DEEP COUPLING LEARNING FOR SOLVING PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-Informed Neural Networks (PINNs) represent a significant advancement in computational methods for solving partial differential equations (PDEs). However, the adoption of deeper neural network architectures presents significant challenges, as they struggle to address differential-related complications that arise during the computation of derivatives over the input of PINNs. These complications extend beyond traditional vanishing and exploding gradients to include vanishing and exploding differentials, with both phenomena becoming more severe as networks grow deeper. By examining the computation graph of derivatives in deep neural networks, we identify key bottlenecks causing numerical instabilities in deep architectures. In response, we introduce a novel approach that utilizes Coupling Layers with carefully regulated spectral norms of Jacobian matrices to stabilize and facilitate deep PINN training, effectively addressing differential-related challenges and improving model stability. Our proposed architecture successfully mitigates the fundamental constraints of deeper PINNs while maximizing their capabilities through consistent differential propagation. Comprehensive evaluations show that our approach surpasses conventional shallow PINN methods and alternative deep PINN designs across a range of challenging problems, particularly in cases featuring high-frequency solution components.

1 INTRODUCTION

Neural networks have revolutionized the field of scientific computing by offering a data-driven approach to solving partial differential equations (PDEs) that govern physical systems. Unlike traditional numerical methods such as finite element analysis (FEA) (Zienkiewicz et al., 2013) and finite difference methods (FDM) (LeVeque, 2007), which often require domain discretization and become computationally expensive for high-dimensional problems. In contrast, neural networks provide a scalable and mesh-free alternative (Lagaris et al., 1998; Han et al., 2018). Physics-Informed Neural Networks(PINNs) (Raissi et al., 2019; Karniadakis et al., 2021) have emerged as a promising framework, embedding physical laws directly into the loss function of neural networks. This integration enables PINNs to tackle both forward and inverse problems without extensive labeled datasets, making them particularly effective across diverse applications including fluid dynamics (Mao et al., 2020), heat transfer (Cai et al., 2021), and quantum mechanics (Raissi et al., 2019). When compared to traditional numerical methods, PINNs demonstrate notable strengths in managing irregular geometries, scaling to high-dimensional problems, and naturally incorporating observational data (Wang et al., 2021a). Nevertheless, despite their promising capabilities, PINNs continue to struggle with achieving comparable accuracy and reliability to established numerical methods, especially when confronted with complex geometrical structures or solutions with high-frequency components (Sirignano and Spiliopoulos, 2018; E and Yu, 2018).

Specifically, high-frequency problems, which are common in wave propagation, turbulence, and oscillatory systems, pose a particular difficulty for neural networks due to their limited ability to capture fine-scale features (Tancik et al., 2020; Krishnapriyan et al., 2021). While traditional numerical methods can handle such problems by refining the computational mesh, PINNs often struggle to converge to accurate solutions without specialized architectural modifications (Raissi et al., 2019; Wang et al., 2021a). This limitation has led to a growing interest in developing techniques to enhance the ability of PINNs to model high-frequency phenomena, which is critical for their broader adoption in applications (Karniadakis et al., 2021; Sirignano and Spiliopoulos, 2018).

Current approaches to addressing high-frequency problems in PINNs predominantly rely on frequency-aware components. For instance, Fourier feature embeddings (Tancik et al., 2020) and sinusoidal activation functions (Sitzmann et al., 2020) have been shown to improve the performance of PINNs on high-frequency tasks. While these training methods are effective for high-frequency PDEs, they often require careful tuning of hyperparameters, such as the frequency range or the scale of the embedding, which can limit their generalization capabilities (Wang et al., 2022a; Glorot and Bengio, 2010). These approaches do not fully exploit the potential of deep architectures, which offer great flexibility in expressing functions with complex frequency in Fourier space.

Here, we hypothesize that a sufficiently deep PINN can accurately solve high-frequency, challenging PDE problems. Though this hypothesis is empirically supported by our experiments (see Section 4), training deeper PINNs is not straightforward. Unlike conventional neural networks, which optimize a loss function based solely on model outputs, PINNs must minimize a composite loss that incorporates both the direct output and partial derivatives derived from the governing equations. This dual objective makes PINNs particularly susceptible to gradient-related issues, such as vanishing gradients, during training (Glorot and Bengio, 2010; He et al., 2015). For instance, backpropagating first-order derivatives through a stacked Multilayer Perceptron (MLP) in PINNs involves repeated multiplication of layer-wise Jacobian matrices, which can result in exponentially diminishing matrix elements and impede training convergence (Wang et al., 2021a; Lu et al., 2021).

While ResNets (He et al., 2015) have been widely used in deep learning to alleviate vanishing gradients via skip connections, their application to PINNs has yielded inconsistent outcomes. Wang et al. (2024a) found that ResNet-based PINNs tend to accumulate errors as network depth increases, especially in problems demanding high precision (Wang et al., 2024a). Thus, there is a clear need for alternative architectural innovations that enable deeper PINN models to achieve high accuracy.

In this work, we introduce a novel network architecture that preserves coherent derivation structures, enabling the training of deep Physics-Informed Neural Networks (PINNs) with high precision. At its core is a **Coupling Block** framework, a concept adapted from architectural designs in normalizing flows (Dinh et al., 2014; 2017). Our **key contributions** are summarized as follows:

1. **Empirical Validation of Depth in PINNs**: Through extensive experiments, we demonstrate that increasing the network depth of PINNs is critical to solving complex physical PDE problems. Deeper architectures offer the expressive capacity required to capture intricate solution features that shallow PINNs fail to represent accurately.
2. **Coupling Layers for Stable Differentiation**: We introduce a novel approach that ensures stable differentiation in deep PINNs via spectral norm regularization of Jacobian matrices and dynamic scaling control in the output layer. As demonstrated in Figure 2, our Coupled-Net architecture effectively mitigates both vanishing gradients — common in MLP-based PINNs — and gradient explosion issues typical of ResNet-based implementations.
3. **Solving PDEs with Diverse Characteristics**: Unlike existing methods that often require problem-specific tuning, such as Fourier features for high-frequency PDEs, our model generalizes robustly across PDEs with varying characteristics. It achieves this without manual hyperparameter adjustments, offering a versatile and practical solution for a broad range of physical systems.

2 PRELIMINARY

2.1 PHYSICS-INFORMED NEURAL NETWORKS

A typical PDE problem consists of two components: the governing equation and boundary conditions. A general PDE defined over a domain Ω with boundary $\partial\Omega$ is:

$$\mathcal{N}[u(\mathbf{x})] = \mathbf{0}, \quad \text{for } \mathbf{x} \in \Omega, \quad \text{and} \quad \mathcal{B}[u(\mathbf{x})] = \mathbf{0}, \quad \text{for } \mathbf{x} \in \partial\Omega, \tag{1}$$

where $\mathcal{N}$ is a series of differential operators, $\mathbf{x}$ represents spatial coordinates, u is an unknown function and $\mathcal{B}$ is a series of constraints at the domain boundaries. PINNs use a neural network $u(\mathbf{x};\boldsymbol{\theta})$[1] to directly approximate the solution $u^*(\mathbf{x})$ with trainable parameters $\boldsymbol{\theta}$. The training process involves minimizing a loss function that incorporates the following two components:

[1] $\mathbf{x}$ encompasses both spatial and temporal dimensions.

1. **PDE Residual Loss ($\ell_{\mathbf{r}}$)**: This measures how well the neural network satisfies the governing equation. It is computed as the mean squared error (MSE) of the PDE residual over a set of collocation points over all derivative constraints in the domain:$\ell_{\mathrm{r}}(\boldsymbol{\theta}) = \frac{1}{N_{\mathrm{r}}} \sum_{i=1}^{N_{\mathrm{r}}} \|\mathcal{N}[u(\mathbf{x}_i;\boldsymbol{\theta})]\|_2^2$, where $\|\cdot\|_2$ is the L_2 norm, $\mathcal{N}[u(\mathbf{x}_i;\boldsymbol{\theta})]$ is a differential operator, generated via automatic differentiation, $\mathbf{x}_i$ are collocation points sampled from Ω, and N_{r} is the number of collocation points.

2. **Boundary Condition Loss ($\ell_{\mathbf{BC}}$)**: This enforces the boundary conditions. It is computed as the MSE of the discrepancy between the network prediction and the boundary condition:$\ell_{\mathrm{BC}}(\boldsymbol{\theta}) = \frac{1}{N_{\mathrm{BC}}} \sum_{i=1}^{N_{\mathrm{BC}}} \|\mathcal{B}[u(\mathbf{x}_i;\boldsymbol{\theta})]\|_2^2$, where $\mathbf{x}_i$ are points sampled from the boundary $\partial\Omega$, and N_{BC} is the number of boundary points. The total loss function used to train the PINN is a weighted sum of these PDE-related losses: $\ell(\boldsymbol{\theta}) = \ell_{\mathrm{r}}(\boldsymbol{\theta}) + \ell_{\mathrm{BC}}(\boldsymbol{\theta})$. By minimizing this loss function, the neural network learns to approximate the solution of the PDE while satisfying the initial and boundary conditions.

2.2 Differentiation Structure Analysis of Deep Networks

The optimization objective of PINNs fundamentally diverges from that of conventional neural networks. In PINNs, the loss function incorporates the physics of the problem, often expressed through PDE residuals. For a first-order PDE, the residual includes terms like the first-order derivative of the network output $u(\mathbf{x};\boldsymbol{\theta})$ with respect to the input $\mathbf{x}$: $\frac{\partial u}{\partial \mathbf{x}}$. Consequently, training a PINN requires computing higher-order gradients, specifically the gradient of such derivatives with respect to the model parameters θ, i.e., $\frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial u}{\partial \mathbf{x}}\right)$. This contrasts sharply with conventional training, where the core computation is the first-order gradient $\frac{\partial u}{\partial \boldsymbol{\theta}}$. This necessity for gradients of derivatives highlights the critical need to analyze the network's differentiation structure within the PINN framework.

To understand the differentiation structure of within the PINN framework, we begin our analysis with the first-order derivatives of an MLP. Consider an MLP with L hidden layers, the network output $u(\mathbf{x};\boldsymbol{\theta})$ for input $\mathbf{x}$ is: $\mathbf{h}_i = f(\mathbf{h}_{i-1})$, for $i \geq 1$, and $\mathbf{h}_0 = \mathbf{x}$, where the i-th layer(or block) transforms its input $\mathbf{h}_{i-1}$ to $\mathbf{h}_i$ and the final output is $u(\mathbf{x};\boldsymbol{\theta})) = \mathbf{h}_L$. The Jacobian matrix of the network output $u(\mathbf{x};\boldsymbol{\theta})$ with respect to the input $\mathbf{x}$ is structured as a product of layer-wise Jacobians:

$$\frac{\partial u(\mathbf{x};\boldsymbol{\theta})}{\partial \mathbf{x}} = \prod_{l=1}^{L} \frac{\partial \mathbf{h}_l}{\partial \mathbf{h}_{l-1}} = \prod_{l=1}^{L} \mathbf{J}_l. \tag{2}$$

This formula implies that if the norms of Jacobians are systematically smaller or larger than one, the resulting derivatives vanish or grow exponentially with depth, which causes the gradients, i.e., $\frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial u}{\partial \mathbf{x}}\right)$, to vanish or explode. In Section 4, we experimentally confirm that this pathology manifests at initialization: MLPs exhibit vanishing derivatives with increasing depth, while ResNet-based architectures show exploding derivative magnitudes. Wang et al. (2024a) further established a probabilistic bound revealing the intrinsic derivative vanishing pathology in MLPs with $\tanh$ activations.

The analysis in Appendix A further demonstrates that the explosion/vanishing of derivatives is distinct from conventional gradient issues, yet it directly triggers these gradient pathologies in PINN training. This highlights the need for network architectures explicitly designed to ensure stable and well-conditioned differentiation.

3 CoupledNet

In Section 2, we begin by analyzing the derivative structures of neural networks, as these derivatives are central to formulating the physical laws governed by PDEs. A key challenge in PINNs is the numerical instability during automatic differentiation, which can be traced to the properties of these derivatives. Existing studies provide crucial insights into this issue: first, the derivative of an MLP is mathematically equivalent to the successive multiplication of its weight matrices (Wang et al., 2024a); second, the spectral norm of these weight matrices governs the scaling effect on feature vectors and the magnitude of changes during gradient updates (Yang et al., 2024). Combining these insights, we deduce that the stability of the entire automatic differentiation process in PINNs is directly influenced by the spectral norms of the hidden layer Jacobian matrices. Therefore, to fundamentally stabilize this process, we propose imposing dual-bound constraints on these spectral norms. This leads to the introduction of CoupledNet, a structure designed to separately constrain the upper and lower bounds of the spectral norms, thereby stabilizing their numerical behavior.

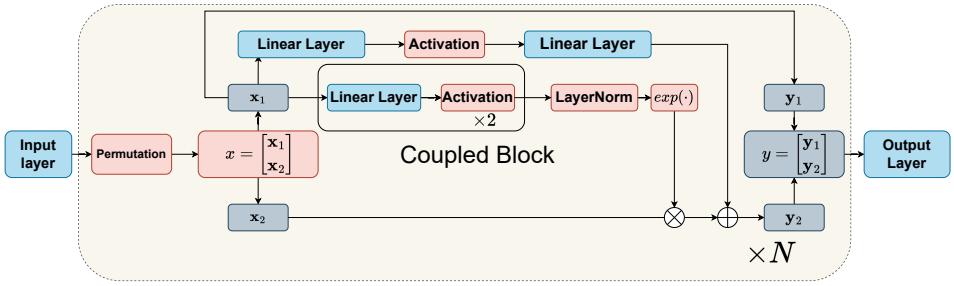


Figure 1: Architecture of CoupledNet: The network consists of (a) an input layer (either linear or Fourier-embedded), which elevates the original coordinates to a high-dimensional embedding space. (b) N stacked Coupled Blocks for highly complex transformation. The notation "$\times 2$" denotes that two sets of linear and activation layers are stacked sequentially. (c) A linear layer maps the final output into the required output dimension.

3.1 Coupled Block

Directly controlling the spectral norm of Jacobian matrices proves highly challenging. Without loss of generality, we assume the widths of different hidden layers are the same, denoted by d, for simplicity. Thus, the Jacobian matrix $\mathbf{J}_l \in \mathbb{R}^{d\times d}$ of each hidden layer is square, except for the input layer and the output layer. Under this assumption, the composite Jacobian determinant across the L hidden layers (without considering the input and output layer) satisfies: $\det\left(\prod_{l=1}^{L}\mathbf{J}_l\right) = \prod_{l=1}^{L}\det\left(\mathbf{J}_l\right)$. The determinant of each Jacobian is precisely the product of its eigenvalues. We utilize the determinant of each Jacobian as an indicator. Mathematically, it can be expressed as

$$\det(\mathbf{J}_l) = \prod_{i=1}^{d} \lambda_{l,i}, \tag{3}$$

where $\lambda_{l,i}$ represents an eigenvalue of $\mathbf{J}_l$. As the lower bound of the spectral norm of the Jacobian matrix is the largest eigenvalue's absolute value, and thus related to the determinant of the Jacobian matrix by (3). This means that one can prevent derivatives from vanishing when L is large by controlling $\det(\mathbf{J}_l) \geq 1$ for each layer l. Therefore, inspired by normalizing flow architectures (Dinh et al., 2017; Kingma and Dhariwal, 2018), we propose CoupledBlock as the core component of CoupledNet, which regulates the spectral norm lower bound of hidden-layer Jacobian matrices by controlling their determinants. Figure 1 presents the detailed model architecture of CoupledNet.

To enhance nonlinear transformations without disrupting the product of determinants, each coupled block first applies a fixed random permutation to the input, then splits it into two same-length partitions $\mathbf{x}_1$ and $\mathbf{x}_2$. The block output $[\mathbf{y}_1; \mathbf{y}_2]$ is computed via:

$$\mathbf{y}_1 = \mathbf{x}_1, \quad \mathbf{y}_2 = \mathbf{s} \odot \mathbf{x}_2 + \mathbf{t}, \tag{4}$$

where $\mathbf{s} = \exp\left(\mathrm{LayerNorm}(\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_1 + \mathbf{b}_1) + \mathbf{b}_2))\right)$ and $\mathbf{t} = \mathbf{W}_4\sigma(\mathbf{W}_3\mathbf{x}_1 + \mathbf{b}_3) + \mathbf{b}_4$. The $\mathrm{LayerNorm}$ has trainable scale parameters and a frozen zero vector as the bias to make the mean of the output $\mathbf{0}$. The number of parameters of a Coupled Block is nearly identical to that of a linear layer with equal width. This architecture prevents derivative vanishing by controlling the determinant of the Jacobian matrix, with theoretical support from the following proposition.

Proposition 3.1 (Spectral Control in CoupledNet Architecture)**.** *The CoupledNet architecture with coupled blocks guarantees controlled spectral properties:*

1. *Each CoupledBlock's Jacobian matrix $\mathbf{J}_l$ satisfies $\|\mathbf{J}_l\|_2 \geq 1$.*
2. *Stacked CoupledBlocks maintain explicit spectral norm lower bound preservation.*

Proof. Proof can be found in Appendix B. □

By strictly enforcing the absolute value of the Jacobian determinant of each coupled block to be 1, we stabilize the eigenvalues of the Jacobian matrices across deep architectures. Thus, CoupledNet avoids derivative vanishing by controlling the spectral norm of the Jacobian matrix.

Moreover, the following theorem proves that CoupledNet has the potential to represent any function, indicating that CoupledNet architecture does not limit the model's expressiveness theoretically.

Proposition 3.2 (Universal Approximation Theorem for CoupledNet)**.** *Let the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-polynomial, bounded, and continuous function (e.g., Sigmoid, Tanh) or a piecewise linear function (e.g., ReLU). For any continuous function $f : K \to \mathbb{R}^m$ defined on a compact set $K \subset \mathbb{R}^n$, and any $\epsilon > 0$, there exists a single-Coupled-Block CoupledNet $F(x)$ such that:*

$$\sup_{x \in K} \|f(x) - F(x)\| < \epsilon,$$

where the number of hidden neurons N is sufficiently large and depends on f, ϵ, and K.

Proof. Proof can be found in Appendix C. □

3.2 Controller for maximum of Jacobian Spectral

We simultaneously seek a simple, effective mechanism to control the maximum spectral norm. However, the presence of permutation layers and the unbounded nature of LayerNorm operations fundamentally prevent rigorous control over the maximum spectral norm of hidden-layer Jacobian matrices. To address this limitation, we develop a two-pronged spectral containment strategy through numerical analysis of differential propagation structures in permutation-free hidden pathways. Specifically, within the CoupledBlock framework, we implement:

- 1. LayerNorm Spectral Containment: Bounding of directional scaling factors during feature normalization,
- 2. Output Layer Constraint: Direct regulation of the output of the last Coupled Block.

For any square matrices A and B, the spectral norm of $A \cdot B$ satisfies $\|AB\| \leq \|A\| \cdot \|B\|$. Since the spectral norm of a permutation matrix is 1, multiplying by a permutation matrix does not affect the maximum control of the spectral norm. Assuming the absence of permutation layers, the diagonal blocks in the Jacobian matrix product of CoupledBlock take the form:

- Upper-left block (preserved dimensions): $[\mathbf{J}]_{1:\frac{n}{2},1:\frac{n}{2}} = \prod_{k=1}^{L} \mathbf{I}_{\frac{n}{2}} = \mathbf{I}_{\frac{n}{2}}$.
- Lower-right block (scaled dimensions):

$$[\mathbf{J}]_{\frac{n}{2}+1:n,\frac{n}{2}+1:n} = \prod_{k=1}^{L} \operatorname{diag}(\mathbf{s}^{(k)}), \text{ where } \left(\prod_{k=1}^{L} \operatorname{diag}(\mathbf{s}^{(k)})\right)_{ii} = \prod_{k=1}^{L} s_i^{(k)}. \quad i > \frac{n}{2}$$

Given that $\mathbf{s}$ is derived through the $\exp(\cdot)$ transformation following LayerNorm, and assuming statistical independence among elements s_i, we impose the probabilistic constraint

$$\mathbb{P}\left(\prod_{k=1}^{L} s_i^{(k)} \leq 2\right) \geq 0.99. \; \forall i \tag{5}$$

Through log-normal distribution analysis, the required LayerNorm standard deviation σ satisfies $\sigma = \frac{\log 2}{z_{0.99} \cdot \sqrt{L}}$, where $z_{0.99} \approx 2.326$ denotes the 99% quantile of the standard normal distribution. We implement this derived σ as the fixed standard deviation parameter in LayerNorm operations, thereby establishing provable upper bounds on the spectral norms of Jacobian matrices within CoupledBlock.

Simultaneously, we analyze the lower-left block of Jacobian matrices. As in CoupledBlock controller, the exponential mapping in the transformation induces dominance of the maximum element in the lower-left Jacobian block by $max(s_i)$. When considering Jacobian matrix products across cascaded blocks, the controlled standard deviation in LayerNorm operations ensures the maximum magnitude in the composite lower-left block can be characterized as a linear combination of layer-wise $max(s_i)$ values. Consequently, we implement a runtime spectral containment protocol:

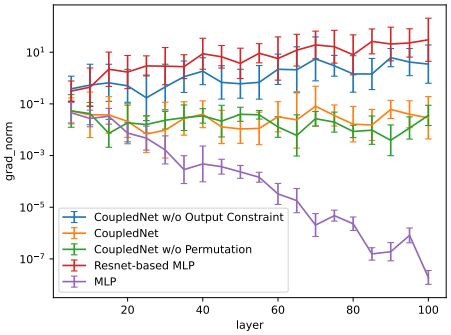


Figure 2: First order derivative norms with different network depth across architectures. Output Constraint stands for Output Layer Constraint in Section 3.2. The solid lines represent the mean values, and the error bars denote $\pm$ one standard deviation in the log domain.

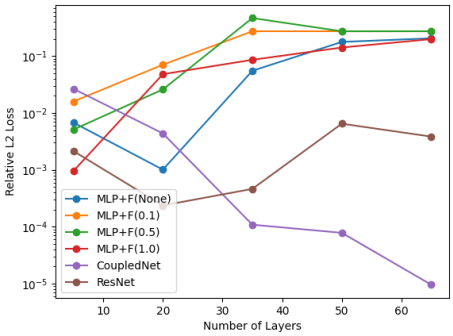


Figure 3: The Figure shows the relative L_2 error of models with different architectures with varying depths on the target function (6). $F(m)$ stands for Fourier feature with scale m, and $F(\text{None})$ means we don't use Fourier feature.

1. Track $\max(\mathbf{s_i})$ during forward propagation for i-th Coupled Block
2. Apply final normalization before output layer: $\mathbf{x} \leftarrow \frac{\mathbf{x}}{\max_i(\max(s_i))*L}$.

The dynamic nature of the scaling parameter during training enables real-time numerical regulation of differentiation operators, contrasting with conventional initialization-based controls that remain static post-initialization. This adaptive mechanism allows continuous stabilization of differentiation structures throughout the training process. This provides CoupledNet's secondary spectral norm control mechanism. However, as this transformation directly modifies pre-activation magnitudes preceding the output layer, it inherently constrains the codomain of solution spaces. We therefore restrict its application to value-range-limited PDE classes.

4 Analysis of Depth-dependent Dynamics in PINN

To validate that CoupledNet can effectively control the norm of model derivatives by regulating spectral norms, as well as to verify that this architecture can enhance stability and strengthen performance in PINN problems through model depth extension, we conducted two experiments.

We first compared the first-order derivative norms of different base model architectures during initialization. Three models were evaluated: MLP, MLP with residual connection (ResNet-based MLP), a widely adopted solution for depth scaling in conventional deep learning (He et al., 2015), and CoupledNet. In the subsequent content, we use ResNet to denote MLP with skip connections, which is described by $\mathbf{h}_i = \sigma(\mathbf{W}\mathbf{h}_{i-1} + \mathbf{b}) + \mathbf{h}_{i-1}$. As shown in Figure 2, the MLP model exhibited an exponential decay of first-order derivatives with increasing depth due to the multiplicative structure of Jacobian matrices from consecutive linear and activation layers. While ResNet addressed vanishing derivatives, it developed derivative explosion issues as depth increased. The numerical instability caused by deeper ResNets also led to training failures in PINNs. In contrast, the CoupledNet architecture demonstrated stable first-order derivative magnitudes across varying depths, regardless of output layer scaling implementation, indicating effective resolution of numerical challenges arising from Jacobian matrix multiplication. Furthermore, since the upper-bound controller proposed in Section 3.2 was designed based on numerical analysis without considering the permutation matrix, we additionally conducted an ablation study by introducing the permutation operation in this experiment. The results show that the permutation operation had negligible influence on the derivative norms, supporting our assumption in Section 3.2 and confirming the rationality of the Jacobian norm upper-bound controller design.

Our experiments have demonstrated that CoupledNet ensures numerical stability in derivative computations during depth escalation by enforcing both upper and lower bounds on the spectral norms of

Jacobian matrices. The following experiment provides empirical evidence that CoupledNet achieves superior performance through architectural deepening. Consider an ODE defined as:

$$\frac{du(x)}{dx} = f(x); u(-2) = u_0; u(2) = u_1, \quad x \in [-2, 2], \tag{6}$$

where $u(x)$ is the solution to be learned, and $f(x)$ is the first derivative of $u(x)$. The analytical solution for $u(x)$ is $u_{\text{analytical}}(x) = \exp(x) + \frac{1}{1+x^2} + \sin(2\pi \cdot 10x)$, which is composed of the sum of three simple functions. It is easy to find that $u_{\text{analytical}}(x)$ contains the high-frequency component $\sin(20\pi x)$, which is generally considered the main failure mode of PINN. In the experiment, the networks are configured with a width of 64 and trained using a batch size of 1,024 for 10,000 steps. The Adam optimizer is used with an initial learning rate of 1×10^{-3} and the learning rate decays following an exponential schedule. We trained models with 5 random seeds for each node in Figure 3 and calculated the average of the relative L_2 losses.

In this experimental configuration, we benchmarked the MLP, ResNet-incorporated MLP, and CoupledNet architectures. To systematically investigate the pathological behaviors induced by network depth escalation in Multilayer Perceptron (MLP) architectures, we implement Fourier feature-enhanced MLP variants to evaluate the hypothesis that spectral embedding could mitigate depth-related pathologies. Figure 3 reveals a consistent pattern: all MLP-derived architectures exhibit progressive deterioration in training capability with increasing depth. This empirical evidence demonstrates that frequency-space transformations fail to compensate for fundamental architectural limitations in deep MLP configurations. Moreover, ResNet implementations achieve a decreasing relative L_2 error with depth of escalation up to 20 layers. However, this improvement trajectory reverses beyond 20 layers, with error metrics exhibiting a positive correlation with additional depth increments. These findings align with the theoretical framework proposed in (Wang et al., 2024a), confirming that skip connections alone cannot resolve the intrinsic pathologies of deep PINN architectures.

As can be seen from Figure 3, CoupledNet fully leverages the superior capacity of deep models in PINN training: the relative L_2 error decreases consistently as the model depth increases. At a depth of 50 layers, CoupledNet achieves prediction accuracy that surpasses several baseline models across all tested depths on the ODE benchmark. We can conclude that, in the training of PINN with high-frequency information, deepening the model by using a reasonable model architecture is an effective way to improve the model's performance.

5 Experimental Results

5.1 High-frequency Experiments

We design this experiment to demonstrate our method's superior capability in handling complex high-frequency PDEs compared to existing approaches. The benchmark problem is defined on a 2D domain $\Omega = [-1, 1] \times [-1, 1]$ with exact solution: $u_{\text{analytical}}(x, t) = \sin\left(30 \cdot 2\pi x^2 + 15\pi t^2\right)$, where the quadratic modulation of spatial coordinates creates high-frequency patterns. The governing first-order vector PDE and boundary conditions are jointly formulated as:

$$\begin{cases} \nabla u = \mathbf{f}(x, t), & (x, t) \in \Omega \\ u(x, t) = \sin\left(30 \cdot 2\pi x^2 + 15\pi t^2\right), & (x, t) \in \partial\Omega \end{cases} \tag{7}$$

where $\nabla u = (\partial u/\partial x, \partial u/\partial t)^\top$ and $\mathbf{f}(x, t)$ is analytically derived from the exact solution.

All models are trained using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 1×10^{-3} and exponential decay (period 2,000 steps, ratio 0.9) for 50,000 iterations on NVIDIA RTX 4090 GPUs. We implemented an experiment using the JAX framework. and the batch size is 1024. We compare two state-of-the-art baselines: the modified MLP in JaxPi (Wang et al., 2023) and PirateNet (Wang et al., 2024a). Both models utilize Fourier features (Wang et al., 2021b), random matrix factorization (Wang et al., 2022b), causal training (Wang et al., 2024b) and NTK reweighting (Wang et al., 2022a) techniques. Hyperparameter grids are searched for all methods (detailed in Appendix E).

The experimental results are shown Table 1. CoupledNet demonstrates superior performance in solving high-frequency PDEs. As shown in Table 1, both the final relative L_2 loss and residual loss

Table 1: Metrics of Three models(Jaxpi, PirateNet, and CoupledNet) on PDEs equation 7 (Best results are highlighted in bold.)

	Pirate	Jaxpi	Pirate w/o NTK	Jaxpi w/o NTK	CoupledNet
Boundary Loss	**1.8e-4**	1.0e-3	1.4e-2	2.4e-2	7.2e-4
Residual Loss	4.98	4.31	1689.5	40.38	**1.4871**
Relative L_2 Loss	0.0334	0.0324	0.0785	0.1022	**0.0216**

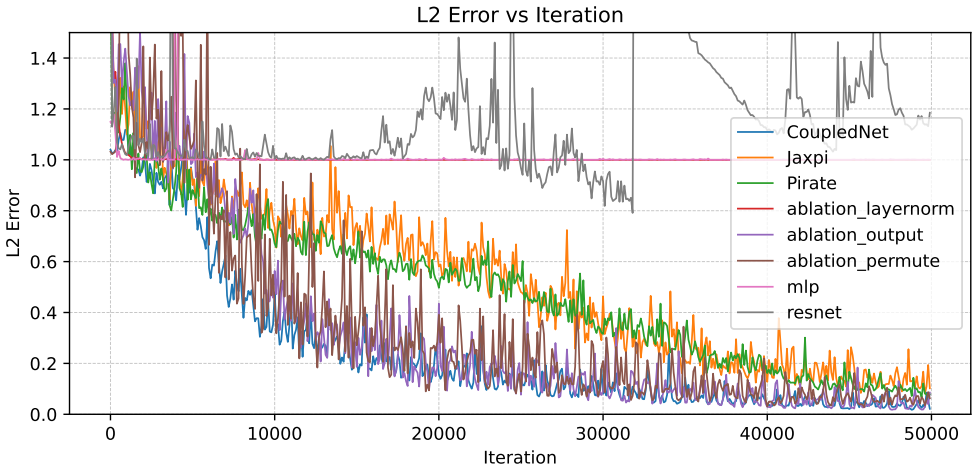


Figure 4: Ablation study on Equation 7. We track the evolution of relative L_2 error across training steps. The study investigates the effects of the two upper-bound controllers and the permutation matrix. CoupledNet exhibits the fastest convergence. "ablation" indicates removal of the specified component from CoupledNet. Specifically, "layernorm" refers to removing the LayerNorm Spectral Containment from Section 3.2, "output" refers to removing the Output Layer Constraint from Section 3.2, and "-permute" refers to removing the permutation layers.

of CoupledNet achieve the lowest values among the three compared models. While its boundary loss is slightly higher than PirateNet's, we attribute this phenomenon to the NTK reweighting technique employed by both PirateNet and Jaxpi during training, which helps maintain appropriate loss weights even with reduced boundary loss magnitudes. Compared with PirateNet and Jaxpi models that do not use NTK re-weighting, CoupledNet has far lower Boundary Loss and Residual Loss than the two. During the hyperparameter search process, we did not perform any reweighting for CoupledNet. However, CoupledNet still achieved a good balance between boundary loss and residual loss. We hypothesize that this advantage stems from CoupledNet's unique differentiation architecture, which intrinsically enhances gradient learning efficiency. Figure can be found in Appendix D.

To further examine the convergence behavior of CoupledNet and assess the effects of its two upper-bound controllers and the permutation matrix, we performed an ablation study using the same PDE setup. We tracked the evolution of the relative L_2 error, initial condition loss, and residual loss over training steps for four models: MLP, ResNet, PirateNet, Jaxpi, and CoupledNet. As shown in Figure 4, Figure 7 and Figure 8, MLP and ResNet fail to converge, which aligns with the pathological behavior predicted by our analysis. CoupledNet exhibits the fastest convergence among all models and naturally maintains a balance between residual and boundary losses, without any reweighting strategy. Notably, when the permutation matrix is removed from CoupledNet, the final relative L_2 error increases to 0.063, confirming the effectiveness and necessity of the permutation matrix design. Although each training step of CoupledNet is slightly slower than that of PirateNet due to its computations, its superior convergence dynamics lead to higher overall training efficiency.

In summary, CoupledNet delivers superior performance on high-frequency PDEs, revealing its inherent potential for addressing the learning challenges posed by high-frequency PDE systems through architectural innovations.

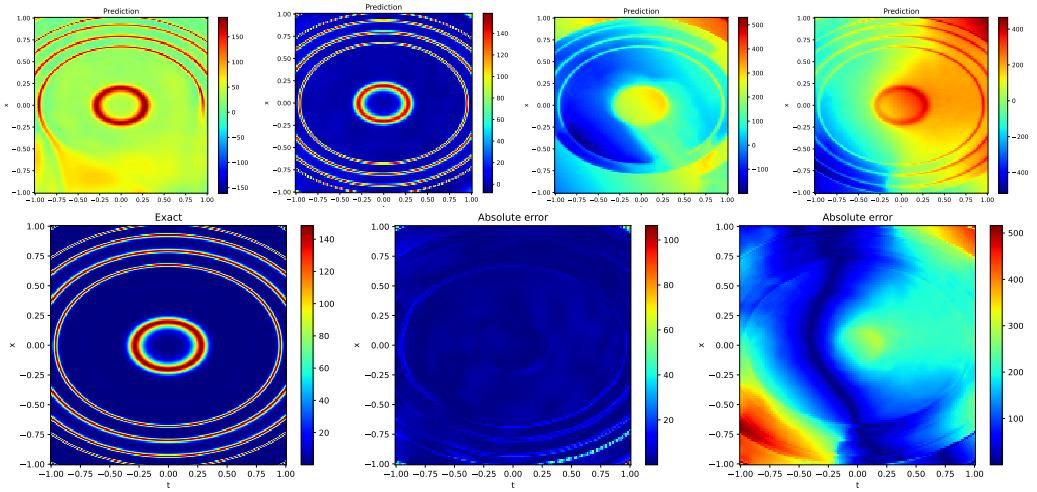


Figure 5: Comparison between CoupledNet and Pirate on the PDE in equation 8. **Top**: From left to right are the prediction results of a 8-layer CoupledNet, a 16-layer CoupledNet, a 9-layer PirateNet, and a 18-layer PirateNet. **Bottom**: From left to right are the analytical solution of the PDE, the absolute error of the predicted solution by the 16-layer CoupledNet, and the absolute error of the predicted solution by the 18-layer PirateNet.

5.2 Comparison Between PirateNet and CoupledNet

As both PirateNet and CoupledNet are depth-enhancing frameworks for PINNs, we conduct a comparative analysis of their architectural efficacy under equivalent depth configurations. To assess their capacity to approximate complex analytic solutions, we still employ a first-order PDE system:

$$\begin{cases} \nabla u = \mathbf{f}(x,t), & (x,t) \in \Omega \\ u(x,t) = e^{5\sin\left(2\pi \sin\left(\pi(x^2+2t^2)\right)\right)}, & (x,t) \in \partial\Omega \end{cases} \tag{8}$$

where $\Omega = [-1,1] \times [-1,1]$, and $u_{\text{analytical}}(x,t) = e^{5\sin\left(2\pi \sin\left(\pi(x^2+2t^2)\right)\right)}$.

In this experiment, both CoupledNet and PirateNet learn with the same hyperparameters, which are presented in Appendix F. Experimental results demonstrate that a 16-layer CoupledNet achieves a relative L_2 error of **0.0718**, outperforming both its 8-layer CoupledNet (**0.5775**), 18-layer PirateNet (**4.067**) and a 9-layer PirateNet (**2.673**). For PDEs that are difficult to learn and require a deeper model to get the solution, CoupledNet is a more suitable architecture than PirateNet. During the experimental process, we found that CoupledNet's computational efficiency was lower than the PirateNet. However, since CoupledNet can solve PDEs that PirateNet cannot handle, we consider this still an acceptable cost.

The models' predictions are shown in Figure 5. Both PirateNet and CoupledNet can obtain more accurate results merely by deepening the model. This phenomenon supplements the conclusion we reached in Section 4: *Deepening the model not only leads to more accurate results in ODE problems but also holds true for PDE problems.*

Discussion on Training Pathologies of PINN Existing studies have identified spectral bias in PINN training, where models preferentially learn low-frequency components over high-frequency features (Rahaman et al., 2018). However, our experimental findings provide new insights into this phenomenon. As visualized in Figure 5, the analytic solution's high-frequency features concentrate within five concentric annuli, separated by low-frequency regions. Both CoupledNet and PirateNet predictions exhibit partial neglect of these low-frequency components during learning.

This observation leads us to hypothesize that attributing PINN training pathologies solely to high-frequency characteristics constitutes an oversimplification. The underlying challenges may involve additional factors beyond frequency domain considerations, including:

Table 2: Results on PDE benchmarks (Best results are highlighted in bold.)

	Advection	Allen-Cahn	Burgers	Grey-Scott	Korteweg-De Vries	Hamilton-Jacobi-Bellman
Jaxpi	6.88e-4	5.37e-5	1.43e-4	6.13	1.96e-3	7.42e-2
PirateNet	4.88e-4	**2.24e-5**	**4.03e-5**	**3.61e-3**	**4.27e-4**	9.32e-2
CoupledNet	**3.64e-4**	9.09e-5	1.22e-4	1.93e-2	1.30e-3	**6.08e-2**

1. Magnitude disparities across different spatial regions of the analytic solution;
2. Numerical characteristics of solution derivatives;
3. Relative scale relationships between the solution and its differential terms.

Our results suggest that frequency-based analysis should be augmented with complementary perspectives to fully understand PINN training dynamics.

5.3 Experiment on conventional PDE Benchmarks

To validate CoupledNet's performance on conventional PDE benchmarks, we conduct experiments using the benchmark suite from Jaxpi (Wang et al., 2023), with full PDE specifications documented in Appendix G. Furthermore, to validate the performance of CoupledNet in high-dimensional PDEs, we evaluated CoupledNet, PirateNet, and Jaxpi on a **100-dimensional** Hamilton-Jacobi-Bellman (HJB) equation from Wang et al. (2022c) at $T = 1.0$. As shown in Table 2, CoupledNet surpasses Jaxpi on five PDEs and achieves the lowest relative L_2 loss on the Advection and HJB equations, despite not obtaining the best numerical results on every benchmark.

Importantly, visualizations of the predicted solutions (see Appendix I) demonstrate that CoupledNet successfully captures the underlying structural features of PDE solutions across all tested problems, even in cases where its quantitative performance is not the best. These visual results confirm that CoupledNet does not exhibit failure modes such as oscillations or divergence, highlighting its robustness and reliability. This observation aligns with our design motivation: to develop a general and stable architecture capable of solving a wide range of PDEs without task-specific modifications.

To further analyze the training dynamics, we investigate the spectral bias issue commonly observed in PINNs. Specifically, we examine the evolution of the neural tangent kernel (NTK) spectrum during the training of the 1D Poisson equation from Wang et al. (2021b). In Section J, Figure 13, Figure 14 and Figure 15 show that, at the early stages of training, the eigenvector corresponding to the top-3 NTK eigenvalue in CoupledNet already exhibits pronounced high-frequency components. In contrast, MLP and PirateNet begin to capture high-frequency features after substantially longer training. This observation indicates that the CoupledNet architecture effectively mitigates the spectral bias problem in PINN training by promoting early learning of high-frequency modes. Such behavior suggests that CoupledNet has the potential to serve as a promising foundation for future research on understanding and alleviating spectral bias in physics-informed learning frameworks.

6 Conclusion

Current deep learning architectures, such as MLPs and ResNets, exhibit initialization pathologies that fundamentally limit their depth scalability. To address this, we propose CoupledNet, a novel physics-informed deep learning architecture. CoupledNet achieves state-of-the-art performance on PDE systems with complex analytical solutions. Although CoupledNet does not achieve state-of-the-art results on all PDE benchmarks, it remains competitively accurate. By addressing the limitations of existing approaches and providing a stable framework for deep PINNs, our work opens new avenues for applying deep learning to complex physical systems.

References

Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The finite element method: Its basis and fundamentals*. Elsevier, 2013.

Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems*. SIAM, 2007.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

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.

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.

Zhiping Mao, Ameya D Jagtap, and George E Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

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.

Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Sifan Wang, Xuhui Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022a.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. URL https://api.semanticscholar.org/CorpusID:206594692.

Lu Lu, Xuhui Meng, Zhiping Mao, and George E Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks, 2024a. URL https://arxiv.org/abs/2402.00326.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv: Learning*, 2014. URL https://api.semanticscholar.org/CorpusID:13995862.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning, 2024. URL https://arxiv.org/abs/2310.17813.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 31:10215–10224, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks, 2023. URL https://arxiv.org/abs/2308.08468.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Sifan Wang, Hanwen Wang, Jacob H. Seidman, and Paris Perdikaris. Random weight factorization improves the training of continuous neural representations, 2022b. URL https://arxiv.org/abs/2210.01274.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421: 116813, 2024b. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2024.116813. URL https://www.sciencedirect.com/science/article/pii/S0045782524000690.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Dräxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:53012119.

Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is l2 physics-informed loss always suitable for training physics-informed neural network? In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022c. Curran Associates Inc. ISBN 9781713871088.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL https://www.sciencedirect.com/science/article/pii/089360809190009T.

A SIMPLE EXAMPLE

The loss function in PINNs typically comprises two parts: a data-driven term (e.g., for boundary/initial conditions) and a physics-driven term (e.g., for the governing ODE/PDE). The gradient computation for the data-driven term mirrors that of traditional neural networks. However, the gradient of the physics-driven term is directly governed by the derivatives of the differential equation. To explicitly illustrate this critical relationship between the network's gradient and the equation's derivatives, we consider a simplified case of a static ODE:

$$\frac{\partial u}{\partial \mathbf{x}} = f(\mathbf{x})$$

The boundary loss is a typical supervised learning problem that is well-established in the deep learning literature. We consider the other part of the loss function in PINNs, using the vanilla residual loss as an example, to show where the difference and difficulty come from:

$$L = \int (\frac{\partial u(\mathbf{x})}{\partial \mathbf{x}} - f(\mathbf{x}))^2 dx \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial u(\mathbf{x}_i)}{\partial \mathbf{x}_i} - f(\mathbf{x}_i) \right)^2$$

The gradient of a single data point of the residual loss is:

$$\frac{\partial L}{\partial \theta} = 2 \left(\frac{\partial u(\mathbf{x}_i)}{\partial \mathbf{x}_i} - f(\mathbf{x}_i) \right) \frac{\partial}{\partial \theta} \left(\frac{\partial u(\mathbf{x}_i)}{\partial \mathbf{x}_i} \right)$$

Considering the gradient of parameters at the i-th layer:

$$\begin{aligned} \frac{\partial L}{\partial \theta_i} &= 2 \left(\frac{\partial u(\mathbf{x}_j)}{\partial \mathbf{x}_j} - f(\mathbf{x}_j) \right) \frac{\partial}{\partial \theta_i} \left(\frac{\partial u(\mathbf{x}_j)}{\partial \mathbf{x}_j} \right) \\ &= r_j \left[\sum_{m=i}^{L} \left(\frac{\partial u}{\partial h_m} \cdot \frac{\partial^2 h_m}{\partial \theta_i \partial h_{m-1}} \cdot \frac{\partial h_{m-1}}{\partial \mathbf{x}} \right) \right], \end{aligned}$$

where $r_j = \frac{\partial u(\mathbf{x}_j)}{\partial \mathbf{x}_j} - f(\mathbf{x}_j)$ represents the residual error.

The gradient depends on both the forward propagation path ($\frac{\partial h_{m-1}}{\partial \mathbf{x}}$) and the backward propagation path ($\frac{\partial u}{\partial h_m}$), connected through the second-order derivative $\frac{\partial^2 h_m}{\partial \theta_i \partial h_{m-1}}$. $\frac{\partial u}{\partial h_m}$ is the product of Jacobian matrices from layer m to the output layer L $\frac{\partial h_{m-1}}{\partial \mathbf{x}}$ is the product of Jacobian matrices from the input layer to layer $m-1$.

For parameters in intermediate layers ($1 < i < L$), the gradient computation involves Jacobian products that span nearly the entire network architecture, in contrast to traditional backpropagation where parameters only face Jacobians from their layer to the output. This analytical result demonstrates that PINN gradients inherently encode richer architectural information than conventional neural networks, as they must account for both the forward flow of input variations and the backward flow of output sensitivities simultaneously.

Therefore, PINN's gradient problem has fundamental differences from traditional well-known gradient problems. For sufficiently deep models, even for parameters close to the output layer, numerical issues in the Jacobian matrices of layers before them will still affect the updates of these parameters.

Therefore, we can conclude that the numerical issues of model derivatives are an important cause of PINN gradient problems. PINN's derivative numerical problems and gradient numerical problems are not independent; when model derivatives are numerically unstable, PINN optimization will not obtain stable gradients.

B PROOF OF PROPOSITION 3.1

Proof. We establish these properties through an analytic examination of the network's differential structure:

Part 1: Spectral norm of Coupled Block.

Consider the coupled block's Jacobian decomposition:

$$\mathbf{J}_l = \mathbf{J'}_l \cdot \mathbf{P}, \text{ with } \mathbf{J'}_l = \begin{bmatrix} \mathbf{I}_l & \mathbf{0} \\ \mathrm{diag}(\mathbf{x}_2)\mathbf{J}_s + \mathbf{J}_t & \mathrm{diag}(\mathbf{s}) \end{bmatrix}. \tag{9}$$

The triangular structure of $\mathbf{J'}_l$ yields determinant:

$$\det(\mathbf{J'}_l) = \prod_{i=1}^{d} s_i = \exp(\sum \log s_i) = \exp(0) = 1, \tag{10}$$

due to the LayerNorm constraint with no bias added, see Section 3.

Since P is a permutation matrix, each layer maintains:

$$|\det(\mathbf{J}_l)| = |\det(\mathbf{J'}_l)||\det(\mathbf{P})| = 1 \tag{11}$$

Therefore, we have

$$\|\mathbf{J}_l\|_2 \geq \rho(\mathbf{J}_l) \geq 1. \tag{12}$$

Part 2: Lower bound of the Spectral norm of the Jacobian of CoupledNet.

For L-layer network(CoupledNet) with total Jacobian $\mathbf{J} = \prod_{l=1}^{L} \mathbf{J}_l$:

$$\det(\mathbf{J}) = \prod_{l=1}^{L} \det(\mathbf{J}_l) = 1. \tag{13}$$

From spectral theory , the spectral radius $\rho(\mathbf{J})$ satisfies:

$$\rho(\mathbf{J}) = \max_{1 \leq k \leq n} |\lambda_k| \geq |\det(\mathbf{J})|^{1/n} = 1. \tag{14}$$

Relating spectral radius to spectral norm:

$$\|\mathbf{J}\|_2 \geq \rho(\mathbf{J}) \geq 1. \tag{15}$$

This completes the proof. □

C PROOF OF PROPOSITION 3.2

Proof. We demonstrate a degenerate case of CoupledNet, which possesses the universal approximation ability, thereby showing that CoupledNet can approximate any continuous function. Let the input layer be a fully-connected layer without bias:

$$\mathbf{x}' = \mathbf{W}_{\text{input}} x, \quad \mathbf{W}_{\text{input}} \in \mathbb{R}^{N \times n} \tag{16}$$

Suppose that the permutation layer of the single-Coupled-Block CoupledNet is an identity function. Then, we have

$$F(x) = \mathbf{W}_{\text{output}} \mathbf{y} + \mathbf{b}_{\text{output}}, \tag{17}$$

where y is the output of the Coupled Block:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \mathbf{y}_1 = \mathbf{x}'_1, \mathbf{y}_2 = \mathbf{s} \odot \mathbf{x}'_2 + \mathbf{t}, \tag{18}$$

where $\mathbf{s} = \exp\left(\mathrm{LayerNorm}(\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}'_1 + \mathbf{b}_1) + \mathbf{b}_2))\right)$, $\mathbf{t} = \mathbf{W}_4\sigma(\mathbf{W}_3\mathbf{x}'_1 + \mathbf{b}_3) + \mathbf{b}_4$, and $\mathbf{x}' = \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \end{bmatrix}$.

Let $\mathbf{W}_2$ be a $\mathbf{0}$ matrix, then we have $\mathbf{s} = \mathbf{1}^{\frac{N}{2}}$. Moreover, let $\mathbf{W}_3$ and $\mathbf{W}_4$ be $\mathbf{I}^{\frac{N}{2}}$ and $\mathbf{b}_4$ equal to $-\mathbf{x}_2'$, we can derive:

$$F(x) = \mathbf{W}_{\text{output}} \cdot \begin{bmatrix} \mathbf{x}_1' \\ \sigma(\mathbf{x}_1' + \mathbf{b}_3) \end{bmatrix} + \mathbf{b}_{\text{output}}. \tag{19}$$

Let $\mathbf{W}_{\text{output}} = \left[\mathbf{0}^{m \times \frac{N}{2}} | \mathbf{W}'_{\text{output}}\right]$, we have

$$F(x) = \mathbf{W}'_{\text{output}} \sigma(\mathbf{W}'_{\text{input}} x + \mathbf{b}_3) + \mathbf{b}_{\text{output}}, \tag{20}$$

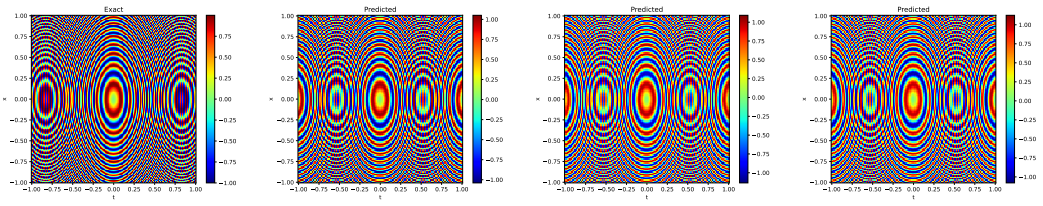
where $\mathbf{W}'_{\text{input}}$ is the upper half of $\mathbf{W}_{\text{input}}$.

By the Universal Approximation Theorem for MLP from (Hornik, 1991), we can drive that for any continuous function $f : K \to \mathbb{R}^m$ defined on a compact set $K \subset \mathbb{R}^n$, and any $\epsilon > 0$, there exists a single-Coupled-Bloscк CoupledNet $F(x)$ such that:

$$\sup_{x \in K} \|f(x) - F(x)\| < \epsilon, \tag{21}$$

where the number of hidden neurons N is sufficiently large and depends on f, ϵ, and K. □

D FIGURE OF SECTION 5.1



(a) Exact (b) CoupledNet (c) Jaxpi (d) PirateNet

Figure 6: Comparison between the solution predicted by three models and the analytical solution on PDE system equation 7.

E HYPER PARAMETER

Table 3: Hyperparameter Search Grid for experiment in Section 5.1 three methods for results in Figure 6.

Parameter	**Value**
Architecture	
Number of layers	9, 18 for PirateNet, 4, 8 for Jaxpi 8, 16 for CoupledNet
Hidden size	128, 256
Activation function	tanh
Fourier embedding scale	NA, 1.0, 2.0, 4.0
Random weight factorization	$(\mu = 0.5, \sigma = 0.1)$, $(\mu = 1.0, \sigma = 0.1)$
Weighting	
Weighting scheme	NA for CoupledNet NA, ntk, grad norm for Jaxpi and PirateNet
Warm up step	0,2000
Causal tolerance	NA, 1.0
Number of Chunks(if use Causal)	32

Table 4: Hyperparameter for experiment in Section 5.1 on PirateNet for results in Figure 6.

Parameter	Value
Architecture	
Number of layers	18
Hidden size	256
Activation function	tanh
Fourier embedding scale	4.0
Random weight factorization	$(\mu = 0.5, \sigma = 0.1)$
Weighting	
Weighting scheme	NTK
Warm up step	0
Causal tolerance	NA
Number of Chunks(if use Causal)	NA

Table 5: Hyperparameter for experiment in Section 5.1 on Jaxpi for results in Figure 6.

Parameter	Value
Architecture	
Number of layers	4
Hidden size	256
Activation function	tanh
Fourier embedding scale	4.0
Random weight factorization	$(\mu = 0.5, \sigma = 0.1)$
Weighting	
Weighting scheme	NTK
Warm up step	0
Causal tolerance	NA
Number of Chunks(if use Causal)	NA

Table 6: Hyperparameter for experiment in Section 5.1 on CoupledNet for results in Figure 6.

Parameter	Value
Architecture	
Number of layers	8
Hidden size	256
Activation function	tanh
Fourier embedding scale	4.0
Random weight factorization	$(\mu = 0.5, \sigma = 0.1)$
Weighting	
Weighting scheme	NA
Warm up step	2000
Causal tolerance	NA
Number of Chunks(if use Causal)	NA

Table 7: Configuration Details of CoupledNet on Advection equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	9
Hidden Dimension	256
Output Dimension	1
Activation Function	tanh
Periodicity	period: (3 * jnp.pi, 1.0) axis: (0, 1) trainable: (True, False)
Fourier Embedding	embed-scale: 1.0 embed-dim: 256
Reparameterization	None
PI Initialization	None
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	2000
Staircase	False
Warmup Steps	5000
Gradient Accumulation Steps	0
Max Training Steps	200000
Batch Size per Device	2048
Weighting Scheme	ntk
Initial Weights	res: 1.0 ics: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	True
Causal Tolerance	1.0
Time window	None
Number of Chunks	32

Table 8: Configuration Details of CoupledNet on Allen-Cahn equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	9
Hidden Dimension	256
Output Dimension	1
Activation	tanh
Periodicity	period: (jnp.pi,) axis: (1,) trainable: (False,)
Fourier Embedding	embed-scale: 2 embed-dim: 256
Nonlinearity	0.0
Reparameterization	None
PI Initialization	None
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	5000
Staircase	False
Warmup Steps	5000
Grad Accum Steps	0
Training Max Steps	300000
Batch Size per Device	4096
Weighting Scheme	ntk
Initial Weights	ics: 1.0 res: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	True
Causal Tol	1.0
Time window	None
Number of Chunks	32

Table 9: Configuration Details of CoupledNet on GS equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	8
Hidden Dimension	256
Output Dimension	2
Activation Function	swish
Periodicity	period: (jnp.pi, jnp.pi) axis: (1, 2) trainable: (False, False)
Fourier Embedding	embed-scale: 1.0 embed-dim: 256
Reparameterization	$(\mu = 0.5, \sigma = 0.1)$
PI Initialization	Ture
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	2000
Staircase	False
Warmup Steps	5000
Gradient Accumulation Steps	0
Max Training Steps	100000
Batch Size per Device	4096
Weighting Scheme	grad_norm
Initial Weights	u_ic: 1.0 v_ic: 1.0 ru: 1.0 rv: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	True
Causal Tolerance	1.0
Time window	10
Number of Chunks	32

Table 10: Configuration Details of CoupledNet on KdV equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	32
Hidden Dimension	256
Output Dimension	1
Activation Function	tanh
Periodicity	period: (jnp.pi,) axis: (1,) trainable: (False,)
Fourier Embedding	embed-scale: 1.0 embed-dim: 256
Reparameterization	None
PI Initialization	None
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	2000
Staircase	False
Warmup Steps	5000
Gradient Accumulation Steps	0
Max Training Steps	200000
Batch Size per Device	4096
Weighting Scheme	grad_norm
Initial Weights	ics: 1.0 res: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	True
Causal Tolerance	0.1
Number of Chunks	16

Table 11: Configuration Details of CoupledNet on Burgers equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	3
Hidden Dimension	256
Output Dimension	1
Activation Function	tanh
Fourier Embedding	embed-scale: 1 embed-dim: 256
Nonlinearity	0.0
Reparameterization	None
PI Initialization	None
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	1000
Staircase	False
Warmup Steps	5000
Max Training Steps	100000
Batch Size per Device	8192
Weighting Scheme	grad_norm
Initial Weights	ics: 1.0 bcs: 1.0 res: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	True
Causal Tolerance	1.0
Number of Chunks	32

F HYPER-PARAMETER IN SECTION 5.2

For CoupledNet and PirateNet, neither of them undergo hyperparameter tuning, and neither NTK reweighting nor Fourier features are used. The hyperparameters are selected based on those employed in PirateNet experiments across multiple PDEs (Wang et al., 2024a). The optimizer chosen is Adam, with an initial learning rate of 1×10^{-3} for 8-layer CoupledNet (9-layer PirateNet) and 1×10^{-4} for 16-layer CoupledNet (18-layer PirateNet), using exponential decay (period 2,000 steps, ratio 0.9). Since the 18-layer PirateNet failed to converge with a learning rate of 1×10^{-4}, we conducted additional experiments using a learning rate of 1×10^{-3}. The width of the hidden layers is 256, the training step is 50000, the activation function is tanh, and the batch size is 512. Because the problem has a large value range, CoupledNet did not use the output layer constraint in this experiment.

G HYPER-PARAMETER IN PDE BENCHMARK

For these benchmarks, we used the experimental setups of Jaxpi and PirateNet (Wang et al., 2023; 2024a). We use the published open source data from Jaxpi as training and testing data in these benchmarks. `https://github.com/PredictiveIntelligenceLab/jaxpi`

G.1 ADVECTION EQUATION HYPER-PARAMETERS

Advection equation used in experiment is:

$$\begin{aligned} &\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad t \in [0,1], x \in (0, 2\pi) \\ &u(0,x) = g(x), \quad x \in (0, 2\pi) \end{aligned}$$

where $c = 80$ and $g(x) = \sin(x)$. Hyper-parameters are shown in Table 7

G.2 ALLEN-CAHN EQUATION

Allen-Cahn equation used in experiment is:

$$\begin{aligned} u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \quad t \in [0,1], x \in [-1,1] \\ u(0,x) = x^2\cos(\pi x) \\ u(t,-1) = u(t,1) \\ u_x(t,-1) = u_x(t,1). \end{aligned}$$

Hyper-parameters are shown in Table 8.

G.3 GREY-SCOTT EQUATION

Grey-Scott equation used in experiment is:

$$\begin{cases} u_t = \epsilon_1 \Delta u + b_1(1-u) - c_1 uv^2, & t \in (0,2), (x,y) \in (-1,1)^2 \\ v_t = \epsilon_2 \Delta v - b_2 v + c_2 uv^2, & t \in (0,2), (x,y) \in (-1,1)^2 \end{cases}$$

subject to the periodic boundary conditions and the initial conditions

$$\begin{cases} u_0(x,y) = 1 - \exp\left(-10\left((x+0.05)^2 + (y+0.02)^2\right)\right), \\ v_0(x,y) = 1 - \exp\left(-10\left((x-0.05)^2 + (y-0.02)^2\right)\right). \end{cases}$$

Hyper-parameters are shown in Table 9.

G.4 KORTEWEG–DE VRIES EQUATION

Korteweg–De equation used in experiment is:

$$\begin{aligned} u_t + \eta u u_x + \mu^2 u_{xxx} = 0, t \in (0,1), x \in (-1,1) \\ u(x,0) = \cos(\pi x) \\ u(t,-1) = u(t,1), \end{aligned}$$

where $\eta = 1$ and $\mu = 0.022$ Hyper-parameters are shown in Table 10.

G.5 BURGER'S EQUATION

Burger's equation equation used in experiment is:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \frac{0.01}{\pi} \frac{\partial^2 u}{\partial x^2}$$

Hyper-parameters are shown in Table 11.

G.6 HAMILTON-JACOBI-BELLMAN EQUATION

Hamilton-Jacobi-Bellman (HJB) equation used in experiment is:

$$\begin{cases} \mathcal{L}_{\text{HJB}} u := \partial_t u(x,t) + \frac{1}{2}\sigma^2 \Delta u(x,t) - \sum_{i=1}^{n} A_i |\partial_{x_i} u|^{c_i} = \varphi(x,t), & (x,t) \in \mathbb{R}^n \times [0,T] \\ \mathcal{B}_{\text{HJB}} u := u(x,T) = g(x), & x \in \mathbb{R}^n \end{cases}$$

where $A_i = \underbrace{(a_i \alpha_i)^{-\frac{1}{\alpha_i - 1}}}_{\text{First term}} - \underbrace{a_i (a_i \alpha_i)^{-\frac{\alpha_i}{\alpha_i - 1}}}_{\text{Second term}} \in (0, +\infty)$ and $c_i = \frac{\alpha_i}{\alpha_i - 1} \in (1, +\infty)$. Hyper-parameters are shown in Table 12.

Table 12: Configuration Details of CoupledNet on HJB equation

Category	**Value**
Architecture Name	CoupledMlp
Number of Layers	9
Hidden Dimension	256
Output Dimension	1
Activation Function	tanh
Fourier Embedding	None
Nonlinearity	0.0
Reparameterization	weight fact (μ=1.0, σ=0.1)
PI Initialization	None
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Epsilon	1e-8
Learning Rate	1e-3
Decay Rate	0.9
Decay Steps	2000
Staircase	False
Warmup Steps	5000
Gradient Accumulation Steps	0
Max Training Steps	50000
Batch Size per Device	512
Weighting Scheme	grad_norm
Initial Weights	res: 1.0 ics: 1.0
Momentum	0.9
Update Every Steps	1000
Use Causal	False
Causal Tolerance	1e-2
Number of Chunks	32

H Ablation study results on High Frequency PDE

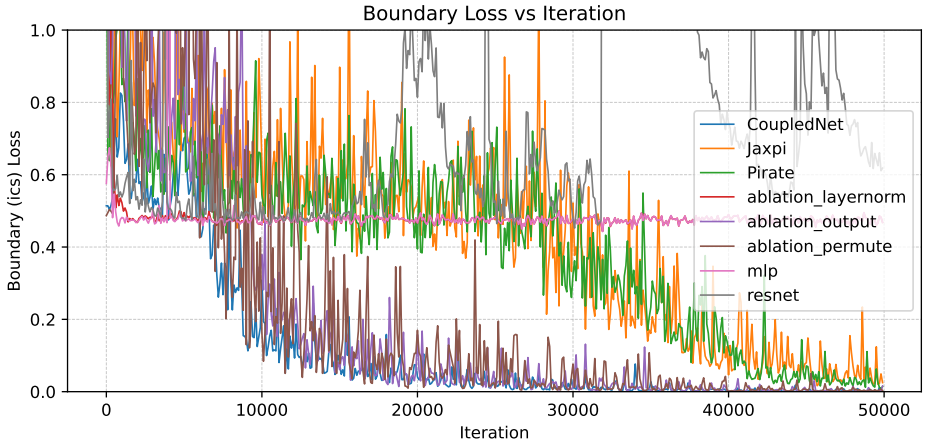


Figure 7: Ablation study on Equation 7 We track the evolution of boundary condition loss across training steps. "ablation" indicates removal of the specified component from CoupledNet. Specifically, "layernorm" refers to removing the LayerNorm Spectral Containment from Section 3.2, "output" refers to removing the Output Layer Constraint from Section 3.2, and "-permute" refers to removing the permutation layers.

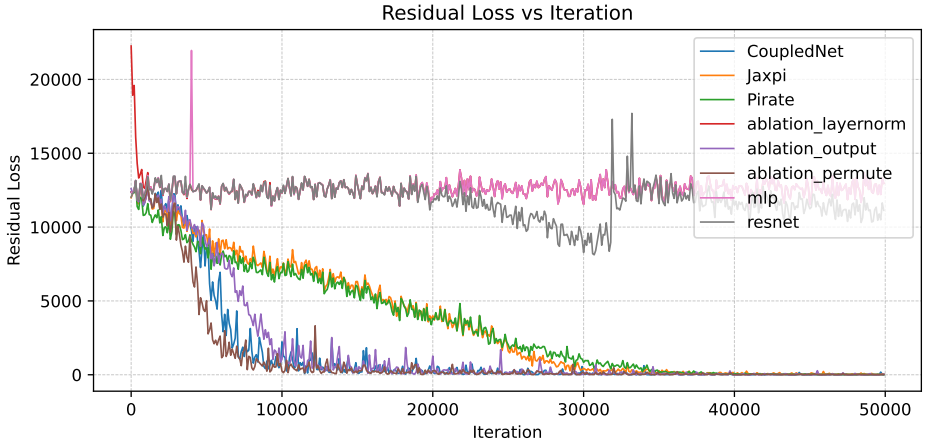


Figure 8: Ablation study on the Equation 7. We track the evolution of residual loss across training steps. "ablation" indicates removal of the specified component from CoupledNet. Specifically, "layernorm" refers to removing the LayerNorm Spectral Containment from Section 3.2, "output" refers to removing the Output Layer Constraint from Section 3.2, and "-permute" refers to removing the permutation layers.

I Jaxpi Benchmark Visualization Results

This section presents the visualization results of **CoupledNet** on three representative partial differential equations (PDEs) from the Jaxpi benchmark, including Advection, Allen–Cahn, and Burgers equations. Each figure shows the predicted solution, analytical solution, and absolute error.

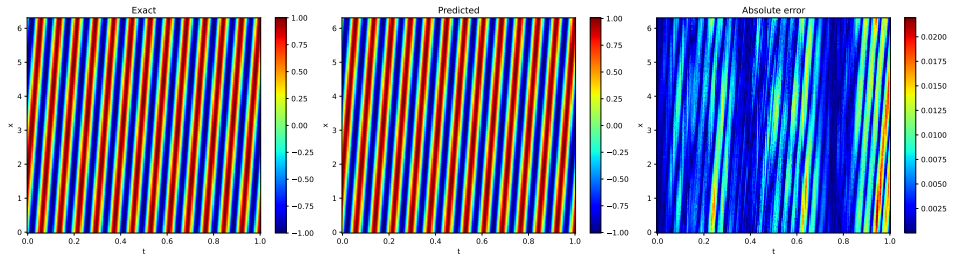


Figure 9: Visualization of CoupledNet predictions, analytical solutions, and absolute errors for the Advection equation.

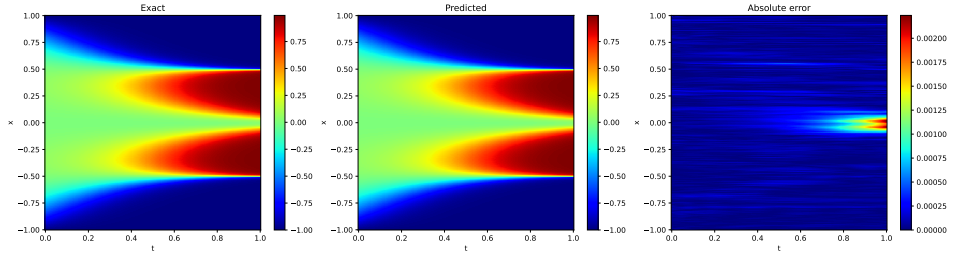


Figure 10: Visualization of CoupledNet predictions, analytical solutions, and absolute errors for the Allen–Cahn equation.

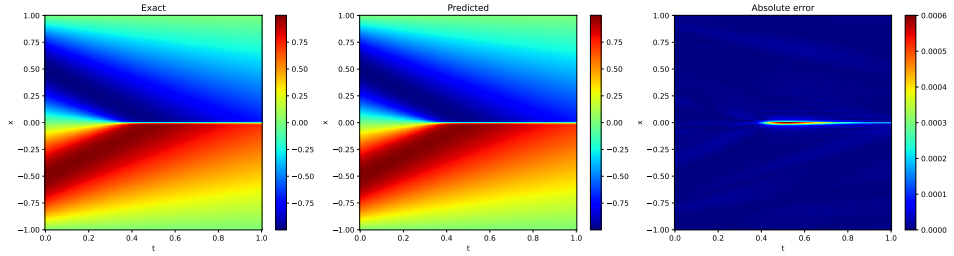


Figure 11: Visualization of CoupledNet predictions, analytical solutions, and absolute errors for the Burgers equation.

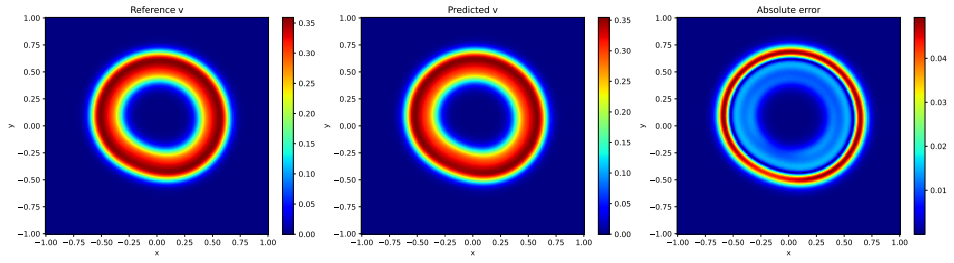


Figure 12: Visualization of CoupledNet predictions, analytical solutions, and absolute errors for the Grey-Scott equation.

J SECTRAL BIAS EXPERIMENTS ON 1D POSSION EQUATION

To analyze the spectral bias in PINN training, we follow the experimental protocol of Wang et al. (2021b). We conduct experiments on the 1D Poisson equation to examine the evolution of the

neural tangent kernel (NTK) spectrum during training. All models—including MLP, PirateNet, and CoupledNet—use two hidden layers with a width of 64 and the same activation function. No additional PINN training techniques (e.g., NTK reweighting, adaptive loss scaling, or residual regularization) are applied to ensure a fair comparison. The NTK matrix is computed at initialization and periodically during training to track the eigenvalues and the corresponding eigenvectors. The frequency spectrum of each eigenvector is visualized as a heatmap to illustrate how different architectures capture frequency components over the course of training.

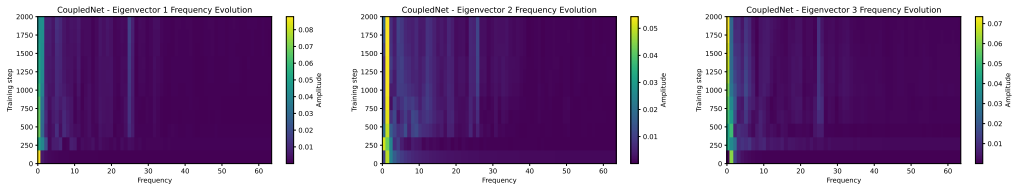


Figure 13: Evolution of frequency components corresponding to the top-3 NTK eigenvectors of CoupledNet.

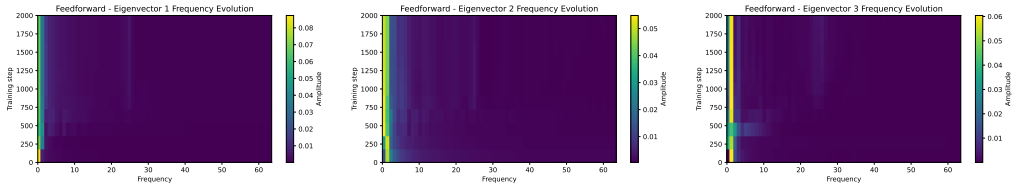


Figure 14: Evolution of frequency components corresponding to the top-3 NTK eigenvectors of MLP.

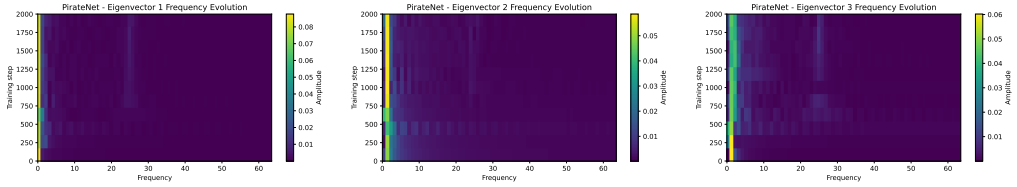


Figure 15: Evolution of frequency components corresponding to the top-3 NTK eigenvectors of PirateNet.

K Verification of the Summation Invariance in Bias-Free Layer Normalization

To verify the numerical stability and structural constraint of CoupledNet, we track the summation of the outputs from the bias-free LayerNorm layers throughout the training process. As shown in Figure 16, the accumulated sum remains bounded within 1.75×10^{-6}, which can be attributed solely to floating-point precision errors. This result confirms that CoupledNet strictly preserves the determinant constraint of the Jacobian matrix, i.e., $\det(\mathbf{J}) = 1$, throughout training.

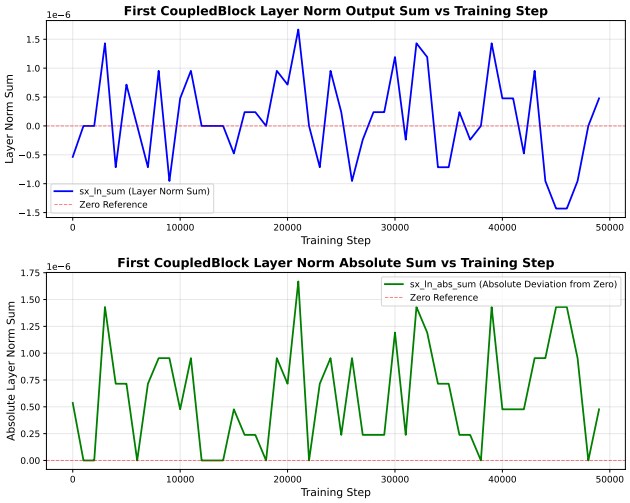


Figure 16: Tracking of the bias-free LayerNorm output summation during CoupledNet training. The values remain bounded within 1.75×10^{-6} due to floating-point errors, confirming that CoupledNet rigorously satisfies the Jacobian determinant constraint $\det(\mathbf{J}) = 1$.

L LLM USAGE

In the preparation of this manuscript, the authors employed DeepSeek-V3.1 for language editing and readability enhancement. The authors conducted thorough review and editing of all AI-assisted content and assume complete responsibility for the publication's content and accuracy.