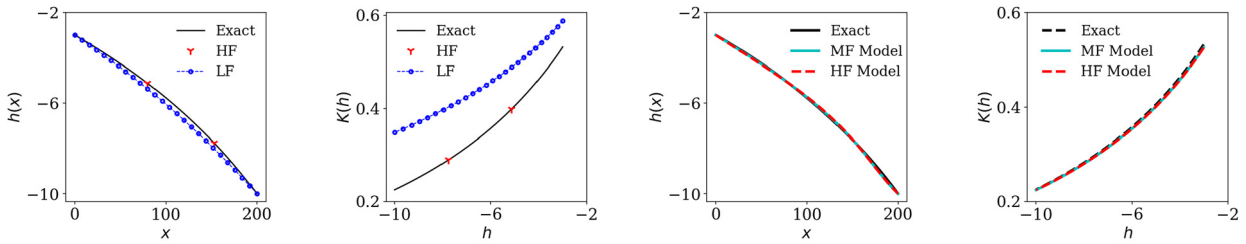

Fig. 6. Parameter inference on multi-fidelity data for unsaturated flow through porous media: (a) low fidelity (LF) and high fidelity (HF) pressure head data used for training, (b) hydraulic conductivity corresponding to low-fidelity and high-fidelity training data, (c) pressure head reconstruction by PECANN model trained on high-fidelity and multi-fidelity data separately, (d) reconstructed hydraulic conductivity by PECANN model trained by high-fidelity and multi-fidelity data separately.

Table 4
Summary of the inferred parameters from using high-fidelity (HF) only or multi-fidelity (MF) data in the learning process averaged over ten different runs. Note that the training time, averaged over 10 independent trials, for our PECANN model is only 4 seconds on a CPU.

Models	Avg. α	$\sigma(\alpha)$	Relative Error(α)	Avg. m	$\sigma(m)$	Relative Error(m)
Ref. [32] with HF data only	0.0440	–	22.22%	0.377	–	4.72%
PECANN with HF data only	**0.0351**	$\mathbf{7.18 \times 10^{-4}}$	**2.58%**	**0.354**	$\mathbf{2.78 \times 10^{-3}}$	**1.78%**
Ref. [32] with MF data	0.0337	7.91×10^{-4}	6.39%	0.349	3.70×10^{-3}	3.06%
PECANN with MF data	**0.0359**	$\mathbf{7.51 \times 10^{-4}}$	**0.30%**	**0.357**	$\mathbf{2.74 \times 10^{-3}}$	**0.86%**
Exact value	0.0360	–	–	0.360	–	–

robustness, efficiency, and accuracy of our framework on an inverse-PDE with multi-fidelity data fusion, we compare our results with the results reported in Meng and Karniadakis [32]. For comparison purposes, we also choose a feed-forward neural network with two hidden layers with 20 neurons per layer as in [32] for their physics-informed neural network trained on high fidelity alone which failed to discover the parameters of interest. However, Meng and Karniadakis [32] constructed customized networks for high fidelity data and low fidelity data separately and then aggregated them together by manually crafted correlations. Therefore, they refer to their approach as composite neural networks. Unlike Meng and Karniadakis [32], we do not need to make any inductive bias about the data and, therefore, use a single network initialized with Xavier initialization technique [51] that we separately train on high-fidelity and multi-fidelity data. This shows the robustness and efficiency of our approach that we can train the same network on multi-fidelity data without the need to design customized networks to process data differently. We let a single network discover and extract features from multi-fidelity data with the help of known physics. We use Adam with its default parameters and 10^{-2} initial learning rate. We set the maximum penalty parameter $\mu_{\max} = 10^4$ and train our network for 2000 epochs total. As for the collocation points, we use the Sobol sequence and generate 400 residual points from across our domain in each epoch. As considered in [32], we assume the flux at the inlet q_0 is known, which allows us to use the integral form of Eq. (31) given as follows,

$$q(x) = -K(h)\frac{dh(x)}{dx} = q_0, \quad \frac{dq(x)}{dx} = 0. \tag{35}$$

Fig. 6(a) and Fig. 6(b) depict the reconstructed pressure head and the corresponding hydraulic conductivity distributions obtained from our PECANN model trained on high-fidelity and multi-fidelity data separately. Compared with the exact solution, it is seen that the inferred results are highly accurate, which shows the robustness and efficiency of our method. Furthermore, in Table 4, we report the average and standard deviation of inferred α and m from our model along with the results from Meng and Karniadakis [32]. The results are over 10 independent trials with random initialization using Xavier [51] scheme.

From Table 4, we observe that our results are significantly outperforming the reported results in [32]. It is worth noting that we are using just a single neural network architecture that is the low-fidelity model in the composite neural network model proposed in [32] and our average CPU training time is only 4 seconds.

5.2. Boundary heat flux identification

In this section, we apply our framework to study an inverse heat conduction problem (IHCP) where boundary conditions are partially accessible. Typically, these problems arise in a plethora of industrial and engineering applications where measurements can only be made in easily-accessible locations or the quantity of interest can be measured indirectly. Unfortunately, inverse problems are ill-posed and ill-conditioned because unknown solutions and parameter values usually have

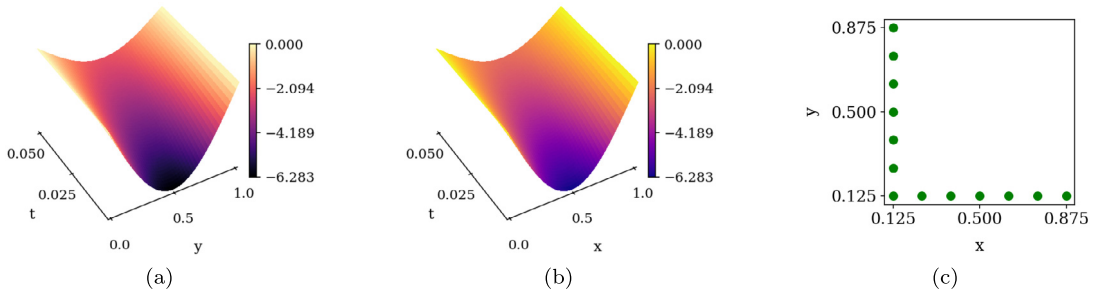

Fig. 7. (a) exact q_x, (b) exact q_y, (c) location of thermocouples.

to be determined from indirect observable data that contains measurement error [63–66]. Here, we aim to identify spatio-temporal boundary heat flux given partial spatio-temporal temperature observations inside the domain as in the work of Wang and Zabaras [66].

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2},\ 0 < x, y < 1, t \in [0, 1], \tag{36a}$$

$$T(x, y, 0) = -2\sin(\pi x)\sin(\pi y),\ 0 \le x, y \le 1, \tag{36b}$$

$$T|_{x=1} = T|_{y=1} = 0,\ 0 < t < 1 \tag{36c}$$

$$\left.\frac{\partial T}{\partial x}\right|_{x=0} = q_x\,(\text{unknown}),\ 0 < t < 1, \tag{36d}$$

$$\left.\frac{\partial T}{\partial y}\right|_{y=0} = q_y,\ (\text{unknown})\ 0 < t < 1, \tag{36e}$$

where q_x and q_y are the unknown heat fluxes to be discovered. As considered in [66], an analytical solution to this problem can be obtained as follows

$$T(x, y, t) = -2\pi\sin(\pi x)\sin(\pi y)e^{-2\pi^2 t}, \tag{37}$$

with the exact heat fluxes as follows

$$q_x = -2\pi\sin(\pi y)e{-}2\pi^2 t \tag{38a}$$

$$q_y = -2\pi\sin(\pi x)e{-}2\pi^2 t. \tag{38b}$$

An exact representation of q_x and q_y are presented in Fig. 7(a) and (b). The inverse problem is to discover q_x and q_y given partial observation from a set of thirteen thermocouples with 0.125 space interval and 0.125 distance to the boundary as shown in Fig. 7(c).

The sampling time interval is taken as $dt = 0.002$. The heat flux history was reconstructed for the time range $t \in [0 : 0.05]$, N = 25, hence, there are 325 observations. Wang and Zabaras [66] represented the unknown flux quantities by parametric linear functions and proposed a Bayesian approach by employing a specialized model of Markov random field (MRF) as prior distribution. Three different cases were considered. Uncertainty in temperature measurements was modeled as stationary zero-mean white noise with standard deviations of $\sigma = 0.005$, $\sigma = 0.01$ and $\sigma = 0.02$. We employ a 3 hidden-layer fully-connected neural network with 30 neurons per layer to learn the temperature field for the entire domain. Our optimizer is LBFGS with its default parameters and *strong-Wolfe* line search function built-in PyTorch framework. We set the limiting penalty parameter $\mu_{\max} = 10^4$ similar to previous problems and we train our network for 10000 epochs. We use Sobol sequences to sample 512 residual points in the domain, 512 points for the Dirichlet boundary conditions, and 512 points for the initial condition only once before training our network. The predictions of our neural network model are shown in Fig. 8. We observe that our network has successfully inferred heat fluxes for all three cases. A summary of the error percentage from our method along with the best results from [66] is provided in Table 5. We observe that our approach has improved the reported results of [66] by a factor of 10 in all three cases.

5.3. Patient-specific tumor growth modeling

In this section, we aim to develop a patient-specific tumor model using noisy magnetic resonance images (MRI). Treatment for tumors involves surgery, radiation, and chemotherapy. Nevertheless, cancer cells may remain after surgery, resulting in recurrence of the tumor and eventual death [67,68]. Therefore, models based on patient-specific information

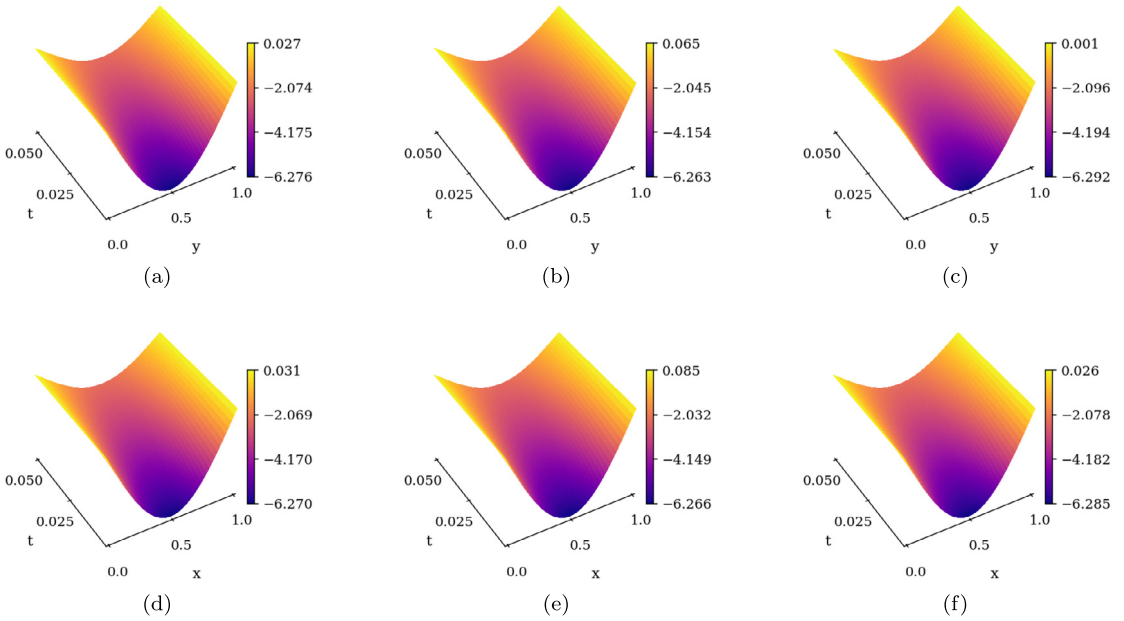

Fig. 8. Heat flux reconstruction: Top row: q_x (a) predicted flux distribution for case I, (b) predicted flux distribution for case II, (c) predicted flux distribution for case III. Bottom row: q_y, (d) predicted flux distribution for case I, (e) predicted flux distribution for case II, (f) predicted flux distribution for case III.

Table 5
q_x reconstruction error by different methods with noisy measurement data.

Models	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.02$
Ref. [66]	4.62%	5.45%	5.75%
PECANN with MF data	**0.53%**	**0.61%**	**0.89%**

are needed to identify tumor cells that may lie beyond the threshold visible to magnetic resonance imaging. Assuming isotropic brain structure and radial symmetry, we can describe tumor cell density evolution using the following non-linear reaction-diffusion type partial differential equation [69–73].

$$\frac{\partial u(r,t)}{\partial t} = D\frac{\partial^2 u(r,t)}{\partial r^2} + \rho u(r,t)(1-u(r,t)),\ \text{in}\ \Omega \times [0,5] \tag{39}$$

$$\frac{\partial u(r,t)}{\partial r} = 0,\ \text{on}\ \partial\Omega \tag{40}$$

$$u(r,0) = \varphi(r),\ \text{in}\ \Omega \tag{41}$$

where $\Omega = \{r \mid 0 \le r \le 10\}$ is the domain with its boundary $\partial\Omega$, $u(r,t)$ is the unknown tumor cell density at time t [year] and distance r [mm]. D is the unknown diffusion coefficient of tumor cells in the brain tissue and ρ is the unknown proliferation coefficient. φ is a point source initial condition. It is assumed that at the time of death $t = 5$, the visually detectable area of tumor volume is equal to a circle of 10 mm in radius. As a proof of concept, we generate synthetic MRI data by solving Eq. (39) in forward mode using finite difference scheme with $\Delta r = 0.0196$, $\Delta t = 10^{-5}$ assuming $D = 0.50$, $\rho = 1.00$ with the following initial condition

$$\varphi(r) = \frac{1}{10}e^{-r},\ \text{in}\ \Omega. \tag{42}$$

Our synthetic data includes two solutions at $t = 1$ and $t = 2$ that simulate patient tumor cell density distribution obtained from MRI of brain scans at the corresponding time states. We further corrupt these data using uncorrelated Gaussian noise with $\sigma = 0.01$. The corresponding noise percent of the data is presented in Table 6.

From Table 6 we observe that our data contain different levels of noise which indicates different levels of fidelity. Finally, we use our corrupted tumor density distribution at $t = 1$ and $t = 2$ as low fidelity data along with Eq. (40) as our boundary constraint (high fidelity data) to infer unknown parameters of Eq. (39). For this problem, we generate $N_\Omega = 512$ residual points to approximate the loss on Eq. (39) and $N_{\partial\Omega} = 512$ to constrain the boundaries only once before training. We also generate 512 points with their labels from our corrupted synthetic brain scan data. We use a feed-forward neural network

Table 6
Error percentage of corrupted MRI data with uncorrelated Gaussian noise with standard deviation $\sigma = 0.01$.

	$u(r, t = 1)$	$u(r, t = 2)$
Noise %	10.34%	5.01%

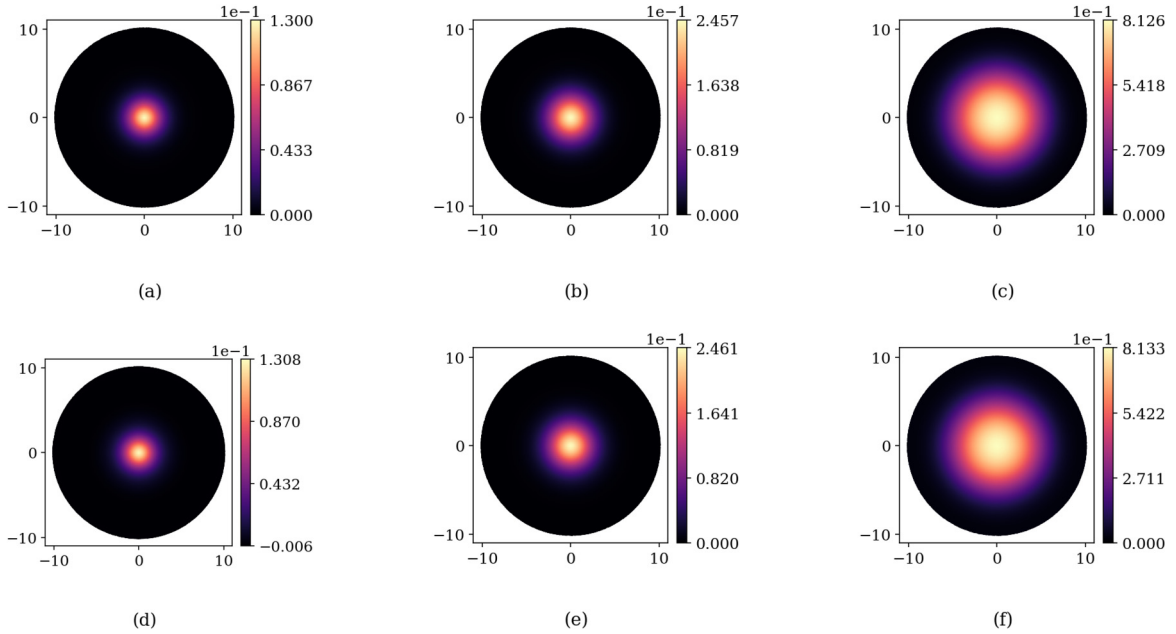

Fig. 9. Natural tumor cell density distribution at different time states. Top row: (a) synthetic brain scan at year one used for training, (b) synthetic brain scan at year two used for training, (c) synthetic brain scan at year five used for testing. Bottom row: (d) reconstructed brain scan data at year one, (e) reconstructed brain scan data at year two, (f) predicted brain scan data at year five.

Table 7
Tumor growth modeling. Summary of inferred parameters using multi-fidelity data in the learning process averaged over ten independent trials with random initialization.

	D	ρ
Exact	0.50	1.00
PECANN with MF data	$0.49 \pm 4.90 \times 10^{-3}$	$1.00 \pm 9.93 \times 10^{-4}$

with two hidden layers each with 10 neurons per layer. Our optimizer is LBFGS with its default parameters and *strong Wolfe* line search function built in PyTorch framework [20]. Our network is trained for 200 epochs with Xavier initialization scheme [51]. We initialize $D \in [0.3285, 0.973]$ and $\rho \in [0.73, 2.92]$ randomly as suggested in [74].

From Fig. 9 we observe that our model not only reconstructs the original data from corrupted noisy data, but also generalizes well to predict the unseen MRI data at the terminal year $t = 5$.

A summary of our inferred parameters is presented on Table 7 over 10 independent trials with random Xavier initialization scheme [51].

6. Conclusion

We have shown that the unconstrained optimization problem formulation pursued in physics-informed neural networks (PINN) is a major source of poor performance when the PINN approach is applied to learn the solution of more challenging multi-dimensional PDEs. We addressed this issue by introducing physics- and equality-constrained artificial neural networks (PECANN), in which we pursue a constrained-optimization technique to formulate the objective function in the first place. Specifically, we adopt the augmented Lagrangian method (ALM) to constrain the PDE solution with its boundary and initial conditions, and with any high-fidelity data that may be available. The objective function formulation in the PECANN model is sufficiently general to admit low-fidelity data to regularize the hypothesis space in inverse problems as well. We applied our proposed method for both forward problems and inverse problems with multi-fidelity data fusion. For all the problems

considered, our PECANN model produced results that are in excellent agreement with exact solutions, while the PINN approach failed to produce acceptable predictions.

It is a common practice to use conventional feed-forward neural networks in the PINN approach. However, these types of neural networks are known to suffer from the so-called vanishing gradient problem, which stalls the learning process. Residual layers (a.k.a. ResNets) that were originally proposed by He et al. [2] tackle the vanishing gradient problem with identity skip connections. In our work, we have modified the original residual layers by restricting them to a single weight layer with a *tanh* activation function and identity skip connections. We find our leaner version of the residual layers to be very effective in improving the accuracy of the PDE predictions for both the original PINN model and our PECANN model.

Our findings suggest that not only the choice of the neural network architecture, but also the optimization problem formulation is crucial in accurately learning PDEs using artificial neural networks. We conjecture that future progress in physics-constrained (informed) learning of PDEs would come from exploring new approaches in the field of non-convex constrained optimization field. Future endeavors could shed light on challenging questions such as: how does the loss landscape of neural networks change with respect to the optimization problem formulation? What is the optimal neural network architecture for PDE learning? And, is there a physics-based approach in searching for optimal architectures?

CRediT authorship contribution statement

Shamsulhaq Basir: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – review & editing. **Inanc Senocak:** Conceptualization, Funding acquisition, Methodology, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1953204 and in part by the University of Pittsburgh, Center for Research Computing through the resources provided.

Appendix A. Additional examples

A.1. One-dimensional Poisson's equation

The aims of this pedagogical example are twofold: First, we thoroughly demonstrate the implementation intricacies of our proposed method and highlight its advantages over the PINN approach. Second, we demonstrate the significant improvement achieved in the model prediction using our modified residual architecture relative to the original residual networks [2].

Let us consider the following one-dimensional Poisson's equation

$$\frac{d^2u}{dx^2} = -(15\pi)^2 \cos(15\pi x), \qquad x \in \Omega, \tag{A.1}$$

$$u(x) = \cos(15\pi x), \qquad x \in \partial\Omega, \tag{A.2}$$

where $\Omega = \{x \,|\, 0 \le x \le 1\}$ and $\partial\Omega$ is its boundary. The exact solution to the above problem is a sinusoidal nonlinear function $u(x) = \cos(15\pi x)$. Considering a neural network solution for the above equation as $\hat{u}(x;\theta)$ parameterized with θ, we write the residual form of this one-dimensional Poisson's equation as follows:

$$\mathcal{F} := \frac{d^2u_\theta}{dx^2} + (15\pi)^2 \cos(15\pi x) \qquad x \in \Omega, \tag{A.3}$$

$$\mathcal{B} := u_\theta(x) - \cos(15\pi x), \qquad x \in \partial\Omega. \tag{A.4}$$

Next, we use the above residual form of this differential equation to construct an objective function as proposed earlier in Eq. (19)

$$\mathcal{L}(\theta) = \sum_{i=1}^{N_\Omega} |\mathcal{F}(x^{(i)})|^2 + \sum_{i=1}^{2} \lambda^{(i)} \phi(\mathcal{B}(x^{(i)})) + \frac{\mu}{2} \sum_{i=1}^{2} |\phi(\mathcal{B}(x^{(i)}))|^2 \tag{A.5}$$

where μ is the penalty parameter and N_Ω is the number residual points sample from Ω at every epoch. $\lambda \in \mathbb{R}^2$ is a vector of Lagrange multipliers for the boundary constraints and ϕ is the quadratic distance function. In contrast to our constrained

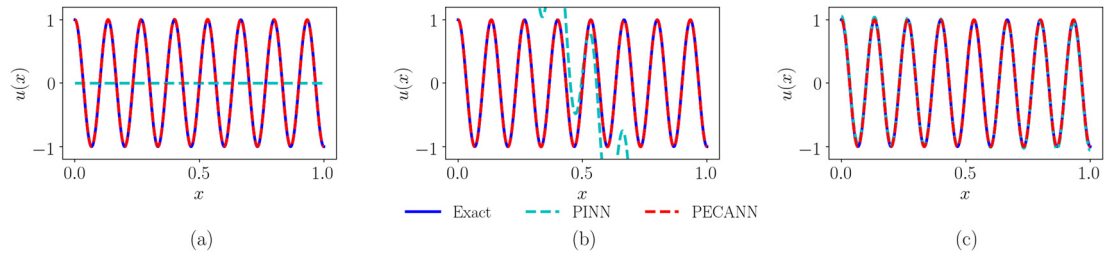

Fig. A.10. Performance comparison of PINN vs PECAN for different neural network architectures: (a) conventional neural network, (b) original residual neural network, (c) modified residual neural network. Note that PECANN approach converged with all network architectures while PINN only converged with our proposed modified residual neural network but with poor norms of errors.

Table A.8
One dimensional Poisson's equation: summary of relative L_2 norms and L_∞ norms for different neural network models trained with different approaches.

	PINN		PECANN	
	Relative L_2	L_∞	L_2	L_∞
Conventional NN	1.00	1.00	$\mathbf{1.73 \times 10^{-3}}$	$\mathbf{1.80 \times 10^{-3}}$
Original Residual NN	4.75	5.84	$\mathbf{3.41 \times 10^{-3}}$	$\mathbf{3.43 \times 10^{-3}}$
Proposed Residual NN	4.63×10^{-2}	5.74×10^{-2}	$\mathbf{1.35 \times 10^{-4}}$	$\mathbf{1.66 \times 10^{-4}}$

optimization with the ALM, the composite objective function adopted in PINNs (i.e. Eq. (4)) yields the following loss function for the current example

$$L(\theta) = \frac{1}{N_\Omega}\sum_{i=1}^{N_\Omega}|\mathcal{F}(x^{(i)})|^2 + \frac{1}{2}\sum_{i=1}^{2}|\mathcal{B}(x^{(i)})|^2 \tag{A.6}$$

Having constructed the objective functions using the constrained-optimization method in the present work and the composite approach adopted in PINNs, we design three networks in such a way that they have the same number of neurons and hidden layers to allow a fair comparison. We use six weight layers with 50 neurons per layer in all three neural network models. More specifically, we have six weight layers in our conventional feed-forward neural network model. Similarly, our second neural network model with the original residual layers has one weight layer in the front with two residual layers and an output weight layer, which makes a total of six weight layers. For our last neural network model with our proposed residual layers, we have a weight layer succeeded by four modified residual layers and an output weight layer that amounts to six weight layers as well. Therefore, all three models have the same number of neurons and the same number of weight layers and are end-to-end trainable. For this problem, the parameters of the network are initialized randomly with the Xavier initialization technique [51]. We use Adam [75] with an initial learning rate of 10^{-2}. We reduce our learning rate by a factor of 0.95 after 100 epochs with no improvement in the objective function. We use the same hyperparameters and train all the models under the same training settings with both objectives as in Eq. (A.5) and Eq. (A.6). We set the limiting penalty parameter $\mu^{\max} = 10^2$. As for the collocation points, we randomly generate $N_\Omega = 654$ residual points from across our domain with uniform probability along with two boundary conditions at each optimization step. The results from all three neural network architectures trained with the PINN and PECANN approaches are juxtaposed in Fig. A.10. We observe from these results that the PINN model with a composite objective function is visibly sensitive to the neural network choice and benefits the most from the adoption of modified residual layers, whereas the PECANN model with equality-constrained optimization is qualitatively less sensitive to the choice of the neural network architecture and performs very well for all three networks. From Table A.8 we observe that for conventional NN and for the original residual neural network the relative L_2 error from our PECANN model is three orders of magnitude lower than the one obtained from our PINN model. However, with our lean residual network, it decreases to two orders of magnitude which demonstrates the impact of our neural network architecture.

A.2. Three-dimensional Poisson's equation

We consider the following non-homogeneous three dimensional Poisson's equation in a cubic domain

$$\nabla^2 u(x, y, z) = f(x, y, z), \qquad (x, y, z) \in \Omega, \tag{A.7}$$

subject to the following boundary conditions

$$u(x, y, z) = g(x, y, z), \qquad (x, y, z) \in \partial\Omega, \tag{A.8}$$

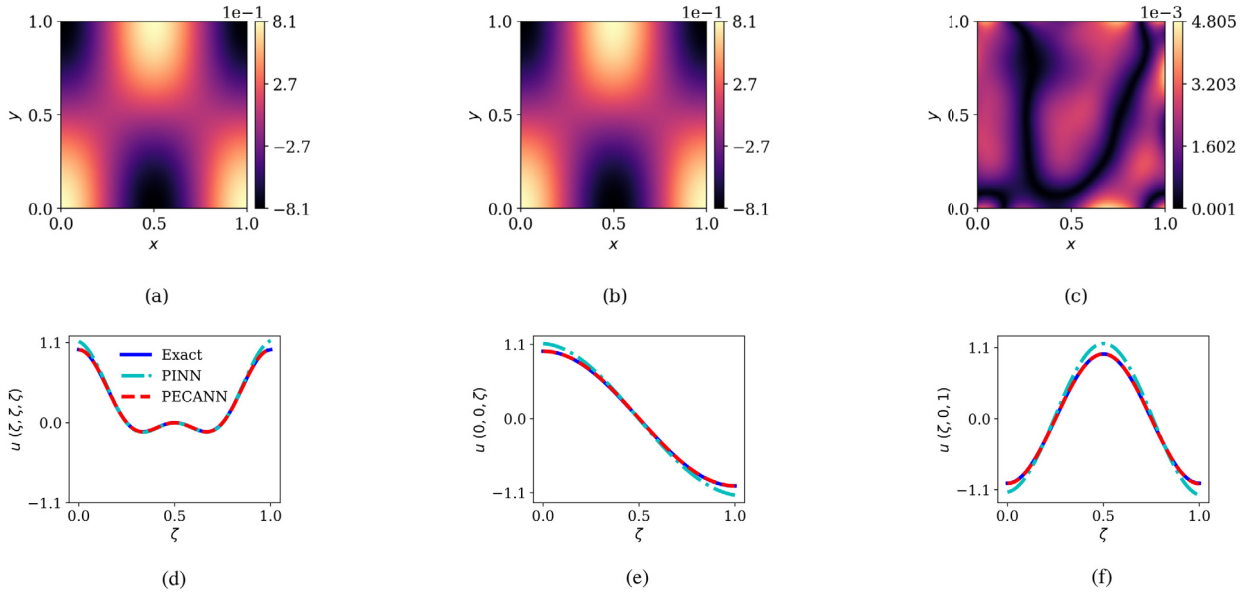

Fig. A.11. Three dimensional Poisson's equation. Top row: cross section view of the solution at $z = 0.2$, (a) exact solution, (b) predicted solution from PECANN model, (c) absolute point-wise error distribution. Bottom row: plots over line obtained from different methods, (d) straight line connecting point $(0, 0, 0)$ to point $(1, 1, 1)$, (e) straight line connecting $(0, 0, 0)$ and $(0, 0, 1)$ points, (f) straight line connecting point $(0, 0, 1)$ to point at $(1, 0, 1)$.

Table A.9
Three dimensional Poisson's equation: summary of relative L_2 norms and L_∞ norms for different neural network models averaged over 10 independent trials with random initialization with Xavier scheme.

Models	Relative L_2	L_∞	N_Ω	$N_{\partial\Omega}$
PINN	$1.09 \times 10^{-1} \pm 1.54 \times 10^{-2}$	$2.31 \times 10^{-1} \pm 4.56 \times 10^{-2}$	256×15000	6×256
PECANN	$\mathbf{2.39 \times 10^{-3} \pm 2.93 \times 10^{-4}}$	$\mathbf{6.21 \times 10^{-3} \pm 1.55 \times 10^{-3}}$	256×15000	6×256

where $\Omega = \{0 \leq x, y, z \leq 1\}$ with its boundary $\partial\Omega$, f and g are known source functions in Ω and on $\partial\Omega$. We manufacture a sinusoidal solution of the following form

$$u(x, y, z) = \cos(2\pi x)\cos(\pi y)\cos(\pi z), \forall (x, y, z) \in \Omega. \tag{A.9}$$

We will use the exact solution eq. (A.9) to evaluate the source functions f and g and solve Eq. (A.7). For this purpose, we use our lean residual neural network with three hidden layers each with 50 neurons. Our optimizer is Adam [75] with its default parameters and 10^{-2} initial learning rate. We also reduce our learning rate by a factor of 0.95 if the objective does not improve after 100 optimization steps. Our network is trained for 15000 epochs with randomly initialized weights according to Xavier scheme [51]. We generate $N_{\partial\Omega} = 6 \times 256$ number of points on the boundaries $\partial\Omega$ only once before training and $N_\Omega = 256$ residual points in the domain Ω at every optimization step. We present a section view of the predicted solution obtained from our PECANN model in Fig. A.11. Since our physics-informed neural network failed to converge as can be seen from the error norms in Table A.9, we did not include a section view of its predicted solution. However, we provide plots over straight lines drawn between two points within the domain. From Fig. A.11(d)-(f) we observe that our PINN model either underpredicted or overpredicted the regions with high gradients and regions close to the boundaries. However, our PECANN model successfully learned the underlying solution. From Table A.9 we observe that for the relative L_2 error from our PECANN model is two orders of magnitude lower than the one obtained from our PINN model which highlights the effectiveness of our method over conventional physics-informed neural networks.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, NIPS 25 (2012) 1097–1105.
[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[3] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (2012) 82–97.
[4] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, CoRR, arXiv:1409.3215, 2014.
[5] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[6] J. Weston, S. Chopra, K. Adams, #TagSpace: semantic embeddings from hashtags, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1822–1827.
[7] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[8] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[9] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (1993) 861–867.
[10] P. Grohs, F. Hornung, A. Jentzen, P. Von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations, arXiv preprint, arXiv:1809.02362, 2018.
[11] J. Darbon, G.P. Langlois, T. Meng, Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures, Res. Math. Sci. 7 (2020).
[12] J. Berner, P. Grohs, A. Jentzen, Analysis of the generalization error: empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations, SIAM J. Math. Data Sci. 2 (2020) 631–657.
[13] M.W.M.G. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (1994) 195–201.
[14] B.P. van Milligen, V. Tribaldos, J.A. Jiménez, Neural network differential equation and plasma equilibrium solver, Phys. Rev. Lett. 75 (1995) 3594–3597.
[15] C. Monterola, C. Saloma, Solving the nonlinear Schrodinger equation with an unsupervised neural network, Opt. Express 9 (2001) 72–84.
[16] D. Parisi, M.C. Mariani, M. Laborde, Solving differential equations with unsupervised neural networks, Chem. Eng. Process. 42 (2003) 715–721.
[17] M. Hayati, B. Karami, Feedforward neural network for solving partial differential equations, J. Appl. Sci. 7 (2007) 2812–2817.
[18] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (1998) 987–1000.
[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USENIX Association, USA, 2016, pp. 265–283.
[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 8024–8035.
[21] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[22] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. USA 115 (2018) 8505–8510.
[23] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[24] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[25] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[26] M. Raissi, Deep hidden physics models: deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (2018) 932–955.
[27] M. Raissi, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. 861 (2019) 119–137.
[28] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 358 (2020) 112623.
[29] A. Yazdani, L. Lu, M. Raissi, G.E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLoS Comput. Biol. 16 (2020) e1007575.
[30] B.M. de Silva, J. Callaham, J. Jonker, N. Goebel, J. Klemisch, D. McDonald, N. Hicks, J.N. Kutz, S.L. Brunton, A.Y. Aravkin, Physics-informed machine learning for sensor fault detection with flight test data, arXiv preprint, arXiv:2006.13380, 2020.
[31] Y. Liu, J.N. Kutz, S.L. Brunton, Hierarchical deep learning of multiscale differential equation time-steppers, arXiv preprint, arXiv:2008.09768, 2020.
[32] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: application to function approximation and inverse pde problems, J. Comput. Phys. 401 (2020) 109020.
[33] A.A. Ramabathiran, P. Ramachandran, SPINN: sparse, physics-based, and partially interpretable neural networks for PDEs, J. Comput. Phys. 445 (2021) 110600.
[34] R. van der Meer, C.W. Oosterlee, A. Borovykh, Optimally weighted loss functions for solving PDEs with neural networks, CoRR, arXiv:2002.06269, 2020.
[35] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv preprint, arXiv:2009.04544, 2020.
[36] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (2021) A3055–A3081.
[37] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508.
[38] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[39] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint, arXiv:1212.5701, 2012.
[40] M.J. Powell, A method for nonlinear constraints in minimization problems, in: R. Fletcher (Ed.), Optimization; Symposium of the Institute of Mathematics and Its Applications, University of Keele, England, 1968, Academic Press, London, New York, 1969, pp. 283–298.
[41] M.R. Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. 4 (1969) 303–320.
[42] M. Bierlaire, Optimization: Principles and Algorithms, EPFL Press, CRC Press, Lausanne, Boca Raton, 2015.
[43] J.R. Martins, A. Ning, Engineering Design Optimization, Cambridge University Press, 2021.
[44] S. Boyd, N. Parikh, E. Chu, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, Now Publishers Inc., 2011.
[45] D.P. Bertsekas, Multiplier methods: a survey, Automatica 12 (1976) 133–145.
[46] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[47] A. Dener, M.A. Miller, R.M. Churchill, T. Munson, C.-S. Chang, Training neural networks under physical constraints using a stochastic augmented Lagrangian approach, arXiv preprint, arXiv:2009.07330, 2020.
[48] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (2021) B1105–B1132.
[49] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European Conference on Computer Vision, Springer, 2016, pp. 630–645.
[50] E. Weinan, A proposal on machine learning via dynamical systems, Commun. Math. Stat. 5 (2017) 1–11.
[51] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y.W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, PMLR, vol. 9, 2010, pp. 249–256.
[52] J.-H. Li, X.-Y. Hu, S.-H. Zeng, J.-G. Lu, G.-P. Huo, B. Han, R.-H. Peng, Three-dimensional forward calculation for loop source transient electromagnetic method based on electric field Helmholtz equation, Chin. J. Geophys. 56 (2013) 4256–4267.
[53] L. Greengard, J. Huang, V. Rokhlin, S. Wandzura, Accelerating fast multipole methods for the Helmholtz equation at low frequencies, IEEE Comput. Sci. Eng. 5 (1998) 32–38.

[54] R.-E. Plessix, A Helmholtz iterative solver for 3D seismic-imaging problems, Geophysics 72 (2007) SM185–SM194.
[55] A. Bayliss, C.I. Goldstein, E. Turkel, The numerical solution of the Helmholtz equation for wave propagation problems in underwater acoustics, Comput. Math. Appl. 11 (1985) 655–665.
[56] J. Nocedal, Updating quasi-Newton matrices with limited storage, Math. Comput. 35 (1980) 773–782.
[57] A.A. Javadi, M.M. Al-Najjar, B. Evans, Flow and contaminant transport model for unsaturated soil, in: Theoretical and Numerical Unsaturated Soil Mechanics, Springer, 2007, pp. 135–141.
[58] N. An, S. Hemmati, Y. Cui, Numerical analysis of soil volumetric water content and temperature variations in an embankment due to soil-atmosphere interaction, Comput. Geotech. 83 (2017) 40–51.
[59] V. Gadi, S. Singh, M. Singhariya, A. Garg, S. Sreedeep, K. Ravi, Modeling soil-plant-water interaction: effects of canopy and root parameters on soil suction and stability of green infrastructure, Eng. Comput. (2018).
[60] M. Hayashi, D.O. Rosenberry, Effects of ground water exchange on the hydrology and ecology of surface water, Ground Water 40 (2002) 309–316.
[61] D. Hillel, Environmental Soil Physics: Fundamentals, Applications, and Environmental Considerations, Elsevier, 1998.
[62] M.T. Van Genuchten, A closed-form equation for predicting the hydraulic conductivity of unsaturated soils, Soil Sci. Soc. Am. J. 44 (1980) 892–898.
[63] J.V. Beck, B. Blackwell, C.R.S. Clair Jr, Inverse Heat Conduction: Ill-posed Problems, James Beck, 1985.
[64] Y. Hon, T. Wei, A fundamental solution method for inverse heat conduction problem, Eng. Anal. Bound. Elem. 28 (2004) 489–495.
[65] S.-Y. Shen, A numerical study of inverse heat conduction problems, Comput. Math. Appl. 38 (1999) 173–188.
[66] J. Wang, N. Zabaras, A Bayesian inference approach to the inverse heat conduction problem, Int. J. Heat Mass Transf. 47 (2004) 3927–3941.
[67] K.R. Swanson, C. Bridge, J. Murray, E.C. Alvord Jr, Virtual and real brain tumors: using mathematical modeling to quantify glioma growth and invasion, J. Neurol. Sci. 216 (2003) 1–10.
[68] A. Giese, R. Bjerkvig, M. Berens, M. Westphal, Cost of migration: invasion of malignant gliomas and implications for treatment, J. Clin. Oncol. 21 (2003) 1624–1636.
[69] R. Jaroudi, Inverse Mathematical Models for Brain Tumour Growth, vol. 1787, Linköping University Electronic Press, 2018.
[70] S. Jbabdi, E. Mandonnet, H. Duffau, L. Capelle, K.R. Swanson, M. Pélégrini-Issac, R. Guillevin, H. Benali, Simulation of anisotropic growth of low-grade gliomas using diffusion tensor imaging, Magn. Reson. Med. 54 (2005) 616–624.
[71] K. Painter, T. Hillen, Mathematical modelling of glioma growth: the use of diffusion tensor imaging (dti) data to predict the anisotropic pathways of cancer invasion, J. Theor. Biol. 323 (2013) 25–39.
[72] E. Özuğurlu, A note on the numerical approach for the reaction–diffusion problem to model the density of the tumor growth dynamics, Comput. Math. Appl. 69 (2015) 1504–1517.
[73] R. Rockne, E. Alvord, J. Rockhill, K. Swanson, A mathematical model for brain tumor response to radiation therapy, J. Math. Biol. 58 (2009) 561–578.
[74] J. Larsson, Solving the Fisher Equation to Capture Tumour Behaviour for Patients with Low Grade Glioma, Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2019.
[75] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), Proceedings of the 3rd International Conference on Learning Representations, 2015.