Newton-PINet: A fast physics-informed neural network with Newton linearization for meta-learning nonlinear PDEs

Yuchen Fan[1,*] **Chang Wei**[1] **Pao-Hsiung Chiu**[2] **Chin Chun Ooi**[2]
Heyang Wang[1] **Jian Cheng Wong**[2,*]
[1]School of Mechanical Engineering, Tianjin University, China
[2]Institute of Advanced Intelligence and Computing, A*STAR, Singapore

Abstract

Scientific machine learning has opened new avenues for solving parameterized partial differential equations (PDEs), enabling models to learn a family of PDEs and generalize to unseen instances. In this context, data-driven operator learning methods typically require large training datasets, while physics-informed neural networks (PINNs) suffer from difficult optimization and limited generalization, especially for nonlinear PDEs. We propose Newton-PINet, a physics-informed network enhanced by Newton linearization, offering an effective meta-learning framework for nonlinear PDEs. Newton-PINet (i) employs a physics-informed multilayer network with skip connections, where the output-layer weights are solved by least squares; (ii) adopts a two-stage learning strategy that first leverages gradient-based training to learn robust representations from the available training tasks, and then performs gradient-free fine-tuning on the output layer for fast task-specific generalization; and (iii) incorporates a Newton linearization method to speed up the least-squares iteration for nonlinear PDE problems. On a challenging nonlinear reaction-diffusion benchmark, Newton-PINet achieves up to three orders of magnitude lower relative error than recent neural solvers, while using 16× fewer training tasks and over an order of magnitude less training time (about 5 minutes versus several hours). This work advances the meta-learning of PINNs toward data-efficient, fast, and generalizable physics solvers. The code and datasets are publicly available at: `https://github.com/fycAIwj/Newton-PINet`.

1 Introduction and related works

Solving partial differential equations (PDEs) is fundamental across diverse fields, including fluid dynamics, climate modeling, and biophysics [Karniadakis et al., 2021, Hagnberger et al., 2025]. Traditional numerical solvers for PDEs often incur prohibitive computational costs, particularly when repeated evaluations are required in applications. Recently, scientific machine learning has emerged as a powerful paradigm, leveraging advances in deep neural networks to deliver fast and accurate approximations of PDE solutions, thereby enabling high-fidelity simulations and real-time predictive capabilities that were previously unattainable [Cai et al., 2021, Cuomo et al., 2022, Hu et al., 2026]. Deep learning for PDEs can be broadly categorized into: data-driven operator learning, physics-informed deep learning, and meta-learning physics-informed neural networks (PINNs).

Data-driven operator learning: Data-driven operator learning focuses on training neural networks to directly map input functions, such as PDE parameters or initial conditions, to their corresponding PDE solutions. Well-known examples of this approach include DeepONet [Lu et al., 2021], Derivative-

*Corresponding authors: fanyuchen@tju.edu.cn (Yuchen Fan); wongj@a-star.edu.sg (Jian Cheng Wong)

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

enhanced-DeepONet [Qiu et al., 2024], Transformer-DeepONet [Wei et al., 2025b], Fourier neural operators (FNO) [Li et al., 2020], Factorized-FNO [Tran et al., 2023], Decomposed-FNO [Li and Ye, 2025], and Tucker-FNO [Zhou et al., 2026]. Once trained, these models can generalize to new parameter settings (e.g., new initial conditions), enabling fast prediction of complex dynamical behaviors. However, they lack explicit physics constraints and interpretability, and their generalization depends on adequacy of labeled training data, which are often expensive to obtain [Li et al., 2024]. For example, Transformer-DeepONet typically requires at least 1,000 different PDE solutions for adequate performance [Wei et al., 2025b].

Physics-informed deep learning: An alternative paradigm, physics-informed deep learning, incorporates governing PDEs and boundary conditions into the training loss function [Wong et al., 2022, Xu et al., 2025, Wei et al., 2025a]. By penalizing violation of physical laws, these models can be trained effectively even in data-sparse regimes, often requiring only initial or boundary conditions. Representative approaches include physics-informed variants of operator learning, such as physics-informed DeepONet [Wang et al., 2021], physics-informed neural operators (PINO) [Li et al., 2024], and HyPINO [Bischof et al., 2025]. Another prominent method is PINN [Raissi et al., 2019], which learns to map the spatiotemporal coordinates to the PDE solution for a given system. While these methods alleviate the need for large labeled datasets, their reliance on physics-informed objectives often results in prohibitively long training time, due to highly nonconvex loss landscapes caused by the stiff PDE constraints [Chiu et al., 2022, Wang et al., 2023, Datar et al., 2024], especially for nonlinear PDEs. For example, training physics-informed DeepONet on a family of Burgers' equations can take more than 9 hours [Wang et al., 2021], while PINNs may require over 24 hours to solve a single Kuramoto-Sivashinsky equation [Wang et al., 2025].

Meta-learning PINNs: Adapting PINN to new PDEs or parameter settings typically requires costly retraining. Transfer learning strategies accelerate convergence by reusing knowledge from related tasks [Wong et al., 2021, Wang et al., 2022]. Parameterized PINNs (P^2INNs) further incorporate problem parameters as network inputs, enabling a single model to represent families of PDEs without labeled data while preserving physical consistency [Cho et al., 2024]. However, P^2INNs still suffer from the optimization challenges of physics-based loss functions, and their adaptation to new tasks remains heavily reliant on gradient-based fine-tuning which is often slow to converge, especially for nonlinear PDEs. Meta-learning methods aim to enable rapid adaptation to unseen PDEs with limited data by learning parameter initializations or update rules [Penwarden et al., 2023, Wong et al., 2026a]. MAML-style approaches learn transferable parameter priors [Liu et al., 2022, Wong et al., 2026b], while other methods leverage latent codes or meta-optimizers for task-specific adaptation [Huang et al., 2022, Cho et al., 2023]. To avoid gradient-based adaptation, Baldwinian-PINN introduces a gradient-free meta-learning strategy based on neuroevolution, evolving shared representations and fine-tuning the output layer via least squares [Wong et al., 2026c]. While highly efficient for linear PDEs, its single-layer architecture limits nonlinear expressiveness. Moreover, its Picard (lagging-of-coefficients) iteration exhibits only linear convergence, leading to slow solution refinement for nonlinear problems [Chiu et al., 2008, Pletcher et al., 2012].

See Appendix A for an extended literature review. In summary, existing deep learning approaches for solving nonlinear PDEs remain constrained by substantial data requirements and/or slow, optimization-heavy adaptation. For this reason, we propose a novel meta-learning PINN framework (Figure 1), termed **Newton-PINet**. The main contributions are as follows:

- We introduce a physics-informed multilayer network with skip connections from early hidden layers to the output, where the final-layer weights are computed using least-squares approach (Tikhonov regularization). This skip-connected Tikhonov regularization PINN architecture improves the model's nonlinear representation capabilities.
- We employ a two-stage learning strategy, where the first stage uses a gradient-based method to meta-learn the nonlinear hidden layers' network weights and essential hyperparameters for task-specific adaptation in an unsupervised or few-shot manner. At inference (test time), task-specific adaptation is confined to the output layer, which can be rapidly updated in a gradient-free manner using Tikhonov regularization.
- We integrate Newton linearization into the Tikhonov-regularized PINN and establish its equivalence to the classical regularized Gauss-Newton method through mathematical derivation, with experiments validating its near-quadratic convergence. The Newton linearization accelerates nonlinear least-squares convergence, providing a more efficient alternative to the

traditional Picard approach during both meta-learning and inference. **We therefore refer to the proposed physics-informed network with Newton linearization as Newton-PINet.**

Results on diverse nonlinear PDE benchmarks show that, compared with recent neural solvers, Newton-PINet significantly reduces the reliance on training data while achieving fast and accurate generalization to new PDE tasks through physics-consistent fine-tuning.

2 Preliminaries

For simplicity, consider a nonlinear PDE with spatial variable x, time t, and solution u defined on $\Omega \times [0, T]$:

$$\text{PDE:} \quad \mathcal{N}_\theta[u(x,t)] = q(x,t) \quad x \in \Omega,\ t \in [0,T], \tag{1a}$$

$$\text{IC:} \quad u(x, t=0) = u_0(x) \quad x \in \Omega, \tag{1b}$$

$$\text{BC:} \quad \mathcal{B}[u(x,t)] = g(x,t) \quad x \in \partial\Omega,\ t \in [0,T]. \tag{1c}$$

where the general differential operator $\mathcal{N}_\theta$ can include both linear and nonlinear combinations of the temporal and spatial derivatives and PDE parameters θ, $q(x,t)$ is the source term, $u_0(x)$ is the initial condition (IC), and $g(x,t)$ is the boundary condition (BC).

Single PDE problem: Standard PINNs aim to solve a single PDE instance (**a single task**) defined by specific PDE parameters, IC, and BC. PINNs are trained by minimizing the discrepancy between Eq. (1) and the model's prediction.

Towards generalizable PINNs: There is growing interest in models capable of generalizing across a set of tasks belonging to some underlying task-distribution $p(\mathcal{T})$, e.g., a family of PDEs spanning different PDE parameters, ICs, and BCs. For meta-learning PINNs, the goal is to learn network initializations using training tasks sampled from $p(\mathcal{T})$ that enable fast, accurate, and physics-aware predictions on unseen scenarios, i.e., any new task from the distribution $\mathcal{T}_i \sim p(\mathcal{T})$ through fine-tuning [Wong et al., 2026a].

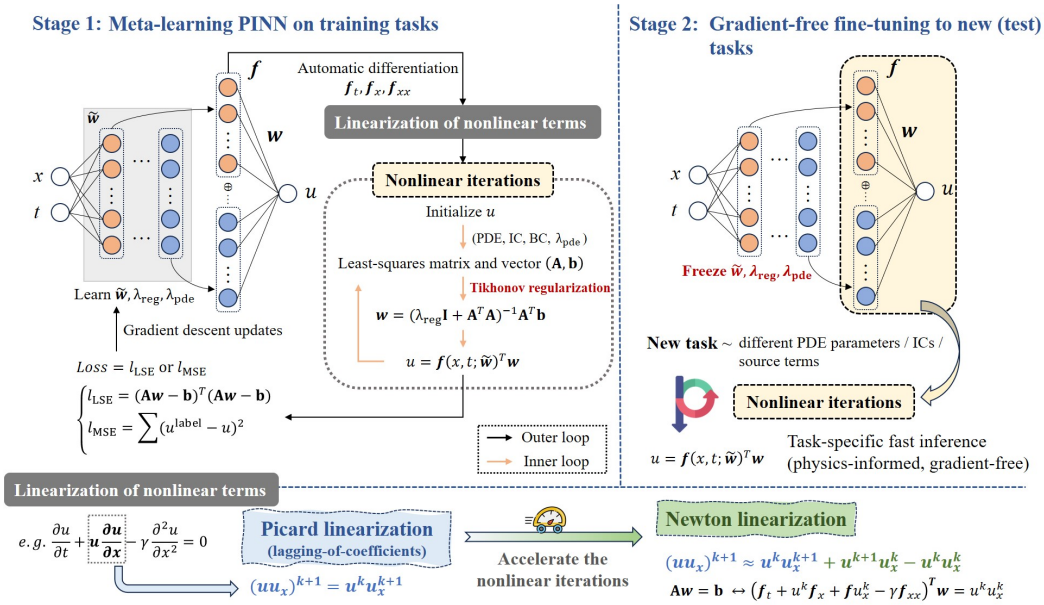


Figure 1: Newton-PINet model framework. Stage 1 (pre-training) meta-learns the task-agnostic parameters $\Theta = \{\tilde{\boldsymbol{w}}, \lambda_{\text{pde}}, \lambda_{\text{reg}}\}$ by gradient descent (outer loop), while the output-layer weights $\boldsymbol{w}$ of each training task are updated by gradient-free Tikhonov-regularized least squares (inner loop); Stage 2 (inference) freezes Θ and adapts only $\boldsymbol{w}$ to new tasks. The bottom panel compares Picard and Newton linearization; the latter accelerates the convergence of nonlinear iterations for solving nonlinear PDEs.

3 Methodology

Skip-connected Tikhonov regularization PINN: We introduce skip connections from all hidden layers to the output layer, with the output-layer weights computed using Tikhonov regularization [Golub et al., 1999]. The skip-connected neural architecture improves expressivity, ensuring stable and accurate least-squares computation by increasing the output-layer width through stacking additional hidden layers, while still maintaining a moderate number of nodes per layer. As shown in Figure 1, we employ the neural networks with $L+1$ layers, where the input is $\mathbf{x} = (x, t)$ (layer 0) and the output is u (layer L). Each hidden layer $l = 1, \ldots, L-1$ contains the same number of neurons (N_n). Sinusoidal feature embeddings are applied at the first layer [Wong et al., 2022], and all hidden layers use $\sin(\cdot)$ activation functions. The pre-final output $\boldsymbol{f}$ is constructed as the concatenation of all hidden activations (skip connections). All trainable network parameters up to the pre-final layer are denoted $\tilde{\boldsymbol{w}}$. The output u can be computed by $u(\mathbf{x}) = \boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \boldsymbol{w}$, where $\boldsymbol{w}$ is the output-layer weights. See Appendix B.1 for a detailed description of the skip-connected model architecture.

The objective of task-specific learning is to determine the best set of $\boldsymbol{w}$ such that $u(\mathbf{x}) = \boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \boldsymbol{w}$ satisfies the PDE, IC, and BC defined in Eq. (1) for a task $\mathcal{T}_i \sim p(\mathcal{T})$. This can be formulated as a physics-based least-squares problem. Given collocation points for PDE residual, points at the initial time, and points on the boundary, together with the loss importance hyperparameter λ_{pde}, the following system is obtained:

$$\begin{bmatrix} \cdots & \lambda_{\text{pde}} \cdot \mathcal{N}_\theta[f_j(x_1^{\text{pde}}, t_1^{\text{pde}}; \tilde{\boldsymbol{w}})] & \cdots \\ & \vdots & \\ \cdots & f_j^i(x_1^{\text{ic}}, 0; \tilde{\boldsymbol{w}}) & \cdots \\ & \vdots & \\ \cdots & \mathcal{B}[f_j(x_1^{\text{bc}}, t_1^{\text{bc}}; \tilde{\boldsymbol{w}})] & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ w_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda_{\text{pde}} \cdot q(x_1^{\text{pde}}, t_1^{\text{pde}}) \\ \vdots \\ u_0(x_1^{\text{ic}}) \\ \vdots \\ g(x_1^{\text{bc}}, t_1^{\text{bc}}) \\ \vdots \end{bmatrix}, \tag{2}$$

which can be compactly written as

$$\mathbf{A}\boldsymbol{w} = \mathbf{b}.$$

Here $\mathbf{A}\boldsymbol{w}$ represents the left-hand side of Eq. (1) at a given set of collocation points, and $\mathbf{b}$ denotes the corresponding right-hand side of Eq. (1). The number of rows in $\mathbf{A}$ and $\mathbf{b}$ equals the total number of collocation points. The number of columns in $\mathbf{A}$ corresponds to the product of the network's pre-final feature dimension and the output dimension. The final output weights $\boldsymbol{w}$ are obtained via Tikhonov regularization: $\arg\min_{\boldsymbol{w}} \ \|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|_2^2 + \lambda_{\text{reg}} \boldsymbol{w}^T \boldsymbol{w}$. λ_{reg} is the Tikhonov regularization parameter whose proper setting improves the conditioning of the problem. The system has a closed-form solution, either $\boldsymbol{w} = (\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ for over-determined system or $\mathbf{A}^T(\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$ for under-determined one, enabling fast, gradient-free updates. See Appendix B.2 for the details.

Note that the Tikhonov-regularized formulation provides a closed-form result in a single computation step for any linear PDE. For nonlinear PDEs, the nonlinear terms must be linearized so that the system can be cast into a linear form. An initial guess is then required, followed by a few **nonlinear iterations** (a process that can also be interpreted as gradient-free fine-tuning) to update $\boldsymbol{w}$ toward the optimal solution. Hereafter, we use N to denote the number of nonlinear iterations.

The proposed Newton-PINet model framework consists of two stages (Figure 1): (i) meta-learning on training tasks and (ii) gradient-free fine-tuning on new tasks.

Meta-learning on training tasks: The meta-learning objective is to optimize the network weights $\tilde{\boldsymbol{w}}$, importance hyperparameter λ_{pde} of PDE loss relative to IC/BC, and the regularization parameter λ_{reg}, collectively denoted Θ, to learn task-agnostic representations spanning a family of PDEs and enable fast generalization to unseen tasks via fine-tuning only the output-layer weights using the Tikhonov regularization update. In our framework, the outer loop of meta-learning updates the learnable parameters via gradient descent with learning rate η, while task-specific adaptation in the inner loop, or generalization, is performed through gradient-free Tikhonov regularization applied to the output-layer weights. The meta-learning objective (outer-loop loss) can be either the physics-based least-squares error $l_{\text{LSE}}(\boldsymbol{w}^*) = \|\mathbf{A}\boldsymbol{w}^* - \mathbf{b}\|_2^2$, or the data-driven mean squared

error $l_{\text{MSE}}(\boldsymbol{w}^*) = \frac{1}{n}\sum_s (u_s^{\text{label}} - \boldsymbol{f}(\mathbf{x};\tilde{\boldsymbol{w}})^T \boldsymbol{w}^*)^2$ given labeled data $\{u_s^{\text{label}}\}_{s=1}^n$ where n is the total number of collocation points (PDE residual, IC, and BC). Here, Tikhonov regularization is used to compute the optimal task-specific output-layer weights $\boldsymbol{w}^*$, enabling the model to specialize to any realization of the task. l_{LSE}-based learning is termed **unsupervised learning**. l_{MSE}-based learning is termed **hybrid learning**, since it couples physics-based Tikhonov updates in the inner loop with data-driven minimization in the outer loop. See Algorithm 1 for the pseudocode and Appendix B.3 for a detailed mathematical description.

Gradient-free fine-tuning to new tasks: After meta-learning, Θ is fixed. For a new task $\mathcal{T}_i \sim p(\mathcal{T})$ with different PDE parameters or IC/BC, Tikhonov regularization is used to compute the task-specific output-layer weights, enabling fast gradient-free adaptation to the new PDE instance. Since this update is independent of gradient-based backpropagation optimizers such as stochastic gradient descent (SGD), the resulting adaptation is extremely fast while remaining physics-compliant. See Algorithm 2 for the pseudocode.

Algorithm 1 : Meta-learning on training tasks

Require: Initialize Θ; $p(\mathcal{T})$; η
Require: $\boldsymbol{\mathcal{C}}$, constructor of least-squares system $(\mathbf{A}, \mathbf{b})$; see Eq. (2)
while not done **do**
  Sample training tasks $\{\mathcal{T}_i\} \sim p(\mathcal{T})$
  for each task $\mathcal{T}_i$ **do**
    Get initial guess $u^{(1)}$
    for $k = 1$ to N **do**
      $(\mathbf{A}, \mathbf{b}) = \boldsymbol{\mathcal{C}}(\mathcal{T}_i, \Theta, u^k)$
      $\boldsymbol{w} = \begin{cases} (\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, & \text{or} \\ \mathbf{A}^T(\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \end{cases}$
      $u^{(k+1)} = \boldsymbol{f}^T\boldsymbol{w}$
    end for
    Compute loss $l_{\mathcal{T}_i}$
  end for
  $\Theta \leftarrow \Theta - \eta\nabla_\Theta \sum_i l_{\mathcal{T}_i}$
end while

Algorithm 2 : Gradient-free fine-tuning to new tasks

Require: Freeze Θ
for each test task $\mathcal{T}_i$ **do**
  Get initial guess $u^{(1)}$
  for $k = 1$ to N **do**
    $(\mathbf{A}, \mathbf{b}) = \boldsymbol{\mathcal{C}}(\mathcal{T}_i, \Theta, u^k)$
    $\boldsymbol{w} = \begin{cases} (\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, & \text{or} \\ \mathbf{A}^T(\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \end{cases}$
    $u^{(k+1)} = \boldsymbol{f}^T\boldsymbol{w}$
  end for
  Store solution $u_{\mathcal{T}_i} = u^{(\text{N}+1)}$
end for
return all task solution $\{u_{\mathcal{T}_i}\}$

Newton linearization: Note that the Tikhonov regularization applies to linear PDEs, where a linear matrix system can be constructed and solved in a single step. For nonlinear PDEs, the nonlinear terms must first be linearized so that the system can be cast into a linear form. The previous approaches typically relied on Picard method (lagging-of-coefficients), which linearizes the nonlinear terms using the solution from the previous iteration and then performs least-squares solves for multiple nonlinear iterations. Although simple, it is only linearly convergent [Dennis Jr and Schnabel, 1996]. To address this limitation, we use the Newton linearization approach to approximate the nonlinear term by Taylor expansion around the current iterate. For a function F of one state variable u, this gives $F(u^{k+1}) = F(u^k) + \frac{dF}{du}\Big|^k (u^{k+1} - u^k) + \text{H.O.T}$, where k is the nonlinear iteration step, and H.O.T denotes the higher-order truncated terms of Taylor expansion. For a function of two variables u and v, the expansion becomes $F(u^{k+1}, v^{k+1}) = F(u^k, v^k) + \frac{\partial F}{\partial u}\Big|^k (u^{k+1} - u^k) + \frac{\partial F}{\partial v}\Big|^k (v^{k+1} - v^k) + \text{H.O.T}$. Based on this principle, the nonlinear terms commonly arising in PDEs can be expressed in a Newton-linearized form amenable to Tikhonov regularization. Table 1 summarizes several representative nonlinear terms derived by our work.

A detailed mathematical derivation is provided in Appendix B.5. In Appendix C, we further establish that the Newton linearization used in the Tikhonov-regularized PINN is essentially equivalent to the classical regularized Gauss-Newton method, with experiments validating its near-quadratic convergence.

Table 1: Newton-linearized expressions of several nonlinear terms derived by our work. m denotes the exponent. k and $k+1$ refer to the current and next nonlinear iterations, respectively.

Nonlinear term	Newton-linearized expression
$(u^m)^{k+1}$	$m(u^k)^{m-1}u^{k+1} + (1-m)(u^k)^m$
$(u^m u_x)^{k+1}$	$m(u^k)^{m-1}u_x^k u^{k+1} + (u^k)^m u_x^{k+1} - m(u^k)^m u_x^k$
$[\sinh(u)]^{k+1}$	$\cosh(u^k)u^{k+1} + \sinh(u^k) - \cosh(u^k)u^k$
$[\exp(u)]^{k+1}$	$\exp(u^k)u^{k+1} + \exp(u^k) - \exp(u^k)u^k$
$[u\ln(u)]^{k+1}$	$(\ln(u^k)+1)u^{k+1} + u^k\ln(u^k) - (\ln(u^k)+1)u^k$

4 Experiment results

We compare the performance of the proposed **Newton-PINet** (Newton linearization), **PINet** (Picard linearization) and purely data-driven deep neural network (DNN) on several representative classes of nonlinear PDEs:

(i) Nonlinear convection-type PDEs: $\frac{\partial u}{\partial t} + \beta u^m \frac{\partial u}{\partial x} - \gamma \frac{\partial^2 u}{\partial x^2} + \delta \frac{\partial^3 u}{\partial x^3} + \sigma \frac{\partial^4 u}{\partial x^4} = 0$,

where $u(x,t)$ is the state variable, $\beta, \gamma, \delta, \sigma$ are PDE parameters, and m denotes the nonlinearity order. We test three 1D time-dependent problems: Burgers, generalized Korteweg-de Vries (KdV), and Kuramoto-Sivashinsky (K-S) equations. In addition, we consider a 2D lid-driven cavity (LDC) flow governed by the Navier-Stokes equations. These problems span nonlinear convection, high-order dispersion/dissipation, and viscous flow with pressure-velocity coupling, providing canonical benchmarks for assessing generalization across different nonlinearities and dimensions.

(ii) Nonlinear forcing-type PDEs: $\frac{\partial u}{\partial t} + \alpha \nabla u - \gamma \Delta u + R(u) = q$,

where $R(u)$ denotes a nonlinear reaction operator, which may take polynomial, hyperbolic, exponential, or logarithmic forms, and q represents an external source term. We consider four 1D time-dependent problems: convection-diffusion-reaction (CDR), Klein-Gordon (K-G), hyperbolic heat, and logarithmic heat equations; two 2D problems: Helmholtz and parametric diffusion-reaction equations; and one 3D problem: parametric diffusion-reaction equation. These equations represent systems where diffusion and wave propagation interact with nonlinear reaction or source terms, making them central to the modeling of chemical kinetics, heat transfer, quantum fields, and complex reaction-diffusion phenomena in physics and engineering.

See Appendix D.1 for detailed descriptions of all PDE problems and meta-learning configurations (summarized in Table 8). See Appendix D.2 for details on data generation, error metrics, and computational setup. Note that, unless otherwise specified, the meta-learning outer-loop loss function for the Newton-PINet is l_{MSE} (hybrid learning). We also adopt a temporal domain decomposition strategy within our meta-learning framework to leverage temporal causality [Wang et al., 2022] as a means of improving accuracy for complex time-dependent PDEs. Each time block is trained to model only short-term dynamics; however, iterative composition across blocks during inference ensures seamless integration over the full temporal horizon. Table 2 below summarizes the performance of the models on the nonlinear PDEs considered in this study.

4.1 Learning to solve PDEs with nonlinear convection term

Burgers' problems: $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \gamma \frac{\partial^2 u}{\partial x^2} = 0$. We first fix the initial condition as $u(x,0) = -\sin(\pi x)$, and vary the viscosity parameter γ in the range $[0.001, 0.05]$ with an increment of 0.001 to generate 50 tasks. Among them, 16 are randomly selected for training, and the remaining 34 are used as test tasks. Under each condition, we keep the number of nonlinear iterations (N) for inner-loop fine-tuning in meta-learning stage consistent with those in generalization stage. We then compare Newton-PINet and PINet across different conditions, with the number of nonlinear iterations (N) ranging from 4 to 12. As shown in Figure 2(a), with an increase of N, Newton-PINet achieves significantly faster convergence during meta-learning compared to PINet. Meanwhile, Newton-PINet achieves lower test errors than PINet. For example, when $\mathrm{N} = 6$, the average MSE across all test tasks of PINet is 1.10×10^{-4}, whereas that of Newton-PINet is 1.79×10^{-6} (see Figure 2(b) for the prediction

Table 2: Summary of model performance on nonlinear PDEs in this study. The reported meta-learning and fine-tuning time correspond to the Newton-PINet. "NaN" entries under PINet indicate numerical overflow during computation. See Appendix D.1 for detailed descriptions of all PDE problems and meta-learning configurations (summarized in Table 8). See Appendix D.2 for details on data generation, error metrics, and computational setup.

Problem	Meta-learning epoch	Meta-learning time cost	Inference time per test task	Test MSE of models		
				DNN	PINet	Newton-PINet
Burgers'	1000	286s	0.035s	1.06e-4	1.10e-4	1.79e-6
Generalized KdV	1000	453s	0.359s	3.74e-2	1.49e-2	1.94e-3
Kuramoto-Sivashinsky	3000	1014s	0.552s	5.03e-1	106	3.45e-2
Convection-diffusion-reaction	1000	203s	0.099s	5.27e-1	4.65e-3	4.32e-5
Klein-Gordon	1000	105s	0.065s	1.04	1.88e-7	2.36e-7
Hyperbolic heat	2000	393s	0.069s	1.06e-1	NaN	1.50e-6
Logarithmic heat	2000	395s	0.067s	2.99e-1	NaN	2.16e-6
Helmholtz	2000	344s	0.067s	5.98	NaN	1.33e-3
Parametric diffusion-reaction (2D)	2000	342s	0.047s	3.43e-1	1.18e-8	1.60e-10
Lid-driven cavity	1000	290s	0.40s	1.42e-1	1.92e-4	1.67e-4
Parametric diffusion-reaction (3D)	2000	1907s	0.104s	7.82e-1	4.58e-5	1.72e-6

plots). These results demonstrate that Newton linearization significantly improves the convergence of meta-learning, which in turn enhances generalization accuracy on test tasks.

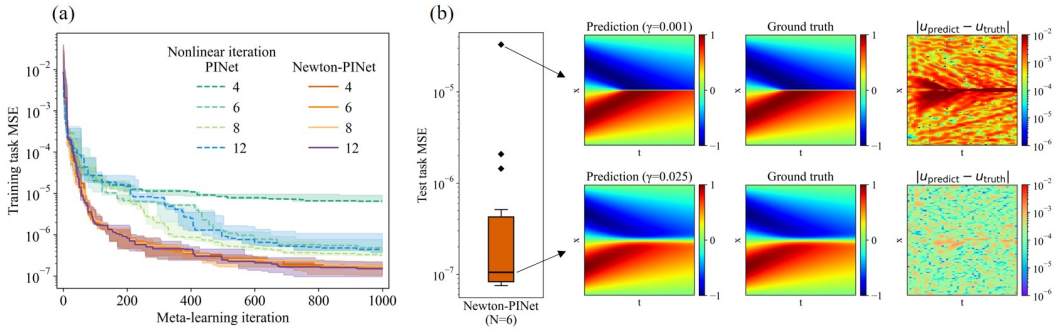


Figure 2: Burgers' problem. (a) Meta-learning convergence of Newton-PINet vs. PINet under different nonlinear iterations (N) (lines and shaded areas: the median convergence path and interquartile ranges of 5 random-seed runs). (b) Boxplot of error distributions across all test tasks and predictions by Newton-PINet (N=6).

Model comparison under varying initial conditions: We evaluate the model under the setting $\gamma = 0.01$ in Burgers' problem, where the initial conditions are sampled from a Gaussian random field $\mathcal{N}(0, 625(-\Delta + 25I)^{-4})$ to construct both training and test tasks. We compare (i) **unsupervised Newton-PINet** (1 time block) and (ii) **hybrid Newton-PINet** (4 time blocks) against popular neural operator baselines in terms of training time and test error. The **unsupervised** operator considered is Physics-Only DeepONet (PO-DeepONet) [Wang et al., 2021]. The **supervised** operators include DeepONet [Lu et al., 2021] and Transformer-DeepONet with Trunk net enhanced by Fourier coefficients (T-DeepONet-TF) [Wei et al., 2025b], while the **hybrid** operator is PINO (a physics-

informed FNO) [Li et al., 2024]. Note that PINO uses a training loss that combines both data-driven and PDE-based physics losses. The results are summarized in Table 3. Compared to PO-DeepONet, unsupervised Newton-PINet reduces training time from 9.25h to 291s while improving test accuracy (in a data-absent scenario). Compared to supervised and hybrid neural operators, hybrid Newton-PINet consistently achieves superior test accuracy, with significantly reduced training cost. More results under different γ configurations are reported in Appendix Table 12.

Table 3: Comparison of training cost and performance at $\gamma = 0.01$ on Burgers' problem.

	Unsupervised		Supervised		Hybrid	
	PO-DeepONet	**Newton-PINet**	DeepONet	T-DeepONet-TF	PINO	**Newton-PINet**
No. training tasks	1000	50	1000	1000	1000	50
Test relative L^2 error	1.38e-2	**3.51e-3**	1.17e-2	2.08e-3	3.80e-3	**5.46e-4**
Training time	9.25h	**291s**	2800s	3041s	1200s	**133s**

Generalized KdV, K-S, and LDC problems: We further evaluate Newton-PINet on more complex nonlinear PDEs, including the generalized KdV equations with higher-order nonlinear terms $u\frac{\partial u}{\partial x}$, $u^2\frac{\partial u}{\partial x}$, $u^3\frac{\partial u}{\partial x}$, the K-S equations with nonlinear term $u\frac{\partial u}{\partial x}$, and the LDC problem governed by the Navier-Stokes equations. These PDEs pose highly challenging benchmarks for PINNs. The state-of-the-art PirateNets+SOAP [Wang et al., 2025] requires over 24 hours of training to solve a single K-S problem and must be retrained for each new problem. In contrast, our **Newton-PINet** completes the meta-learning stage in *under half an hour*, while single-task adaptation takes *less than one second*, demonstrating a clear advantage in computational efficiency. The results show that Newton-PINet achieves test errors of $\mathrm{MSE} = 1.94 \times 10^{-3}$ on the generalized KdV, $\mathrm{MSE} = 3.45 \times 10^{-2}$ on the K-S, and $\mathrm{MSE} = 1.67 \times 10^{-4}$ on the LDC problem, outperforming PINet and DNN under the same configuration. Figures 5, 6, 7 show the prediction results of these problems. This demonstrates that our Newton linearization can improve the model performance across a variety of nonlinear convective PDEs. Note that the regularization parameter (λ_{reg}) is meta-learned rather than manually tuned, and is shown to be robust across diverse test tasks (Appendix F.1). The initial guess (u_{guess}) may affect convergence and Appendix F.2 provides simple strategies to mitigate this sensitivity. In addition, since the Tikhonov solution depends on $\mathbf{A}$, Appendix F.3 analyzes its impact on inference time, showing that larger $\mathbf{A}$ increases the computational cost.

4.2 Learning to solve PDEs with nonlinear forcing term

Convection-diffusion-reaction (CDR) problem: $\frac{\partial u}{\partial t} + \alpha\frac{\partial u}{\partial x} - \gamma\frac{\partial^2 u}{\partial x^2} - \rho_1 u + \rho_2 u^2 + \rho_3 u^3 = 0$. We generate 40 training and 160 test tasks by varying PDE parameters and initial conditions: $\alpha = 1$, $\gamma \in \{0.005, 0.01, 0.05\}$, $\rho_1, \rho_2 \in \{0, 1, 3, 5\}$, $\rho_3 = 5$, and $u(x, t = 0) = \sum_{j=1}^{J} A_j \sin(l_j x + \phi_j)$, where $J = 5$, $A_j \in [0.1, 0.5]$, $l_j \in \{1, 2, 3, 4\}$, and $\phi_j \in [-\pi, \pi]$. We adopt a 4-time-block decomposition during meta-training and inference to improve accuracy. Figure 3 (a) and (b) show the meta-learning process and test results. It can be seen that PINet converges well during meta-learning, and its test error decreases as nonlinear iteration (N) increases. In contrast, Newton-PINet converges much faster and achieves lower test error. As shown in Figure 3 (c) and (d), for a comparable error level, Newton-PINet ($\mathrm{N} = 4$) requires less than half the meta-training and fine-tuning time of PINet ($\mathrm{N} = 12$). These results indicate that Newton linearization can substantially accelerate nonlinear iterations without significantly introducing additional time cost in either meta-learning or inference. More prediction results under various test conditions are provided in Appendix Figure 8. Note that different time-blocking strategies can affect model inference performance, as discussed in Appendix F.4. As shown in Figure 3, the predictions exhibit drift at temporal block boundaries. We can mitigate this drift by increasing the initial condition weight, as detailed in Appendix F.5.

Additional nonlinear forcing-type PDEs: We evaluate Newton-PINet on several other challenging nonlinear forcing-type PDEs: Klein-Gordon, hyperbolic/logarithmic heat, Helmholtz, and parametric diffusion-reaction equations. The results show that Newton-PINet outperforms the DNN and PINet baselines on nearly all problems. Notably, for problems with hyperbolic or logarithmic nonlinearities, the Picard approach used by PINet often fails to converge or leads to unstable training, whereas Newton-PINet remains stable and accurate. Additional results for these problems are provided in

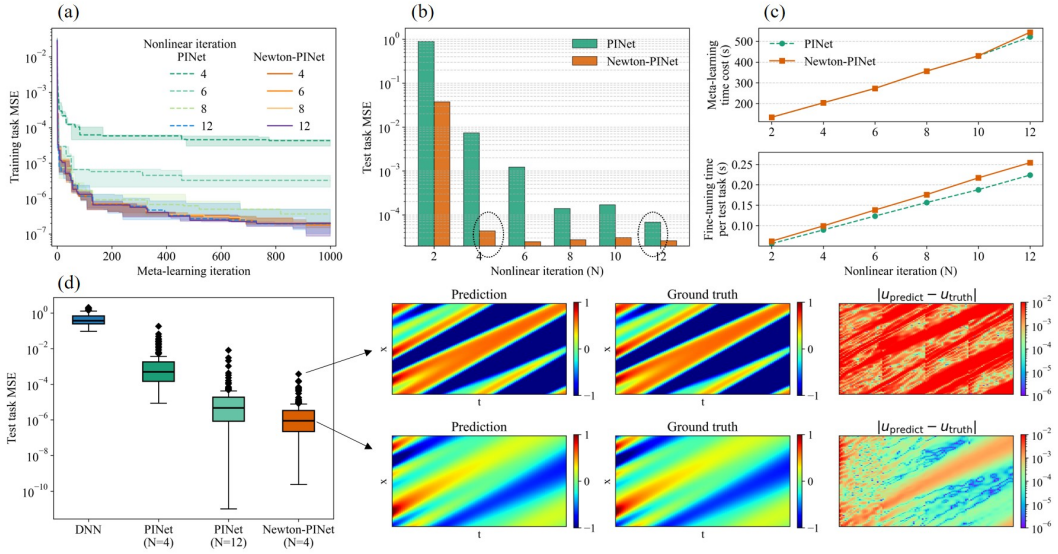


Figure 3: Convection-diffusion-reaction (CDR) problem. (a) Meta-learning convergence of Newton-PINet vs. PINet under different nonlinear iterations (N). (b,c) Test task MSE and runtime aggregated from 5 runs. In (b), the two circled regions correspond to Newton-PINet (N=4) and PINet (N=12), respectively. (d) Error distribution boxplot across all test tasks for different models; the prediction fields correspond to Newton-PINet (N=4).

Appendix D.3 and Figure 9. We further demonstrate the model's applicability to different boundary condition types in Appendix F.6.

Model generalization to unseen boundary conditions and irregular geometries: We verify that the model, pretrained only on samples with *Neumann* boundary conditions, can effectively generalize to unseen tasks with *Dirichlet* and mixed boundary conditions (see Appendix D.3 and Table 9). Furthermore, the model, pretrained only on samples with regular geometry, can effectively generalize to test cases with complex and varying irregular geometries, as detailed in Appendix D.3, Tables 10 and 11, Figs. 11 and 12.

Model comparison on a nonlinear reaction-diffusion problem: We further evaluate our method on a challenging 1D nonlinear reaction-diffusion problem $\frac{\partial u}{\partial t} - \gamma \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0$ [Krishnapriyan et al., 2021]. In this benchmark, the initial condition is fixed, while the PDE parameters are sampled over $\gamma \in [1, 5]$ and $\rho \in [-5, 5]$. The benchmark setup and baseline results are taken from the physics-informed neural solver (PI-neural-solver) [Boudec et al., 2025], which is trained on 800 labeled tasks and evaluated on 200 unseen tasks. In contrast, our **Newton-PINet**, under a hybrid training scheme, uses only 50 labeled tasks, i.e., *16× fewer tasks*, and is evaluated on the same test set. For a fair comparison, both models are trained and evaluated on an NVIDIA A6000 GPU. As shown in Table 4, Newton-PINet achieves a test MSE that is approximately three orders of magnitude lower than that of the PI-neural-solver, while requiring only 301 seconds (5 minutes) of total meta-training wall-clock time, compared to several hours for the baseline. These results highlight the *high data efficiency and computational efficiency* of Newton-PINet. More detailed comparisons with additional baselines, as well as model performance under different GPU configurations, are provided in Appendix E.2 and E.3. Model prediction results are presented in Appendix Figure 13.

Comparison with other meta-learning PINN and neural operator methods: We further compare the performance of Newton-PINet with state-of-the-art gradient-based meta-learning PINNs [Penwarden et al., 2023] and gradient-free Baldwinian-PINN [Wong et al., 2026c]. Detailed results are provided in Appendix E.4 and Table 15. On a parametric diffusion-reaction problem, Newton-PINet achieves approximately a *179× improvement in generalization accuracy* while requiring *7000×*

Table 4: Model comparison on a 1D nonlinear reaction-diffusion problem.

Model	No. training tasks	Test relative MSE	Training time	Inference time
PI-neural-solver [Boudec et al., 2025]	800	2.91e-4	4h30m	0.284s
Newton-PINet (Ours)	50	1.72e-7	301s	0.080s

less fine-tuning time compared with gradient-based meta-learning PINNs. Moreover, relative to Baldwinian-PINN, Newton-PINet reduces test MSE by 2$\sim$5 orders of magnitude across benchmark problems and shortens the task-specific fine-tuning time by roughly one order of magnitude. These results highlight Newton-PINet's superior generalization performance and computational speed. We next evaluate the performance of our method and state-of-the-art neural operators on the newly introduced PDE benchmarks, including the generalized KdV, K-S, CDR, and LDC problems. The results show that Newton-PINet achieves higher generalization accuracy than neural operators under identical training configurations. Detailed results are provided in Appendix E.5 and Table 17. Furthermore, Appendix E.6 compares the inference time of Newton-PINet with the runtime of classical numerical solvers, demonstrating substantial speedups achieved by our model while maintaining comparable accuracy.

In Appendix F.7, we demonstrate that Newton-PINet benefits significantly from skip connections, which enhance robustness to depth, width, and mesh resolution, thereby reducing the need for extensive hyperparameter tuning. In Appendix G, we discuss Newton-PINet's scalability. Briefly, since the features are generated by a neural representation, each row of $\mathbf{A}$ is dense rather than sparse in the numerical sense. The computational cost primarily depends on the expressiveness of pre-final layer features. By meta-learning a compact yet informative representation, the least-squares solve remains efficient even for large collocation sets.

5 Limitations & Outlook

Current experiments focus on 1D+time, 2D, and 3D cases. For more complex 2D/3D+time or multi-physics systems, larger, denser, and more ill-conditioned $\mathbf{A}$ matrix, combined with GPU memory limitations, hinders efficient solution. Future work will focus on developing low-rank strategies and scalable solvers for efficiently solving dense neural feature-based least-squares problems, thereby enhancing overall model performance. Additionally, for long-time prediction of chaotic systems such as K-S, we currently require 15 temporal blocks for accurate inference. Enabling single-shot prediction or reducing dependence on temporal decomposition remains an open challenge.

6 Conclusions

Newton-PINet has been validated on 1D, 2D, and 3D PDEs, as well as across diverse boundary conditions and domain geometries, covering a broad range of challenging nonlinear PDE operators. Newton-PINet demonstrates superior computational efficiency compared with state-of-the-art neural operators and meta-learning PINN methods. The main strengths of our model can be summarized as follows. **(1) Computational efficiency**: task adaptation in both meta-learning and inference is performed via least-squares updates of only the output layer. This gradient-free fine-tuning is faster than typical methods such as SGD, requiring just one step for linear PDEs and a few iterations for nonlinear PDEs. The Newton linearization further accelerates nonlinear solves. **(2) Data efficiency**: because the least-squares fine-tuning for new-task generalization is physics-informed, the meta-learning stage of our model requires very few labeled training samples and can converge in only a few epochs. **(3) Accuracy**: Tikhonov regularization stabilizes the least-squares solve, reducing ill-conditioning and improving generalization accuracy. Altogether, Newton-PINet achieves high generalization accuracy while requiring an order of magnitude fewer training tasks than state-of-the-art baselines. Task-specific inference on new tasks is also orders of magnitude faster compared to gradient-based meta-learning PINNs. These advantages make Newton-PINet a practical and scalable framework for learning large families of nonlinear PDEs in few-shot and real-time scenarios.

Acknowledgments and Disclosure of Funding

This research was in part supported by the National Research Foundation, Singapore under its AI-for-Science (AI4S) Catalytic Grant (Award No.: NRF-AI4SCT_2025-0007), and in part supported by the National Research Foundation, Singapore through the AI Singapore Programme, under the project "AI-based urban cooling technology development" (Award No.: AISG3-TC-2024-014-SGKR). The authors declare no competing interests.

References

Rafael Bischof, Michal Piovarči, Michael A Kraus, Siddhartha Mishra, and Bernd Bickel. Hypino: Multi-physics neural operators via hyperpinns and the method of manufactured solutions. *arXiv preprint arXiv:2509.05117*, 2025.

Lise Le Boudec, Emmanuel De Bézenac, Louis Serrano, Ramon Daniel Regueiro-Espino, Yuan Yin, and Patrick Gallinari. Learning a neural solver for parametric pde to enhance physics-informed methods. 2025. URL `https://openreview.net/forum?id=jqVj8vCQsT`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Pao-Hsiung Chiu and Hee Joo Poh. Development of an improved divergence-free-condition compensated coupled framework to solve flow problems with time-varying geometries. *International Journal for Numerical Methods in Fluids*, 93(1):44–70, 2021.

Pao-Hsiung Chiu, Tony WH Sheu, and Reui-Kuo Lin. An effective explicit pressure gradient scheme implemented in the two-level non-staggered grids for incompressible navier–stokes equations. *Journal of Computational Physics*, 227(8):4018–4037, 2008.

Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:11219–11231, 2023.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized pdes. *arXiv preprint arXiv:2408.09446*, 2024.

Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Eric C Cyr, Mamikon A Gulian, Ravi G Patel, Mauro Perego, and Nathaniel A Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In *Mathematical and Scientific Machine Learning*, pages 512–536. PMLR, 2020.

Chinmay Datar, Taniya Kapoor, Abhishek Chandra, Qing Sun, Erik Lien Bolager, Iryna Burak, Anna Veselovska, Massimo Fornasier, and Felix Dietrich. Fast training of accurate physics-informed neural networks without gradient descent. *arXiv preprint arXiv:2405.20836*, 2024.

John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

Suchuan Dong and Zongwei Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387: 114129, 2021.

Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide, 2014.

Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. Tikhonov regularization and total least squares. *SIAM journal on matrix analysis and applications*, 21(1):185–194, 1999.

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. In *Machine learning in modeling and simulation: methods and applications*, pages 219–254. Springer, 2023.

Jan Hagnberger, Daniel Musekamp, and Mathias Niepert. CALM-PDE: Continuous and Adaptive Convolutions for Latent Space Modeling of Time-dependent PDEs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022.

Jingren Hou, Hong Wang, Pengyu Xu, Chang Gao, Huafeng Liu, and Liping Jing. Learning neural operators from partial observations via latent autoregressive modeling. *arXiv preprint arXiv:2601.15547*, 2026.

Peiyan Hu, Haodong Feng, Hongyuan Liu, Tongtong Yan, Wenhao Deng, Tianrun Gao, Rong Zheng, Haoren Zheng, Chenglei Yu, Chuanrui Wang, Kaiwen Li, Zhi-Ming Ma, Dezhi Zhou, Xingcai Lu, Dixia Fan, and Tailin Wu. Realpdebench: A benchmark for complex physical systems with real-world data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=y3oHMcoItR`. Oral Presentation.

Xiang Huang, Zhanhong Ye, Hongsheng Liu, Shi Ji, Zidong Wang, Kang Yang, Yang Li, Min Wang, Haotian Chu, Fan Yu, et al. Meta-auto-decoder for solving parametric partial differential equations. *Advances in Neural Information Processing Systems*, 35:23426–23438, 2022.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Jungeun Kim, Kookjin Lee, Dongeun Lee, Sheo Yon Jhin, and Noseong Park. Dpm: A novel training method for physics-informed neural networks in extrapolation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8146–8154, 2021.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Varun Kumar, Somdatta Goswami, Katiana Kontolati, Michael D Shields, and George Em Karniadakis. Synergistic learning with multi-task deeponet for efficient pde problem solving. *Neural Networks*, 184:107113, 2025.

Kangjie Li and Wenjing Ye. D-fno: A decomposed fourier neural operator for large-scale parametric partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 436:117732, 2025.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Xu Liu, Xiaoya Zhang, Wei Peng, Weien Zhou, and Wen Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, 34(17):14511–14534, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3 (3):218–229, 2021.

Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M Kirby. A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes. *Journal of Computational Physics*, 477:111912, 2023.

Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC press, 2012.

Yuan Qiu, Nolan Bridges, and Peng Chen. Derivative-enhanced deep operator network. *Advances in Neural Information Processing Systems*, 37:20945–20981, 2024.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Tony WH Sheu and RK Lin. Newton linearization of the incompressible navier–stokes equations. *International Journal for Numerical Methods in Fluids*, 44(3):297–312, 2004.

Tony WH Sheu and RK Lin. On a high-order newton linearization method for solving the incompressible navier–stokes equations. *International journal for numerical methods in engineering*, 62(11):1559–1578, 2005.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=tmIiMPl4IPa`.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. *Journal of Machine Learning Research*, 25(402):1–51, 2024.

Sifan Wang, Ananyae Kumar Bhartari, Bowen Li, and Paris Perdikaris. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. *arXiv preprint arXiv:2502.00604*, 2025.

Chang Wei, Yuchen Fan, Jian Cheng Wong, Chin Chun Ooi, Heyang Wang, and Pao-Hsiung Chiu. Ffv-pinn: A fast physics-informed neural network with simplified finite volume discretization and residual correction. *Computer Methods in Applied Mechanics and Engineering*, 444:118139, 2025a.

Zhi-Feng Wei, Wenqian Chen, and Panos Stinis. Efficient transformer-inspired variants of physics-informed deep operator networks. *arXiv preprint arXiv:2509.01679*, 2025b.

Jian Cheng Wong, Abhishek Gupta, and Yew-Soon Ong. Can transfer neuroevolution tractably solve your differential equations? *IEEE Computational Intelligence Magazine*, 16(2):14–30, 2021.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3):985–1000, 2022.

Jian Cheng Wong, Abhishek Gupta, Chin Chun Ooi, Pao-Hsiung Chiu, Jiao Liu, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Evo-pinn frontiers and opportunities. *IEEE Computational Intelligence Magazine*, 21(1):16–36, 2026a.

Jian Cheng Wong, Isaac Yin Chung Lai, Pao-Hsiung Chiu, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Transferable physics-informed representations via closed-form head adaptation. *arXiv preprint arXiv:2604.21761*, 2026b.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, Pao-Hsiung Chiu, Joshua Shao Zheng Low, My Ha Dao, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Advancing generalizability by the baldwin effect. *IEEE Transactions on Evolutionary Computation*, 2026c.

Chenhui Xu, Dancheng Liu, Amir Nassereldine, and Jinjun Xiong. Fp64 is all you need: Rethinking failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 2025.

Guancheng Zhou, Zelin Zeng, Yisi Luo, Qi Xie, and Deyu Meng. Tucker-fno: Tensor tucker-fourier neural operator and its universal approximation theory. In *The Fourteenth International Conference on Learning Representations*, 2026.

Appendix

In this Appendix, we provide comprehensive materials to facilitate a deeper understanding of our study.

- Appendix A provides an extended review of the literature.
- Appendix B presents a detailed description of the Newton-PINet model, including the meta-learning and gradient-free fine-tuning framework, and the Newton linearization method.
- Appendix C provides theoretical and experimental evidence showing that Newton linearization used in the Tikhonov-regularized PINN achieves near-quadratic convergence.
- Appendix D describes the data generation, computational configurations, and additional experimental results.
- Appendix E provides a comprehensive comparison of model performance on public datasets and newly introduced nonlinear benchmark problems.
- Appendix F provides ablation studies covering meta-learning hyperparameters, factors influencing inference cost, model architecture design, and related aspects.
- Appendix G presents an analysis of the computational complexity of the model and discusses its limitations and failure modes.

A Extended review of related work

In this section, we provide an extended review of the literature and clarify its relation to our work.

- Existing PINN architectures, such as DPM and PirateNets [Kim et al., 2021, Wang et al., 2024], employ standard residual connections (e.g., $x + f(x)$) to improve gradient flow and stabilize training. In contrast, our approach adopts a skip-connected design that concatenates features from all hidden layers and feeds them into the output layer, followed by a least-squares solve. This design aims to construct a richer basis space, where each hidden layer progressively contributes additional nonlinear features, thereby enhancing representation capacity and improving the conditioning and accuracy of the physics-informed Tikhonov solution, rather than merely stabilizing training.
- Least-squares Gradient Descent (LSGD) [Cyr et al., 2020] combines least-squares formulations with gradient-based optimization, focusing on data-driven single-task learning with linear structures. In contrast, our method extends least-squares formulations to meta-learning for nonlinear PDEs through a physics-informed Tikhonov framework. Specifically, the system is constructed from physical constraints (PDE residuals, ICs, and BCs), and Newton linearization transforms nonlinear residuals into local linear least-squares problems. This enables efficient gradient-free inner-loop adaptation with inherent enforcement of physical laws, which is absent in LSGD.
- Methods such as Almost No Inner Loop (ANIL) [Raghu et al., 2019] improve efficiency by restricting adaptation to the last layer. While sharing this idea, these approaches rely on gradient-based updates. In contrast, our method performs closed-form, physics-constrained least-squares updates via Tikhonov regularization. By eliminating inner-loop gradient descent and exploiting the algebraic structure of PDEs, our approach enables fast and stable adaptation to new tasks, fundamentally differing from conventional data-driven meta-learning methods.

Solving complex nonlinear PDEs via meta-learning presents fundamental challenges that require entirely different algorithmic considerations from traditional data-driven methods like LSGD or ANIL. By synergistically integrating basis-enriching skip-connections, Newton linearization, and physics-constrained Tikhonov least-squares, our work proposes a distinct physics-informed meta-learning framework tailored to the algebraic structure of physical systems, which is firmly backed by both theoretical and empirical analyses.

B Newton-PINet model

This study presents a skip-connected Tikhonov regularization PINN model for meta-learning nonlinear PDEs, enhanced with Newton linearization, to enable efficient few-shot learning and fast generalization, hereafter referred to as Newton-PINet.

B.1 Skip-connected PINN architecture

Building upon the conventional feed-forward MLP architecture of PINNs, we introduce skip connections from all hidden layers to the output layer, with the output-layer weights computed using Tikhonov regularization.

The skip-connected neural architecture enables us to provide better expressivity to ensure stable and accurate least-squares computation by increasing the width of the output layer through the stacking of additional hidden layers, even while maintaining a moderate number of nodes per layer. In addition, the concatenation mechanism can be interpreted as expanding a richer basis space: each added hidden layer introduces new nonlinear features, analogous to incorporating higher-order terms in a polynomial or Chebyshev-type basis, leading to a more expressive representation with reduced truncation effects. A complementary viewpoint is that skip connections also help stabilize optimization—much like in ResNets—by mitigating vanishing-gradient issues in deeper architectures.

This architecture resembles extreme learning machines (ELMs) in its stacked structure [Dong and Li, 2021]. However, in traditional ELMs, hidden-layer weights are randomly initialized, which makes training deeper architectures increasingly difficult and can compromise the accuracy of the solution. In contrast, our framework employs a meta-learning strategy to update the hidden-layer weights instead of relying on random initialization. As a result, the network depth can be increased appropriately to improve nonlinear representation capacity while maintaining stable and accurate generalization.

We employ the Tikhonov-regularized PINN with $L+1$ layers, where the input is $\mathbf{x} \in \mathbb{R}^{D_{in}}$ (layer 0) and the output is $\mathbf{u} \in \mathbb{R}^{D_o}$ (layer L). Here, D_{in} and D_o denote the dimensionality of the input and output variables, respectively. Each hidden layer $l = 1, \ldots, L-1$ contains the same number of neurons (N_n). Sinusoidal feature embeddings are applied at the first layer [Wong et al., 2022], and all hidden layers use $\sin(\cdot)$ activation functions. Skip connections from all hidden layers to the output layer are employed to allow each hidden layer to contribute nonlinearly to the final representation without additional weights in the concatenation.

The transformation and activation in the first hidden layer are defined as

$$z_j^1 = \sum_{d=1}^{D_{in}} W_{j,d}^1 x_d + b_j^1, \quad f_j^1(\mathbf{x}) = \sin(2\pi z_j^1), \quad j = 1, \ldots, N_n, \tag{3}$$

where j and d denote neurons in the current and previous layers respectively, $W^1 \in \mathbb{R}^{N_n \times D_{in}}$, and $b^1 \in \mathbb{R}^{N_n}$. We incorporate Sinusoidal feature embeddings in the first layer to enhance the representation of high-frequency components.

For subsequent hidden layers $l = 2, \ldots, L-1$, we have

$$z_j^l = \sum_{d=1}^{N_n} W_{j,d}^l f_d^{l-1} + b_j^l, \quad f_j^l(\mathbf{x}) = \sin(z_j^l), \quad j = 1, \ldots, N_n, \tag{4}$$

where $W^l \in \mathbb{R}^{N_n \times N_n}$ and $b^l \in \mathbb{R}^{N_n}$.

All trainable network parameters up to the pre-final layer are denoted as $\tilde{\boldsymbol{w}} = [W^l, b^l],\ l = 1, \ldots, L-1$.

The pre-final output is constructed as the concatenation of all hidden activations (skip connections):

$$\boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T = \left[f_1^1, \ldots, f_{N_n}^1,\ f_1^2, \ldots, f_{N_n}^2,\ \ldots,\ f_1^{L-1}, \ldots, f_{N_n}^{L-1}\right], \tag{5}$$

which is then connected to the output layer with D_o dimension, so that $\boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \in \mathbb{R}^{D_o \times N_n (L-1)}$. Then, we need to flatten it, resulting in $\boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \in \mathbb{R}^{1 \times D_o N_n (L-1)}$.

The output u can be computed by

$$u(\mathbf{x}) = \sum_j w_j f_j = \boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \boldsymbol{w}, \tag{6}$$

where $\boldsymbol{w}^T = [\dots w_j \dots] \in \mathbb{R}^{1 \times D_o N_n (L-1)}$ denotes the output-layer weights.

B.2 Physics-based least-squares formulation

The objective of task-specific learning is to determine the best set of $\boldsymbol{w}$ such that $u(\mathbf{x}) = \boldsymbol{f}(\mathbf{x}; \tilde{\boldsymbol{w}})^T \boldsymbol{w}$ satisfies the PDE, IC, and BC, for a task ($\mathcal{T}_i$):

$$\text{PDE:} \quad \mathcal{N}_\theta[u(x,t)] = q(x,t) \quad x \in \Omega,\ t \in [0,T], \tag{7a}$$

$$\text{IC:} \quad u(x, t=0) = u_0(x) \quad x \in \Omega, \tag{7b}$$

$$\text{BC:} \quad \mathcal{B}[u(x,t)] = g(x,t) \quad x \in \partial\Omega,\ t \in [0,T], \tag{7c}$$

where $\mathcal{N}_\theta, q, u_0, g$ denote the differential operator with PDE parameters θ, the source term, IC, and BC, respectively.

This leads to a physics-based least-squares formulation. Let $\mathbf{x} = (x,t)$. Given collocation points for the PDE residual, $(x_n^{\text{pde}}, t_n^{\text{pde}}),\ n = 1, \dots, n_{\text{pde}}$; points at the initial time, $(x_n^{\text{ic}}, 0),\ n = 1, \dots, n_{\text{ic}}$; and points on the boundary, $(x_n^{\text{bc}}, t_n^{\text{bc}}),\ n = 1, \dots, n_{\text{bc}}$, where n_{pde}, n_{ic}, and n_{bc} denote the numbers of the corresponding points, respectively. The loss importance hyperparameter λ_{pde} is further introduced, and the following system is obtained:

$$\begin{bmatrix} \cdots & \lambda_{\text{pde}} \cdot \mathcal{N}_\theta[f_j(x_1^{\text{pde}}, t_1^{\text{pde}}; \tilde{\boldsymbol{w}})] & \cdots \\ & \vdots & \\ \cdots & \lambda_{\text{pde}} \cdot \mathcal{N}_\theta[f_j(x_{n_{\text{pde}}}^{\text{pde}}, t_{n_{\text{pde}}}^{\text{pde}}; \tilde{\boldsymbol{w}})] & \cdots \\ \cdots & f_j^i(x_1^{\text{ic}}, 0; \tilde{\boldsymbol{w}}) & \cdots \\ & \vdots & \\ \cdots & f_j(x_{n_{\text{ic}}}^{\text{ic}}, 0; \tilde{\boldsymbol{w}}) & \cdots \\ \cdots & \mathcal{B}[f_j(x_1^{\text{bc}}, t_1^{\text{bc}}; \tilde{\boldsymbol{w}})] & \cdots \\ & \vdots & \\ \cdots & \mathcal{B}[f_j(x_{n_{\text{bc}}}^{\text{bc}}, t_{n_{\text{bc}}}^{\text{bc}}; \tilde{\boldsymbol{w}})] & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ w_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda_{\text{pde}} \cdot q(x_1^{\text{pde}}, t_1^{\text{pde}}) \\ \vdots \\ \lambda_{\text{pde}} \cdot q(x_{n_{\text{pde}}}^{\text{pde}}, t_{n_{\text{pde}}}^{\text{pde}}) \\ u_0(x_1^{\text{ic}}) \\ \vdots \\ u_0(x_{n_{\text{ic}}}^{\text{ic}}) \\ g(x_1^{\text{bc}}, t_1^{\text{bc}}) \\ \vdots \\ g(x_{n_{\text{bc}}}^{\text{bc}}, t_{n_{\text{bc}}}^{\text{bc}}) \end{bmatrix}, \tag{8}$$

which can be compactly written as

$$\mathbf{A}\boldsymbol{w} = \mathbf{b}.$$

Here $\mathbf{A}\boldsymbol{w}$ represents the left-hand side of Eq. (7) at a given set of collocation points, $\mathbf{b}$ denotes the corresponding right-hand side of Eq. (7).

Therefore, using the PDE, IC, and BC, the physics-informed matrix $\mathbf{A}$ and vector $\mathbf{b}$ can be assembled. Concretely, the pre-final representation $\boldsymbol{f}$ is obtained via forward propagation, and its derivatives with respect to (x,t) in $\mathcal{N}_\theta$ are computed through automatic differentiation (AD).

For a single task, i.e., a single PDE instance, the number of rows in $\mathbf{A}$ and $\mathbf{b}$ equals the total number of collocation points, including those for the PDE residual, IC, and BC. The number of columns in $\mathbf{A}$ corresponds to the product of the pre-final feature dimension $N_n(L-1)$ and the output dimension D_o. This results in $\mathbf{A} \in \mathbb{R}^{(n_{\text{pde}}+n_{\text{ic}}+n_{\text{bc}}) \times D_o N_n (L-1)}$ and $\mathbf{b} \in \mathbb{R}^{(n_{\text{pde}}+n_{\text{ic}}+n_{\text{bc}}) \times 1}$.

The final output weights $\boldsymbol{w}$ are obtained via Tikhonov regularization:

$$\arg\min_{\boldsymbol{w}} \ \|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|_2^2 + \lambda_{\text{reg}}\, \boldsymbol{w}^T \boldsymbol{w}, \tag{9}$$

where $\lambda_{\text{reg}} \geq 0$ denotes the regularization parameter. Compared with ordinary least squares (OLS), which minimizes only the residual norm $\|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|_2^2$, Tikhonov regularization introduces an additional penalty term $\lambda_{\text{reg}}\, \boldsymbol{w}^T \boldsymbol{w}$. This term suppresses large weights, stabilizes the solution, and improves the conditioning of the problem.

Note that for linear PDE, the closed-form Tikhonov-regularized solution of Eq. (8) can be obtained in a single step:

$$\boldsymbol{w} = \begin{cases} (\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, & \text{if Eq. (8) is over-determined,} \\ \mathbf{A}^T(\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}, & \text{if Eq. (8) is under-determined.} \end{cases} \tag{10}$$

For nonlinear PDEs, the nonlinear terms must first be linearized so that the system can be cast in the form of Eq. (8). For example, in the Burgers' equation: $u_t + uu_x - \gamma u_{xx} = 0$, we can compute

$$u = \boldsymbol{f}^T\boldsymbol{w}, \quad u_t = \frac{\partial u}{\partial t} = \boldsymbol{f}_t^T\boldsymbol{w}, \quad u_x = \frac{\partial u}{\partial x} = \boldsymbol{f}_x^T\boldsymbol{w}, \quad u_{xx} = \frac{\partial^2 u}{\partial x^2} = \boldsymbol{f}_{xx}^T\boldsymbol{w},$$

where the pre-final representation $\boldsymbol{f}$ and its derivatives can be obtained via forward propagation and AD. However, the nonlinear term uu_x cannot be directly expressed as a combination of $\boldsymbol{f}$ (or $\boldsymbol{f}_x$) and $\boldsymbol{w}$, which motivates the need for linearization. Under Newton linearization, the nonlinear term can be approximated as

$$(uu_x)^{k+1} \approx u^{k+1}u_x^k + u^k u_x^{k+1} - u^k u_x^k,$$

where k and $k+1$ denote the current and next nonlinear iteration step, respectively. Given initial guesses for u and u_x (e.g., setting u and u_x at all time steps equal to the initial field), the corresponding entries in $\mathbf{A}$ and $\mathbf{b}$ are updated as:

$$\begin{aligned} \mathcal{N}_\theta\big[f_j^{k+1}(\mathbf{x}_n^{\text{pde}}; \tilde{\boldsymbol{w}})\big] = &\frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}}; \tilde{\boldsymbol{w}})}{\partial t} + u^k(\mathbf{x}_n^{\text{pde}})\frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}}; \tilde{\boldsymbol{w}})}{\partial x} \\ &+ f_j^{k+1}(\mathbf{x}_n^{\text{pde}}; \tilde{\boldsymbol{w}})\frac{\partial u^k(\mathbf{x}_n^{\text{pde}})}{\partial x} - \gamma\frac{\partial^2 f_j^{k+1}(\mathbf{x}_n^{\text{pde}}; \tilde{\boldsymbol{w}})}{\partial x^2}, \end{aligned} \tag{11}$$

$$q(\mathbf{x}_n^{\text{pde}}) = u^k(\mathbf{x}_n^{\text{pde}})\frac{\partial u^k(\mathbf{x}_n^{\text{pde}})}{\partial x}. \tag{12}$$

This leads to an iterative nonlinear solve, where Eq. (10) is repeatedly solved to update $\boldsymbol{w}$ toward the optimal solution.

How to make Tikhonov solver tractable? The task adaptation is formulated as a Tikhonov-regularized least-squares problem, whose tractability mainly depends on the size of $\mathbf{A}$, the Tikhonov regularization parameter (λ_{reg}), the PDE-loss importance hyperparameter (λ_{pde}), and the choice of the initial field during nonlinear iterations.

- The matrix $\mathbf{A}$ is constructed from the network features at each sampling point of the PDE residuals/ICs/BCs, which are generated by a multilayer neural representation with skip connections. The distribution of these features directly determines the conditioning of the solve. In addition, when the system requires a large number of sampling points, the number of rows in $\mathbf{A}$ increases, making it more difficult to solve. Since the Tikhonov update only involves $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{D_o N_n(L-1) \times D_o N_n(L-1)}$, the computational cost mainly depends on the network feature dimension. Thus, the practical scaling for large collocation sets may derive more from the fact that one can still control the cost of the solve through meta-learning a better (yet minimal) set of pre-final layer features.
- To avoid laborious hyperparameter tuning, both λ_{reg} and λ_{pde} are meta-learned jointly with the network parameters.
- For nonlinear problems, an inappropriate initial field may slow convergence. Therefore, we use the initial condition as u_{guess} (setting u_{guess} at all time steps equal to the initial field) for time-dependent problems and employ temporal domain decomposition for more complex dynamics, ensuring that each block starts closer to the true solution, which improves the stability of the least-squares updates.

The specific discussion on the scalability of the model is provided in Appendix G.

B.3 Meta-learning PINN on training tasks

The trainable hyperparameters during meta-learning include network weights ($\tilde{\boldsymbol{w}}$), PDE loss importance hyperparameter (λ_{pde}), and regularization parameter (λ_{reg}), collectively denoted as Θ. These components are crucial for achieving optimal performance in task-specific physics-informed learning.

The meta-learning objective is to optimize Θ to learn task-agnostic representations spanning a family of PDEs, enabling fast generalization to unseen tasks via fine-tuning only the output-layer weights using the Tikhonov regularization update. In our framework, the outer loop of meta-learning updates the learnable parameters via gradient descent, while task-specific adaptation in the inner loop, or generalization, is performed through gradient-free Tikhonov regularization applied to the output-layer weights. See Algorithm 1 for the pseudocode workflow.

We consider a distribution over tasks $p(\mathcal{T})$ that we want our model to be able to adapt to. The meta-objective (loss) may consist solely of physics-based least-squares error (l_{LSE}, unsupervised), data-driven mean squared error (l_{MSE}, few-shot training), or a combination of both:

$$\min_{\Theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathbb{E}_{\tilde{\boldsymbol{w}} \sim p_\theta(\tilde{\boldsymbol{w}})} \left[l_{\text{LSE}}(\boldsymbol{w}^*) + l_{\text{MSE}}(\boldsymbol{w}^*) \right]. \tag{13}$$

Here, Tikhonov regularization is used to obtain the optimal output-layer weights $\boldsymbol{w}^*$ that allow the model to specialize to any realization of task $\mathcal{T}_i \sim p(\mathcal{T})$ for the given network's $\tilde{\boldsymbol{w}} \sim p_\theta(\tilde{\boldsymbol{w}})$.

l_{LSE} and l_{MSE} are

$$l_{\text{LSE}}(\boldsymbol{w}^*) = (\mathbf{A}\boldsymbol{w}^* - \mathbf{b})^T(\mathbf{A}\boldsymbol{w}^* - \mathbf{b}), \tag{14}$$

$$l_{\text{MSE}}(\boldsymbol{w}^*) = \frac{1}{n}\sum_{s=1}^{n}(u_s^{\text{label}} - \sum_j w_j^* f_j(x_s, t_s; \tilde{\boldsymbol{w}}))^2, \tag{15}$$

given labeled data $\{u_s^{\text{label}}\}_{s=1}^n$ (n denotes the total number of collocation points including PDE residual, IC, and BC constraints for a task).

We refer to l_{LSE}-based learning as **unsupervised learning**, and to l_{MSE}-based learning as **hybrid learning** (a combination of physics-based Tikhonov updates and data-driven loss). Unless otherwise stated, for meta-learning loss (outer-loop), our model does not consider the combined loss formulation $l_{\text{LSE}} + l_{\text{MSE}}$, as in some cases the combined objective exhibits poorer generalization performance compared with using l_{MSE} alone.

In practice, we observe that for simple nonlinear PDEs (e.g., Burgers' equation), training with l_{LSE} alone already leads to strong generalization. However, for more complex PDEs, incorporating a small amount of labeled data through the l_{MSE} term becomes essential for both stability and accuracy.

B.4 Gradient-free fine-tuning to new tasks

After meta-learning, the network weights ($\tilde{\boldsymbol{w}}$), the PDE-loss importance hyperparameter (λ_{pde}), and the regularization parameter (λ_{reg}) are fixed. For a new task with different PDE parameters or IC/BC conditions, Tikhonov regularization is used to compute the output-layer weights ($\boldsymbol{w}$), enabling fast gradient-free adaptation to the new PDE instance. See Algorithm 2 for the pseudocode workflow. Since this update is independent of gradient-based backpropagation optimizers such as stochastic gradient descent (SGD), the resulting adaptation is extremely fast while remaining physics-compliant.

B.5 Linearization for nonlinear PDEs

It should be noted that the Tikhonov regularization is directly applicable to linear PDEs, where a linear matrix system can be constructed and solved in a single step. For nonlinear PDEs, the nonlinear terms must first be linearized so that the system can be cast into a linear form. The previous approaches typically relied on Picard method (lagging-of-coefficients), which linearizes the nonlinear terms using the solution from the previous iteration and then performs least-squares solves for multiple nonlinear iterations.

Picard iterations. Take the Burgers' equation as an example: $u_t + uu_x - \gamma u_{xx} = 0$. The Picard method approximately linearizes the nonlinear term as: $(uu_x)^{k+1} = u^k u_x^{k+1}$, where k and $k+1$ denote the current and next iteration steps, respectively. An initial guess for u is provided, e.g., setting u at all time steps equal to the initial field. Under this formulation, the entries of the least-squares

matrix $\mathbf{A}$ and vector $\mathbf{b}$ in Eq. (8) corresponding to the $(k+1)$-th iteration, n-th PDE collocation point, and the j-th output-layer neuron are given by:

$$\mathcal{N}_\theta\left[f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\tilde{\boldsymbol{w}})\right] = \frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\tilde{\boldsymbol{w}})}{\partial t} + u^k(\mathbf{x}_n^{\text{pde}})\frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\tilde{\boldsymbol{w}})}{\partial x} - \gamma\frac{\partial^2 f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\tilde{\boldsymbol{w}})}{\partial x^2} \tag{16}$$

$$q(\mathbf{x}_n^{\text{pde}}) = 0, \tag{17}$$

where f_j denotes the pre-final output of the neural network. Note that u^k here is the known solution, either the initial guess or the solution at the k-th iteration.

The final output weights $\boldsymbol{w}^T = [\ldots, w_j, \ldots]$ are obtained via Tikhonov regularization-based nonlinear iterations. At each iteration, the solution is updated by $u(\mathbf{x}) = \sum_j w_j f_j$, which then serves as the new guess for the next iteration. This procedure is repeated until $\boldsymbol{w}$ satisfies a prescribed convergence criterion or the maximum number of iterations is reached.

Newton linearization iterations. The Picard approach suffers from slow convergence of nonlinear PDE solutions due to its linear convergence speed. To overcome these limitations, we use the Newton linearization approach.

Let $F(u^{k+1})$ denote the nonlinear term as a function of the state variable u. In Newton linearization, a first-order Taylor expansion is performed around the current iterate u^k, yielding

$$F(u^{k+1}) \quad = F(u^k) + \left.\tfrac{dF}{du}\right|^k (u^{k+1} - u^k) + \text{H.O.T}, \tag{18}$$

where "H.O.T" denotes the higher-order truncated terms.

According to Eq. (18), we can derive a general expression for the nonlinear terms u^m with m denoting the exponent that commonly appear in PDEs, as follows:

$$\begin{aligned}(u^m)^{k+1} &\approx (u^m)^k + m(u^{m-1})^k(u^{k+1} - u^k)\\ &= m(u^k)^{m-1}u^{k+1} + (1-m)(u^k)^m\end{aligned} \tag{19}$$

The corresponding expressions for $m = 2$ and $m = 3$ are as follows:

$$(u^2)^{k+1} \approx 2u^{k+1}u^k - u^k u^k \tag{20}$$

$$(u^3)^{k+1} \approx 3u^{k+1}u^k u^k - 2u^k u^k u^k \tag{21}$$

For nonlinear terms involving multiple state variables, i.e., $F(u^{k+1}, v^{k+1})$, the corresponding Taylor expansion is given by:

$$F(u^{k+1}, v^{k+1}) = F(u^k, v^k) + \left.\tfrac{\partial F}{\partial u}\right|^k (u^{k+1} - u^k) + \left.\tfrac{\partial F}{\partial v}\right|^k (v^{k+1} - v^k) + \text{H.O.T} \tag{22}$$

Based on Eq. (22), the general Newton linearization form for nonlinear convective terms of arbitrary order can be expressed as:

$$\begin{aligned}(u^m u_x)^{k+1} &\approx (u^m)^k u_x^k + u_x^k\left[(u^m)^{k+1} - (u^m)^k\right] + (u^m)^k(u_x^{k+1} - u_x^k)\\ &= (u^m)^{k+1}u_x^k + (u^m)^k u_x^{k+1} - (u^m)^k u_x^k\\ &= \left[m(u^k)^{m-1}u^{k+1} + (1-m)(u^k)^m\right]u_x^k + (u^k)^m u_x^{k+1} - (u^k)^m u_x^k\\ &= m(u^k)^{m-1}u_x^k u^{k+1} + (1-m)(u^k)^m u_x^k + (u^k)^m u_x^{k+1} - (u^k)^m u_x^k\\ &= m(u^k)^{m-1}u_x^k u^{k+1} + (u^k)^m u_x^{k+1} - m(u^k)^m u_x^k\end{aligned} \tag{23}$$

Accordingly, the corresponding expressions for the nonlinear convective terms with $m = 1$, $m = 2$, and $m = 3$ are given by:

$$(uu_x)^{k+1} \approx u^{k+1}u_x^k + u^k u_x^{k+1} - u^k u_x^k \tag{24}$$

$$(u^2 u_x)^{k+1} \approx 2u^k u_x^k u^{k+1} + (u^k)^2 u_x^{k+1} - 2(u^k)^2 u_x^k \tag{25}$$

$$(u^3 u_x)^{k+1} \approx 3(u^k)^2 u_x^k u^{k+1} + (u^k)^3 u_x^{k+1} - 3(u^k)^3 u_x^k \tag{26}$$

Therefore, the nonlinear term in the Burgers' equation under Newton linearization can be expressed as $(uu_x)^{k+1} \approx \underline{u^{k+1}u_x^k} + u^k u_x^{k+1} - \underline{u^k u_x^k}$, where the underlined terms originate from higher-order corrections that refine the classical Picard linearization: $(uu_x)^{k+1} = u^k u_x^{k+1}$. Given initial guesses for u and u_x (e.g., setting u and u_x at all time steps equal to the initial field), the corresponding entries in $\mathbf{A}$ and $\mathbf{b}$ are updated as:

$$\begin{aligned}\mathcal{N}_\theta\big[f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\,\tilde{\boldsymbol{w}})\big] = &\frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\,\tilde{\boldsymbol{w}})}{\partial t} + u^k(\mathbf{x}_n^{\text{pde}})\frac{\partial f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\,\tilde{\boldsymbol{w}})}{\partial x} \\ &+ f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\,\tilde{\boldsymbol{w}})\frac{\partial u^k(\mathbf{x}_n^{\text{pde}})}{\partial x} - \gamma\frac{\partial^2 f_j^{k+1}(\mathbf{x}_n^{\text{pde}};\,\tilde{\boldsymbol{w}})}{\partial x^2}\end{aligned} \tag{27}$$

$$q(\mathbf{x}_n^{\text{pde}}) = u^k(\mathbf{x}_n^{\text{pde}})\frac{\partial u^k(\mathbf{x}_n^{\text{pde}})}{\partial x}. \tag{28}$$

The Newton linearization method substantially accelerates the convergence of Tikhonov regularization-based nonlinear iterations due to its near-quadratic convergence property (see Appendix C for a proof). Consequently, it reduces the time complexity of meta-learning and enhances the speed of fine-tuning and generalization on new tasks.

Similarly, the Newton-linearized expressions for other nonlinear terms, such as hyperbolic, exponential, and logarithmic forms, are given as follows:

$$\begin{aligned}[\sinh(u)]^{k+1} &\approx \sinh(u^k) + \cosh(u^k)(u^{k+1} - u^k) \\ &= \cosh(u^k)u^{k+1} + \sinh(u^k) - \cosh(u^k)u^k\end{aligned} \tag{29}$$

$$\begin{aligned}[\cosh(u)]^{k+1} &\approx \cosh(u^k) + \sinh(u^k)(u^{k+1} - u^k) \\ &= \sinh(u^k)u^{k+1} + \cosh(u^k) - \sinh(u^k)u^k\end{aligned} \tag{30}$$

$$\begin{aligned}[\tanh(u)]^{k+1} &\approx \tanh(u^k) + \big(1 - \tanh^2(u^k)\big)(u^{k+1} - u^k) \\ &= \big(1 - \tanh^2(u^k)\big)u^{k+1} + \tanh(u^k) \\ &\quad -\big(1 - \tanh^2(u^k)\big)u^k\end{aligned} \tag{31}$$

$$\begin{aligned}[\exp(u)]^{k+1} &\approx \exp(u^k) + \exp(u^k)(u^{k+1} - u^k) \\ &= \exp(u^k)u^{k+1} + \exp(u^k) - \exp(u^k)u^k\end{aligned} \tag{32}$$

$$\begin{aligned}[u\ln(u)]^{k+1} &\approx u^k\ln(u^k) + (\ln(u^k) + 1)(u^{k+1} - u^k) \\ &= (\ln(u^k) + 1)u^{k+1} + u^k\ln(u^k) - (\ln(u^k) + 1)u^k\end{aligned} \tag{33}$$

C Convergence analysis of the Newton-PINet

Newton linearization has long been employed in numerical PDE solvers [Sheu and Lin, 2004, 2005, Chiu et al., 2008], e.g., to linearize the convective term in the Navier-Stokes equations. We integrate Newton linearization into the Tikhonov-regularized PINN, resulting in Newton-PINet, which significantly improves the performance of meta-learning nonlinear PDEs. This appendix provides theoretical analysis and experimental validation showing that the Newton-PINet optimization procedure is essentially equivalent to the classical regularized Gauss-Newton method and achieves near-quadratic convergence.

C.1 Nonlinear system formulation

The output u and its spatial derivative u_x at the n-th collocation point in the Tikhonov-regularized PINN are given by

$$u(\mathbf{x}_n) = \sum_j f_j(\mathbf{x}_n) w_j, \quad u_x(\mathbf{x}_n) = \sum_j f_{x,j}(\mathbf{x}_n) w_j \quad \Rightarrow \quad u = \boldsymbol{\Phi}\boldsymbol{w}, \quad u_x = \boldsymbol{\Phi}_x \boldsymbol{w}, \tag{34}$$

where the pre-final features $\{f_j\}$ are generated by hidden-layer transformations followed by smooth activation functions, and are thus inherently bounded and continuously differentiable. Here, $\boldsymbol{w}^T = [\ldots, w_j, \ldots]$ denotes the output-layer weights, $\Phi_{nj} = f_j(\mathbf{x}_n)$, and $(\Phi_x)_{nj} = f_{x,j}(\mathbf{x}_n)$.

To provide a simple illustration, we consider a nonlinear equation $F = uu_x$ defined on the domain $x \in [0, 1]$, subject to the boundary conditions $u(0) = 0$ and $u(1) = 1$. The PDE residuals evaluated at the collocation points $\{x_n\}$ are then given by:

$$r_n(\boldsymbol{w}) = u(x_n)\, u_x(x_n) = (\boldsymbol{\Phi}\boldsymbol{w})_n\, (\boldsymbol{\Phi}_x \boldsymbol{w})_n. \tag{35}$$

The residuals corresponding to the boundary conditions are

$$r_0(\boldsymbol{w}) = u(0) - 0 = \boldsymbol{\Phi}(0,:)\boldsymbol{w}, \qquad r_1(\boldsymbol{w}) = u(1) - 1 = \boldsymbol{\Phi}(1,:)\boldsymbol{w} - 1. \tag{36}$$

We consider the nonlinear system of equations in a general form: $\mathbf{A}(\boldsymbol{w})\, \boldsymbol{w} = \mathbf{b}$.

By stacking all the residuals, we obtain the unified nonlinear system:

$$\begin{aligned} F(\boldsymbol{w}) &= \mathbf{A}(\boldsymbol{w})\, \boldsymbol{w} - \mathbf{b} \\ &= \begin{bmatrix} u \odot u_x \\ \boldsymbol{\Phi}(0,:)\boldsymbol{w} \\ \boldsymbol{\Phi}(1,:)\boldsymbol{w} - 1 \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\Phi}\boldsymbol{w}) \odot (\boldsymbol{\Phi}_x \boldsymbol{w}) \\ \boldsymbol{\Phi}(0,:)\boldsymbol{w} \\ \boldsymbol{\Phi}(1,:)\boldsymbol{w} - 1 \end{bmatrix} = \mathbf{0}, \end{aligned} \tag{37}$$

where $\odot$ denotes the elementwise product.

C.2 Classical Gauss-Newton method

In our experiments, the matrix $\mathbf{A}$ is predominantly overdetermined. For example, in the Burgers' equation, $\mathbf{A} \in \mathbb{R}^{6810 \times 512}$. In this setting, the Newton-PINet optimization can be characterized as the Gauss-Newton method [Dennis Jr and Schnabel, 1996]. Specifically, the goal of the Gauss-Newton method is to minimize a nonlinear least-squares objective function:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2} \|F(\boldsymbol{w})\|_2^2 = \frac{1}{2} \sum_{i=1}^{m} F_i(\boldsymbol{w})^2. \tag{38}$$

The Gauss-Newton method provides an iterative scheme to minimize this objective by approximating the true Hessian matrix $\nabla^2 \mathcal{L}(\boldsymbol{w})$. While the exact Hessian is given by $\nabla^2 \mathcal{L}(\boldsymbol{w}) = J_F(\boldsymbol{w})^T J_F(\boldsymbol{w}) + \sum_i F_i(\boldsymbol{w}) \nabla^2 F_i(\boldsymbol{w})$, the Gauss-Newton method neglects the second-order term $\sum_i F_i \nabla^2 F_i$, assuming it is small. This yields the approximated Hessian $H_{\mathrm{GN}} \approx J_F(\boldsymbol{w})^T J_F(\boldsymbol{w})$.

Setting the gradient to zero yields the Gauss-Newton update step $\Delta \boldsymbol{w} = \boldsymbol{w}^{k+1} - \boldsymbol{w}^k$, which is determined by solving the normal equations:

$$\big(J_F(\boldsymbol{w}^k)^T J_F(\boldsymbol{w}^k)\big)(\boldsymbol{w}^{k+1} - \boldsymbol{w}^k) = -J_F(\boldsymbol{w}^k)^T F(\boldsymbol{w}^k). \tag{39}$$

This update is mathematically equivalent to solving the following linearized overdetermined system in the least-squares sense:

$$\boxed{J_F(\boldsymbol{w}^k)\, \boldsymbol{w}^{k+1} = J_F(\boldsymbol{w}^k)\, \boldsymbol{w}^k - F(\boldsymbol{w}^k).} \tag{40}$$

C.3 Our Newton-PINet: Newton linearization used in Tikhonov-regularized PINN

In our Newton-PINet framework, the nonlinear PDE operators are first linearized at the continuous level (using Newton expansion), and the resulting discretized system is written in the linear form $\mathbf{A}(\boldsymbol{w}^k)\, \boldsymbol{w}^{k+1} = \mathbf{b}^k$. This linear system is then solved iteratively using linear least squares.

Using the Newton expansion for uu_x (as shown in Eq. (24)) as an illustrative example, the linearization of $(uu_x)^{k+1}$ is:

$$(uu_x)^{k+1} \approx u^{k+1}u_x^k + u^k u_x^{k+1} - u^k u_x^k, \tag{41}$$

where k and $k+1$ denote the current and next iteration steps, respectively. Defining $u^k = \boldsymbol{\Phi}\boldsymbol{w}^k$ and $u_x^k = \boldsymbol{\Phi}_x\boldsymbol{w}^k$, we have:

$$\begin{aligned}(uu_x)^{k+1} &= (\boldsymbol{\Phi}\boldsymbol{w})^{(k+1)} \odot (\boldsymbol{\Phi}_x\boldsymbol{w})^{(k+1)} \\ &\approx (\boldsymbol{\Phi}_x\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}\boldsymbol{w}^{(k+1)}) + (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^{(k+1)}) - (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k).\end{aligned} \tag{42}$$

Through matrix manipulations, this becomes:

$$(\boldsymbol{\Phi}_x\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}\boldsymbol{w}^{(k+1)}) = \operatorname{diag}(\boldsymbol{\Phi}_x\boldsymbol{w}^k)\,(\boldsymbol{\Phi}\boldsymbol{w}^{(k+1)}) = \big(\operatorname{diag}(\boldsymbol{\Phi}_x\boldsymbol{w}^k)\boldsymbol{\Phi}\big)\boldsymbol{w}^{(k+1)},$$

$$(\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^{(k+1)}) = \operatorname{diag}(\boldsymbol{\Phi}\boldsymbol{w}^k)\,(\boldsymbol{\Phi}_x\boldsymbol{w}^{(k+1)}) = \big(\operatorname{diag}(\boldsymbol{\Phi}\boldsymbol{w}^k)\boldsymbol{\Phi}_x\big)\boldsymbol{w}^{(k+1)}.$$

Consequently, the constructed matrix $\mathbf{A}(\boldsymbol{w}^k)$ and the right-hand side vector $\mathbf{b}^k$ are explicitly given by:

$$\mathbf{A}(\boldsymbol{w}^k) = \begin{bmatrix} \big(\operatorname{diag}(\boldsymbol{\Phi}_x\boldsymbol{w}^k)\boldsymbol{\Phi}\big) + \big(\operatorname{diag}(\boldsymbol{\Phi}\boldsymbol{w}^k)\boldsymbol{\Phi}_x\big) \\ \boldsymbol{\Phi}(0,:) \\ \boldsymbol{\Phi}(1,:) \end{bmatrix}, \quad \mathbf{b}^k = \begin{bmatrix} (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k) \\ 0 \\ 1 \end{bmatrix}. \tag{43}$$

It can be verified that our $\mathbf{A}(\boldsymbol{w}^k)$ and $\mathbf{b}^k$ correspond exactly to the classical Gauss-Newton update:

$$\boxed{J_F(\boldsymbol{w}^k)\,\boldsymbol{w}^{k+1} = J_F(\boldsymbol{w}^k)\boldsymbol{w}^k - F(\boldsymbol{w}^k),}$$

i.e.,

$$\mathbf{A}(\boldsymbol{w}^k) = J_F(\boldsymbol{w}^k), \tag{44}$$

$$\begin{aligned}\mathbf{b}^k &= J_F(\boldsymbol{w}^k)\,\boldsymbol{w}^k - F(\boldsymbol{w}^k) \\ &= \begin{bmatrix} (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k) + (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k) - (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k) \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} (\boldsymbol{\Phi}\boldsymbol{w}^k) \odot (\boldsymbol{\Phi}_x\boldsymbol{w}^k) \\ 0 \\ 1 \end{bmatrix}.\end{aligned} \tag{45}$$

Furthermore, by integrating Tikhonov regularization $\lambda_{\text{reg}}\|\boldsymbol{w}\|^2$ into our linear solvers to improve conditioning, the objective function is effectively penalized. The corresponding normal equations solved in our algorithm become:

$$\big(J_F(\boldsymbol{w}^k)^T J_F(\boldsymbol{w}^k) + \lambda_{\text{reg}}\mathbf{I}\big)\boldsymbol{w}^{k+1} = J_F(\boldsymbol{w}^k)^T\mathbf{b}^k, \tag{46}$$

which categorizes our Newton-PINet approach as a **regularized Gauss-Newton method**.

The convergence of the classical regularized Gauss-Newton method depends on both the magnitude of the residuals and the strength of the regularization. Near-quadratic convergence can be achieved when the residuals are sufficiently small and the regularization term is sufficiently weak relative to the Gauss-Newton Hessian approximation. Newton-PINet exhibits near-quadratic convergence, similar to the regularized Gauss-Newton method, only when the meta-learned neural representation is sufficiently accurate, the initial guess is not too far from the solution, and the regularization coefficient is not too large.

C.4 PINet: Picard linearization used in Tikhonov-regularized PINN

As a baseline model, we employ Picard linearization in the PINet framework. For the Picard linearization, also referred to as the "lagging-of-coefficients" approach, one factor in the nonlinear term uu_x is kept from the previous iteration k. Specifically, the linearization of $(uu_x)^{k+1}$ is given by $(uu_x)^{k+1} \approx u^k\, u_x^{k+1}$, which can be written in matrix form as

$$\begin{aligned}(uu_x)^{k+1} &\approx (\boldsymbol{\Phi}\boldsymbol{w})^k \odot (\boldsymbol{\Phi}_x\boldsymbol{w})^{(k+1)} \\ &= \big(\operatorname{diag}(\boldsymbol{\Phi}\boldsymbol{w}^k)\boldsymbol{\Phi}_x\big)\boldsymbol{w}^{(k+1)}.\end{aligned} \tag{47}$$

Consequently, in the linear system $\mathbf{A}(\boldsymbol{w}^k)\,\boldsymbol{w}^{k+1} = \mathbf{b}^k$, the corresponding matrix and right-hand side are

$$\mathbf{A}(\boldsymbol{w}^k) = \begin{bmatrix} \big(\operatorname{diag}(\boldsymbol{\Phi}\boldsymbol{w}^k)\boldsymbol{\Phi}_x\big) \\ \boldsymbol{\Phi}(0,:) \\ \boldsymbol{\Phi}(1,:) \end{bmatrix}, \quad \mathbf{b}^k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{48}$$

Since the Picard method only updates one factor at each iteration while keeping the other fixed, the convergence rate is generally linear rather than quadratic.

C.5 Experimental evaluation of convergence

The convergence behavior of Newton-PINet is further verified experimentally. The residual norm $|\mathbf{A}\boldsymbol{w} - \mathbf{b}|$ and the empirical order of convergence (EOC), $\mathrm{EOC} = \frac{\log(\|e_{k+1}\|/\|e_k\|)}{\log(\|e_k\|/\|e_{k-1}\|)}$, are evaluated for both Newton and Picard iterations on the nonlinear Burgers problem. As shown in Table 5, using the initialization $u_{\mathrm{guess}}(t) = u_{t=0}$ for all time steps, Newton-PINet achieves an observed EOC of 1.7–2.0, a smaller final residual (2.98×10^{-3}), and a substantially lower MSE (4.59×10^{-7}) than Picard (EOC ≈ 1.2), indicating near-quadratic convergence when the initialization is sufficiently close to the solution. Table 6 further shows that poorer initializations, i.e., larger ℓ_2 distances from the ground-truth solution u_{gt}, lead to a lower convergence order of Newton-PINet. As discussed in Appendix F.2, this sensitivity can be mitigated by initializing all time steps with the initial field and employing temporal block decomposition for more challenging nonlinear time-dependent problems such as the K-S equation.

Table 5: Convergence process comparison between Newton and Picard iterations

	Newton			Picard		
Iter.	$\|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|$	EOC	MSE	$\|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|$	EOC	MSE
1	1.27×10^{-2}	–	4.91×10^{-2}	8.64×10^{-3}	–	2.65×10^{-2}
2	5.62×10^{-2}	–	3.28×10^{-3}	6.30×10^{-3}	–	4.48×10^{-3}
3	7.26×10^{-3}	1.98	2.71×10^{-5}	6.01×10^{-3}	1.26	5.15×10^{-4}
4	2.98×10^{-3}	1.73	4.69×10^{-7}	4.54×10^{-3}	1.26	5.20×10^{-5}
5	2.98×10^{-3}	1.85	4.59×10^{-7}	4.08×10^{-3}	1.16	1.50×10^{-5}

Table 6: Impact of initial guess on the convergence

		Newton			Picard		
Initial guess	$\|u_0 - u_{\mathrm{gt}}\|_2$	$\|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|$	EOC	MSE	$\|\mathbf{A}\boldsymbol{w} - \mathbf{b}\|$	EOC	MSE
$u(t) = u_{t=0}$	28.59	2.98×10^{-3}	1.85	4.59×10^{-7}	4.08×10^{-3}	1.16	1.50×10^{-5}
Zero field	48.93	2.98×10^{-3}	1.63	4.62×10^{-7}	4.60×10^{-3}	1.26	5.19×10^{-5}
Constant field	94.73	1.33	0.53	1.53×10^{-1}	4.07×10^{-1}	0.87	6.01×10^{-2}

The empirical results (Table 7 below) show that increasing λ_{reg} degrades the observed empirical order of convergence (EOC), transitioning Newton-PINet from near-quadratic convergence to linear/sublinear convergence. When λ_{reg} is sufficiently small, it mainly serves to improve numerical

conditioning. However, as $\lambda_{\text{reg}}\mathbf{I}$ begins to dominate, the regularized update $(\mathbf{A}^T\mathbf{A} + \lambda_{\text{reg}}\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b}$ approaches $\frac{1}{\lambda_{\text{reg}}}\mathbf{A}^T\mathbf{b}$, transforming the Newton update into a heavily damped gradient-descent step and thereby degrading the convergence order.

Table 7: Impact of λ_{reg} on convergence order (EOC) and MSE

		Newton		Picard	
Scale	λ_{reg}	EOC	MSE	EOC	MSE
$1\times$	7.58×10^{-4}	1.85	4.59×10^{-7}	1.16	1.50×10^{-5}
$10\times$	7.58×10^{-3}	1.80	8.90×10^{-7}	1.16	1.45×10^{-5}
$10^2\times$	7.58×10^{-2}	1.55	1.27×10^{-5}	1.15	3.22×10^{-5}
$10^3\times$	7.58×10^{-1}	0.93	3.19×10^{-4}	1.08	4.52×10^{-4}
$10^4\times$	7.58	0.85	6.64×10^{-3}	0.96	5.99×10^{-3}
$10^5\times$	7.58×10^{1}	0.86	4.64×10^{-2}	0.89	3.91×10^{-2}

D Problem description and experimental results

D.1 Description of nonlinear PDE problems

(i) Nonlinear convection-type PDEs

Burgers' equation (1D + time)

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \gamma\frac{\partial^2 u}{\partial x^2} = 0, \quad x \in [-1, 1],\, t \in [0, 1]. \tag{49}$$

a) Varying PDE parameter:

The temporal and spatial resolutions are set to 51×129 with periodic boundary conditions. We fix the initial condition as $u(x, 0) = -\sin(\pi x)$, and vary the viscosity parameter γ within the range $[0.001, 0.05]$ with an increment of 0.001 to generate 50 tasks; 16 of them are randomly selected as training tasks, while the remaining 34 are used for testing. The model architecture consists of 4 hidden layers with 128 neurons per layer. Time block decomposition is not employed (a single time block), and the step of the fine-tuning (nonlinear) iterations is set to 6.

b) Varying initial condition:

The temporal and spatial resolutions are set to 101×101 with periodic boundary conditions. Four distinct viscosity values $\gamma \in \{0.1, 0.01, 0.001, 0.0001\}$ are considered. For each γ, the initial conditions are sampled from a Gaussian random field $\mathcal{N}(0, 625(-\Delta + 25I)^{-4})$, consistent with Wang et al. [2021]. We generate 50 training tasks and 1000 test tasks for each viscosity, and experiments are conducted independently for each γ. The model architecture consists of 4 hidden layers with 128 neurons per layer, and the nonlinear iteration step is set to 4. We evaluate two model variants: unsupervised Newton-PINet (1 time block) and hybrid Newton-PINet (4 time blocks).

Generalized Korteweg-de Vries (KdV) equation (1D + time)

$$\frac{\partial u}{\partial t} + \beta_1 u\frac{\partial u}{\partial x} + \beta_2 u^2\frac{\partial u}{\partial x} + \beta_3 u^3\frac{\partial u}{\partial x} + \delta\frac{\partial^3 u}{\partial x^3} = 0, \quad x \in [-1, 1],\, t \in [0, 1] \tag{50}$$

The temporal and spatial resolutions are set to 101×257 with periodic boundary conditions. We consider three different forms of nonlinear convection terms, corresponding to three combinations of $(\beta_1, \beta_2, \beta_3)$. For each combination, 15 tasks are randomly generated by varying δ, leading to a total of 45 tasks. Specifically:

$$\text{KdV: } (\beta_1, \beta_2, \beta_3) = (1, 0, 0), \quad \delta \in [0.032^2, 0.12^2],$$

$$M\text{-KdV: } (\beta_1, \beta_2, \beta_3) = (0, 1, 0), \quad \delta \in [0.02^2, 0.06^2],$$

$$G\text{-KdV: } (\beta_1, \beta_2, \beta_3) = (0, 0, 1), \quad \delta \in [0.08^2, 0.15^2].$$

Among them, 10 tasks are randomly chosen for training and the remaining 35 tasks for testing. The model architecture consists of 4 hidden layers with 256 neurons per layer. 6 time blocks are employed, and the nonlinear iteration step is set to 6.

Kuramoto-Sivashinsky (K-S) equation (1D + time)

$$\frac{\partial u}{\partial t}+\beta u\frac{\partial u}{\partial x}-\gamma\frac{\partial^2 u}{\partial x^2}+\sigma\frac{\partial^4 u}{\partial x^4}=0,\quad x\in[0,2\pi],\, t\in[0,1] \tag{51}$$

The simulated data have a temporal and spatial resolution of 251×509, which is downsampled to 63×129 for training and evaluation. Periodic boundary conditions are imposed. The fixed coefficients are set as $\beta=100/16$ and $\gamma=100/16^2$. The parameter σ varies within the range $[200/16^4,\, 300/16^4]$, along with varying initial conditions:

$$u(x,0)=\sum_{j=1}^{J}A_j\sin\left(\frac{2\pi l_j x}{L}+\phi_j\right),$$

where $J=5$, $L=1$, $A_j\in[-0.8,0.8]$, $l_j\in\{0,1,2,3,4\}$, and $\phi_j\in[-\pi,\pi]$. In total, 150 tasks are generated, of which 100 are used for training and 50 for testing. The model architecture consists of 4 hidden layers with 256 neurons per layer. 15 time blocks are employed, and the nonlinear iteration step is set to 4.

Lid-driven cavity (LDC) equations (2D)

The 2D steady incompressible Navier-Stokes equations in LDC problem are given by

$$\begin{cases}\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0,\\ u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+\frac{\partial p}{\partial x}-\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0,\\ u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+\frac{\partial p}{\partial y}-\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)=0,\end{cases} \tag{52}$$

where $(x,y)\in[0,1]^2$ and Re is the Reynolds number.

The simulated data are generated on a 200×200 grid and then downsampled to 51×51 for training and testing. The boundary conditions are

$$\text{top lid: } u=1,v=0;\quad \text{other walls: } u=v=0.$$

The Re ranges from 1 to 1000 (specifically 1, 5, 10, 25, 50, 80, 100, ..., 1000 with a step of 50 from 100 to 1000), yielding 25 flow conditions. We use Re = 1, 50, 200, 400, 600, 800 as the training set and the remaining cases as the test set. The model consists of 4 hidden layers with 128 neurons per layer. With three output variables (u, v, p), the resulting weight dimension of the output layer (i.e., the number of columns in matrix $\mathbf{A}$) is $128\times 4\times 3=1536$. The nonlinear iteration step is set to 8.

(ii) Nonlinear forcing-type PDEs

Convection-diffusion-reaction (CDR) equation (1D + time)

$$\frac{\partial u}{\partial t}+\alpha\frac{\partial u}{\partial x}-\gamma\frac{\partial^2 u}{\partial x^2}-\rho_1 u+\rho_2 u^2+\rho_3 u^3=0,\quad x\in[0,1],\, t\in[0,1]. \tag{53}$$

The simulated data have a temporal and spatial resolution of 201×257, which are downsampled to 101×129 for training and evaluation. Periodic boundary conditions are imposed. The parameters are set as $\alpha=1$, $\gamma\in\{0.005,0.01,0.05\}$, $\rho_1,\rho_2\in\{0,1,3,5\}$, and $\rho_3=5$. The initial conditions are generated from

$$u(x,0)=\sum_{j=1}^{J}A_j\sin(l_j x+\phi_j),$$

where $J=5$, $A_j\in[0.1,0.5]$, $l_j\in\{1,2,3,4\}$, and $\phi_j\in[-\pi,\pi]$.

In total, 200 tasks are generated, with 40 used for training and 160 for testing. The model architecture consists of 4 hidden layers with 128 neurons per layer. 4 time blocks are employed, and the nonlinear iteration step is set to 4.

Klein-Gordon equation (1D + time)

$$\frac{\partial^2 u}{\partial t^2} - \gamma \frac{\partial^2 u}{\partial x^2} + u^3 = q, \quad x \in [0,1],\, t \in [0,1] \tag{54}$$

$$\text{ICs:} \quad \frac{\partial u}{\partial t}(x,0) = 0.$$

The solution is defined as $u(x,t) = k_1 x \cos(k_2 \pi t) + k_3 (xt)^3$, which is used to derive the corresponding Dirichlet BCs, and source term q. The spatio-temporal grid resolution is set to 32×32 for meta-learning, while a finer grid of 128×128 is used for testing. The parameters vary within $\gamma, k_1, k_3 \in [0.5, 3]$ and $k_2 \in [1, 7]$. A total of 80 tasks are generated, with 16 used for meta-learning and 64 for testing. The model network consists of 4 hidden layers with 128 neurons per layer. 1 time block is employed, and the nonlinear iteration step is set to 4.

Hyperbolic heat equation (1D + time)

$$\frac{\partial u}{\partial t} - \gamma \frac{\partial^2 u}{\partial x^2} + k_1 \sinh(u) + k_2 \cosh(u) + k_3 \tanh(u) = q, \quad x \in [-1,1],\, t \in [0,1]. \tag{55}$$

The solution is defined as $u(x,t,k_1,k_2,k_3) = \sin(k_1 k_2 \pi x)\cos(k_3 \pi x)\, e^{-\pi t^2}$, which is used to derive the corresponding ICs, Dirichlet BCs, and source term q. The spatio-temporal grid resolution is set to 32×32 for meta-learning, while a finer grid of 128×128 is used for testing. The parameters vary within $\gamma \in [0.2, 3]$ and $k_1, k_2, k_3 \in [0.2, 2.5]$. A total of 80 tasks are generated, with 16 used for training and 64 for testing. The model network consists of 2 hidden layers with 450 neurons per layer. 1 time block is employed, and the nonlinear iteration step is set to 4.

Logarithmic heat equation (1D + time)

$$\frac{\partial u}{\partial t} - \gamma \frac{\partial^2 u}{\partial x^2} + k_1 u \, \log(k_2 u) + k_3 e^{k_4 u} = q, \quad x \in [-1,1],\, t \in [0,1]. \tag{56}$$

The solution is defined as $u(x,t,k_2,k_4) = \big(\sin(k_2 \pi x) + 1.5\big) e^{-k_4 \pi x^2} e^{-\pi t^2}$, which is used to derive the corresponding ICs, Dirichlet BCs, and source term q. The spatio-temporal grid resolution is set to 32×32 for meta-learning, while a finer grid of 128×128 for testing. The parameters vary within $\gamma, k_1, k_3 \in [0.5, 5]$ and $k_2, k_4 \in [0.5, 2]$. A total of 80 tasks are generated, with 16 used for meta-learning and 64 for testing. The model network consists of 2 hidden layers with 450 neurons per layer. 1 time block is employed, and the nonlinear iteration step is set to 4.

Helmholtz equation (2D)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - 100u + 10\cosh(u) = q, \quad x, y \in [0, 1.5]. \tag{57}$$

The solution is defined as $u(x,y) = 4\cos(\alpha_1 \pi x^2)\cos(\alpha_2 \pi y^2)$, which is used to derive the corresponding Dirichlet BCs, and source term q. The spatial grid resolution is set to 32×32 for meta-learning, while a finer grid of 128×128 is used for testing. The parameters vary within $\alpha_1, \alpha_2 \in [1, 3]$. A total of 80 tasks are generated, with 20 used for meta-learning and 60 for testing. The model network consists of 2 hidden layers with 450 neurons per layer. The nonlinear iteration step is set to 4.

Parametric diffusion-reaction equation (2D)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u(1 - u^2) = q, \quad x, y \in [-1, 1]. \tag{58}$$

The solution is defined as $u(x,y,a_1,a_2,w_1,w_2,w_3,w_4) = a_1 \tanh(w_1 x)\tanh(w_2 y) + a_2 \sin(w_3 x)\sin(w_4 y)$, which is used to derive the corresponding Dirichlet BCs and source term q. The spatial grid resolution is set to 32×32 for meta-learning, while a finer grid of 128×128 is used for testing. The parameters vary within $a_1, a_2 \in [0.1, 1]$ and $w_1, w_2, w_3, w_4 \in [1, 5]$. 17 tasks are generated for meta-learning and 100 for testing. The model network consists of 2 hidden layers with 450 neurons per layer. The nonlinear iteration step is set to 4.

Parametric diffusion-reaction equation (3D)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + u(1-u^2) = q, \quad x, y, z \in [-1, 1]. \tag{59}$$

The solution is defined as $u(x, y, z, a_1, a_2, w_1, w_2, w_3, w_4, w_5, w_6) = a_1 \tanh(w_1 x)\tanh(w_2 y)\tanh(w_3 z) + a_2 \sin(w_4 x)\sin(w_5 y)\sin(w_6 z)$, which is used to derive the corresponding Dirichlet boundary conditions and source term q. The spatial grid resolution is set to $32 \times 32 \times 32$ for both meta-training and testing. The PDE parameters are sampled from $a_1, a_2 \in [0.1, 1]$ and $w_1, w_2, w_3, w_4, w_5, w_6 \in [1, 5]$. 17 tasks are generated for meta-learning and 100 for testing. The network consists of two hidden layers with 450 neurons per layer, and the number of nonlinear iterations is set to 4.

The meta-learning configurations for each problem are summarized in Table 8.

Table 8: Summary of meta-learning parameter configurations on nonlinear PDEs in this study. For time-dependent PDEs, the mesh size is reported as $t \times x$, whereas for time-independent PDEs it is reported as $x \times y\,(\times z)$. "Mesh size" and "No. A rows" refer to settings in the meta-learning stage unless otherwise marked with "(test)", which denotes the inference stage. "Dim." denotes the dimension, and "No." stands for "number of". "–" indicates that the time-blocking strategy is not applicable for time-independent problems.

Problem	Mesh size (test)	Dim. $n_{\text{pde}} + n_{\text{ic}} + n_{\text{bc}}$; No. $\mathbf{A}$ rows (test)	Dim. $\boldsymbol{w}$; No. $\mathbf{A}$ columns	No. train/ test tasks	Nonlinear iteration	No. time blocks
Burgers' (1D + time)	51×129	6810	512	16/34	6	1
Generalized KdV (1D + time)	101×257	4402	1024	10/35	6	6
Kuramoto-Sivashinsky (1D + time)	63×129	649	1024	100/50	4	15
Convection-diffusion-reaction (1D + time)	101×129	3404	512	40/160	4	4
Klein-Gordon (1D + time)	32×32 (128×128)	1148 (16892)	512	16/64	4	1
Hyperbolic heat (1D + time)	32×32 (128×128)	1148 (16892)	900	16/64	4	1
Logarithmic heat (1D + time)	32×32 (128×128)	1148 (16892)	900	16/64	4	1
Helmholtz (2D)	32×32 (128×128)	1148 (16892)	900	20/60	4	–
Parametric diffusion -reaction (2D)	32×32 (128×128)	1148 (16892)	900	17/100	4	–
Lid-driven cavity (2D)	51×51	7595	1536	6/19	8	–
Parametric diffusion -reaction (3D)	32×32×32	38536	900	17/100	4	–

D.2 Data generation, error metrics, and computational setup

Datasets for Burgers, generalized KdV, K-S, and CDR problems are generated using the `Chebfun` package [Driscoll et al., 2014] with spectral Fourier discretization and a fourth-order exponential time-differencing Runge-Kutta scheme (ETDRK4) [Cox and Matthews, 2002]. Datasets for LDC problem are generated using a computational fluid dynamics (CFD) method [Chiu and Poh, 2021]. For the remaining benchmark problems, analytical solutions are available.

We evaluate the generalization accuracy of the models using both mean squared error (MSE) and relative L^2 error. In addition, for the 1D nonlinear diffusion-reaction problem, we adopt the relative

MSE metric, consistent with the baseline study [Boudec et al., 2025] for comparison (see Table 13 and Figure 13).

$$\text{MSE} = \frac{1}{n}\sum_{s=1}^{n}\left(u_s^{\text{label}} - u_s\right)^2, \tag{60}$$

$$\text{Relative } L^2 = \frac{\left\|u^{\text{label}} - u\right\|_2}{\left\|u^{\text{label}}\right\|_2}, \tag{61}$$

$$\text{Relative MSE} = \frac{\sum_{s=1}^{n}\left(u_s^{\text{label}} - u_s\right)^2}{\sum_{s=1}^{n}\left(u_s^{\text{label}}\right)^2}, \tag{62}$$

where u^{label} and u denote the ground-truth and predicted solutions for a given PDE instance, and n is the total number of collocation points, including those for the PDE residual, IC/BC constraints.

In this study, all models are implemented in the JAX framework [Bradbury et al., 2018]. For the KdV, K-S, CDR, and LDC problems, model training is parallelized across four Tesla V100 GPUs; all other models are trained on a single Tesla V100 GPU. Inference for all models is performed on a single Tesla V100 GPU.

D.3 Summary of model performance on nonlinear PDE problems

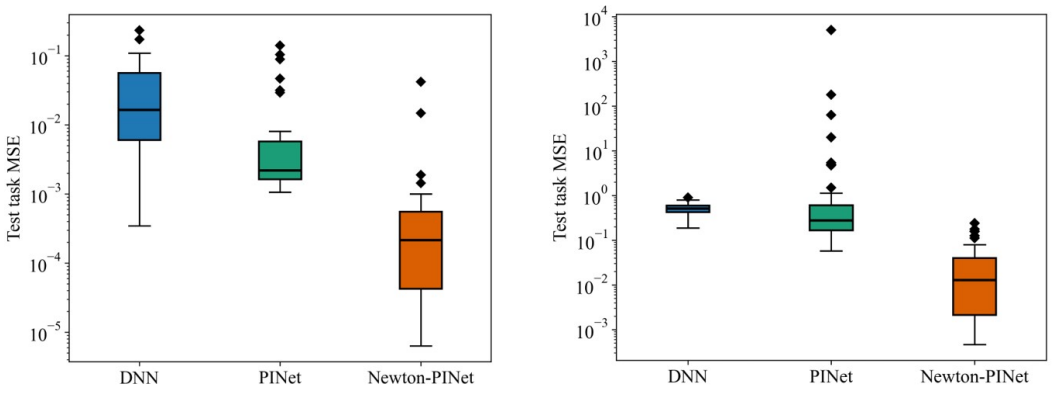
Table 2 in the main text summarizes the model performance on each problem. For each problem, besides the **Newton-PINet** model proposed in this study, two additional baseline models were tested: one that uses the Picard method, referred to as **PINet**, and a purely data-driven **DNN** baseline which incorporates PDE parameters as inputs. Note that the tables do not include comparisons with other benchmark models on problems such as the Burgers' equation with varying initial conditions; the detailed results can be found in Table 12, Table 13, and Table 15.

As shown, Newton-PINet achieves superior test accuracy across nearly all nonlinear PDEs compared to DNN and PINet, except for the Klein-Gordon equation. The Klein-Gordon equation involves second-order temporal dynamics, making the nonlinear iterations more sensitive to the quality of the initial guess. Initializing all time steps with the initial field $u(t = 0)$ may still be relatively far from the true solution, leading to a slight degradation in convergence for Newton-PINet. However, the difference in prediction accuracy is negligible (MSE: 1.88×10^{-7} vs. 2.36×10^{-7}). Notably, for hyperbolic and logarithmic heat problems, the Picard approach employed by PINet fails to converge (leading to NaN values during meta-learning), whereas Newton-PINet remains stable and achieves accurate generalization.

Prediction results of generalized KdV, K-S, and LDC problems: Due to the coexistence of nonlinear convection terms and high-order dispersion operators, solving the generalized KdV and K-S equations is particularly challenging for PINN-based models. To mitigate these difficulties, we adopt a larger number of temporal domain partitions to stabilize training and improve convergence (6 time blocks for the generalized KdV and 15 time blocks for the K-S equations). As shown in Table 2 and Figure 4, Newton-PINet achieves test errors of $\text{MSE} = 1.94 \times 10^{-3}$ on the generalized KdV, $\text{MSE} = 3.45 \times 10^{-2}$ on the K-S, and $\text{MSE} = 1.67 \times 10^{-4}$ on the LDC problem, outperforming PINet and DNN under the same configuration. Figures 5, 6, 7 show the prediction results of the Newton-PINet model on test tasks for the generalized KdV, K-S, and LDC problems, respectively.

Prediction results of nonlinear forcing-type PDEs: Figures 8, 9, 10 show the prediction results of Newton-PINet on some test cases for the convection-diffusion-reaction (CDR) and other nonlinear forcing-type PDEs described in Appendix D.1.

Model generalization to unseen boundary conditions: We first investigate the generalization capability of Newton-PINet to unseen boundary condition (BC) based on the 1D+time hyperbolic heat problem defined in Eq. (55). The model is meta-trained on 16 tasks with *Neumann* boundary conditions and subsequently evaluated on 64 unseen tasks with *Dirichlet* and mixed BC. As shown in Table 9, the Newton-PINet model can effectively generalize to these out-of-distribution boundary conditions that were not encountered during meta-training.



(a) Generalized Korteweg-de Vries (KdV) (b) Kuramoto-Sivashinsky (K-S)

Figure 4: MSE distributions across all test tasks for different models on the generalized KdV and K-S problems.

Table 9: Model generalization to unseen boundary condition (BC) for the 1D+time hyperbolic heat problem.

	Dirichlet BC	Mixed BC
Inference time (s)	0.067	0.071
MSE	$(1.42 \pm 4.54) \times 10^{-5}$	$(1.49 \pm 4.48) \times 10^{-5}$
Relative L^2	$(5.11 \pm 11.30) \times 10^{-3}$	$(5.47 \pm 11.30) \times 10^{-3}$

Model generalization to unseen irregular geometries: We further evaluate the generalization capability of Newton-PINet on unseen irregular geometries using two PDE problems, including the 1D+time hyperbolic heat equation and the 3D parametric diffusion-reaction equation.

(1) 1D+time hyperbolic heat problem

For the hyperbolic heat equation defined in Eq. (55), the model is meta-trained on regular geometry with 16 different PDE parameter configurations and evaluated on 64 unseen test cases with irregular curved geometries and simultaneously varying PDE parameters. During testing, the spatial domain is replaced by a time-dependent curved geometry with symmetric moving boundaries:

$$x_l(t) = -1 - A\sin(2\pi f t), \qquad x_u(t) = 1 + A\sin(2\pi f t), \tag{63}$$

where the deformation amplitude and frequency are randomly sampled from $A \in [0.1, 0.5]$ and $f \in [1.0, 5.5]$, respectively.

The adaptation is performed solely through physics-informed least-squares fine-tuning of the output layer, without retraining the network parameters. The prediction results are shown in Table 10 and Figure 11.

Table 10: Generalization to unseen irregular geometries and parameters for the 1D+time hyperbolic heat problem.

Metric	Value
Inference time (s)	0.074
MSE	$(1.50 \pm 3.23) \times 10^{-5}$
Relative L^2	$(6.12 \pm 11.40) \times 10^{-3}$

(2) 3D parametric diffusion-reaction problem

We further examine the generalization capability of Newton-PINet using the 3D parametric diffusion-reaction problem (Eq. (59)). The model is meta-trained on regular geometry with 17 different PDE

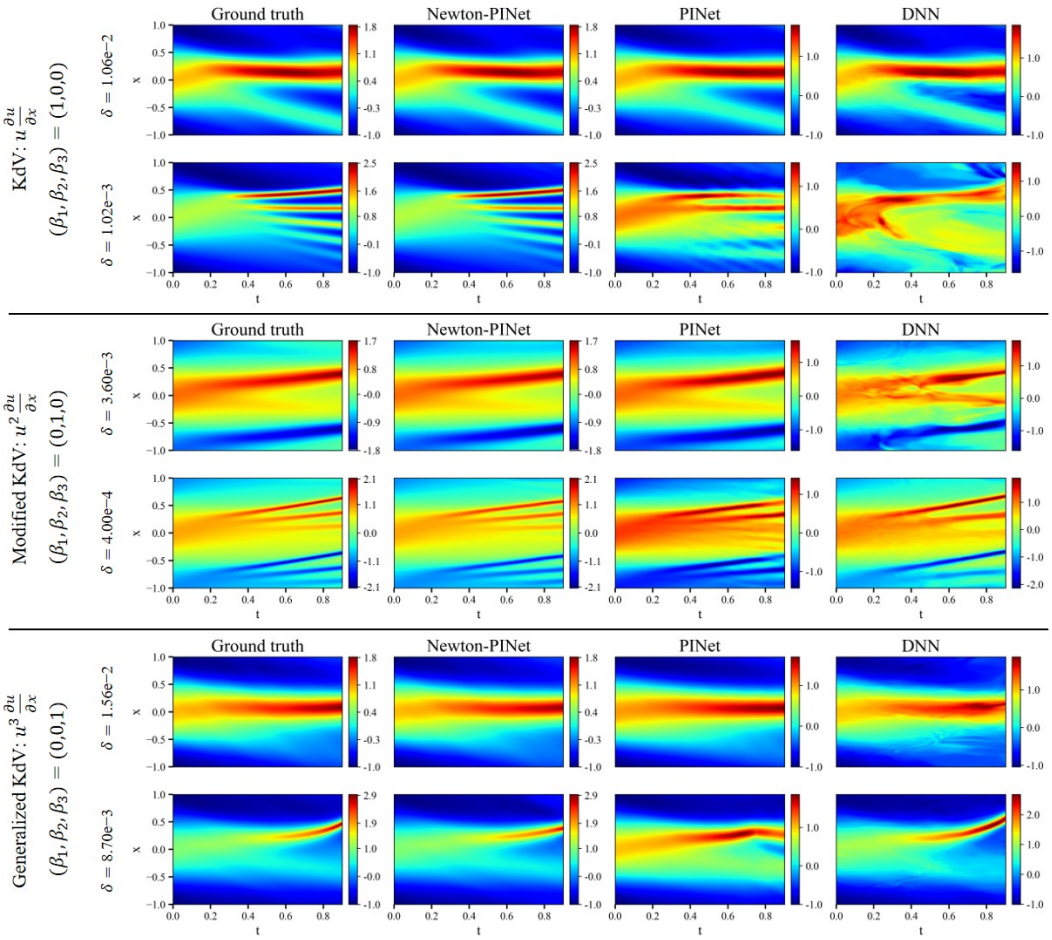


Figure 5: Prediction results of Newton-PINet for the generalized KdV problem.

parameter configurations and evaluated on 100 unseen test cases with irregular domains of diverse shapes and simultaneously varying PDE parameters. The irregular geometries are generated through a spherical-domain perturbation procedure. Specifically, sampling points are defined by radius $r \in [0.05, 1.0]$, azimuth $\theta \in [0, 2\pi)$, and polar angle $\phi \in [0.05, \pi - 0.05]$ to avoid polar singularities. A perturbation scale is then defined as

$$\text{scale} = 1.0 + A\cos(k_\theta\theta + \varphi_\theta)\sin(k_\phi\phi + \varphi_\phi), \tag{64}$$

where $A \in [0.1, 0.3]$ controls the perturbation amplitude, $k_\theta, k_\phi \in 2, 3, 4, 5$ determine the spatial frequencies, and $\varphi_\theta, \varphi_\phi$ are randomly sampled phases. The perturbed points are mapped to Cartesian coordinates according to

$$\begin{cases} x = r \cdot \text{scale} \cdot \sin(\phi)\cos(\theta), \\ y = r \cdot \text{scale} \cdot \sin(\phi)\sin(\theta), \\ z = r \cdot \text{scale} \cdot \cos(\phi). \end{cases} \tag{65}$$

The outermost shell ($r = 1.0$) of the $32 \times 32 \times 32$ grid is extracted to impose the Dirichlet boundary conditions. This procedure preserves the topology of the original domain while introducing diverse irregular boundary shapes.

The test results are summarized in Table 11, with representative predictions shown in Figure 12.

Overall, these results demonstrate that Newton-PINet can generalize to unseen boundary conditions and irregular domain geometries while maintaining high prediction accuracy and computational efficiency. Notably, the adaptation to these out-of-distribution settings requires only physics-informed fine-tuning of the output layer, without retraining the network parameters.

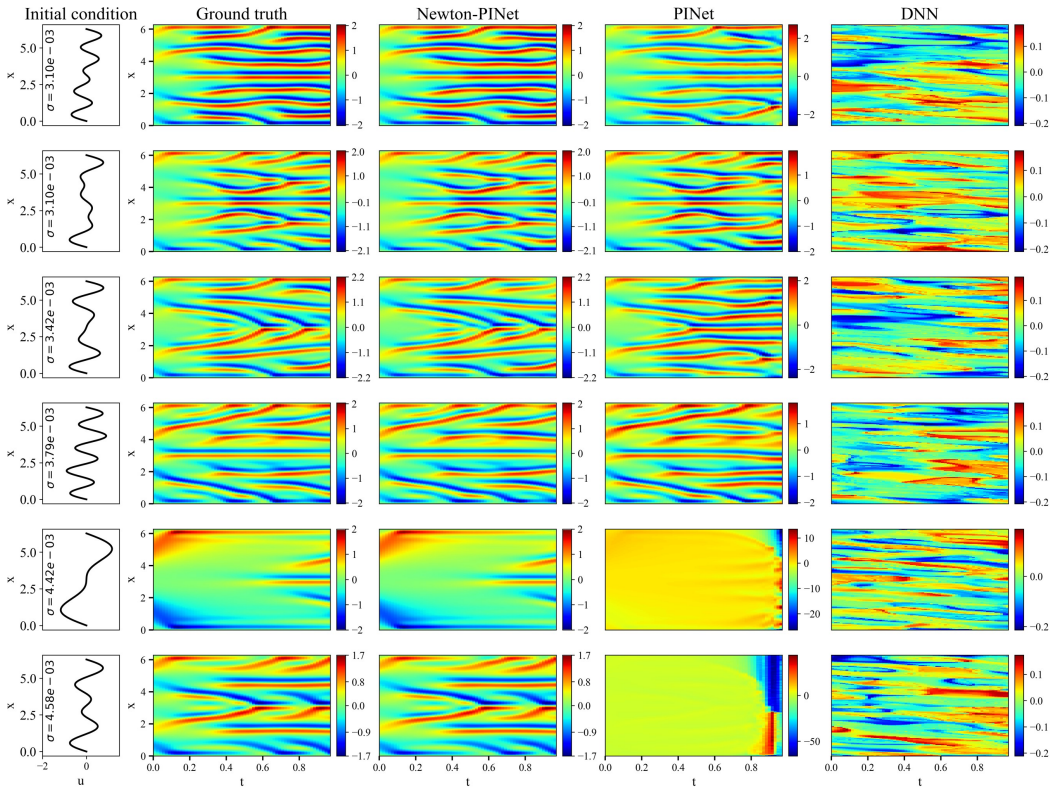


Figure 6: Prediction results of Newton-PINet for the K-S problem.

Table 11: Generalization to unseen irregular geometries and parameters for the 3D parametric diffusion-reaction problem.

Metric	Value
Inference time (s)	0.104
MSE	$(5.56 \pm 7.39) \times 10^{-6}$
Relative L^2	$(9.21 \pm 4.09) \times 10^{-3}$

E Comprehensive comparison of model performance on nonlinear benchmark problems

E.1 Model comparison under varying initial conditions on Burgers' problem

We evaluate the model on four settings with $\gamma \in \{0.1, 0.01, 0.001, 0.0001\}$ in Burgers' problem, where, for each γ, the initial conditions are sampled from a Gaussian random field $\mathcal{N}(0, 625(-\Delta + 25I)^{-4})$ to generate 50 training and 1000 test tasks. Experiments are conducted independently for each γ. We compare (i) **unsupervised Newton-PINet** (1 time block) and (ii) **hybrid Newton-PINet** (4 time blocks) against popular neural operator baselines in terms of training time and test error. The **unsupervised** operator considered is Physics-Only DeepONet (PO-DeepONet) [Wang et al., 2021]. The **supervised** operators include DeepONet [Lu et al., 2021], Transformer-DeepONet with Trunk net enhanced by Fourier coefficients (T-DeepONet-TF) [Wei et al., 2025b], and FNO [Li et al., 2020], while the **hybrid** operator is PINO (a physics-informed FNO) [Li et al., 2024]. Note that PINO uses a training loss that combines both data-driven and PDE-based physics losses. The results are summarized in Table 12. Compared to PO-DeepONet, unsupervised Newton-PINet reduces training time from 9.25h to 291s while improving test accuracy across all γ experiments (in a data-absent scenario). Compared to supervised and hybrid neural operators, including the state-of-

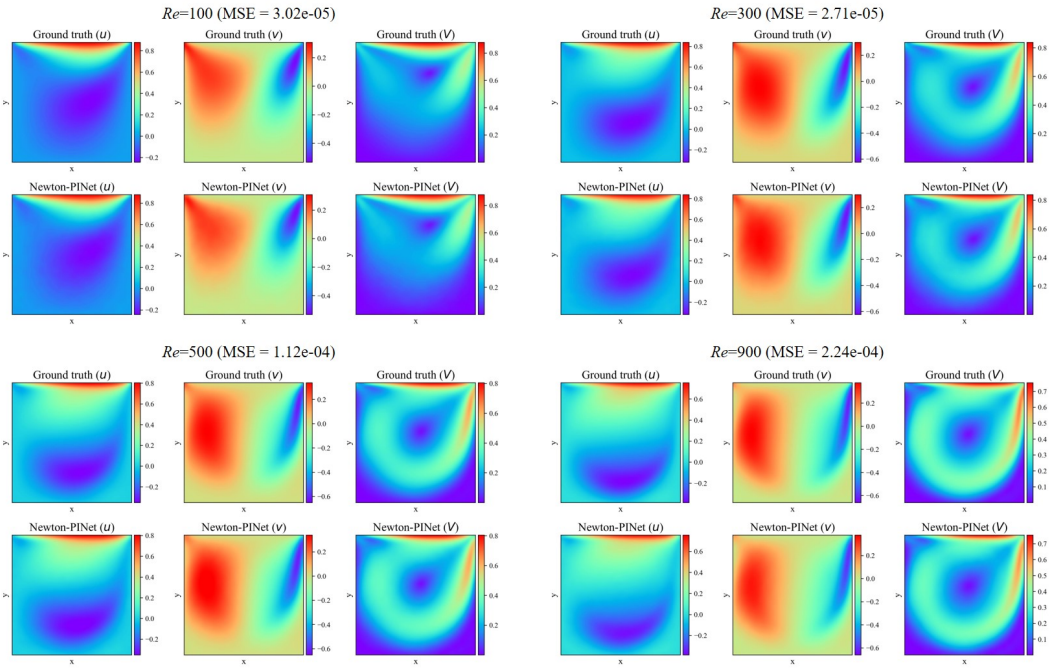


Figure 7: Prediction result of Newton-PINet for the 2D lid-driven cavity problem. Velocity magnitude is computed as $V = \sqrt{u^2 + v^2}$.

the-art T-DeepONet-TF and PINO, Newton-PINet consistently achieves superior test accuracy under $\gamma = 0.1, 0.01, 0.0001$, with significantly reduced training cost (10x less time). Our model consistently achieves higher accuracy while requiring the least training time, whether fully unsupervised (without data) or with a few labeled samples.

Table 12: Model comparison on the 1D Burgers' problem. The initial conditions for generating all the training and test data are drawn from GRF $\sim \mathcal{N}(0, 625(-\Delta + 25I)^{-4})$. The lowest errors are highlighted in bold. "–" denotes results not reported in the references.

	Model	No. training tasks	Test relative L^2 error (training time)			
			$\gamma = 0.1$	$\gamma = 0.01$	$\gamma = 0.001$	$\gamma = 0.0001$
Unsupervised	PO-DeepONet [Wang et al., 2021]	1000	–	1.38e-2 (9.25h)	2.16e-1 –	2.48e-1 –
	Newton-PINet (Ours)	50	1.33e-3 (291s)	3.51e-3 (291s)	1.37e-1 (291s)	2.01e-1 (291s)
Supervised	DeepONet [Wei et al., 2025b]	1000	–	1.17e-2 (2800s)	2.30e-1 (2620s)	2.88e-1 (2660s)
	T-DeepONet-TF [Wei et al., 2025b]	1000	–	2.08e-3 (3041s)	**9.81e-3** (2569s)	1.18e-1 (3466s)
	FNO [Li et al., 2020]	1000	1.39e-2	–	–	–
Hybrid	PINO [Li et al., 2024]	1000	–	3.80e-3 (1200s)	–	–
	Newton-PINet (Ours)	50	**1.13e-3** (133s)	**5.46e-4** (133s)	4.66e-2 (133s)	**9.15e-2** (133s)

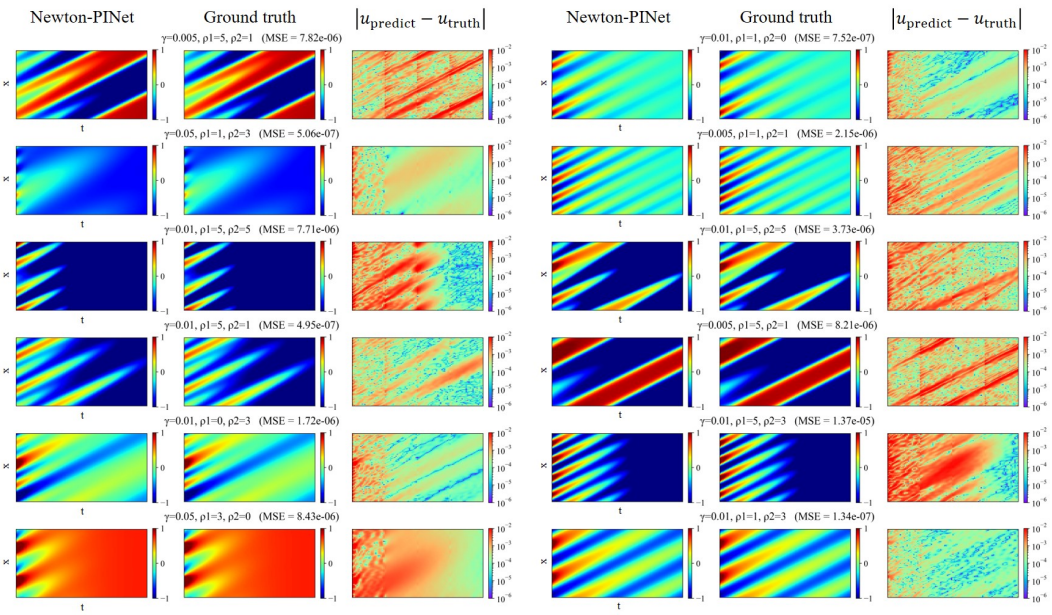


Figure 8: Prediction results of Newton-PINet for the CDR problem.

E.2 Model comparison under varying PDE parameters on a nonlinear reaction-diffusion problem

We further evaluate our method on a widely studied 1D nonlinear reaction-diffusion problem $\frac{\partial u}{\partial t} - \gamma \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0$ [Krishnapriyan et al., 2021]. The benchmark configuration and baseline model results are all taken from [Boudec et al., 2025]. The benchmark uses a fixed Gaussian initial condition, with PDE parameters sampled over $\gamma \in [1, 5]$ and $\rho \in [-5, 5]$. The baseline models are grouped into two categories: (i) **unsupervised**, including instance-wise PINNs trained with L-BFGS (PINNs+L-BFGS) or Adam+L-BFGS (PINNs-multi-opt), parametric PINNs (PPINNs) [Boudec et al., 2025] and P^{2}INNs [Cho et al., 2024] that incorporate PDE parameters as inputs, and Physics-Only DeepONet (PO-DeepONet) [Wang et al., 2021]; and (ii) **hybrid**, which combine supervised and physics-informed approaches, such as PI-DeepONet [Goswami et al., 2023], PINO [Li et al., 2024], and the physics-informed neural solver (PI-neural-solver) [Boudec et al., 2025]. These baselines were trained with 800 tasks and tested on 200 unseen tasks. It can be observed that, among the baseline models, purely physics-based unsupervised training tends to struggle with highly nonlinear problems, whereas hybrid approaches leverage large amounts of labeled data to improve representation learning, thereby enhancing test accuracy. In contrast, our hybrid **Newton-PINet** *achieves state-of-the-art performance using only 50 training tasks*, i.e., *16× fewer tasks*, and is evaluated on the same test set using a 4-time-block decomposition. For a fair comparison, both models are trained and evaluated on an NVIDIA A6000 GPU. As shown in Table 13, Newton-PINet reaches a test MSE roughly three orders of magnitude lower than the best-performing baseline (PI-neural-solver) while requiring only 301 seconds of total meta-training wall-clock time, compared with several hours for the baselines. These results highlight the *high data efficiency and computational efficiency* of Newton-PINet. Model prediction results are provided in Appendix Figure 13.

E.3 Model comparison under different GPU configurations

Following the problem setup in Appendix E.2, Table 14 further reports the model performance under different GPU configurations. It can be observed that Newton-PINet consistently outperforms PI-neural-solver on both A6000 and V100 GPUs.

In addition, a comparison of Newton-PINet on A6000 and V100 GPUs shows that training time differs across the two GPUs, while inference time remains comparable. This is likely because GPU

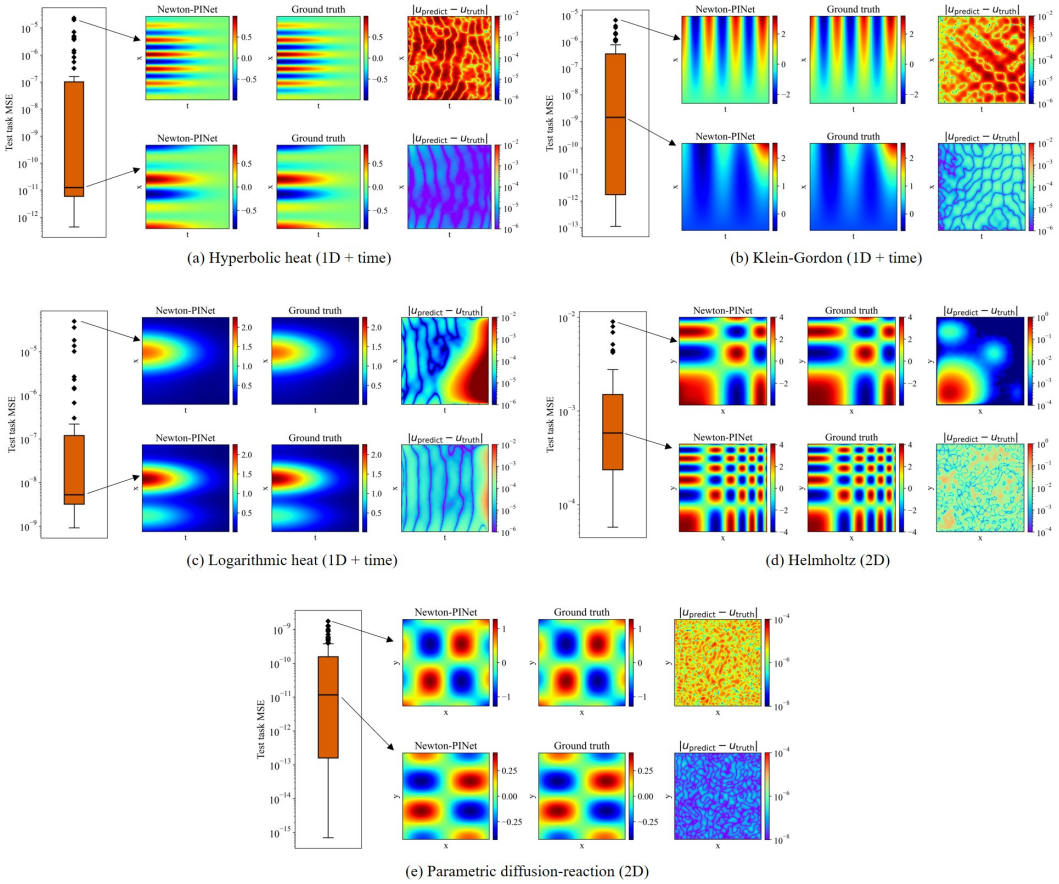


Figure 9: Prediction results of Newton-PINet for the other 5 nonlinear PDE problems.

Table 13: Model comparison on the 1D nonlinear reaction-diffusion problem. Baseline results are from Boudec et al. [2025]. Time is reported in hours (h), minutes (m), and seconds (s). Best and second-best are bold and underlined, respectively.

	Model	No. training tasks	Test relative MSE	Training time	Inference time
Unsupervised	PINNs+L-BFGS	800	6.13e-1	–	369s
	PINNs-multi-opt		7.57e-1	–	16.5s
	PPINNs		3.94e-1	4h15m	0.291s
	P^2INNs		5.69e-1	11h	0.676s
	PO-DeepONet		4.10e-1	3h30m	0.438s
Hybrid	PI-DeepONet	800	7.90e-2	3h30m	0.443s
	PINO		4.21e-4	<u>1h10m</u>	0.519s
	PI-neural-solver		<u>2.91e-4</u>	4h30m	<u>0.284s</u>
	Newton-PINet (Ours)	50	**1.72e-7**	**301s**	**0.080s**

configurations mainly affect the efficiency of gradient-based training, whereas our gradient-free Tikhonov-based adaptation is much less sensitive to such differences.

E.4 Model comparison with other meta-learning PINN methods

We further compare the performance of Newton-PINet with state-of-the-art gradient-based meta-learning PINNs [Penwarden et al., 2023] and Baldwinian-PINNs [Wong et al., 2026c]. For a fair comparison, we train an unsupervised Newton-PINet with a meta-learning loss of LSE and a hybrid

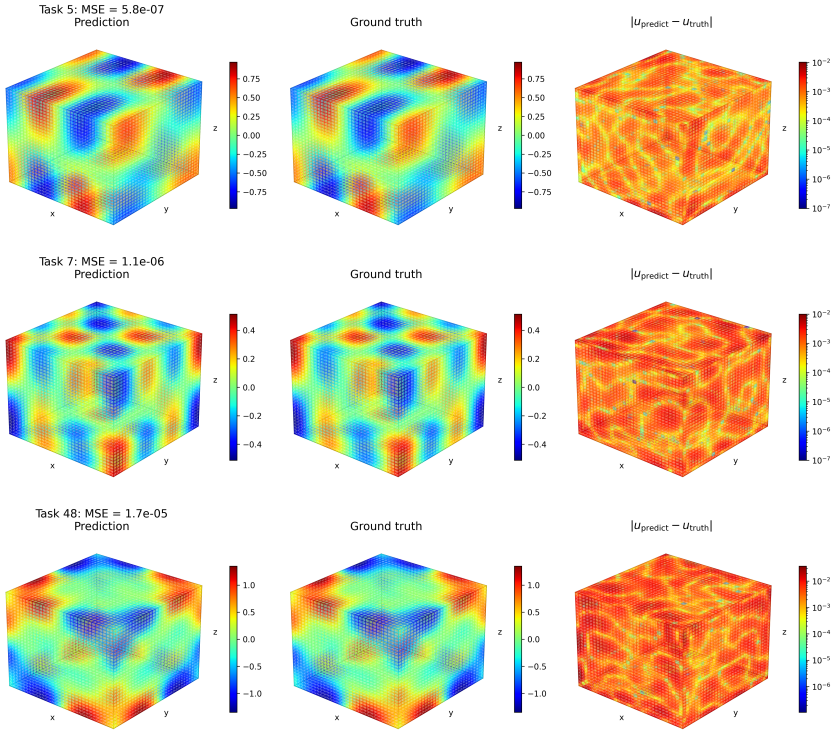


Figure 10: Prediction results of Newton-PINet for the 3D parametric diffusion-reaction problem.

Table 14: Comparison of model performance under different GPU configurations.

Model	No. training tasks	Test relative MSE	Training time	Inference time
PI-neural-solver (A6000-GPU)	800	2.91e-4	4h30m	0.284s
Newton-PINet (A6000-GPU)	50	1.72e-7	301s	0.080s
Newton-PINet (V100-GPU)	50	1.71e-7	119s	0.084s

Newton-PINet with a meta-learning loss of MSE, considering a single time-block setting. Table 15 summarizes the generalization performance and computational cost of Newton-PINet relative to these meta-learned PINN models. Compared to unsupervised gradient-based meta-PINNs, the unsupervised Newton-PINet consistently achieves lower test errors and shorter inference times. For instance, on a parametric diffusion-reaction problem, Newton-PINet exhibits approximately a 179× improvement in generalization accuracy while requiring 7000× less computational time. Against gradient-free Baldwinian-PINN, which employs Picard to handle PDE nonlinear terms, Newton-PINet which employs the Newton linearization method reduces test MSE by 2∼5 orders of magnitude across the benchmark problems and shortens single-task fine-tuning time by roughly one order of magnitude.

We also note that, in some cases, the unsupervised Newton-PINet achieves slightly higher accuracy than the hybrid Newton-PINet. This occurs for relatively simple nonlinear PDEs, where using an LSE-based meta-learning loss facilitates better convergence and generalization than MSE. Conversely, for more complex nonlinear PDEs, such as the KdV and K-S equations, adopting an MSE-based meta-learning loss leads to much better performance.

E.5 Model comparison with other neural operators on the newly introduced datasets

In the above experiments, we validated the performance of our model on several previously reported challenging datasets. We next further evaluate our method on the newly introduced and more complex PDE benchmarks in this study, including the generalized KdV, Kuramoto-Sivashinsky (K-S),

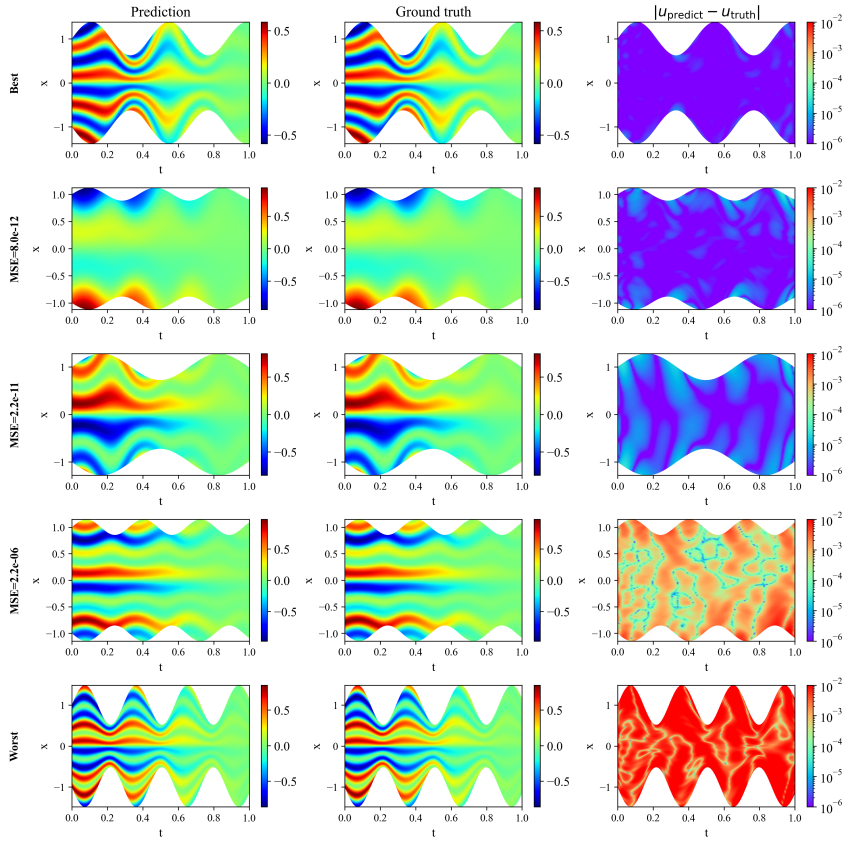


Figure 11: Prediction results of Newton-PINet for the hyperbolic heat problem across varying domain shapes and PDE parameters, with meta-training performed on regular geometry.

convection-diffusion-reaction (CDR), and the lid-driven cavity (LDC) nonlinear problems. Specifically, we compare the proposed hybrid Newton-PINet with several state-of-the-art neural operator methods, including DeepONet [Lu et al., 2021], Physics-Only DeepONet (PO-DeepONet) [Wang et al., 2021], Multi-task DeepONet (MT-DeepONet) [Kumar et al., 2025], FNO [Li et al., 2020], PINO [Li et al., 2024], and Tucker-FNO [Zhou et al., 2026]. The hyperparameter settings of these baseline models are summarized in Table 16. For each problem, all baseline models and our method are trained and evaluated using the same training and test datasets under identical NVIDIA V100 GPU configurations. The comparison results are presented in Table 17. It can be observed that our model maintains a moderate parameter scale while consistently achieving the lowest test error across all considered benchmarks. Moreover, for the generalized KdV, CDR, and LDC problems, Newton-PINet requires substantially less training time than the competing methods. For the K-S problem, however, the optimization remains relatively challenging for our model, leading to a comparatively longer training time. Overall, on these newly introduced datasets, existing state-of-the-art neural operator methods exhibit relatively poor generalization performance under limited training data settings, primarily because they lack physics-based fine-tuning mechanisms and their generalization capability heavily relies on large amounts of training data. In contrast, the proposed Newton-PINet is able to maintain high generalization accuracy even in data-sparse regimes.

E.6 Runtime comparison with classical numerical solvers

The goal is not to replace classical solvers for solving a single PDE instance, but to provide an efficient framework for *offline meta-training followed by online real-time adaptation* for parameterized PDE families. Unlike classical methods, which solve each new PDE configuration from scratch, the pretrained Newton-PINet can directly adapt to new configurations through fast inference while achieving comparable accuracy to numerical solvers.

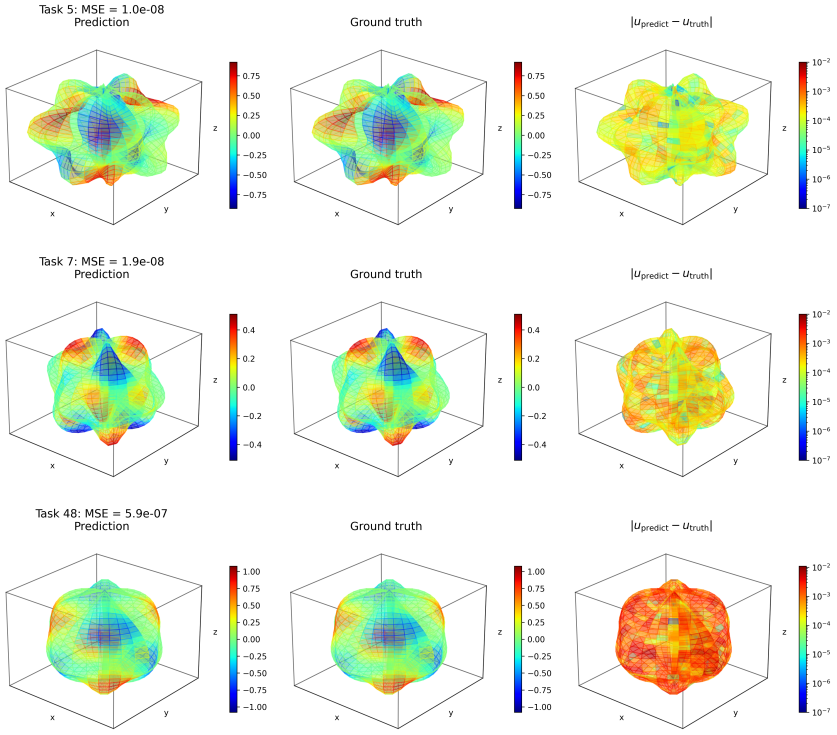


Figure 12: Prediction results of Newton-PINet for the 3D parametric diffusion-reaction equation across varying domain shapes and PDE parameters, with meta-training performed on regular geometry.

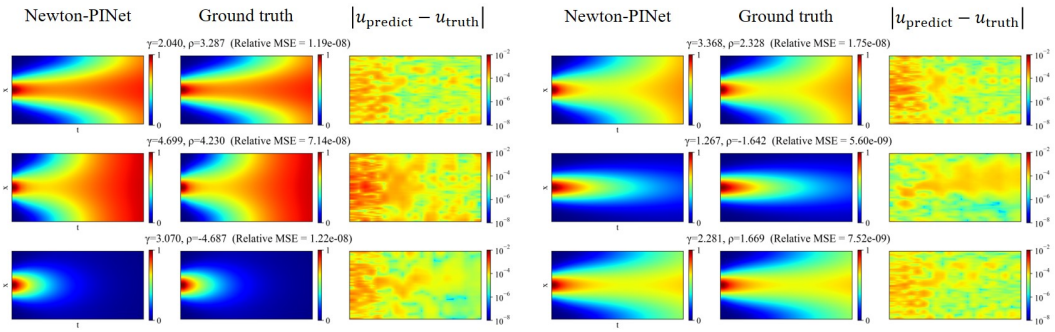


Figure 13: Prediction results of Newton-PINet for the 1D nonlinear reaction-diffusion benchmark problem.

For example, for the Burgers' equation, a standard Gauss-Newton solver requires approximately 23 s for solving a new instance from scratch, whereas Newton-PINet completes adaptation within only 0.035 s. For the CDR problem, the `Chebfun` spectral solver (MATLAB) requires approximately 8 s, compared with 0.1 s for Newton-PINet. For the 2D lid-driven cavity flow, a CFD solver requires more than 30 s (excluding meshing), while Newton-PINet achieves prediction within 0.4 s with an MSE of only 1.67×10^{-4} compared with the CFD solution.

Table 15: Model comparison on nonlinear benchmark problems. Baselines are reported from Penwarden et al. [2023], Wong et al. [2026c]. "–" denotes results not reported in the references.

Meta-learning PINNs			Nonlinear benchmark problems (number of test tasks)			
			Heat (64)	Allen-Cahn (32)	Diffusion -reaction (64)	Parametric diffusion -reaction (100)
Unsupervised	Random	Error[1]	0.0052	0.015	0.011	0.0022
		Time[2]	156	496	1073	612
	MAML	Error[1]	0.0045	–	–	–
		Time[2]	188	–	–	–
	Center	Error[1]	0.0045	0.012	0.0095	0.0018
		Time[2]	96	201	426	494
	Multitask	Error[1]	0.0048	0.012	0.009	0.0017
		Time[2]	49	120	243	431
	LMC	Error[1]	0.0049	0.011	0.0091	0.0015
		Time[2]	59	120	302	428
	RBF	Error[1]	0.0044	0.012	0.0081	0.0017
		Time[2]	35	68	280	375
	Polynomial	Error[1]	0.0046	0.012	0.0085	0.0018
		Time[2]	38	44	249	496
	Newton-PINet (Ours)	Rel. L^2	4.0e-6	4.2e-7	1.4e-4	9.5e-6
		Time[3]	0.06	0.044	0.047	0.049
Hybrid	Baldwinian-PINN	Rel. L^2	6.0e-4	9.5e-4	1.3e-4	1.9e-4
		MSE	1.3e-7	2.0e-7	1.2e-8	1.6e-8
		Time[3]	0.87	1.19	0.58	0.6
	Newton-PINet (Ours)	Rel. L^2	7.3e-6	3.7e-6	1.4e-4	1.3e-5
		MSE	1.7e-11	2.5e-12	6.8e-10	1.6e-10
		Time[3]	0.062	0.044	0.047	0.047

[1] Relative L^2 errors obtained after 500 iterations of fine-tuning using ADAM followed by L-BFGS, as described in Penwarden et al. [2023].
[2] Only the L-BFGS optimization time (s) is reported in Penwarden et al. [2023].
[3] Mean inference time per test task (s), on a Tesla V100 GPU.

F Ablation studies

F.1 Robustness of the meta-learned Tikhonov regularization parameter

In our method, the Tikhonov regularization parameter (λ_{reg}) is not manually tuned but meta-learned, enabling the model to identify a stable and task-agnostic regularization level across all training tasks. Once learned, λ_{reg} remains fixed during test-time adaptation.

To evaluate its stability, using the Burgers' problem as an example, we additionally perform per-task optimization by initializing λ_{reg} from the meta-learned value and further updating it for each test task with 300 epochs of gradient-based updates. As shown in Table 18, the meta-learned λ_{reg} remains close to the per-task optimized values across a wide range of test tasks, and their generalization performance differs only marginally. This indicates that the meta-learned hyperparameter exhibits strong stability and low sensitivity during test-time adaptation.

F.2 Effect of initial guess in nonlinear iterations on model accuracy

We conduct empirical ablations to evaluate how the initial field guess (u_{guess}) in nonlinear iterations influences model accuracy. Let's quantify the distance between u_{guess} and the ground truth solution (u_{label}) using the ℓ_2 norm ($\|u_{\text{guess}} - u_{\text{label}}\|_2$). To assess sensitivity to initialization, we test several representative choices for u_{guess}: setting u_{guess} at all time steps equal to the initial field ($u_{t=0}$), all zeros field, all ones field, and normal random field.

Table 16: Summary of baseline model architectures and hyperparameter configurations.

Model	Network architecture	Activation	Initialization	Optimizer / LR	Special components
DeepONet	Branch: 4 layers $\times$ 100 units Trunk: 4 layers $\times$ 100 units	tanh	He uniform	Adam, Cosine decay $10^{-3} \to 10^{-6}$	Learnable global bias
PO-DeepONet	Branch: 4 layers $\times$ 100 units Trunk: 4 layers $\times$ 100 units	tanh	Glorot normal	Adam, Cosine decay $10^{-3} \to 10^{-6}$	AD-based physics residual
MT-DeepONet	Branch: [100, 100] $\to$ 100 (latent) Trunk: [100, 100, 100] $\to$ 100 (latent)	ReLU Leaky ReLU	Glorot normal	Adam, Cosine decay $10^{-3} \to 10^{-6}$	Layer normalization No activation on final layer
FNO	4 spectral conv blocks width 32, modes 12×12 Projection: $32 \to 128 \to 1$	GELU	Uniform	Adam, Cosine decay $10^{-3} \to 10^{-6}$	Frequency truncation Complex multiplication Residual connections
PINO	4 spectral conv blocks width 32, modes 12×12 Projection: $32 \to 128 \to 1$	GELU	Uniform	Adam, Cosine decay $10^{-3} \to 10^{-6}$	Spectral derivative Data + physics loss Residual connections
Tucker-FNO	4 Tucker-spectral blocks width 32, modes 12, rank 32 Projection: $32 \to 128 \to 1$	GELU	Normal / Complex normal	Adam, Cosine decay $10^{-3} \to 10^{-6}$	Tucker core tensor contraction Residual connections

Table 17: Model comparison on the newly introduced benchmarks. Best and second-best results are highlighted in bold and underlined, respectively.

Problem	Model	No. model parameters	No. training / test tasks	Training time (s)	Test MSE
Generalized KdV	DeepONet	87101		642	8.24e-2
	PO-DeepONet	87101		4437	5.01e-1
	MT-DeepONet	77400		<u>588</u>	9.13e-2
	FNO	2376321	10 / 35	639	<u>6.12e-2</u>
	PINO	2376321		657	9.63e-2
	Tucker-FNO	1067137		954	9.12e-2
	Newton-PINet (Ours)	198146		**453**	**1.94e-3**
Kuramoto-Sivashinsky	DeepONet	74001		<u>916</u>	5.07
	PO-DeepONet	74001		2334	5.16e-1
	MT-DeepONet	64300		**901**	5.36
	FNO	2372129	100 / 50	934	<u>3.89e-1</u>
	PINO	2372129		1438	4.39e-1
	Tucker-FNO	1062945		1726	4.26e-1
	Newton-PINet (Ours)	198146		1014	**3.45e-2**
Convection-diffusion reaction	DeepONet	74401		1190	7.03e-1
	PO-DeepONet	74401		2587	2.18e-1
	MT-DeepONet	64700		1078	1.87e-1
	FNO	2372257	40 / 160	1572	1.55e-1
	PINO	2372257		<u>609</u>	<u>1.42e-1</u>
	Tucker-FNO	1063073		1179	2.03e-1
	Newton-PINet (Ours)	49922		**203**	**4.32e-5**
Lid-driven cavity	DeepONet	61403		<u>660</u>	1.17e-1
	PO-DeepONet	61403		2825	1.34
	MT-DeepONet	57443		681	8.44e-2
	FNO	2368259	6 / 19	1158	<u>9.98e-4</u>
	PINO	2368259		1305	1.06e-1
	Tucker-FNO	1059075		1353	3.17
	Newton-PINet (Ours)	49922		**290**	**1.67e-4**

As shown in Table 19, for the time-dependent Burgers' problem, using the u_{label} as u_{guess} leads to the lowest error, representing the convergence limit of our method. The $u_{t=0}$ strategy keeps u_{guess} close to the u_{label} (lowest non-zero distance) and achieves the best accuracy among the evaluated initialization choices. As shown in the Figure 14, the converged error starting from $u_{t=0}$ is comparable to that obtained when using u_{label} as the initial guess. For the more chaotic K-S system, attempting to solve a single time-block with the initialization of u_0 proved very challenging. This is because a simple initial condition can evolve into highly complex structures over time, causing this u_{guess} to deviate from u_{label}. To address this, we employed a 15-block temporal decomposition strategy. The block-wise inference generally makes the u_{guess} for each sub-block closer to the u_{label},

Table 18: Sensitivity of the meta-learned λ_{reg} to task adaptation in test inference. For meta-learning, λ_{reg} is learned during meta-training and kept fixed during testing. For Per-task optimization, λ_{reg} is initialized from the meta-learned model parameters and then optimized for each test task through gradient-based updates.

Method		γ (test tasks of Burgers' problem)					
		0.001	0.003	0.004	0.006	...	0.047
Meta-learning	λ_{reg}	7.57e-4					
	MSE	3.30e-5	2.07e-6	1.44e-6	4.59e-7	...	7.54e-8
Per-task optimization	λ_{reg}	2.07e-5	4.34e-4	4.81e-4	3.26e-4	...	9.79e-5
	MSE	3.21e-5	2.01e-6	1.44e-6	4.69e-7	...	7.07e-8

thereby improving the convergence of the least-squares solve. As Table 19 shows, the test errors under different initial guesses are comparable. This indicates that the temporal block decomposition mitigates the sensitivity of the least-squares solution to the choice of u_{guess}. We further tested the time-independent lid-driven cavity (LDC) problem and found that zero initialization achieves the lowest test error.

Our experiments show that for the strongly nonlinear/chaotic K-S equations, even a relatively small initialization distance (10–20) can lead to significantly worse convergence, whereas tasks like the Burgers' equation and LDC problems are much less sensitive. Crucially, our empirical observations unveil a few strategies to mitigate this sensitivity to initialization. Setting u_{guess} at all time steps equal to the initial field ($u_{t=0}$) can help provide high accuracy for time-dependent problems, while a temporal block decomposition strategy may be employed for increasingly more complex nonlinear time-dependent systems like the K-S problem.

Table 19: Effect of different initial guesses for the nonlinear iterations on test errors.

Problem	Metrics	Initial guess for the nonlinear iteration (u_{guess})				
		Ground truth	All $u_{t=0}$ (ours)	All zeros	All ones	Normal random
Burgers'	Distance	0	26.4	45.9	93.2	93.6
	MSE	1.24e-6	1.24e-6	1.24e-6	1.11e-04	4.84e-3
	Relative L^2	1.04e-3	1.04e-3	1.04e-3	9.61e-03	4.96e-2
K-S	Distance	0	3.3	13.3	23.9	24.4
	MSE	2.84e-2	2.83e-2	2.91e-2	2.91e-2	3.08e-2
	Relative L^2	1.76e-1	1.76e-1	1.79e-1	1.79e-1	1.86e-1
LDC	Distance	0	–	10.5	51.7	51.3
	MSE	1.58e-4	–	1.67e-4	1.76e-3	1.86e-4
	Relative L^2	5.54e-2	–	5.61e-2	1.32e-1	5.69e-2

F.3 Effect of the number of columns in A on inference time

The scalability of the Tikhonov solution in Newton-PINet is determined by the matrix $\mathbf{A}$. Its rows grow with the total number of PDE, IC, and BC residual samples ($n_{\text{pde}}, n_{\text{ic}}, n_{\text{bc}}$). Its column size is determined by the product of the output dimension D_o and the pre-final feature dimension (i.e., the total number of neurons in the $L-1$ hidden layers with each of width N_n due to skip connections). This results in $\mathbf{A} \in \mathbb{R}^{(n_{\text{pde}}+n_{\text{ic}}+n_{\text{bc}})\times D_o N_n(L-1)}$.

However, the Tikhonov update only involves $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{D_o N_n(L-1)\times D_o N_n(L-1)}$. We conduct experiments on the 2D steady Navier-Stokes equations for the lid-driven cavity (LDC) problem to evaluate how inference time varies with D_o. For this LDC case, $N_n = 128$ and $L = 5$ are fixed, $D_o = 3$ corresponds to predicting (u, v, p), while $D_o = 2$ and $D_o = 1$ correspond to predicting (u, v) with

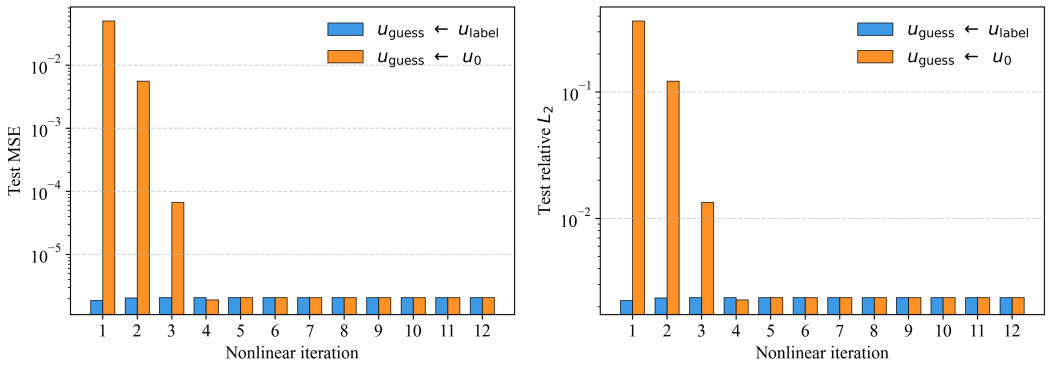


Figure 14: Convergence comparison of single-task generalization starting from the true field versus u_0 as initial guesses.

known pressure p, and u with known (p, v), respectively. Table 20 reports the inference time for different D_o.

We further examine the cost increase due to the column size by varying N_n (Table 21). The results show that increasing either D_o or N_n enlarges $\mathbf{A}$, thus increasing the cost of the Tikhonov update and task inference time.

Table 20: Inference time versus D_o, with $N_n = 128$ and $L = 5$ fixed.

	$D_o = 1$	2	3
Number of $\mathbf{A}$ columns	512	1024	1536
Inference time (s)	0.25	0.31	0.40

Table 21: Inference time versus N_n, with $D_o = 3$ and $L = 5$ fixed.

	$N_n = 64$	128	256	512
Number of $\mathbf{A}$ columns	768	1536	3072	6144
Inference time (s)	0.29	0.40	0.76	2.11

F.4 Effect of time-blocking strategies on test error

We evaluate the effect of different numbers of time blocks on the average test error taking the convection-diffusion-reaction (CDR) problem as an example. As shown in Table 22, without temporal decomposition (i.e., using a single time block), the model achieves fast inference but suffers from large errors. This is because the initial guess in the nonlinear iteration may deviate significantly from the true solution, causing poor least-squares accuracy. To address this issue, we apply a temporal domain decomposition strategy: starting from the known initial condition, the final prediction of each block serves as the initial guess for the next. This ensures that each block starts more closely to the true dynamics, enabling reliable long-horizon prediction. As shown in the table, increasing the number of blocks can substantially reduce error, though at the cost of longer inference time. For this problem, a 4-block configuration is adopted to balance the accuracy and efficiency.

F.5 Effect of initial condition weight on drift at temporal block boundaries

As can be seen in Figure 3, the CDR prediction results exhibit noticeable drift at temporal block boundaries. Upon investigation, the drift at boundaries is actually due to an under-weighted enforcement of the initial condition (IC) relative to the PDE and boundary condition (BC) in Tikhonov regularization during the meta-learning and test time. After the ablation study, as shown in Table 23,

Table 22: Ablation study for different time-blocking strategies.

Metrics	Number of blocks						
	1	2	**4**	6	10	15	20
Test MSE	3.56e-3	4.46e-3	**9.57e-6**	2.93e-6	9.07e-7	1.05e-6	5.78e-7
Test relative L^2	2.23e-1	3.96e-2	**3.56e-3**	1.80e-3	1.13e-3	1.12e-3	1.08e-3
Inference time per task (s)	3.74e-2	6.82e-2	**1.01e-1**	4.04e-1	6.77e-1	1.26	1.66

we found that increasing the IC weight (λ_{ic}) from 1 to 2 can reduce the test error. Figure 15 illustrates that this IC weight adjustment can effectively mitigate the drift at block boundaries.

Table 23: Effect of initial condition (IC) weight in the Tikhonov regularization on test errors.

Errors	Weight parameter for initial condition (λ_{ic})				
	1	1.5	2	5	10
Test MSE	9.57e-6	6.56e-6	5.75e-6	5.77e-6	6.46e-6
Test relative L^2	3.56e-3	2.89e-3	2.67e-3	2.67e-3	2.80e-3

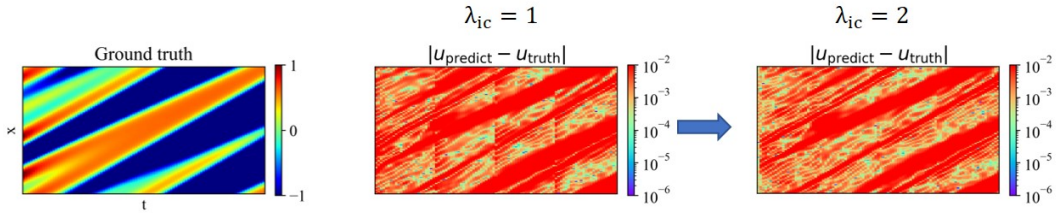


Figure 15: Mitigation of drift at block boundaries via adjustment of the initial condition (IC) weight in the Tikhonov regularization (4-time-block for CDR inference).

F.6 Model performance under other types of boundary conditions

As shown in Table 24, we test the model performance for Dirichlet, Neumann, and mixed BCs on the hyperbolic heat and the Klein-Gordon problems. As shown in Table 25, we also evaluate the model for discontinuous BCs on the 2D lid-driven cavity Navier-Stokes problem, where the velocity field exhibits corner discontinuities at the moving lid. These experiments demonstrate that the proposed framework is both theoretically and practically applicable to a broad range of BC types, beyond periodic and Dirichlet settings.

F.7 Ablation studies on model architecture

Newton-PINet employs a skip-connected multi-layer architecture with Tikhonov regularization, Sinusoidal feature embeddings at the input, and $\sin(\cdot)$ activations in all hidden layers. In this section, we perform ablation studies to examine the impact of these design choices and other hyperparameters. All test errors in the following tables are aggregated from 5 individual runs.

Effect of network depth: We first study the impact of network depth using the Burgers' and CDR problems, with and without skip connections (Table 26). For comparison, Sinusoidal feature embeddings are always included at the input, and all hidden layers adopt $\sin(\cdot)$ activations. The number of neurons per layer is set to 512, 256, 170, and 128 for depths of 1∼4, respectively, so that the last-layer width remains around 512 when skip connections are included. The results show that

Table 24: Model performance under different boundary conditions.

Problem	Dirichlet			Neumann			Mixed		
	Time[1]	MSE	Rel. L^2	Time[1]	MSE	Rel. L^2	Time[1]	MSE	Rel. L^2
Heat	0.067	1.69e-6	1.76e-3	0.071	1.80e-5	5.84e-3	0.086	5.35e-5	1.04e-2
Klein-Gordon	0.064	2.36e-7	3.86e-4	0.084	2.34e-6	1.16e-3	0.082	4.05e-6	1.13e-3

1. Inference time per test task (s)

Table 25: Model performance for 2D lid-driven cavity (LDC) under discontinuous boundary conditions.

Problem	Inference time per test task (s)	Test MSE	Test Relative L^2
2D LDC	0.40	2.30e-4	6.04e-2

shallow networks (1 layer) fail to achieve satisfactory accuracy in both problems. Increasing depth without skip connections often deteriorates generalization. In contrast, with skip connections, the model consistently maintains high accuracy across all depths, confirming the stabilizing effect of skip connections and their robustness to depth variations. This property greatly simplifies hyperparameter tuning compared to standard PINNs.

Table 26: Test relative L^2 errors of model with (w/) and without (w/o) skip connections under different network depths.

Problem	Skip connections	Network depth			
		1 layer	2 layers	3 layers	4 layers
Burgers'	w/o	1.04e-2 ±1.22e-2	3.75e-4 ±1.50e-3	4.77e-5 ±1.20e-4	5.02e-5 ±1.05e-4
	w/	1.04e-2 ±1.22e-2	1.43e-5 ±7.21e-5	1.93e-6 ±9.24e-6	1.79e-6 ±7.98e-6
CDR	w/o	4.70e-3 ±1.62e-3	7.51e-5 ±7.46e-5	6.80e-4 ±7.30e-4	2.12e-3 ±1.97e-3
	w/	4.70e-3 ±1.62e-3	5.43e-5 ±4.58e-5	5.64e-5 ±5.88e-5	4.32e-5 ±4.18e-5

Effect of network width: Next, we fix the depth to 4 layers and compare the effect of different per-layer widths, with and without skip connections (Table 27). Across both Burgers' and CDR problems, incorporating skip connections consistently improves accuracy under almost all width settings. An exception occurs for the case where all hidden layers have 256 neurons, where skip connections slightly degrade performance—likely due to the substantially increased final-layer width, which makes the Tikhonov regularization computation numerically more challenging. Overall, skip connections improve robustness to width variations.

Effect of mesh resolution: We then examine performance under different mesh resolutions, with consistent resolutions applied during meta-training and testing (Table 28). With skip connections, the model exhibits stronger adaptability to resolution changes, particularly in the CDR problem. Notably, different resolutions alter the size of the matrices, which can influence numerical stability. Nevertheless, skip connections help preserve accuracy under these settings.

Effect of activation functions and Sinusoidal embeddings: We further compare activation functions under varying depths, with skip connections included (Table 29). For $\tanh(\cdot)$ and Gaussian activations, Sinusoidal embeddings are not used, while for $\sin(\cdot)$ activations they are applied at

Table 27: Test relative L^2 errors with (w/) and without (w/o) skip connections under different network widths (from top to bottom, corresponding to layer 1$\sim$4).

Problem	Skip connections	Number of nodes in each layer						
		256 256 256 256	128 128 128 128	64 64 64 64	32 32 32 32	16 16 16 16	16 32 64 128	128 64 32 16
Burgers'	w/o	9.13e-6 ±1.87e-5	5.02e-5 ±1.05e-4	1.01e-3 ±2.06e-3	1.24e-3 ±2.12e-3	1.77e-3 ±2.38e-3	4.87e-4 ±8.03e-4	1.14e-3 ±1.95e-3
	w/	2.19e-6 ±7.43e-6	1.79e-6 ±7.98e-6	4.12e-6 ±1.44e-5	8.14e-5 ±1.11e-4	2.03e-3 ±2.78e-3	3.17e-4 ±5.66e-4	5.41e-6 ±1.66e-5
CDR	w/o	3.88e-4 ±3.44e-4	2.12e-3 ±1.97e-3	4.80e-3 ±3.79e-3	5.98e-2 ±8.14e-2	1.05e-1 ±6.29e-2	1.34e-3 ±8.42e-4	1.24e-1 ±7.24e-2
	w/	1.10e-3 ±2.20e-3	4.32e-5 ±4.18e-5	2.25e-4 ±2.14e-4	7.73e-4 ±4.77e-4	6.48e-3 ±8.32e-3	2.02e-4 ±1.82e-4	4.55e-4 ±3.47e-4

Table 28: Test relative L^2 errors with (w/) and without (w/o) skip connections under different mesh resolutions

Problem	Skip connections	Mesh resolution (t, x)				
		11×29	13×33	17×43	26×65	51×129
Burgers'	w/o	4.17e-5 ±4.57e-5	7.19e-6 ±1.52e-5	2.23e-5 ±2.30e-5	1.27e-5 ±1.74e-5	5.02e-5 ±1.05e-4
	w/	2.57e-4 ±7.24e-4	5.54e-6 ±1.62e-5	2.80e-5 ±4.61e-6	1.49e-6 ±7.00e-6	1.79e-6 ±7.98e-6
		11×29	21×49	41×69	71×99	101×129
CDR	w/o	6.34e-2 ±9.09e-2	3.08e-3 ±2.42e-3	2.26e-3 ±1.78e-3	1.51e-3 ±1.16e-3	2.12e-3 ±1.97e-3
	w/	3.72e-2 ±3.85e-2	3.27e-3 ±4.55e-3	4.03e-5 ±1.44e-5	3.36e-5 ±3.13e-5	4.32e-5 ±4.18e-5

the input. The combination of Sinusoidal embeddings with $\sin(\cdot)$ consistently outperforms other activations across different depths, highlighting the effectiveness of this design choice for nonlinear PDE learning.

Effect of weight initialization and learning rate: During meta-learning, we update the hidden-layer weights ($\tilde{\boldsymbol{w}}$) and other learning parameters via gradient descent. Here, we investigate the impact of weight initialization schemes and learning rate (η) settings on the model generalization (Table 30). Results indicate that He initialization together with a smaller initial learning rate leads to the most stable and accurate generalization performance, whereas larger learning rates often cause unstable training or accuracy degradation.

In summary: Newton-PINet benefits significantly from skip connections, which enhance robustness to depth, width, and mesh resolution, thereby reducing the need for extensive hyperparameter tuning. Sinusoidal feature embeddings with $\sin(\cdot)$ activations further boost accuracy across depths, and careful initialization and learning rate selection improve stability. Collectively, these results explain why Newton-PINet maintains reliable generalization across diverse nonlinear PDE problems.

Table 29: Test relative L^2 errors of models with different activation functions under varying network depths

Problem	Activation function	Network depth			
		1 layer	2 layers	3 layers	4 layers
Burgers'	tanh	1.07e-2 ±1.30e-2	1.30e-4 ±2.42e-4	6.39e-4 ±2.54e-3	7.13e-5 ±1.60e-4
	Gaussian	1.14e-2 ±1.31e-2	4.64e-3 ±8.95e-3	8.27e-6 ±3.06e-5	3.28e-6 ±1.26e-5
	sin	1.04e-2 ±1.22e-2	1.43e-5 ±7.21e-5	1.93e-6 ±9.24e-6	1.79e-6 ±7.98e-6
CDR	tanh	3.94e-2 ±1.33e-2	1.84e-3 ±2.18e-3	6.09e-3 ±1.16e-2	4.16e-2 ±8.27e-2
	Gaussian	3.60e-2 ±2.11e-2	5.62e-3 ±5.25e-3	6.62e-4 ±7.16e-4	2.57e-4 ±2.01e-4
	sin	4.70e-3 ±1.62e-3	5.43e-5 ±4.58e-5	5.64e-5 ±5.88e-5	4.32e-5 ±4.18e-5

Table 30: Test relative L^2 errors under different weight initialization methods and initial learning rates

Problem	Weight initialization	Initial learning rate		
		0.0005	0.005	0.05
Burgers'	Xavier	7.57e-5 ±1.62e-4	2.83e-6 ±9.33e-6	2.09e-3 ±5.54e-3
	He	1.57e-5 ±7.89e-5	1.79e-6 ±7.98e-6	2.69e-3 ±4.93e-3
CDR	Xavier	2.62e-2 ±2.17e-2	5.79e-4 ±4.69e-4	1.11e-3 ±2.20e-3
	He	2.78e-4 ±3.27e-4	4.32e-5 ±4.18e-5	1.68e-2 ±3.20e-2

G Practical scaling discussion of Newton-PINet

G.1 Effectiveness of Tikhonov regularization

The scalability of the Tikhonov solution is determined by the matrix $\mathbf{A}$. Its rows grow with the total number of PDE, IC, and BC residual samples. Its column size is determined by the product of the pre-final feature dimension and the output dimension (D_o). The pre-final feature dimension equals the total number of neurons in the $L-1$ hidden layers (each of width N_n) due to the skip connections. This results in $\mathbf{A} \in \mathbb{R}^{(n_{\text{pde}}+n_{\text{ic}}+n_{\text{bc}})\times D_o N_n (L-1)}$.

For 2D/3D or multi-physics systems (e.g., Navier-Stokes), the increase in the number of PDE equations requires more residual sampling points, while a higher output dimensionality enlarges the number of columns, making $\mathbf{A}$ larger and more challenging to solve, particularly during Newton iterations for nonlinear problems.

Our strategy is to moderately control the pre-final feature dimension, for example, by decreasing the number of neurons in each layer, thereby reducing the number of columns of $\mathbf{A}$ and ensuring that the least-squares problem remains reasonably overdetermined. Since the Tikhonov update $\boldsymbol{w} = (\lambda_{\text{reg}}\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ involves only $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{D_o N_n (L-1)\times D_o N_n (L-1)}$, its computational cost depends primarily on the feature dimension, as opposed to the potentially very large number of residual samples in high-dimensional or multiphysics domains. This keeps the Tikhonov solver tractable as long as the feature dimension is properly controlled. In our experiments, the Newton-PINet achieves high accuracy without requiring a large pre-final feature dimension (i.e., no. features

$\ll$ no. collocation points). Appendix F.3 investigates the effect of the column size of $\mathbf{A}$ on task inference time. The results show that increasing either the output dimension or the number of network width enlarges $\mathbf{A}$, thereby increasing the cost of the Tikhonov update and the task inference time.

Although the computational cost of the Tikhonov update can be effectively controlled, the cost of constructing $\mathbf{A}$ may become considerable and dominate the overall inference time when the number of residual samples is large (e.g., in high-dimensional systems). Nevertheless, even in such cases, the proposed gradient-free adaptation can be more efficient than gradient-based fine-tuning in PINNs.

G.2 Effectiveness for solving parametric PDEs

Our model is well suited for solving parametric PDEs, as the number of residual samples need not be large and tasks can be efficiently mini-batched. The parameters define different tasks and can be large but they are not directly used as network inputs. Instead, our model learns shared representations across tasks. During training, we avoid sampling residual points across parameter inputs as in P^{2}INNs [Cho et al., 2024], and instead construct $\mathbf{A}$ directly from spatiotemporal points for least-squares adaptation. So, the number of residual samples depends only on the spatiotemporal dimension, not the parameter dimension, which avoids unnecessary growth in the size of $\mathbf{A}$. Table 15 demonstrates that the model achieves superior performance on a parametric PDE problem compared to other meta-learning PINN methods.

G.3 Grid-free nature and resolution adaptability

Similar to PINNs and neural operators, our model is grid-free: no mesh construction is required; collocation points can be sampled arbitrarily, and the trained model can give predictions at arbitrary locations, even on finer grids. For example, in the five benchmark cases (Tables 2 and 8), trained on a coarse 32×32 grid, the model achieves accurate zero-shot generalization on a 128×128 grid.

G.4 Time-blocking strategy

In addition to the matrix size, the scalability of our model for nonlinear problems also depends on the quality of the initial guess for the iterative updates. Similar to the theoretical behavior of the Newton method, if the initial guess deviates too far from the true solution, more Newton iterations are required to achieve reasonable convergence. This challenge becomes particularly severe for time-dependent chaotic systems, such as the Kuramoto-Sivashinsky (K-S) equation, where even simple initial conditions can quickly evolve into highly complex spatiotemporal patterns. Consequently, it is difficult to obtain an initial guess close to the solution. We can adopt the temporal domain decomposition strategy: starting from the known initial condition, the final prediction of a previous block is used as the initial guess for the next block. This ensures that each block starts more closely to the true dynamics, improving the stability and accuracy of the least-squares updates without using large Newton iterations, and enabling reliable long-horizon prediction.

For complex time-dependent PDEs, temporal domain decomposition is a practical and effective strategy across both AI-based and classical methods. For PINNs methods, training PINNs over the entire spatiotemporal domain in a single-shot usually fails. As noted in prior work [Wang et al., 2023], gradient-based training tends to fit later-time dynamics before resolving initial conditions, which can lead to premature convergence (and incorrect solutions), especially for chaotic systems such as K-S. Consequently, many recent PINNs adopt temporal decomposition, typically at the expenses of high computational cost. Separately, many neural PDE solvers (e.g., neural operators) [Hou et al., 2026, Han et al., 2022] also rely on autoregressive (one-step) inference for long-time prediction. This strategy is consistent with classical numerical methods, where solutions are advanced sequentially in time to ensure stability.

G.5 Low-rank strategy

Although the task adaptation step is formulated as a least-squares problem, the matrix $\mathbf{A}$ in our method is not sparse in the traditional numerical PDE solver sense. Because the features are generated by a multilayer neural representation with skip connections, each row of $\mathbf{A}$ contains dense network features rather than the typical discretization-induced sparsity and structure. As a consequence, the rank and conditioning of our least-squares system is determined jointly by (i) the density and

distribution of collocation points, (ii) the Tikhonov parameter, and more importantly (iii) the quantity and expressiveness of hidden-layer weights that shape the feature space. **In this instance, the practical scaling for large collocation sets may derive more from the fact that one can still control the cost of the solve through meta-learning a better (yet minimal) set of pre-final layer features.** For this reason, low-rank or iterative solvers designed for large sparse PDE matrices (e.g., LU decomposition or Krylov solvers) may not offer clear advantages in our setting. We experimented with SVD-based solvers, LU, and Cholesky factorizations; none achieved better accuracy-stability trade-offs than the Tikhonov formulation used in our method. A more comprehensive investigation of scalable low-rank or iterative solvers tailored to dense neural-feature matrices is an interesting direction for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state that this work advances the meta-learning of PINNs toward data-efficient, fast, and generalizable physics solvers.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We analyze the model's computational complexity and discuss its limitations, failure modes, and future research directions in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Appendix C provides theoretical and experimental evidence showing that Newton linearization in the Tikhonov-regularized PINN achieves near-quadratic convergence.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix D details the data generation process, computational configurations, and experimental settings to ensure reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The Github link to the code and datasets is provided in the Abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide detailed descriptions of the experimental settings and implementations in Section 4 and Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We evaluate prediction errors using MSE, relative L^2, and relative MSE. For public benchmarks, we use the same metrics as prior work for fair comparison (e.g., Tables 12, 13, and 15). We report test error boxplots (Figs. 3, 4, and 9) and mean $\pm$ standard deviation over five runs for ablation studies (Tables 26, 27, 28, 29 and 30).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detail the computational hardware configuration in Appendix D.2, and summarize the experimental setups and runtimes for all cases in Appendix D.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work complies with the NeurIPS Code of Ethics. It introduces a generalizable meta-learning PINN framework for solving nonlinear PDEs. All experiments use synthetic data only, with no real-world or personal information involved. The method poses no foreseeable risk of misuse, is reproducible using public benchmarks and standard tools, and can be trained efficiently on standard hardware.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This work advances scientific understanding by introducing a generalizable meta-learning PINN for solving nonlinear PDEs commonly arising in scientific and engineering applications. As such, no direct societal impact is expected.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work advances scientific understanding by introducing a generalizable meta-learning PINN for solving nonlinear PDEs. Therefore, any misuse can be mostly excluded.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Libraries such as the Chebfun package and the JAX framework are appropriately referenced in the paper.

    Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The GitHub repository containing the code and datasets is linked in the Abstract. A Readme file is provided in the repository to describe the model's contributions and provide configuration instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM was used only for writing, editing, and formatting purposes.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.