Neural Tangent Kernel Analysis to Probe Convergence in Physics-informed Neural Solvers: PIKANs vs. PINNs

Salah A. Faroughi[a,*], Farinaz Mostajeran[a]

[a]*Energy & Intelligence Lab, Department of Chemical Engineering, University of Utah, Salt Lake City, Utah 84112, USA*

Abstract

Physics-informed Kolmogorov–Arnold Networks (PIKANs), and in particular their Chebyshev-based variants (cPIKANs), have recently emerged as promising models for solving partial differential equations (PDEs). However, their training dynamics and convergence behavior remain largely unexplored both theoretically and numerically. In this work, we aim to advance the theoretical understanding of cPIKANs by analyzing them using Neural Tangent Kernel (NTK) theory. Our objective is to discern the evolution of kernel structure throughout gradient-based training and its subsequent impact on learning efficiency. We first derive the NTK of standard cKANs in a supervised setting, and then extend the analysis to the physics-informed context. We analyze the spectral properties of NTK matrices, specifically their eigenvalue distributions and spectral bias, for four representative PDEs: the steady-state Helmholtz equation, transient diffusion and Allen–Cahn equations, and forced vibrations governed by the Euler–Bernoulli beam equation. We also conduct an investigation into the impact of various optimization strategies, e.g., first-order, second-order, and hybrid approaches, on the evolution of the NTK and the resulting learning dynamics. Results indicate a tractable behavior for NTK in the context of cPIKANs, which exposes learning dynamics that standard physics-informed neural networks (PINNs) cannot capture. Spectral trends also reveal when domain decomposition improves training, directly linking kernel behavior to convergence rates under different setups. To the best of our knowledge, this is the first systematic NTK study of cPIKANs, providing theoretical insight that clarifies and predicts their empirical performance.

Keywords: Physics-informed Neural Networks, Kolmogorov-Arnold Network, Chebyshev Polynomials, Domain-scaling, Variable-scaling, Scientific Machine Learning

1. Introduction

Modeling systems characterized by diverse phenomena occurring across different scales is a computational challenge. Some systems possess dynamics spanning nanometers to kilometers spatially, and nanoseconds to years temporally. Such problems arise in a range of real-world applications, including nuclear reactor design and safety [1], catalysts and functional materials [2, 3], combustion dynamics [4, 5], synthetic biology and bioengineering [6], battery systems [7], and power grid infrastructures [8]. More complex examples are found in multiphase reacting flows within porous media [9] and Earth systems modeling under uncertainty [10, 11], both of which can exhibit dynamics over vastly extended spatial and temporal scales. In order to investigate such systems, multifaceted mathematics are needed to capture the system in its entirety while integrating all components [12–14]. This approach typically leads to a set of coupled partial or ordinary differential equations (PDEs and ODEs), cross-scale closure models, stochastic processes, and uncertainty analytics. When modeling these systems, especially in an inverse context where extensive optimization is required to identify targeted properties or infer unknown characteristics [15, 16], rapid convergence and high accuracy become essential. Classical numerical methods, such as finite difference, finite element, and finite volume schemes, are effective, but their computational cost scales poorly with domain size, $\Omega \times [0, T]$, due to stability restrictions (e.g., CFL conditions) and the need for fine resolution to capture localized phenomena [17–19].

To overcome the computational limitations of classical solvers, physics-informed neural networks (PINNs) have been introduced as a mesh-free alternative that incorporates the governing PDEs directly into the loss function of a deep neural network [20–23]. Over the past few years, PINNs have demonstrated remarkable success across a wide range of scientific and engineering problems, showcasing their potential as general-purpose solvers for forward and inverse modeling. Notable applications include modeling turbulent fluid flows [24–27] and viscoelastic fluid flows [28, 29], solving cardiovascular flow problems [30–32], simulating seismic wave propagation [33–35], and identifying material parameters in solid mechanics [36–39]. These studies highlight the strength of PINNs in leveraging limited data and physical constraints to infer complex dynamics, especially in scenarios where traditional solvers are either too costly or inapplicable. Despite their theoretical appeal and

*Corresponding author: salah.faroughi@utah.edu

success on benchmark problems, PINNs often perform poorly on large domains, particularly when the solution exhibits high-frequency oscillations, sharp interfaces, or multiscale structures [40–42]. Several studies have investigated the underlying failure modes of PINNs and attempted to characterize their learning dynamics [43–46]. These limitations have motivated the development of alternative neural architectures designed to improve expressivity, training stability, and convergence in the presence of complex physical behavior.

To overcome the limitations of MLP-based PINNs, recent advances have introduced alternative architectures grounded in the Kolmogorov–Arnold representation theorem [47, 48]. Among them, Kolmogorov–Arnold Networks (KANs) [49–51] replace fixed nonlinear activations with learnable univariate functions, often implemented using splines or low-order polynomials. This formulation provides finer control over spectral bias and improves approximation of localized or high-frequency features [52]. When integrated with the physics-informed training paradigm, the resulting physics-informed KANs (PIKANs) offer enhanced expressivity while remaining grounded in physical laws [52–54]. In particular, the Chebyshev-based variant, cPIKAN, leverages orthogonal polynomial bases to improve stability and interpretability in learning tasks [55, 56]. However, the use of Chebyshev polynomials necessitates input normalization to the $[-1, 1]$ interval for numerical stability, especially in large or multi-scale domains [55]. To this end, Scaled-cPIKAN [54] introduces a scaling strategy for spatial variables and residual losses that enhances convergence and accuracy, as verified across multiple benchmark problems. By aligning the architectural flexibility of KANs with domain-informed normalization, this approach effectively addresses the performance bottlenecks of vanilla PINNs in extended spatial domains. Also, in [54], the authors further investigate the theoretical necessity of variable scaling by analyzing the Neural Tangent Kernel (NTK) structure [57], leveraging the mathematical simplicity of Chebyshev polynomials to establish a clear justification for their scaling strategy. These developments set the stage for a deeper theoretical understanding of training dynamics in cPIKANs, particularly through the lens of the NTK analysis, which we want to explore in this paper.

The NTK was introduced by Jacot et al. [57] as a theoretical framework to analyze the training dynamics of infinitely-wide neural networks under gradient descent. They showed that, in the infinite-width limit, neural networks behave like kernel methods, where the NTK converges to a deterministic, constant kernel during training. This allows the study of learning dynamics in function space, rather than parameter space, and connects convergence properties to the eigenstructure of the NTK. Their work also demonstrated that the speed of convergence depends on the eigenvalues of the NTK, with faster convergence along directions corresponding to larger eigenvalues. Building on this, [58] investigated the validity of NTK theory for finite-width networks. Their results revealed that the NTK approximation often fails for deep or improperly initialized networks, where gradients may vanish or explode. They further showed that NTK behavior depends critically on the depth-to-width ratio and the initialization regime [59], whether the network operates in the ordered phase, chaotic phase, or at the edge of chaos. In particular, they proved that the variability of the NTK grows exponentially with depth in the chaotic and the edge of chaos regimes, undermining the assumption of a constant NTK during training. Inspired by these findings, Wang et al. [60] applied NTK analysis to PINNs. They derived the NTK for PINNs and showed that, under infinite-width assumptions, it converges to a deterministic kernel that remains nearly constant during training. However, they identified a critical limitation: an imbalance in convergence rates across different loss components, driven by spectral bias. To mitigate this issue, they proposed an adaptive gradient descent algorithm that uses NTK eigenvalues to balance learning dynamics. Saadat et al. [61] extended these ideas to PINNs for linear advection-diffusion equations and demonstrated that variations in advection speed or diffusion coefficient can cause training failures. Their work highlighted that PINNs often struggle to learn initial or boundary conditions when PDE parameters dominate. They also suggested strategies such as adaptive loss weighting and periodic activation functions to address these challenges.

In this work, we aim to advance the theoretical understanding of cPIKANs by studying their training behavior through the NTK framework. We focus on how the use of Chebyshev polynomial bases, combined with input normalization strategies, shapes the NTK eigenstructure, and how these factors affect the learning efficiency and generalization of cPIKANs on large or multi-scale domains. Understanding these relationships is essential for designing architectures that balance expressiveness, stability, and computational efficiency when modeling complex physical systems with sharp interfaces or high-frequency features. Our analysis begins by deriving explicit expressions for the NTK of cKANs in the standard, non-physics-informed setting. We study how the choice of basis functions and network parameters influences the NTK structure for finite-width networks. We then extend the analysis to the physics-informed case, deriving the NTK for cPIKANs, where PDE residuals are included in the loss. We provide a detailed breakdown of the NTK into components that reflect the effects of physics-informed terms. We support our theoretical findings with experiments on various PDE examples. First, we compare the NTK behavior of cKANs with standard MLPs. Then, we study cPIKANs trained with different optimization algorithms such as Adam and L-BFGS, and compare the impact of Chebyshev and B-spline basis functions on the NTK. We also analyze how the NTK structure varies across different subdomains within a single problem. Overall, this work bridges the gap between empirical observations and theoretical understanding in cPIKANs. The insights provided here can guide the development of more robust and efficient physics-informed neural architectures for complex physical modeling tasks.

2. Preliminary

2.1. Kolmogorov-Arnold Networks (KANs)

Kolmogorov's superposition theorem, introduced in 1957 [47] and refined by Arnold [48, 62, 63], states that any continuous multivariate function can be expressed as a combination of continuous single-variable functions. This result was extended by Lorentz [64, 65], Sprecher [66, 67], and Friedman [68], though early constructions had issues such as the non-smoothness of the inner functions [69]. Later, Kůrková [70, 71] addressed these concerns using sigmoidal approximations, while Sprecher and Köppen proposed numerical and structural improvements [72, 73], culminating in a constructive proof by Braun and Griebel [74]. This Kolmogorov-Arnold Representation Theorem forms a theoretical basis for universal function approximation [75–77] and continues to influence the design of neural architectures that use compositions (depth) and summations (width) of simple univariate functions to approximate complex mappings [78–81].

Early implementations of the Kolmogorov-Arnold representation in neural networks followed the original depth-2, width-$(2n+1)$ structure [73, 82], but faced practical limitations due to non-smooth inner functions and inefficient training. Recent advances [83, 84] have renewed interest in this approach by incorporating gradient-based optimization and emphasizing interpretability, particularly in scientific domains. These developments have led to the emergence of Kolmogorov-Arnold Networks (KANs) [50, 51, 85, 86], which explicitly implement the decomposition,

$$f(x_1, \ldots, x_d) = \sum_{k=0}^{2d} \varphi_k \left(\sum_{j=1}^{d} \psi_{k,j}(x_j) \right), \tag{1}$$

using trainable univariate functions $\psi_{k,j}$ and φ_k within neural architectures. While this formulation corresponds to a depth-2 network, recent work [49] extends KANs to multi-layer architectures by composing univariate functions, for example,

$$\phi_k = \phi_k^{(L)} \circ \cdots \circ \phi_k^{(1)}, \tag{2}$$

which improves both smoothness and representational capacity. In this setting, a KAN with L layers can be expressed as a composition of layer-wise basis functions,

$$f(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)(x), \tag{3}$$

where each $\Phi_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$ consists of univariate transformations applied element-wise, and the outputs are recursively computed as $x_{\ell+1} = \Phi_\ell(x_\ell)$. This generalized formulation retains the theoretical foundation of Kolmogorov's superposition while harnessing the expressive power of deep architectures to achieve greater flexibility and efficiency.

Kolmogorov-Arnold Networks rely on carefully chosen univariate basis functions for approximation. B-splines are a common choice [], offering local control through piecewise polynomial functions defined on a grid,

$$\phi(x) = w_b\, b(x) + w_s \sum_i c_i B_i(x), \quad b(x) = \frac{x}{1 + e^{-x}}, \tag{4}$$

where $B_i(x)$ are spline basis functions defined by the grid size g and polynomial order k, with trainable parameters $\boldsymbol{\theta} = \{c_i, w_b, w_s\}$. While effective for structured data, spline-based KANs scale poorly with increasing model complexity, requiring $O(N_l N_n^2 (k+g))$ parameters. To enhance the modeling of fine-grained and hierarchical features, researchers have developed wavelet-based KANs (Wav-KAN) [87]. These leverage multiscale wavelet expansions to efficiently capture localized patterns, with the number of trainable parameters scaling as $O(3N_l N_n^2)$. For problems requiring smooth interpolation, radial basis function (RBF) networks [88] provide an effective alternative, maintaining the same parameter complexity while offering superior approximation properties for continuous functions.

An increasingly favored approach for KAN design involves the use of Chebyshev polynomials [55] as a globally orthogonal basis [89, 90], offering strong theoretical guarantees in approximation and efficient parameter usage. In the Chebyshev-KAN (cKAN) framework, the nonlinear mapping is expressed via a polynomial expansion,

$$\phi(x) = \sum_{n=0}^{k} c_n\, T_n(x), \tag{5}$$

where $T_n(x)$ denotes the nth Chebyshev polynomial of the first kind, recursively defined as,

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_n(x) = 2\,x\,T_{n-1}(x) - T_{n-2}(x). \tag{6}$$

This orthogonal basis enables stable and efficient function approximation, especially for smooth or oscillatory inputs, while reducing the parameter count to $|\boldsymbol{\theta}| \sim O(N_l N_n^2 k)$. For faster evaluation, a trigonometric form $T_n(x) = \cos(n \arccos(x))$ is sometimes used, though care must be taken near the domain boundaries $x \in [-1, 1]$ to avoid numerical instability [54]. To ensure numerical stability and input compatibility, inputs are normalized

using tanh activations across layers [55, 56, 91], leading to the composite architecture,

$$f_{\text{cKAN}}(\boldsymbol{x}) = (\boldsymbol{\Phi}_L \circ \tanh \circ \cdots \circ \tanh \circ \boldsymbol{\Phi}_1 \circ \tanh)(\boldsymbol{x}). \tag{7}$$

The Chebyshev basis thus offers a computationally efficient and theoretically grounded method for representing high-order nonlinearities in KANs, particularly beneficial in applications involving high-frequency content or rapidly varying target functions.

2.2. Neural Tangent Kernel (NTK)

The Neural Tangent Kernel, introduced in the seminal work by Jacot et al. [57], provides a powerful theoretical framework for understanding the training dynamics of neural networks. NTK theory studies fully-connected neural networks in the infinite-width limit, where the number of neurons in each hidden layer tends to infinity [57, 92]. In this regime, it has been shown that the network's behavior during training can be approximated by a linear model derived from the first-order Taylor expansion around its initialization [57, 92, 93]. Under such conditions and a specific parameter initialization, the NTK remains constant throughout training, rendering the training process equivalent to deterministic kernel regression. This insight reveals that training an over-parameterized neural network with gradient descent can be viewed through kernel methods, providing a clearer understanding of why such networks often generalize well despite their complexity.

In [54], the authors showed that scaling spatial variables is crucial for stable and efficient learning in the cKAN framework using NTK analysis. Their study revealed that the spectral properties of the NTK matrix strongly influence the training dynamics and highlighted the role of proper scaling in achieving robust model performance. Let us consider the problem of minimizing the squared loss function,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2, \tag{8}$$

over a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \subset [-1, 1]^{d_{\text{in}}} \times \mathbb{R}$, where f denotes the output of a cKAN parameterized by $\boldsymbol{\theta}$. The continuous version of gradient descent, known as gradient flow, is given by,

$$\frac{d\boldsymbol{\theta}(\tau)}{d\tau} = -\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}(\tau)) = -\sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))\, \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta}). \tag{9}$$

Throughout this work, τ denotes the gradient flow time, a continuous-time variable that represents the evolution of the network parameters under gradient descent. For simplicity, we refer to it as training time. Using the chain rule, the time evolution of the network output can be expressed as,

$$\frac{df(\boldsymbol{x}_i; \boldsymbol{\theta})}{d\tau} = \left[\frac{df(\boldsymbol{x}_i; \boldsymbol{\theta})}{d\boldsymbol{\theta}}\right] \cdot \left[\frac{d\boldsymbol{\theta}(\tau)}{d\tau}\right] = -\sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))\, [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta})]^T\, [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta})]\,. \tag{10}$$

The Neural Tangent Kernel matrix associated with the cKAN model is defined as,

$$(\boldsymbol{K}_{\text{ntk}}(\tau))_{i,j} = \left\langle \frac{\partial f(\boldsymbol{x}_i; \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{x}_j; \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}} \right\rangle. \tag{11}$$

$\boldsymbol{K}_{\text{ntk}}(\tau)$ is a positive semi-definite block matrix that captures the sensitivity of each output component with respect to changes in model parameters. Finally, the evolution of the network predictions $\boldsymbol{v}(\tau) = (f(\boldsymbol{x}_1; \boldsymbol{\theta}(\tau)), \ldots, f(\boldsymbol{x}_N; \boldsymbol{\theta}(\tau))) \in \mathbb{R}^N$ follows the differential equation,

$$\frac{d\boldsymbol{v}(\tau)}{d\tau} = -\boldsymbol{K}_{\text{ntk}}(\tau) \cdot (\boldsymbol{v}(\tau) - \boldsymbol{y}), \tag{12}$$

where $\boldsymbol{y} = (y_1, \ldots, y_N)$ represents the target outputs. This formulation reveals how the NTK governs the dynamics of training in the cKAN framework. Assume that the output of the cKAN model can be represented as a Chebyshev polynomial expansion,

$$f(x; \boldsymbol{\theta}) = \sum_{n=0}^{k} c_n T_n(x), \tag{13}$$

where k is the number of terms in the expansion, and $\boldsymbol{\theta} = \{c_n\}_{n=0}^k$ denotes the corresponding coefficients. Since the partial derivative of the cKAN output in Eq. (13) with respect to each coefficient c_n is simply the corresponding Chebyshev polynomial,

$$\frac{\partial f}{\partial c_n} = T_n(x) \quad \text{for all } n = 0, \ldots, k, \tag{14}$$

the Neural Tangent Kernel takes the form,

$$(\boldsymbol{K}_{\text{ntk}}(\tau))_{i,j} = \sum_{n=0}^{k} T_n(x_i)\, T_n(x_j). \tag{15}$$

This shows that the NTK matrix is independent of the parameters c_n and remains constant throughout training,

$$\boldsymbol{K}^*_{\text{ntk}} = \boldsymbol{K}_{\text{ntk}}(\tau) \quad \text{for all } \tau. \tag{16}$$

Therefore, under the assumption in Eq. (13), the NTK does not evolve during training [54].

3. Theoretical Development

This section develops the theoretical framework needed to analyze the NTK associated with the cPIKAN model. We recently studied the NTK of cKAN using a non-nested approximation that simplifies the treatment of input scaling and enables analytical analysis [54]. While this approach provides useful insight into the influence of the Chebyshev basis on the NTK structure, it does not capture the full complexity of the nested cKAN architecture. To address this gap, we first investigate the NTK associated with cKANs in a nested data-driven setting; a theoretical analysis that lays the foundation for the second part, where we extend the analysis to cPIKANs by incorporating physics-informed constraints into the NTK framework.

3.1. Step I: NTK for cKANs

This section analyzes the behavior of $\boldsymbol{K}_{\text{ntk}}(\tau)$ under the use of nested approximation. We consider a cKAN with a single hidden layer consisting of N neurons, input dimensionality d, Chebyshev polynomial degree k, and a scalar output. For each normalized input coordinate $\tilde{x}_i = \tanh(x_i)$, $i = 1, \ldots, d$, the Chebyshev polynomials of the first kind up to degree k are computed recursively as,

$$T_0(\tilde{x}_i) = 1, \quad T_1(\tilde{x}_i) = \tilde{x}_i, \quad T_n(\tilde{x}_i) = 2\,\tilde{x}_i\, T_{n-1}(\tilde{x}_i) - T_{n-2}(\tilde{x}_i), \quad n = 2, \ldots, k, \tag{17}$$

and the output of each hidden neuron j is given by,

$$h_j(\boldsymbol{x}) = \sum_{i=1}^{d} \sum_{n=0}^{k} w^{(1)}_{i,j,n} \cdot T_n(\tilde{x}_i), \quad j = 1, \ldots, N, \tag{18}$$

where $w^{(1)}_{i,j,n}$ are trainable coefficients associated with input coordinate i, neuron j, and polynomial degree n. The final output f is computed as a linear combination of the transformed hidden neurons,

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{j=1}^{N} \sum_{n=0}^{k} w^{(2)}_{j,1,n} \cdot T_n(\tanh(h_j(\boldsymbol{x}))), \tag{19}$$

where $w^{(2)}_{j,1,n}$ are trainable coefficients associated with hidden neuron j. The full set of trainable parameters is denoted as $\boldsymbol{\theta} = \{w^{(1)}_{i,j,n}, w^{(2)}_{j,1,n}\}$. We can also show that,

$$\frac{\partial h_j(\boldsymbol{x})}{\partial w^{(1)}_{i,j,n}} = T_n(\tilde{x}_i), \quad \frac{\partial f(\boldsymbol{x}; \boldsymbol{\theta})}{\partial h_j} = \big(1 - \tanh^2(h_j(\boldsymbol{x}))\big) \cdot \sum_{n=0}^{k} w^{(2)}_{j,1,n}\, T'_n\big(\tanh(h_j(\boldsymbol{x}))\big), \tag{20}$$

where $T'_n(\cdot)$ denotes the derivative of the Chebyshev polynomial T_n with respect to its argument. Therefore, the derivative of the output with respect to the first-layer coefficients is,

$$\frac{\partial f(\boldsymbol{x}; \boldsymbol{\theta})}{\partial w^{(1)}_{i,j,n}} = T_n(\tilde{x}_i) \cdot \big(1 - \tanh^2(h_j(\boldsymbol{x}))\big) \cdot \sum_{m=0}^{k} w^{(2)}_{j,1,m}\, T'_m\big(\tanh(h_j(\boldsymbol{x}))\big). \tag{21}$$

and, the derivative of the output with respect to the second-layer coefficients is,

$$\frac{\partial f(\boldsymbol{x}; \boldsymbol{\theta})}{\partial w^{(2)}_{j,1,n}} = T_n\big(\tanh(h_j(\boldsymbol{x}))\big). \tag{22}$$

Given that the coefficients $w^{(1)}_{i,j,n}$ and $w^{(2)}_{j,1,m}$ are independently and identically distributed as standard normal random variables,

$$w^{(1)}_{i,j,n},\ w^{(2)}_{j,1,m} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1), \tag{23}$$

and assuming that the input values $\boldsymbol{x} = (x_1, \ldots, x_d)$ are fixed, the hidden pre-activation for neuron j is given by,

$$h_j(\boldsymbol{x}) = \sum_{i=1}^{d} \sum_{n=0}^{k} w_{i,j,n}^{(1)} \, T_n(\tanh(x_i)) \sim \mathcal{N}(0,\ \sigma_{h_j}^2), \tag{24}$$

where the variance $\sigma_{h_j}^2$ is determined by,

$$\sigma_{h_j}^2 = \sum_{i=1}^{d} \sum_{n=0}^{k} \left[T_n(\tanh(x_i)) \right]^2. \tag{25}$$

Since h_j is a Gaussian random variable, the transformed value $z_j = \tanh(h_j)$ is a smooth and bounded function of a Gaussian variable. While z_j is no longer Gaussian, its distribution remains symmetric, centered at zero, and unimodal. The network output f can thus be computed as a linear combination of the second-layer coefficients $w_{j,1,n}^{(2)}$, weighted by the Chebyshev polynomials evaluated at z_j,

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \sum_{n=0}^{k} w_{j,1,n}^{(2)} \cdot T_n(z_j(\boldsymbol{x})), \tag{26}$$

where $z_j(\boldsymbol{x}) = \tanh(h_j(\boldsymbol{x})) \in (-1, 1)$, implying that each Chebyshev term $T_n(z_j)$ is bounded within $[-1, 1]$. Given z_j, the values $T_n(z_j)$ are deterministic, so each product $w_{j,1,n}^{(2)} \cdot T_n(z_j)$ is a Gaussian random variable with zero mean and variance $T_n(z_j)^2$. Therefore, conditioned on the values of $\{z_j\}$, the output f is normally distributed,

$$f \mid \{z_j\} \sim \mathcal{N}\left(0,\ \sum_{j=1}^{N} \sum_{n=0}^{k} T_n(z_j)^2\right). \tag{27}$$

Since the variables z_j themselves are random, the marginal distribution of f is a Gaussian mixture, meaning that f has a distribution whose variance depends on the particular realization of $\{z_j\}$. In expectation, the variance of f is given by,

$$\mathbb{V}[f] = \mathbb{E}_{\{z_j\}} \left[\sum_{j=1}^{N} \sum_{n=0}^{k} T_n(z_j)^2 \right], \tag{28}$$

that means the output f has zero mean, i.e., $\mathbb{E}[f] = 0$, and a variance that depends on the input coordinate values $\tilde{x}_i = \tanh(x_i)$, the network width N, the Chebyshev polynomial degree k, and how the nonlinearity tanh transforms the Gaussian pre-activations h_j. In practice, due to the summation over many (weakly dependent) terms, the output f will approximately follow a Gaussian distribution by the Central Limit Theorem [94–96], especially as the number of neurons N increases. For notational simplicity in the following derivations, we use $\boldsymbol{x}$ and $\boldsymbol{x}'$ to denote the full input vectors, corresponding to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in Eq. (11), while x_i or x_j refer to individual coordinates within these vectors.

Theorem 1. *Let $f(\boldsymbol{x}; \boldsymbol{\theta})$ denote the output of a cKAN with one hidden layer of width N, where all coefficients are initialized as independent standard normal random variables. The expected NTK between two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ is given by,*

$$\mathbb{E}[\boldsymbol{K}_{ntk}(\boldsymbol{x}, \boldsymbol{x}')] = N \cdot \left[\sum_{n=0}^{k} C_n(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i=1}^{d} \sum_{n=0}^{k} T_n(\tilde{x}_i) \cdot T_n(\tilde{x}_i') \cdot D(\boldsymbol{x}, \boldsymbol{x}') \right], \tag{29}$$

where $T_n(\cdot)$ denotes the Chebyshev polynomial of degree n, and $\tilde{x}_i = \tanh(x_i)$ represents the transformed input coordinate. The term $C_n(\boldsymbol{x}, \boldsymbol{x}')$ captures the correlation between the Chebyshev features of hidden activations and is defined as,

$$C_n(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[T_n(\tanh(h)) \cdot T_n(\tanh(h'))], \tag{30}$$

where the pair (h, h') follows a bivariate normal distribution with zero mean and covariance matrix,

$$\begin{bmatrix} \sigma^2(\boldsymbol{x}) & \rho(\boldsymbol{x}, \boldsymbol{x}') \\ \rho(\boldsymbol{x}, \boldsymbol{x}') & \sigma^2(\boldsymbol{x}') \end{bmatrix}, \tag{31}$$

and the second term, $D(\boldsymbol{x}, \boldsymbol{x}')$, accounts for the contribution of gradients through the activation function and is given by,

$$D(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\left[(1 - \tanh^2(h))(1 - \tanh^2(h')) \sum_{m=0}^{k} T_m'(\tanh(h)) \cdot T_m'(\tanh(h')) \right], \tag{32}$$

where (h, h') has the same distribution as above.

The proof of Theorem 1 can be found in Appendix A.

Remark 1. *The expected NTK increases linearly with the number of hidden neurons N. It is composed of two main components. The first part, known as the second-layer kernel, involves the terms C_n, which measure how similar two inputs are after passing through the nonlinear activation function* tanh *and being expanded using Chebyshev polynomials. The second part, referred to as the first-layer kernel, involves the term D, which captures how sensitive the network output is to changes in the first-layer coefficients; this sensitivity is expressed through derivatives of the Chebyshev polynomials. Importantly, both components depend only on the input data and their statistical properties, not on the specific random initialization of the network coefficients.*

As training progresses, the parameter vector $\boldsymbol{\theta}(\tau)$ deviates from its initial random value due to updates from gradient descent. Understanding how much the NTK changes over time, referred to as NTK drift, is crucial for analyzing the learning dynamics of finite-width cKANs.

Theorem 2. *Let $f(\boldsymbol{x}; \boldsymbol{\theta})$ be a cKAN parameterized by weights $\boldsymbol{\theta} \in \mathbb{R}^P$, and let the empirical NTK matrix at time τ be defined as,*

$$\boldsymbol{K}_{ntk}^{(\tau)}(\boldsymbol{x}, \boldsymbol{x}') := \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta}(\tau)), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}'; \boldsymbol{\theta}(\tau)) \rangle . \tag{33}$$

Assume the following:

(I) The parameters evolve under gradient flow,

$$\frac{d\boldsymbol{\theta}}{d\tau} = -\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}(\tau)), \tag{34}$$

where $\mathcal{L}$ is a smooth loss function defined in Eq. (8)*.*

(II) The gradients and Hessians of the network outputs are uniformly bounded,

$$\sup_{\boldsymbol{\theta}, \boldsymbol{x}} \|\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta})\| \leq B_1, \quad \sup_{\boldsymbol{\theta}, \boldsymbol{x}} \left\|\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{x}; \boldsymbol{\theta})\right\| \leq B_2, \tag{35}$$

for some constants $B_1, B_2 > 0$.

Then, the NTK matrix remains close to its initialization for a finite training time τ,

$$\|\boldsymbol{K}_{ntk}(\tau) - \boldsymbol{K}_{ntk}(0)\| \leq C \cdot \|\boldsymbol{\theta}(\tau) - \boldsymbol{\theta}(0)\| , \tag{36}$$

where $C = 2B_1B_2$. In particular, if the parameter change $\|\boldsymbol{\theta}(\tau) - \boldsymbol{\theta}(0)\|$ vanishes as the width $N \to \infty$, the NTK remains asymptotically constant during training.

The proof of Theorem 2 can be found in Appendix B.

Remark 2. *Theorem 2 implies that under gradient flow and boundedness conditions, the NTK matrix of a cKAN model remains nearly constant throughout training, especially in the infinite-width limit. This stability is a key assumption in many theoretical analyses of neural network training dynamics and justifies using the NTK at initialization to approximate the learning behavior of the model. In particular, cKAN, which is based on Chebyshev polynomial expansions, satisfies these conditions due to its smooth structure and bounded derivatives.*

3.2. Step II: NTK for cPIKANs

This section aims to derive the NTK formulation for the cPIKAN framework. Consider a well-posed partial differential equation defined over a bounded domain $\Omega \subset \mathbb{R}^d$, given by,

$$\mathcal{N}[u(\boldsymbol{x})](\boldsymbol{x}) = h(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \tag{37}$$

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega \tag{38}$$

where $\mathcal{N}$ is a differential operator, and $u : \Omega \to \mathbb{R}$ is the unknown function to be learned, with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$. For time-dependent problems, time t is included as an additional coordinate in $\boldsymbol{x}$, and Ω represents the combined space-time domain. In such cases, the initial condition is treated as a special case of a Dirichlet boundary condition. Following the vanilla physics-informed formulation, we approximate the solution u using a deep neural network $u(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the trainable parameters of the model. The PDE residual at any input $\boldsymbol{x}$ is defined as,

$$r(\boldsymbol{x}; \boldsymbol{\theta}) := \mathcal{N}[u](\boldsymbol{x}; \boldsymbol{\theta}) - h(\boldsymbol{x}). \tag{39}$$

The network parameters are optimized by minimizing a loss function composed of two parts: the data loss and the residual loss. Specifically, the total loss is given by,

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{N_d} \left|u(\boldsymbol{x}_i^d;\boldsymbol{\theta}) - g(\boldsymbol{x}_i^d)\right|^2}_{\mathcal{L}_d(\boldsymbol{\theta})} + \underbrace{\frac{1}{2}\sum_{i=1}^{N_r} |r(\boldsymbol{x}_i^r;\boldsymbol{\theta})|^2}_{\mathcal{L}_r(\boldsymbol{\theta})} \tag{40}$$

where $\mathcal{L}_d(\boldsymbol{\theta})$ is the loss from data or boundary/initial conditions, and $\mathcal{L}_r(\boldsymbol{\theta})$ is the residual loss from the PDE. Here, N_d and N_r denote the number of training points sampled from the boundary and/or initial conditions and the interior of the domain, respectively, with corresponding datasets defined as,

$$\mathcal{D}_d = \left\{(\boldsymbol{x}_i^d, g(\boldsymbol{x}_i^d))\right\}_{i=1}^{N_d}, \quad \mathcal{D}_r = \left\{(\boldsymbol{x}_i^r, h(\boldsymbol{x}_i^r))\right\}_{i=1}^{N_r}, \tag{41}$$

where $\boldsymbol{x}_i^d \in \partial\Omega$ and $\boldsymbol{x}_i^r \in \Omega$ indicate the sampled input locations for the data and residual loss terms, respectively.

To optimize the neural network, we seek the parameter vector $\boldsymbol{\theta}$ that minimizes the loss function $\mathcal{L}(\boldsymbol{\theta})$. A widely used approach for this task is gradient descent, which iteratively updates the parameters in the direction of the negative gradient of the loss. When the learning rate is taken to be infinitesimally small, the discrete update rule of gradient descent can be approximated by a continuous-time process known as gradient flow [57, 97]. In this regime, the evolution of the parameters is governed by the differential equation,

$$\frac{d\boldsymbol{\theta}(\tau)}{d\tau} = -\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}(\tau)) = -\nabla_{\boldsymbol{\theta}}\left(\mathcal{L}_d(\boldsymbol{\theta}(\tau)) + \mathcal{L}_r(\boldsymbol{\theta}(\tau))\right), \tag{42}$$

where τ is training time. This equation describes how the parameters $\boldsymbol{\theta}$ evolve smoothly along the direction of steepest descent of the loss function.

Lemma 1. *Let the training data be given by Eq.* (41)*. Under the gradient flow dynamics defined in Eq.* (42)*, the evolution of the network outputs $u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau))$ and $\mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau))$ with respect to the training time τ follows,*

$$\frac{d}{d\tau}\begin{bmatrix} u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau)) \\ \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau)) \end{bmatrix} = -\begin{bmatrix} K_{uu}(\tau) & K_{ur}(\tau) \\ K_{ru}(\tau) & K_{rr}(\tau) \end{bmatrix}\begin{bmatrix} u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau)) - g(\boldsymbol{x}_i^d) \\ \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau)) - h(\boldsymbol{x}_i^r) \end{bmatrix}, \tag{43}$$

where the kernel blocks $K_{uu}(\tau)$, $K_{ur}(\tau)$, $K_{ru}(\tau)$, and $K_{rr}(\tau)$ are components of the empirical NTK, given by inner products of the parameter gradients of the network outputs and residuals. Specifically,

$$\begin{aligned}
(K_{uu}(\tau))_{i,j} &= \left\langle \frac{\partial u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}}, \frac{\partial u(\boldsymbol{x}_j^d;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}} \right\rangle, \\
(K_{rr}(\tau))_{i,j} &= \left\langle \frac{\partial \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}}, \frac{\partial \mathcal{N}[u](\boldsymbol{x}_j^r;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}} \right\rangle, \\
(K_{ru}(\tau))_{i,j} &= \left\langle \frac{\partial \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}}, \frac{\partial u(\boldsymbol{x}_j^d;\boldsymbol{\theta}(\tau))}{\partial\boldsymbol{\theta}} \right\rangle,
\end{aligned} \tag{44}$$

where $\langle\cdot\rangle$ denotes the inner product taken over all neural network parameters $\boldsymbol{\theta} = \{c_n\}_{n=0}^k$, which represent the coefficients in the Chebyshev polynomial expansion.

The proof of Lemma 1 follows the same reasoning as the NTK analysis for PINNs presented in [60]. Accordingly, the matrix,

$$\boldsymbol{K}_{\text{ntk}}(\tau) = \begin{bmatrix} K_{uu}(\tau) & K_{ur}(\tau) \\ K_{ru}(\tau) & K_{rr}(\tau) \end{bmatrix}, \tag{45}$$

is referred to as the NTK of the cPIKAN. It characterizes the sensitivities of both the cKAN outputs $u(\boldsymbol{x}_i^d;\boldsymbol{\theta})$ and the physics-informed residuals $\mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\theta}$. These kernels capture how perturbations in the network parameters influence the model outputs and their corresponding residuals across the training data. The evolution equation,

$$\frac{d\boldsymbol{\psi}(\tau)}{d\tau} = -\boldsymbol{K}_{\text{ntk}}(\tau)\cdot(\boldsymbol{\psi}(\tau) - \boldsymbol{\mathcal{G}}) \tag{46}$$

describes a linearized training dynamic that holds broadly for physics-informed models under gradient flow, regardless of the specific neural network architecture used (e.g., fully connected networks, cKANs, Fourier networks). Here, $\boldsymbol{\psi}(\tau)$ is the stacked vector of network outputs corresponding to different components of the problem, including PDE residuals, initial and boundary conditions, and possibly data observations; $\boldsymbol{\mathcal{G}}$ contains

the corresponding target values; and $\boldsymbol{K}_{\text{ntk}}(\tau)$ is the NTK matrix, given by,

$$\boldsymbol{K}_{\text{ntk}}(\tau) = \begin{bmatrix} K_{\psi_1,\psi_1}(\tau) & K_{\psi_1,\psi_2}(\tau) & \cdots & K_{\psi_1,\psi_m}(\tau) \\ K_{\psi_2,\psi_1}(\tau) & K_{\psi_2,\psi_2}(\tau) & \cdots & K_{\psi_2,\psi_m}(\tau) \\ \vdots & \vdots & \ddots & \vdots \\ K_{\psi_m,\psi_1}(\tau) & K_{\psi_m,\psi_2}(\tau) & \cdots & K_{\psi_m,\psi_m}(\tau) \end{bmatrix}, \tag{47}$$

where each block $K_{\psi_i,\psi_j}(\tau)$ encodes the inner product of gradients of the network outputs ψ_i and ψ_j with respect to the model parameters. This formulation is flexible and extends naturally to a wide class of PDEs, including time-dependent and nonlinear problems, with varied boundary and initial conditions. It provides a unified view of training dynamics across different network types and physics constraints.

4. Results & Discussion

In this section, we present the results of four numerical experiments designed to evaluate the accuracy, computational efficiency, and learning dynamics of cPIKAN architectures, with particular emphasis on the eigenvalue behavior of NTK matrices under various settings. Since the computational domains in our experiments extend beyond the unit scale, we employ the scaled cPIKAN method as proposed in [54]. Each subsection targets a distinct aspect of model behavior or optimization strategy. Specifically, the experiments include: (I) a comparison between PINN and cPIKAN on the diffusion equation, (II) an evaluation of different optimization methods and model variants on the Helmholtz equation, (III) an analysis of NTK structure in temporally decomposed subdomains for the Allen–Cahn equation, and (IV) the evolution of learned hyperparameters in the context of a forced vibration problem. For each case, we report the network architecture, number of trainable parameters $|\boldsymbol{\theta}|$, relative $\mathcal{L}^2$ error (defined as $(\|u_{\text{pred}} - u_{\text{exact}}\|_2)/(\|u_{\text{exact}}\|_2)$), and the average computation time per training iteration, measured on a workstation equipped with an NVIDIA RTX 6000 Ada GPU. In addition, we provide visualizations of the ground truth, predicted solution, and absolute error over the computational domain, along with training curves for the loss and relative $\mathcal{L}^2$ error per iteration. The eigenvalue spectra of the NTK matrices, denoted by $\lambda(K_{\psi_i,\psi_j}(\tau))$, are also presented to offer further insights into convergence and learning dynamics. We note that in Experiments 4.1–4.3, the NTK matrix is defined according to Lemma 1 and Eq. (45). However, in Experiment 4.4, due to the presence of distinct boundary and initial conditions, the formulation of the NTK differs and is discussed separately within that example.

4.1. Experiment I: PINN and cPIKAN (Diffusion Equation)

In this example, we aim to investigate and compare the behavior of the eigenvalues of the NTK matrix for two approaches: the scaled versions of the PINN and the cPIKAN, as introduced in [54]. To this end, we consider the one-dimensional diffusion equation defined as,

$$\begin{aligned} &u_t(x,t) - D\, u_{xx}(x,t) = 0, \quad x \in [-6,6], t \in (0,1], \\ &u(-6,t) = u(6,t) = 0, \qquad t \in (0,1], \\ &u(x,0) = \sin(\pi x), \qquad x \in [-6,6], \end{aligned} \tag{48}$$

where D is the diffusion coefficient. The ground truth solution for this problem is known and given $u(x,t) = \sin(\pi x)\exp(-\pi^2 D t)$. This setup allows us to explore how the NTK spectrum evolves during training and how it reflects the learning dynamics and inductive biases of the two methods.

Table 1: Comparison of network configurations, number of trainable parameters $|\boldsymbol{\theta}|$, relative $\mathcal{L}^2$ errors (RE), and average computation time per iteration (in milliseconds) for solving the diffusion equation in Experiment 4.1 with diffusion coefficient $D = 0.1$. All models use equal residual and data loss weights.

Method	(N_l, N_n, k)	$\lvert\boldsymbol{\theta}\rvert$	$(N_{\text{r}}, N_{\text{d}})$	RE	Time
cPIKAN	(2, 8, 5)	462	(2000, 800)	$\mathbf{2.18 \times 10^{-3}}$	6
Parameter-based analysis					
PINN-I	(2, 19, -)	456	(2000, 800)	8.62×10^{-3}	2
Computation-time-based comparison					
PINN-II	(5, 100, -)	40802	(40000, 800)	1.46×10^{-2}	10

Before analyzing the eigenvalues of the NTK matrices, we present a summary of the training configurations and performance metrics for the compared models in Table 1. The PINN architecture is evaluated in two

settings: PINN-I is configured to match the number of trainable parameters with cPIKAN (parameter-based comparison), while PINN-II is designed to have a similar computation time per iteration (time-based comparison). As shown, cPIKAN achieves the lowest relative $\mathcal{L}^2$ error (2.18×10^{-3}) while maintaining a moderate computational cost (6 ms per iteration), offering approximately 4 $\times$ higher accuracy than PINN-I and nearly 7$\times$ higher accuracy than PINN-II, despite significantly fewer parameters and lower sampling requirements. These results highlight the superior efficiency and precision of cPIKAN in solving the diffusion problem.

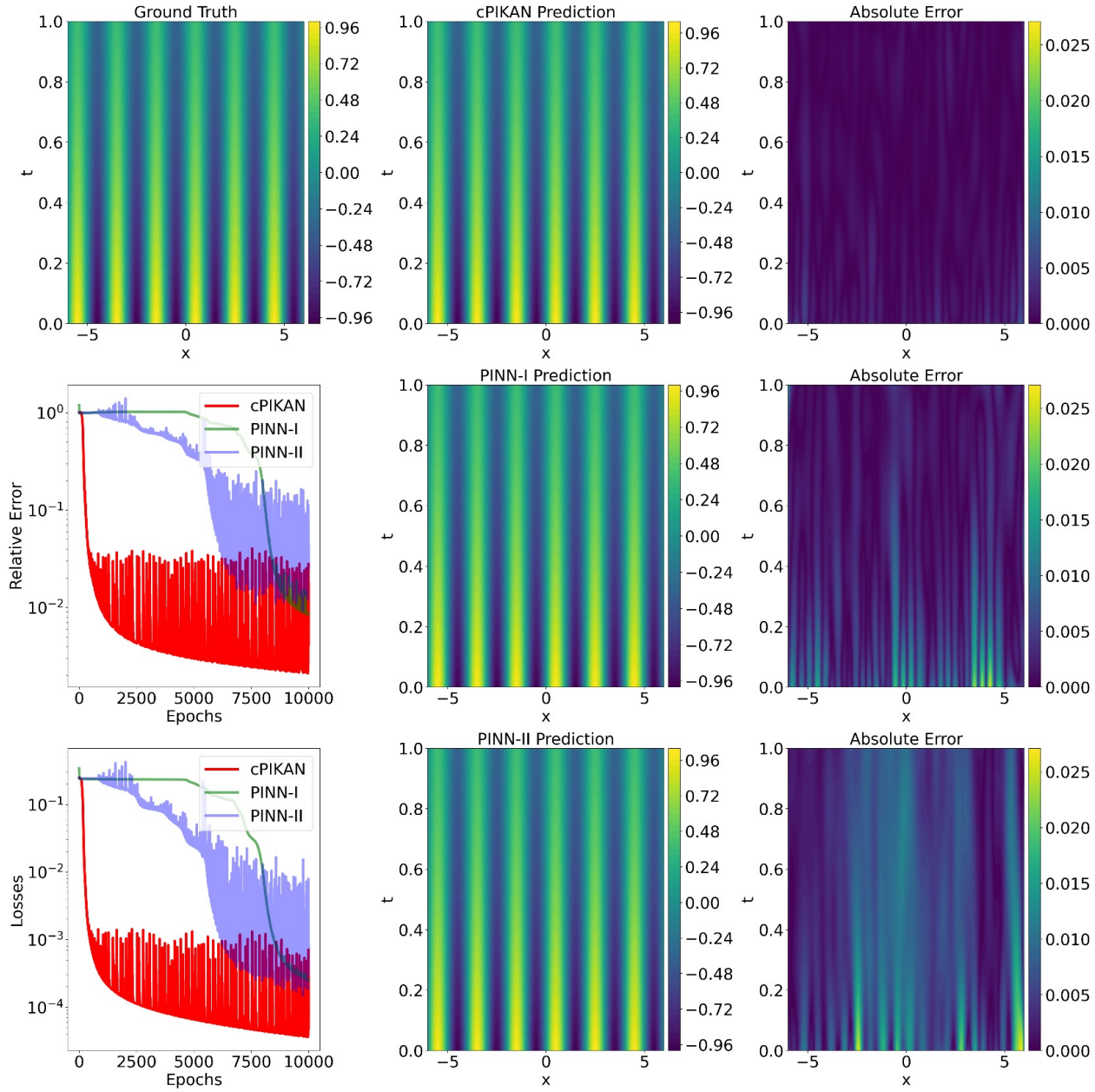

Figure 1: Comparison of the predicted solutions for the diffusion equation in Experiment 4.1, demonstrating that cPIKAN yields significantly higher accuracy, with a maximum absolute error of only 7.52×10^{-3}, compared to PINN-I and PINN-II, whose errors exceed 2.5×10^{-2}. cPIKAN also exhibits faster convergence, lower final relative $\mathcal{L}^2$ error, and more stable training dynamics. These findings highlight the superior performance of cPIKAN in both predictive accuracy and optimization efficiency for solving the diffusion problem.

In Fig. 1, we evaluate and compare the performance of cPIKAN, PINN-I, and PINN-II in solving the one-dimensional diffusion equation. The predicted solutions and corresponding absolute error distributions indicate that cPIKAN achieves the highest accuracy, with a maximum absolute error of 7.52×10^{-3}, which is substantially lower than that of PINN-I (2.53×10^{-2}) and PINN-II (2.70×10^{-2}). The plots in the bottom-left of the figure depict the evolution of the relative $\mathcal{L}^2$ error and the total training loss as functions of training epochs. These curves show that cPIKAN not only attains the smallest final relative $\mathcal{L}^2$ error but also demonstrates faster convergence and greater training stability compared to the standard PINN configurations. This highlights the effectiveness of cPIKAN in both accuracy and optimization dynamics in solving Experiment 4.1.

Figure 2 illustrates the temporal evolution of the NTK eigenvalue spectra during training for three architectures: cPIKAN, PINN-I, and PINN-II. Each columns represent the eigenvalues of the full NTK matrix K_{ntk}, the data-data submatrix K_{uu}, and the residual-residual submatrix K_{rr}, respectively. For all models,

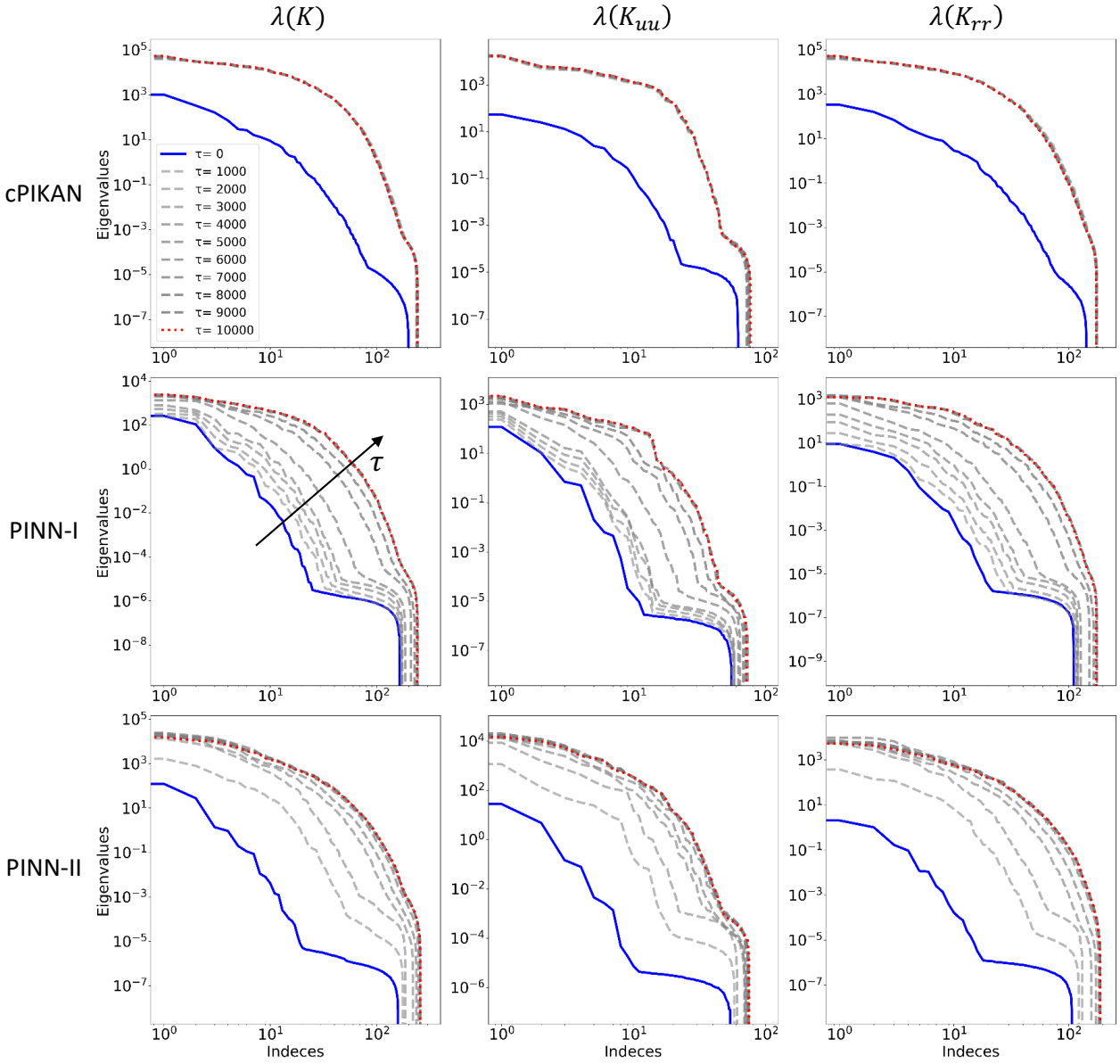

Figure 2: Evolution of the NTK eigenvalue spectra during training for the diffusion equation in Experiment 4.1. The NTK spectrum in cPIKAN gradually converges during training, indicating stable learning dynamics and effective optimization. In contrast, the spectra for PINN-I and PINN-II remain dispersed, suggesting unstable behavior and poor information flow. These differences highlight the improved convergence and robustness of the cPIKAN model.

the spectra are plotted over a range of training iterations from $\tau = 0$ (blue) to $\tau = 10^4$ (red). The plots reveal distinct spectral dynamics across models. Notably, cPIKAN maintains a significantly broader and more persistent spectrum throughout training. This indicates enhanced expressivity and better signal propagation through both data and residual losses, facilitating stable convergence. In contrast, the spectra of PINN-I and PINN-II rapidly collapse towards lower magnitudes, especially in the residual blocks. This decay reflects limited capacity to represent gradient information and suggests poor learning signal transmission in deeper iterations. Moreover, the eigenvalue spectrum of K_{rr} for cPIKAN retains higher magnitudes across more modes compared to PINN-I and PINN-II. This implies that cPIKAN better preserves the diversity of learning directions in the residual space, which contributes to its superior performance in both accuracy and convergence behavior observed in earlier figures.

4.2. Experiment II: Different Optimizations (Helmholtz Equation)

In this example, we assess the performance of various models, including the scaled version of cPIKAN, standard PINN, and the B-spline-based PINN (bPIKAN) introduced in [54], for solving the two-dimensional Helmholtz equation with homogeneous Dirichlet boundary conditions,

$$\begin{aligned} &u_{xx}(x,y) + u_{yy}(x,y) + \kappa^2 u(x,y) = f(x,y), \quad (x,y) \in \Omega, \\ &u(x,y) = 0, \qquad\qquad\qquad\qquad\quad (x,y) \in \partial\Omega, \end{aligned} \tag{49}$$

where the domain is $\Omega = [-4,4]^2$. The source term is defined as $f(x,y) = \left(\kappa^2 - (a_1^2 + a_2^2)\pi^2\right)\sin(a_1\pi x)\sin(a_2\pi y)$, with the corresponding exact solution $u(x,y) = \sin(a_1\pi x)\sin(a_2\pi y)$, which satisfies the prescribed boundary conditions. The parameters a_1, a_2, and κ control the oscillatory behavior of the solution. In this test, we use $(a_1, a_2, \kappa) = (1.0, 1.0, 1.0)$, yielding a smooth yet non-trivial benchmark. In addition to comparing model architectures, we also investigate the effect of different optimization strategies (e.g., ADAM, LBFGS, a hybrid scheme) on the NTK spectra and training dynamics of cPIKAN. The LBFGS optimizer is configured with a strong Wolfe line search to promote stable and efficient convergence.

Table 2: Comparison of network configurations, total number of trainable parameters $|\boldsymbol{\theta}|$, relative $\mathcal{L}^2$ errors (RE), and per-iteration computational time (in milliseconds) for solving the Helmholtz equation described in Experiment 4.2. All models are trained with equal residual and data loss weights and use $N_\mathrm{r} = N_\mathrm{d} = 4000$ training points. The first three rows correspond to models trained with the ADAM optimizer. The bottom section reports the performance of cPIKAN under alternative optimization strategies, including LBFGS and a hybrid ADAM+LBFGS scheme.

Method	(N_l, N_n, k)	$\lvert\boldsymbol{\theta}\rvert$	RE	Time
cPIKAN	(4,25,3)	7800	$\mathbf{6.61 \times 10^{-2}}$	18
PINN	(4, 50, -)	7850	2.65×10^{0}	4
bPIKAN	(4, 16, 3) & 5 grid	7344	1.67×10^{0}	30
Other Optimization of cPIKAN				
LBFGS	(4,25,3)	7800	$\mathbf{4.76 \times 10^{-3}}$	18
ADAM + LBFGS	(4,25,3)	7800	$\mathbf{5.03 \times 10^{-3}}$	18

Table 2 summarizes the performance of different models in solving the Helmholtz equation introduced in Experiment 4.2, comparing their accuracy, computational efficiency, and network complexity. The first part of this table includes results for cPIKAN, PINN, and bPIKAN, all trained using the ADAM optimizer. Among these, cPIKAN achieves the best accuracy (relative $\mathcal{L}^2$ error of 6.61×10^{-2}) at a moderate computational cost of 18 ms per iteration. In contrast, standard PINN is significantly faster (4 ms per iteration) but exhibits much lower accuracy. The bPIKAN model yields a relatively high error, which may stem from its sensitivity to domain scaling; it likely requires different configurations or learning strategies to handle problems defined over larger domains effectively.

The lower section of the table presents the results for cPIKAN under alternative optimization schemes, including LBFGS and a hybrid ADAM+LBFGS approach. Both methods yield substantial improvements in accuracy, reducing the error by over an order of magnitude compared to the ADAM-only version, while maintaining the same per-iteration computational cost. This highlights the potential of optimization strategy in enhancing the training efficiency and precision of physics-informed models.

Figure 3 depicts a comparison of three different models, cPIKAN, PINN, and bPIKAN, for solving the Helmholtz equation (Experiment 4.2). Among the three, cPIKAN demonstrates accurate performance, closely matching the ground truth both qualitatively and quantitatively, with negligible absolute error. PINN can capture the general structure of the solution, including the location of peaks and troughs, but it significantly misestimates their amplitudes, leading to larger absolute errors. In contrast, bPIKAN fails to solve the problem accurately under the given settings, as its prediction deviates substantially from the true solution in both shape and scale. The plot of relative $\mathcal{L}^2$ error over training epochs shows that cPIKAN achieves the lowest error and exhibits a consistent downward trend, indicating stable convergence. PINN shows slower convergence with a higher final error, and bPIKAN maintains a high relative $\mathcal{L}^2$ error throughout, suggesting poor learning. The loss curves reinforce this behavior: cPIKAN maintains the lowest loss, converging smoothly, while PINN and bPIKAN show higher and less stable losses, with bPIKAN showing clear signs of underfitting or poor training dynamics.

In Fig. 4, the evolution of the eigenvalue spectra of the NTK matrix is shown for cPIKAN (top row), PINN (middle row), and bPIKAN (bottom row) throughout the training process. For cPIKAN, the spectra of all NTK-related matrices remain stable and converge smoothly across training iterations. This consistent spectral behavior aligns with the model's strong performance observed earlier. In contrast, for PINN, the

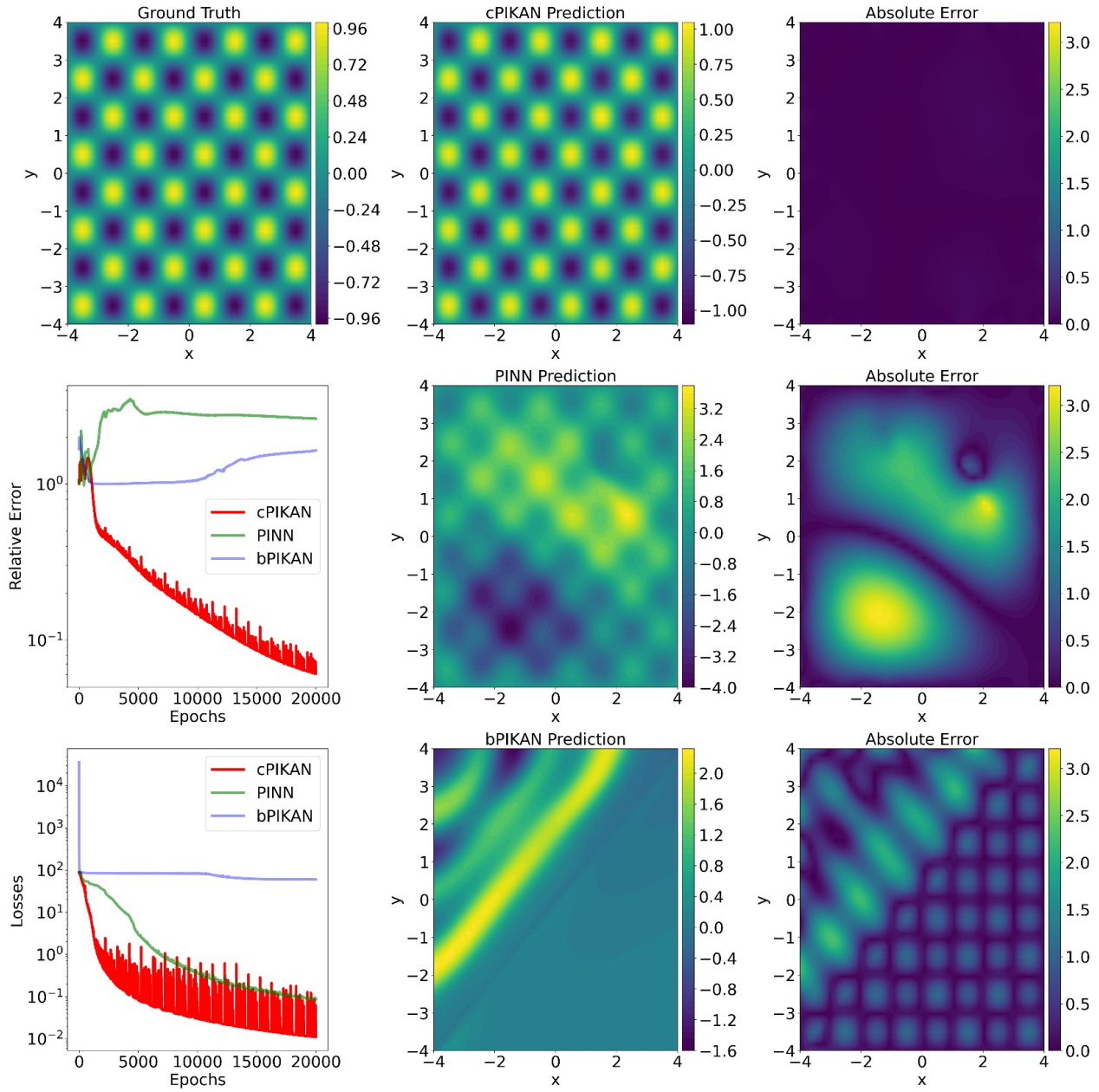

Figure 3: Comparison of the predicted solutions for the Helmholtz equation in Experiment 4.2, showing that cPIKAN produces highly accurate predictions with minimal absolute error, closely matching the ground truth. PINN captures the overall structure but significantly misestimates the amplitude, while bPIKAN fails to approximate the solution correctly. The training curves further confirm these findings. cPIKAN achieves the lowest relative $\mathcal{L}^2$ error and loss with stable convergence, whereas PINN converges more slowly with higher error, and bPIKAN shows poor learning behavior and fails to converge effectively.

spectra start to stabilize only after several thousand training steps, and convergence is only observed toward the final epochs. This delayed spectral alignment is consistent with its slower convergence and moderate accuracy. For bPIKAN, the eigenvalue spectra are highly disordered and do not show clear convergence, indicating instability in training. This chaotic behavior supports the earlier observation that bPIKAN fails to accurately approximate the solution to the Helmholtz equation under the given configuration.

We examine the behavior of the NTK matrix under different optimization strategies for the cPIKAN method, as shown in Figs. 5-6. The corresponding network architecture used in all experiments is summarized in Table 2. Figure 5 compares the predicted solutions of the Helmholtz equation using three optimizers: ADAM, LBFGS, and a combination of ADAM followed by LBFGS. Visually, the predictions from LBFGS and ADAM+LBFGS show excellent agreement with the ground truth, with very low absolute error across the domain. In contrast, ADAM alone yields a noticeably higher absolute error, particularly near the center of the domain. The relative $\mathcal{L}^2$ error plot shows that ADAM+LBFGS achieves the lowest error and fastest convergence, followed by LBFGS. ADAM alone converges more slowly and to a higher final error. This trend is also reflected in the loss curves: both LBFGS and ADAM+LBFGS exhibit smooth and steep loss decay, while ADAM shows oscillations and slower reduction.

Figure 6 shows the evolution of the eigenvalue spectra of the NTK matrix during the training process for solving the Helmholtz equation, evaluated at different training steps. It compares three optimization strategies:

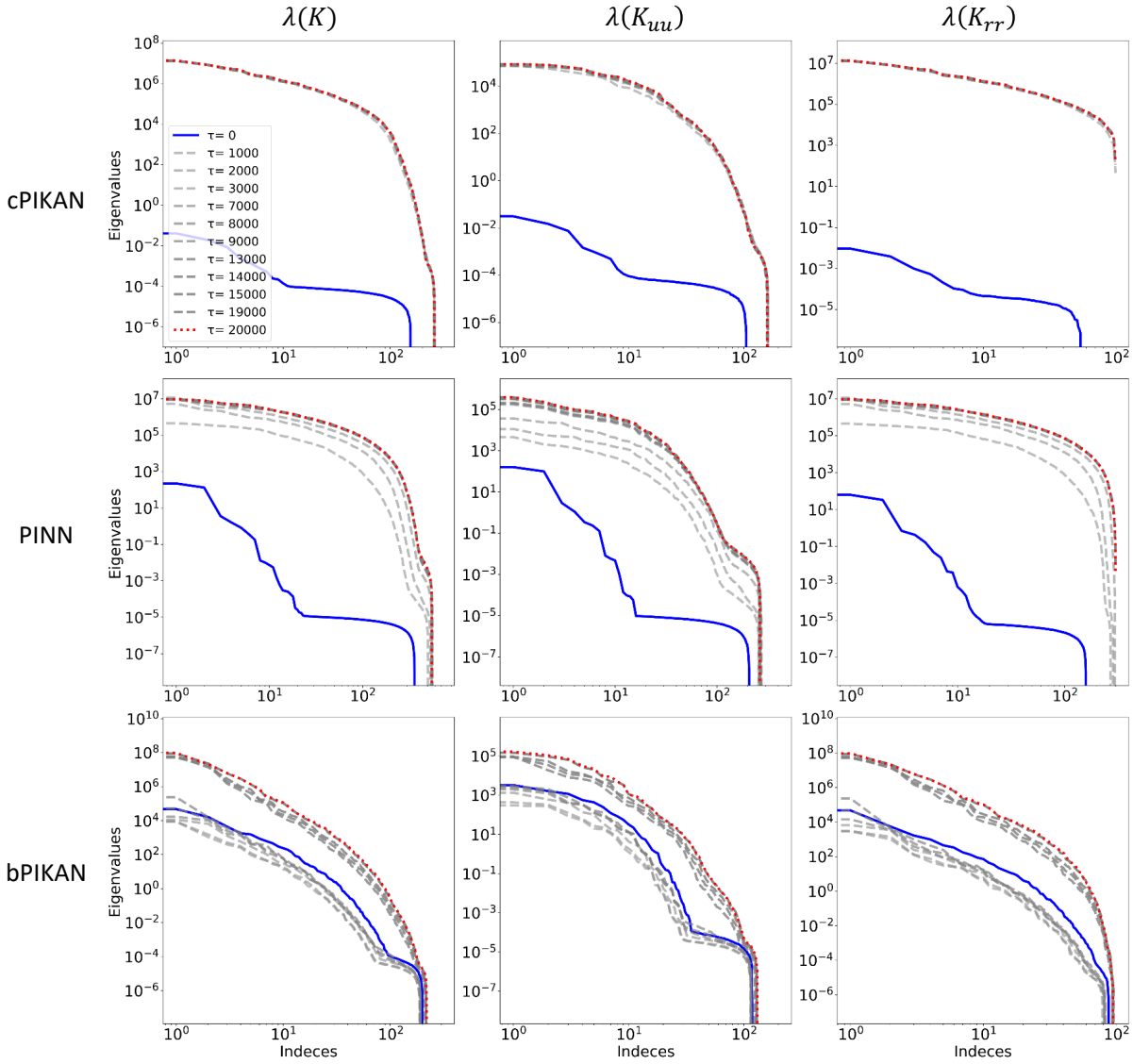

Figure 4: Evolution of the NTK eigenvalue spectra during training for the Helmholtz equation in Experiment 4.2. The spectra for cPIKAN remain stable and gradually converge throughout training, reflecting well-conditioned dynamics and supporting its strong predictive performance. PINN shows delayed spectral stabilization, consistent with its slower convergence and moderate accuracy. In contrast, bPIKAN exhibits disordered and non-converging spectra, indicating unstable learning and poor approximation capability.

ADAM (left), L-BFGS (center), and hybrid started with ADAM and followed by LBFGS (right). While all methods lead to changes in the NTK spectrum over time, LBFGS and hybrid approaches demonstrate a more rapid and structured shift in the eigenvalues, especially in the early stages of training. This behavior indicates effective optimization and better conditioning of the NTK matrix, which aligns with their strong performance in terms of both loss and relative $\mathcal{L}^2$ error. Although ADAM shows a smoother and more gradual evolution of the eigenvalues, its final solution is less accurate than the other two methods. This suggests that a well-conditioned NTK spectrum alone does not guarantee high solution accuracy; the choice of optimizer plays a crucial role in how effectively the model explores the solution space.

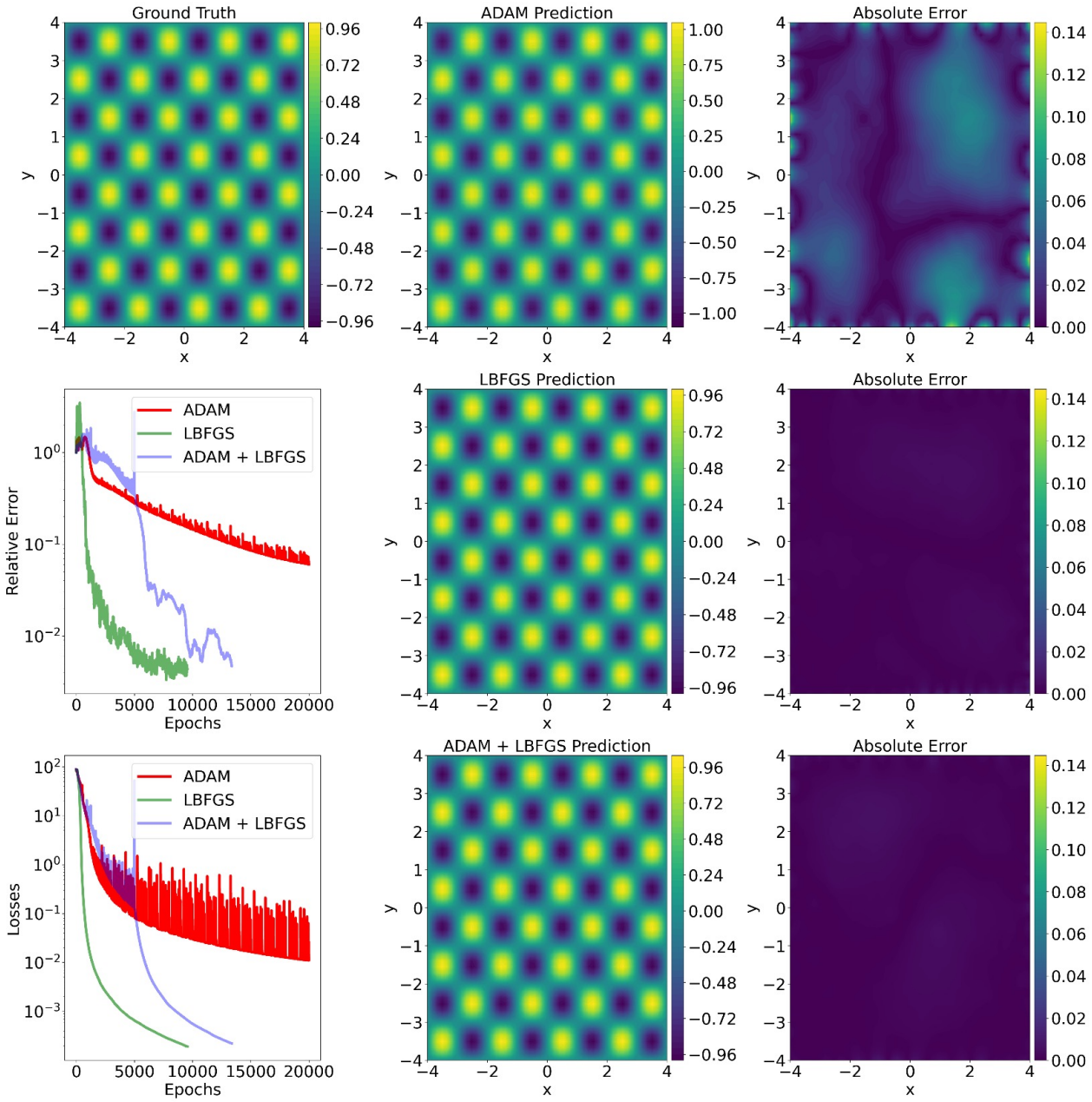

Figure 5: Comparison of predicted solutions for the Helmholtz equation using different optimization strategies in Experiment 4.2. It is shown that combining ADAM with LBFGS leads to the most accurate and stable solution, achieving the lowest absolute and relative errors. LBFGS alone also performs well, while ADAM alone results in higher errors, especially near the domain center. Training curves confirm these observations. LBFGS converges fastest and most smoothly, followed by ADAM+LBFGS, whereas ADAM shows slower convergence and unstable loss behavior. These results highlight the importance of optimizer choice for achieving reliable and accurate training in cPIKAN.

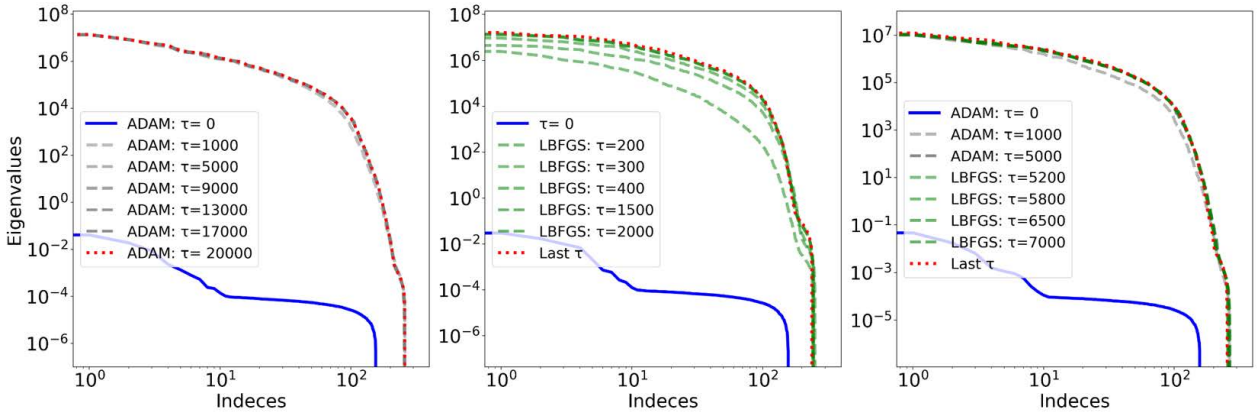

Figure 6: Evolution of the NTK eigenvalue spectra during training for the Helmholtz equation in Experiment 4.2, using three different optimization strategies.

4.3. Experiment III: NTK Behavior in Subdomains (Allen-Cahn Equation)

In this example, we investigate the behavior of the NTK when solving the Allen–Cahn equation using the cPIKAN method. We aim to explore whether splitting the domain into smaller temporal subdomains can lead to a more convergent or structured NTK matrix. This could provide valuable insight into the potential of domain decomposition for improving learning stability and efficiency in physics-informed models.
We consider the Allen–Cahn equation, which is commonly used to model phase separation in multi-component systems. The equation is given by,

$$\begin{aligned} u_t - D\,u_{xx} + 5\left(u^3 - u\right) = 0, \quad & x \in [-6, 6],\, t \in (0, 1], \\ u(-6, t) = u(6, t) = -1, \quad & t \in (0, 1], \\ u(x, 0) = (x/6)^2 \cos\left(\pi\, x/6\right), \quad & x \in [-6, 6], \end{aligned} \tag{50}$$

where u is the state variable and $D = 10^{-4}$ is the diffusion coefficient. The nonlinear term $5(u^3 - u)$ governs the local phase dynamics. The initial and boundary conditions are chosen to ensure well-posedness.

Table 3: Network configurations, number of trainable parameters $|\boldsymbol{\theta}|$, number of training data, relative $\mathcal{L}^2$ errors (RE), and per-iteration computational time (in milliseconds) for solving the Allen–Cahn equation (Experiment 4.3) using the cPIKAN method. Each row corresponds to a different number of temporal subdomains. The listed settings are applied to each subdomain individually.

# Subdomains	(N_l, N_n, k)	$\lvert\boldsymbol{\theta}\rvert$	$(N_\mathrm{r}, N_\mathrm{d})$	RE	Time
1 Sub.	(4, 27, 3)	9072	(8000, 6200)	5.09×10^{-1}	40
2 Subs.	(4, 20, 3)	5040	(4000, 3200)	7.90×10^{-3}	20
4 Subs.	(4, 15, 3)	2880	(2000, 1700)	6.21×10^{-3}	20

In this experiment, the temporal domain is divided into multiple subdomains to investigate the effect of domain decomposition on training performance using the cPIKAN method. The time interval is split into 1, 2, or 4 equal subdomains. Each subdomain, denoted by Ω_i, is rescaled following the scaling described in [54], allowing the learning process to be applied consistently within each interval. The predicted solution at the final time of subdomain Ω_i is used as the initial condition for the next subdomain Ω_{i+1}, and this procedure continues until the final subdomain is reached.

The results for different numbers of subdomains are reported in Table 3, along with the corresponding network configurations. The networks are designed such that the total number of trainable parameters and training data remains consistent across the entire domain. For example, when the domain is split into 2 subdomains, each subdomain uses 4000 residual points and a network with 5040 parameters. This results in a total of 8000 residual points and 10080 parameters, which is comparable to the case with 1 subdomain (8000 residual points and 9072 parameters).

As shown in the table, using 2 or 4 subdomains significantly reduces the relative $\mathcal{L}^2$ error compared to training a single network over the full domain. In particular, the error drops from 5.09×10^{-1} (1 subdomain) to 7.90×10^{-3} (2 subdomains) and further to 6.21×10^{-3} (4 subdomains). Additionally, the required runtime per iteration is reduced from 40 milliseconds in the single-domain case to 20 milliseconds when using subdomain decomposition, demonstrating both improved accuracy and computational efficiency. In Fig. 7, we evaluate the performance of the proposed cPIKAN method when applied to the Allen–Cahn equation using different numbers of temporal subdomains. The predicted solutions and corresponding absolute errors demonstrate that increasing the number of subdomains significantly improves accuracy. For example, the maximum absolute error decreases from 1.44 in the single-domain case to 0.09 with 2 subdomains and further to 0.0675 with 4 subdomains. This improvement is also reflected in the relative error plots (left column), where the error curves for the multi-subdomain cases show a consistent downward trend, reaching lower magnitudes than the single-domain counterpart. Notably, in the 2- and 4-subdomain settings, the relative error sharply drops at the start of each subdomain, indicating efficient learning within localized temporal regions. These observations confirm that temporal domain decomposition not only enhances convergence but also results in a more accurate and stable solution across the entire spatiotemporal domain.

Figures 8 and 9 illustrate the evolution of the NTK eigenvalue spectra during training of the cPIKAN method for the Allen–Cahn equation. As the number of temporal subdomains increases, the NTK matrices become increasingly well-conditioned and converge more rapidly. This behavior highlights how domain decomposition mitigates spectral bias and enhances the learning dynamics of cPIKAN. The strategy reduces the complexity of the target function in each sub-network, leading to more stable training and improved predictive accuracy. This confirms that spectral bias is a critical factor limiting performance in the full domain case, and domain decomposition offers a practical remedy, especially in diffusion-dominated regimes like Allen–Cahn.

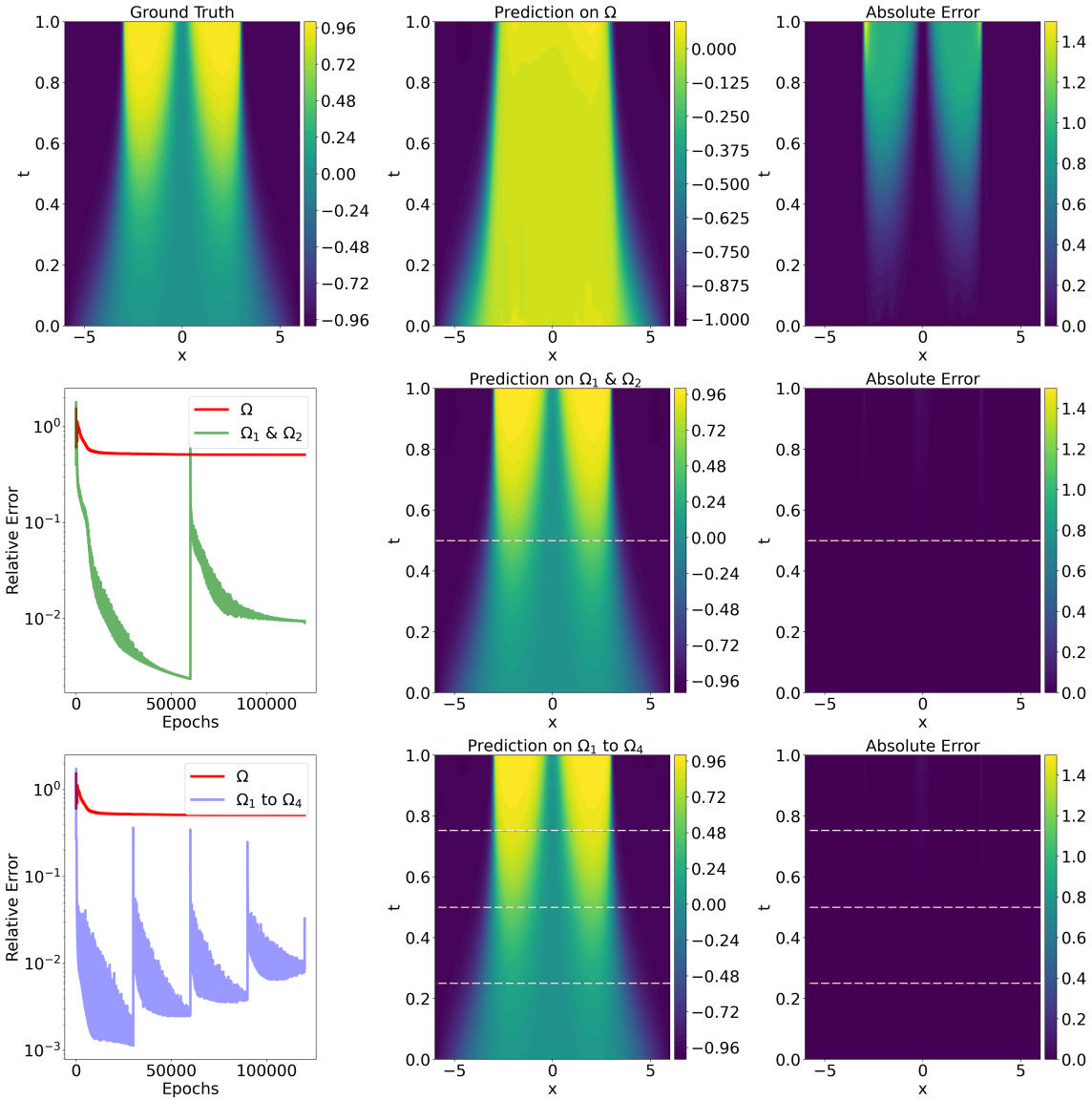

Figure 7: Comparison of the predicted solutions for the Allen–Cahn equation (Experiment 4.3) using the cPIKAN method with varying numbers of temporal subdomains. It is shown that increasing the number of subdomains significantly enhances prediction accuracy and convergence. Specifically, the maximum absolute error drops from 1.44 (single domain) to 0.09 and 0.0675 with 2 and 4 subdomains, respectively. The corresponding relative error curves reveal faster and more stable convergence in the multi-subdomain settings, especially evident at the onset of each subdomain.

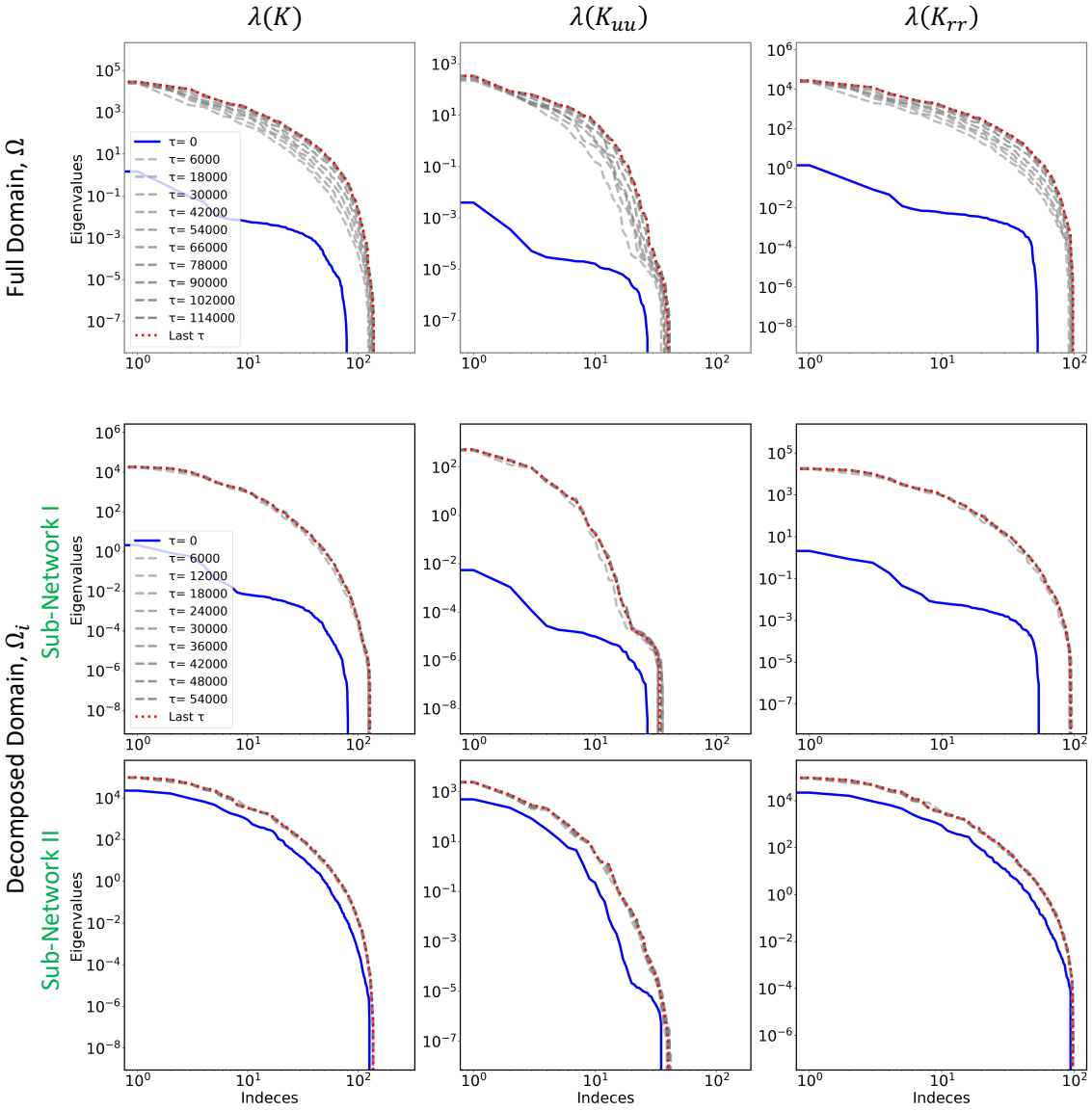

Figure 8: Evolution of the NTK eigenvalue spectra during training of the cPIKAN method for solving the Allen–Cahn equation (Experiment 4.3). The comparisons show that increasing the number of temporal subdomains leads to faster spectral convergence and better conditioning of key NTK blocks. This improved behavior reflects more efficient learning dynamics and reduced spectral bias.

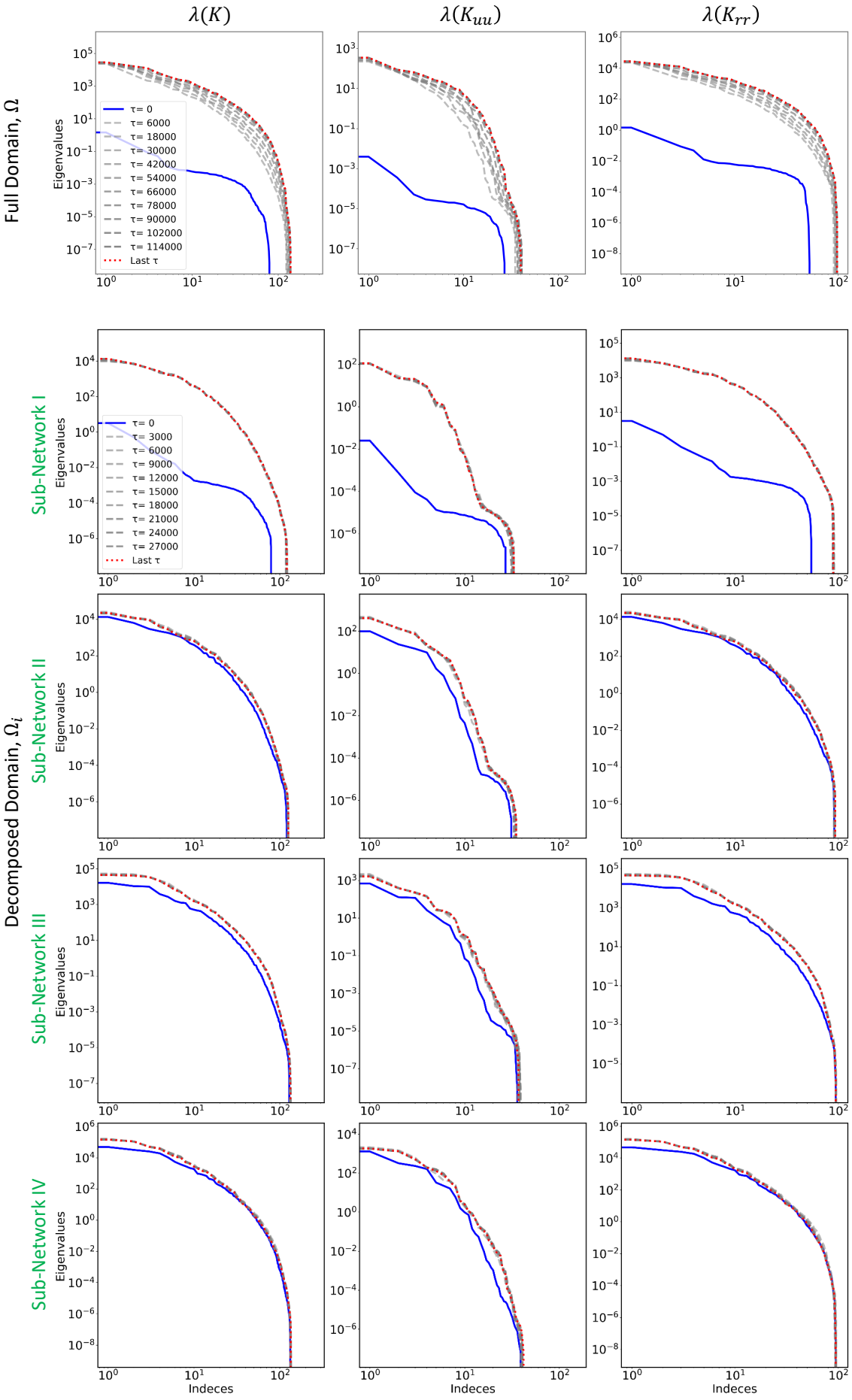

Figure 9: Evolution of the NTK eigenvalue spectra during training of the cPIKAN method for solving the Allen–Cahn equation (Experiment 4.3). The comparison shows that increasing the number of temporal subdomains leads to faster spectral convergence and better conditioning of key NTK blocks. This improved behavior reflects more efficient learning dynamics and reduced spectral bias.

4.4. Experiment IV: High-Order Dynamics and Enriched NTK Structure (forced vibration Equation)

We now consider a forced vibration problem governed by the Euler–Bernoulli beam equation with damping. This example is used to highlight the structural complexity of the associated NTK matrix, which differs from previous cases due to the inclusion of higher-order temporal and spatial derivatives. The governing equation is given by,

$$D_f \frac{\partial^4 u(x,t)}{\partial x^4} + c_d \frac{\partial u(x,t)}{\partial t} + \rho_l \frac{\partial^2 u(x,t)}{\partial t^2} = p(x,t), \quad (x,t) \in (0,l) \times (0,T], \tag{51}$$

where D_f is the flexural stiffness, c_d is the damping coefficient, ρ_l is the mass per unit length, p is the external excitation force, and $u(x,t)$ is the displacement field. This equation is subject to the following boundary and initial conditions,

$$\begin{cases} u(0,t) = u(l,t) = 0, \\ \dfrac{\partial^2 u}{\partial x^2}(0,t) = \dfrac{\partial^2 u}{\partial x^2}(l,t) = 0, \end{cases} \quad t \in [0,T], \quad \text{and} \quad \begin{cases} u(x,0) = 0, \\ \dfrac{\partial u}{\partial t}(x,0) = 0, \end{cases} \quad x \in (0,l). \tag{52}$$

The parameters of the beam used in this example are as follows: $l = 1$ m, $\rho_l = 1$ kg/m, $D_f = 2$ N.m^2, $c_d = 3$ N.s/m^2. The excitation force is $p(x,t) = P \sin(\pi x/l)\cos(2\pi f t)$, with an amplitude of $P = 0.1$ N/m and a frequency of $f = 2.7$ Hz.

In contrast to the previous examples, the NTK matrix in this case includes additional diagonal blocks corresponding to the temporal derivative u_t and the second-order spatial derivative u_{xx}, in addition to the standard terms K_{uu} and K_{rr}. Within the physics-informed learning framework, we define the stacked network output $\boldsymbol{\psi}(\tau)$, the corresponding target vector $\boldsymbol{\mathcal{G}}$, and the associated NTK matrix $\boldsymbol{K}_{\text{ntk}}(\tau)$ in Eq. (46) as follows,

$$\boldsymbol{\psi}(\tau) = \begin{bmatrix} u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau)) \\ u_t(\boldsymbol{x}_i^0;\boldsymbol{\theta}(\tau)) \\ u_{xx}(\boldsymbol{x}_i^b;\boldsymbol{\theta}(\tau)) \\ \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau)) \end{bmatrix}, \quad \boldsymbol{\mathcal{G}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{p}(\boldsymbol{x}_i^r) \end{bmatrix}, \quad \boldsymbol{K}_{\text{ntk}}(\tau) = \begin{bmatrix} K_{u,u} & K_{u,u_t} & K_{u,u_{xx}} & K_{u,r} \\ K_{u_t,u} & K_{u_t,u_t} & K_{u_t,u_{xx}} & K_{u_t,r} \\ K_{u_{xx},u} & K_{u_{xx},u_t} & K_{u_{xx},u_{xx}} & K_{u_{xx},r} \\ K_{r,u} & K_{r,u_t} & K_{r,u_{xx}} & K_{r,r} \end{bmatrix}, \tag{53}$$

in which the diagonal blocks are computed as inner products of gradients with respect to the model parameters,

$$\begin{aligned} (K_{u,u})_{i,j} &= \left\langle \frac{\partial u(\boldsymbol{x}_i^d;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}}, \frac{\partial u(\boldsymbol{x}_j^d;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}} \right\rangle, \qquad (K_{u_{xx},u_{xx}})_{i,j} = \left\langle \frac{\partial u_{xx}(\boldsymbol{x}_i^b;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}}, \frac{\partial u_{xx}(\boldsymbol{x}_j^b;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}} \right\rangle, \\ (K_{u_t,u_t})_{i,j} &= \left\langle \frac{\partial u_t(\boldsymbol{x}_i^0;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}}, \frac{\partial u_t(\boldsymbol{x}_j^0;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}} \right\rangle, \quad (K_{r,r})_{i,j} = \left\langle \frac{\partial \mathcal{N}[u](\boldsymbol{x}_i^r;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}}, \frac{\partial \mathcal{N}[u](\boldsymbol{x}_j^r;\boldsymbol{\theta}(\tau))}{\partial \boldsymbol{\theta}} \right\rangle. \end{aligned} \tag{54}$$

leading to a larger and more structured NTK, capturing richer dynamics of the system and posing a more complex challenge for the learning process.

Table 4: Network configurations, number of trainable parameters $|\boldsymbol{\theta}|$, number of residual training data, number of epochs in training each subnetwork, relative $\mathcal{L}^2$ errors (RE), and per-iteration computational time (in milliseconds) for solving the forced vibration equation (Experiment 4.4) when $T = 10$ (s) using the (a) cPIKAN and (b) PINN method. Each row corresponds to a different number of temporal subdomains. The listed settings are applied to each subdomain individually. Each model is trained with 400 points for initial conditions and 400 for boundary conditions.

(a) cPIKAN method						
# Subdomains	(N_l, N_n, k)	$\lvert\boldsymbol{\theta}\rvert$	N_{r}	# epochs	RE	Time
1 Sub.	(4, 47, 5)	40608	40000	10^6	7.64×10^0	660
2 Subs.	(4, 34, 5)	21420	20000	5×10^5	9.57×10^0	240
4 Subs.	(4, 24, 5)	10800	10000	25×10^4	3.55×10^{-2}	83
8 Subs.	(4, 17, 5)	5508	5000	125×10^3	7.06×10^{-3}	74
(b) PINN method						
# Subdomains	(N_l, N_n)	$\lvert\boldsymbol{\theta}\rvert$	N_{r}	# epochs	RE	Time
8 Sub.	(4, 40)	5080	5000	125×10^3	7.79×10^{-1}	16

Similar to the previous experiment (Experiment 4.3), the temporal domain in this experiment is also divided into multiple subdomains, each denoted by Ω_i and individually rescaled. The solution at the final time of Ω_i is used as the initial condition for Ω_{i+1}, enabling sequential training across subdomains. Table 4 summarizes the performance of the cPIKAN and PINN methods in solving the forced vibration equation over

a time interval $[0, T]$ with final time $T = 10$ seconds, using different numbers of temporal subdomains. The settings for each subnetwork are chosen such that the total number of parameters, training data, and training iterations across the entire domain remain consistent. For the cPIKAN method, increasing the number of subdomains leads to a significant improvement in accuracy and a notable reduction in training time. According to the results, the cPIKAN method performs poorly when using only 1 (i.e., full domain) or 2 subdomains, as indicated by the large relative errors. In these cases, the network fails to provide an accurate approximation of the solution, despite requiring a substantial amount of training time. The performance improves significantly when the domain is divided into 4 subdomains, leading to both lower error and reduced computational time. Further decomposition into 8 subdomains yields additional benefits. Specifically, the relative error decreases by approximately 80% compared to the 4-subdomain case, and the total training time is reduced by about 11%. This demonstrates that increasing the number of subdomains not only enhances accuracy but also improves training efficiency. This highlights the advantage of temporal decomposition in enhancing both accuracy and time. When using 8 subdomains, the cPIKAN method delivers a solution that is approximately 99% more accurate than the standard PINN approach. Although cPIKAN requires more training time, this additional cost results in a dramatic improvement in accuracy, making it a highly effective choice for problems where precision is critical. These results demonstrate that cPIKAN successfully achieve high-resolution solutions that PINNs cannot match, even under the same training settings per subdomain.

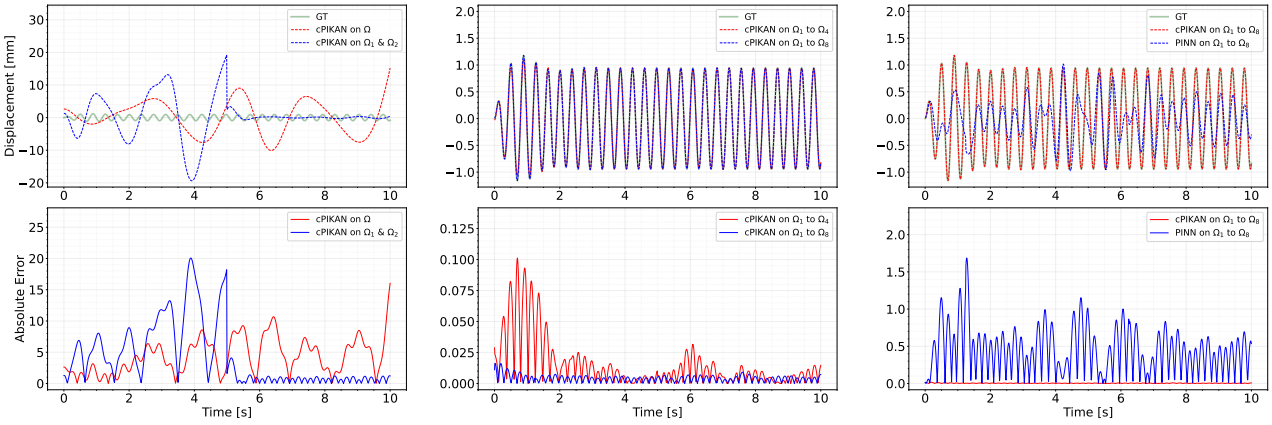

Figure 10: Comparison of predicted solutions for the forced vibration equation (Experiment 4.4) using cPIKAN with varying numbers of temporal subdomains, demonstrating that increasing the number of subdomains dramatically improves accuracy. While configurations with 1 or 2 subdomains yield large errors (up to 20.07), using 4 and 8 subdomains reduces the maximum absolute error to 0.102 and 0.016, respectively, an 84% improvement. Moreover, cPIKAN with 8 subdomains outperforms PINN with the same decomposition by approximately 98%, confirming that the temporal domain decomposition strategy enhances both stability and predictive precision, especially in problems involving oscillatory dynamics.

Figure 10 compares the performance of the cPIKAN method with 1, 2, 4, and 8 temporal subdomains, as well as the PINN method with 8 subdomains. In terms of accuracy, the maximum absolute error for cPIKAN with 1 and 2 subdomains is very large, 16.01 and 20.07, respectively, indicating that the network fails to provide a reliable solution in these cases. However, the error drops significantly when the number of subdomains increases. For 4 subdomains, the maximum error decreases to 0.102, and with 8 subdomains, it reaches as low as 0.016. This represents an 84% improvement in accuracy from 4 to 8 subdomains within the cPIKAN framework. Compared to PINN with 8 subdomains, which yields a maximum error of 0.779, the 8-subdomain cPIKAN solution is approximately 98% more accurate, highlighting the benefit of the domain decomposition strategy.

Rather than illustrating the progression of the diverse eigenvalues of the NTK matrices, we introduce an alternative metric dubbed as the spectral entropy of the NTK matrix, calculated as,

$$\text{Spectral Entropy} = -\sum_i \frac{|\lambda_i|}{\sum_j |\lambda_j|} \log\left(\frac{|\lambda_i|}{\sum_j |\lambda_j|}\right), \tag{55}$$

where λ_i are the eigenvalues of the NTK matrix. We adopt the concept of spectral entropy from Shannon's information theory [98], introduced in the 1940s to quantify uncertainty and information content in communication systems. Shannon's information theory defines entropy as a measure of unpredictability or information content in a signal, which later became a cornerstone in data compression [99–101] and machine learning [102, 103] and leveraged as cross-entropy in classification models using deep learning [96, 104]. In Eq. (55), by applying Shannon's information theory to the normalized eigenvalues, we compute the spectral entropy that captures the transient and convergence behavior of the NTK spectrum. This helps evaluate how well the network approximates the target function across localized regions of the domain. Based on Eq.(55), a low spectral entropy indicates that the network's learning dynamics are focused along a few dominant directions, while a high entropy implies more distributed and less structured learning behavior.

Figure 11 presents a comprehensive analysis of the training behavior and kernel complexity of the cPIKAN

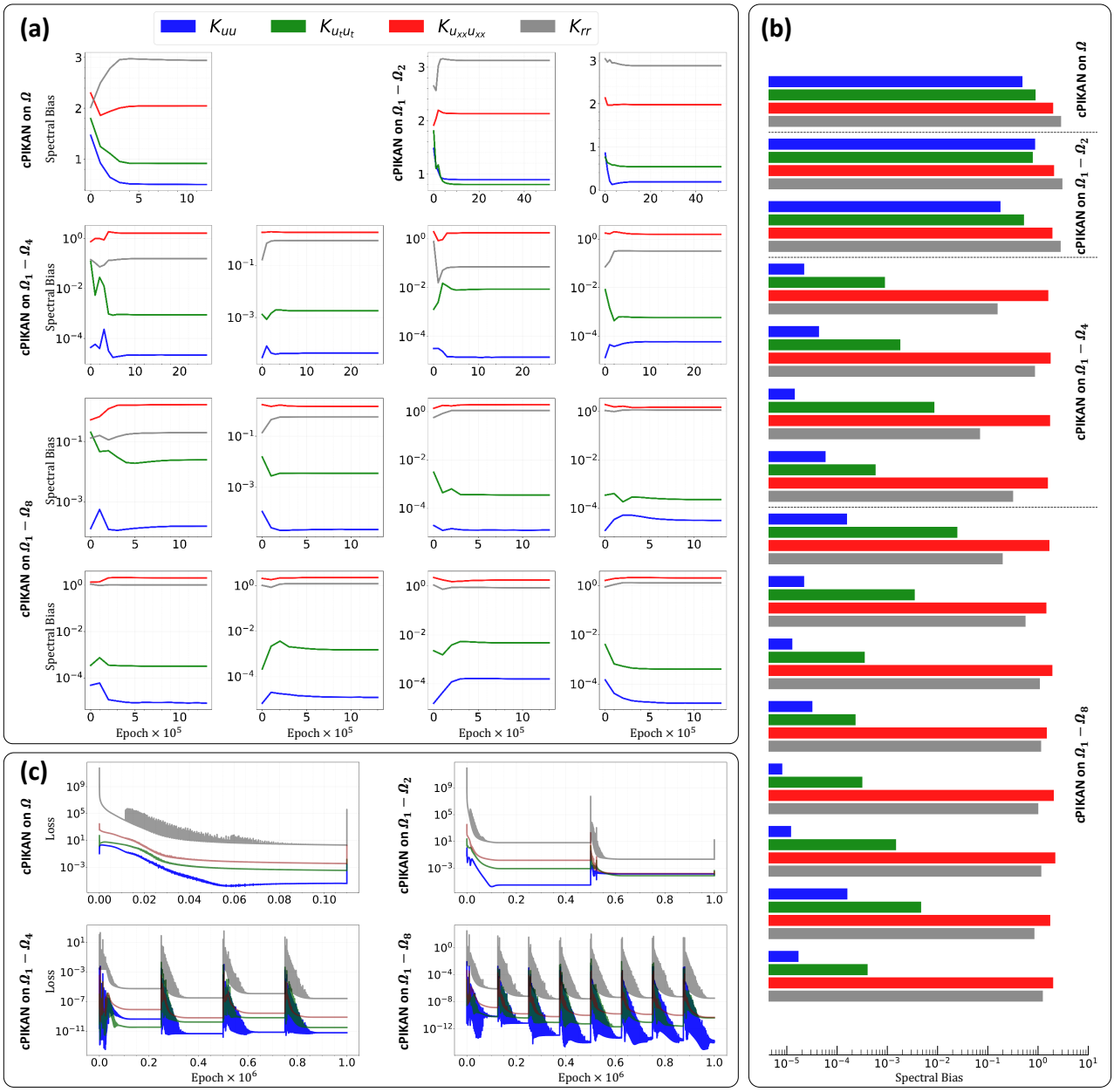

Figure 11: Comparison of the training dynamics and spectral entropy behavior of the cPIKAN method for the forced vibration equation (Experiment 4.4) under varying temporal subdomain configurations. Panels (a) and (b) shows that increasing the number of subdomains consistently reduces the spectral entropy of key NTK components, particularly K_{uu} and $K_{u_t u_t}$, indicating lower kernel complexity and more structured learning dynamics. This reduction becomes more pronounced as the number of subdomains increases from 1 to 8. Additionally, the training loss curves shown in Panel (c) reveal faster and more stable convergence for configurations with more subdomains. These findings demonstrate that temporal decomposition not only improves training efficiency but also enhances the generalization capacity of cPIKAN by promoting simpler and more focused representations in the neural tangent kernel.

method when applied to the forced vibration problem under different subdomain configurations. Panel (a) illustrates the evolution of spectral entropy for various NTK components throughout training, showing how these components converge over time. The results indicate that dividing the domain into subdomains generally leads to a noticeable reduction in spectral entropy for most NTK components, suggesting a simplification of the learning dynamics and more efficient training in structured domains. Panel (b) shows the final spectral entropy values after training, clearly revealing that entropy decreases as the number of subdomains increases, especially for the K_{uu} and $K_{u_t u_t}$ components. This supports the idea that subdomain decomposition helps to reduce kernel complexity and may enhance training stability. Finally, panel (c) presents the training loss curves, which confirm consistent convergence across all configurations. Notably, as the number of subdomains increases, the training process becomes slightly faster and more stable, with the 8-subdomain configuration showing the most efficient convergence behavior.

Figure 12 compares the training performance and kernel behavior of the cPIKAN and PINN methods for the forced vibration problem, both using the same division into 8 temporal subdomains. The loss plots on the left show that cPIKAN converges more quickly and consistently than PINN, with sharper drops in loss values and more regular training dynamics. On the right, the mean spectral entropy plots indicate that

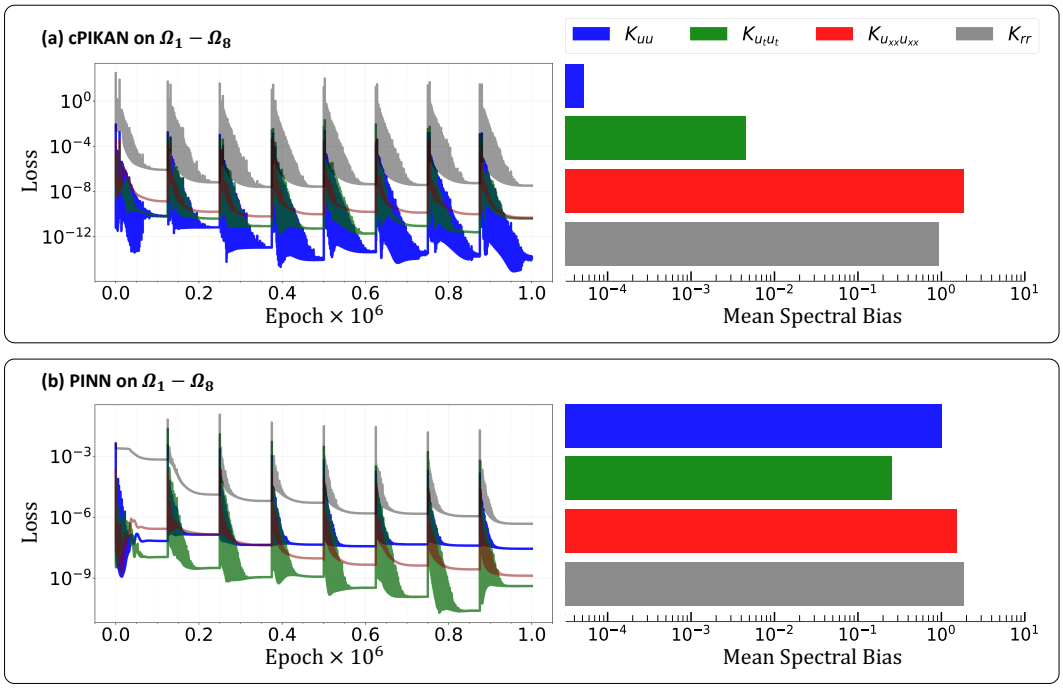

Figure 12: Comparison of training dynamics and spectral bias between (a) cPIKAN and (b) PINN for the forced vibration problem (Experiment 4.4) with 8 temporal subdomains. The comparisons show that cPIKAN achieves faster and more stable convergence, as evidenced by sharper and smoother loss reduction. Additionally, cPIKAN exhibits consistently lower mean spectral entropy across key NTK components, especially K_{uu} and $K_{u_t u_t}$, indicating reduced kernel complexity and enhanced representation of solution features. These findings highlight the advantage of cPIKAN in capturing the underlying physics more efficiently and accurately than PINN, particularly in problems with oscillatory or high-frequency behavior.

cPIKAN has lower mean spectral entropy for the key solution-related NTK components, especially K_{uu} and $K_{u_t u_t}$, suggesting that it better captures high-frequency features of the solution. These observations highlight cPIKAN's improved ability to learn the underlying physics of the problem more efficiently and accurately compared to PINN.

5. Future Outlook

In the future, we envision three key directions to expand this work. First, a deeper theoretical understanding is needed to determine how the NTK of cPIKANs behaves with respect to the model's complexity, specifically, the polynomial degree and the number of layers. As shown in [58], for MLPs initialized in a regime that leads to gradient explosion across layers, the NTK does not follow classical theory. Instead of remaining stable, the kernel appears random at initialization and evolves significantly during training. This challenges the traditional assumption that the NTK remains nearly constant throughout training in sufficiently wide networks. Investigating whether similar phenomena occur in cKANs and how the Chebyshev structure influences such dynamics remains an open question. Second, in this work, we examined the eigenvalue distribution and spectral entropy of the NTK matrix to gain insight into the learning dynamics of cPIKAN. While these metrics offer a useful first look into the structure and evolution of the kernel, they do not fully explain how the NTK influences model accuracy or generalization. A promising direction for future research is to develop additional theoretical measures that more directly link the NTK spectrum to the model's prediction error. For example, analyzing how the alignment between the NTK and the target function changes during training, or investigating how the energy concentration in the leading eigenmodes relates to final performance, could provide a deeper understanding of the learning process. Such studies may lead to the development of new error estimators or performance predictors derived from NTK properties, offering a stronger theoretical basis for architecture design and training decisions. Third, we observed that as the number of boundary and initial conditions increases, such as in the forced vibration problem, the NTK matrix grows in size, leading to significant computational overhead. This issue becomes more pronounced in problems involving coupled PDE systems or high-dimensional physics-based models. To make the NTK-based analysis tractable in such cases, future work should explore matrix decomposition techniques or develop alternative learning metrics that capture the essence of learning dynamics at a lower computational cost.

6. Conclusions

This study investigated the training behavior of Chebyshev-based physics-informed Kolmogorov–Arnold Networks (cPIKANs) through the framework of Neural Tangent Kernel (NTK) theory. We first derived and analyzed the NTK for cKANs in the supervised learning setting, providing a theoretical foundation for

understanding how the kernel structure governs convergence dynamics. Building on this, we extended the analysis to cPIKANs using numerical experiments across a variety of PDEs, including the diffusion equation, Helmholtz equation, Allen–Cahn equation, and a forced vibration problem. Through these case studies, we observed that the NTK matrices associated with cPIKANs exhibit structured and tractable behavior during training. The spectral analysis revealed consistent patterns, such as eigenvalue concentration and reduced entropy, that directly correlate with faster convergence and improved generalization. These findings demonstrate that the NTK framework not only explains the empirical advantages of cPIKANs, but also provides predictive insight into their learning dynamics across different PDE settings.

In the one-dimensional diffusion equation, we compared cPIKAN with two PINN baselines and found that cPIKAN achieved up to 7 times higher accuracy and significantly faster convergence. This performance gain was associated with a more informative and well-conditioned NTK eigenvalue spectrum, which exhibited slower decay and greater stability throughout training. These spectral characteristics enhanced the tractability of the learning dynamics and allowed the model to retain and propagate useful gradient information more effectively. In the two-dimensional Helmholtz equation, we compared cPIKAN, PINN, and bPIKAN, and found that cPIKAN achieved an error approximately three times smaller than PINN and five times smaller than bPIKAN. NTK analysis showed that cPIKAN maintained a more stable and better-conditioned eigenvalue spectrum, resulting in improved convergence and accuracy. Additionally, using LBFGS or a hybrid ADAM+LBFGS optimizer, the error was reduced by nearly an order of magnitude compared to ADAM alone, highlighting the important influence of optimization strategy on NTK dynamics and model performance. The Allen-Cahn experiment shows that splitting the time domain into 2 or 4 subdomains improves the NTK conditioning and convergence, reducing the relative error by over 98% and the maximum absolute error by over 95%, compared to training on the full domain. In the final experiment, we considered a forced vibration problem governed by a high-order PDE, which led to a larger and more structured NTK matrix due to the presence of higher order derivative terms. Results show that cPIKAN with 8 temporal subdomains achieves over 98% lower maximum error compared to PINN, while also reducing spectral entropy of key NTK blocks by a factor of three, indicating more focused and efficient learning dynamics. This highlights the dual benefit of temporal decomposition in both reducing NTK complexity and improving model accuracy and convergence stability.

7. Acknowledgements

S.A.F. acknowledges the support by the U.S. Department of Energy's Office of Environmental Management (award no.: DE-EM0005314).

8. Conflict of Interest

The authors declare no conflict of interest.

References

[1] D. R. Gaston, C. J. Permann, J. W. Peterson, A. E. Slaughter, D. Andrš, Y. Wang, M. P. Short, D. M. Perez, M. R. Tonks, J. Ortensi, et al., Physics-based multiscale coupling for full core nuclear reactor simulation, Annals of Nuclear Energy 84 (2015) 45–54.

[2] D. A. Cullen, K. C. Neyerlin, R. K. Ahluwalia, R. Mukundan, K. L. More, R. L. Borup, A. Z. Weber, D. J. Myers, A. Kusoglu, New roads and challenges for fuel cells in heavy-duty transportation, Nature energy 6 (5) (2021) 462–474.

[3] T. Zhou, R. Gani, K. Sundmacher, Hybrid data-driven and mechanistic modeling approaches for multiscale material and process design, Engineering 7 (9) (2021) 1231–1238.

[4] C. Xi, L. Rongchao, T. Ye, T. Yongqi, L. Ao, M. Daofeng, Z. Haibo, Engineering design and numerical design for chemical looping combustion reactor: A review, Energy Reviews (2024) 100100.

[5] J. Lee, D. Jun, B. Chun, S. M. Mousavi, B. J. Lee, S. A. Faroughi, Large eddy simulation of the effects of radiative heat loss on combustion instability prediction, Acta Astronautica 217 (2024) 312–322.

[6] D. I. Fotiadis, A. I. Sakellarios, V. T. Potsika, Multiscale Modelling in Biomedical Engineering, John Wiley & Sons, 2023.

[7] H. Yu, L. Zhang, W. Wang, K. Yang, Z. Zhang, X. Liang, S. Chen, S. Yang, J. Li, X. Liu, Lithium-ion battery multi-scale modeling coupled with simplified electrochemical model and kinetic monte carlo model, Iscience 26 (9) (2023).

[8] X. Gao, B. Knueven, J. D. Siirola, D. C. Miller, A. W. Dowling, Multiscale simulation of integrated energy system and electricity market interactions, Applied Energy 316 (2022) 119017.

[9] N. M. Pawar, R. Soltanmohammadi, S. Faroughi, S. A. Faroughi, Geo-guided deep learning for spatial downscaling of solute transport in heterogeneous porous media, Computers & Geosciences 188 (2024) 105599.

[10] S. K. Mahjour, G. Liguori, S. A. Faroughi, Selection of representative general circulation models under climatic uncertainty for western north america, Journal of Water and Climate Change 15 (2) (2024) 686–702.

[11] S. K. Mahjour, J. P. Tiefenbacher, S. A. Faroughi, Select representative general circulation model-runs using enveloped-based technique, Journal of Climate (2025).

[12] A. Heinlein, A. A. Howard, D. Beecroft, Multifidelity domain decomposition-based physics-informed neural networks and operators for time-dependent problems, Mathematical Optimization for Machine Learning: Proceedings of the MATH+ Thematic Einstein Semester 2023 (2025) 79.

[13] X. Zhou, Y. Liu, M. Ali, M. He, A multilevel-multiphysics modeling and simulation approach for multichip electronics, Applied Thermal Engineering (2025) 125738.

[14] B. G. Van Willigen, M. B. van der Hout-van der Jagt, W. Huberts, F. N. van de Vosse, A multiscale mathematical model for fetal gas transport and regulatory systems during second half of pregnancy, International Journal for Numerical Methods in Biomedical Engineering 41 (1) (2025) e3881.

[15] S. A. Faroughi, N. M. Pawar, C. Fernandes, M. Raissi, S. Das, N. K. Kalantari, S. Kourosh Mahjour, Physics-guided, physics-informed, and physics-encoded neural networks and operators in scientific computing: Fluid and solid mechanics, Journal of Computing and Information Science in Engineering 24 (4) (2024) 040802.
[16] D. Kim, J. Lee, A review of physics informed neural networks for multiscale analysis and inverse problems, Multiscale Science and Engineering 6 (1) (2024) 1–11.
[17] O.-H. E. Oladayo, O. Joshua, Stability analysis of explicit finite difference methods for neutral stochastic differential equations with multiplicative noise, Asian Research Journal of Current Science 7 (1) (2025) 12–21.
[18] Ö. Oruç, A. Esen, F. Bulut, Numerical solution of the rosenau-kdv-rlw equation via combination of a polynomial scaling function collocation and finite difference method, Mathematical Methods in the Applied Sciences (2025).
[19] J. Weiss, M. Knezevic, Effects of element type on accuracy of microstructural mesh crystal plasticity finite element simulations and comparisons with elasto-viscoplastic fast fourier transform predictions, Computational Materials Science 240 (2024) 113002.
[20] M. Raissi, P. Perdikaris, G. 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 (2019) 686–707.
[21] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125–141.
[22] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440.
[23] F. Mostajeran, R. Mokhtari, Deepbhcp: Deep neural network algorithm for solving backward heat conduction problems, Computer Physics Communications 272 (2022) 108236.
[24] S. Hanrahan, M. Kozul, R. D. Sandberg, Studying turbulent flows with physics-informed neural networks and sparse data, International Journal of Heat and Fluid Flow 104 (2023) 109232.
[25] S. Jang, M. Jadidi, S. Rezaeiravesh, A. Revell, Y. Mahmoudi, Physics-informed neural network for turbulent flow reconstruction in composite porous-fluid systems, Machine Learning: Science and Technology 5 (3) (2024) 035030.
[26] S. Yazdani, M. Tahani, Data-driven discovery of turbulent flow equations using physics-informed neural networks, Physics of Fluids 36 (3) (2024).
[27] A. Gafoor CTP, S. Kumar Boya, R. Jinka, A. Gupta, A. Tyagi, S. Sarkar, D. N. Subramani, A physics-informed neural network for turbulent wake simulations behind wind turbines, Physics of Fluids 37 (1) (2025).
[28] M. Mahmoudabadbozchelou, G. E. Karniadakis, S. Jamali, nn-pinns: Non-newtonian physics-informed neural networks for complex fluid modeling, Soft Matter 18 (1) (2022) 172–185.
[29] S. Thakur, M. Raissi, A. M. Ardekani, Viscoelasticnet: A physics informed neural network framework for stress discovery and model selection, Journal of Non-Newtonian Fluid Mechanics 330 (2024) 105265.
[30] E. Kharazmi, Z. Zhang, G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, Computer Methods in Applied Mechanics and Engineering 374 (2021) 113547.
[31] A. Arzani, S. T. Dawson, Data-driven cardiovascular flow modelling: examples and opportunities, Journal of the Royal Society Interface 18 (175) (2021) 20200802.
[32] X. Zhang, B. Mao, Y. Che, J. Kang, M. Luo, A. Qiao, Y. Liu, H. Anzai, M. Ohta, Y. Guo, et al., Physics-informed neural networks (pinns) for 4d hemodynamics prediction: An investigation of optimal framework based on vascular morphology, Computers in Biology and Medicine 164 (2023) 107287.
[33] M. Rasht-Behesht, C. Huber, K. Shukla, G. E. Karniadakis, Physics-informed neural networks (pinns) for wave propagation and full waveform inversions, Journal of Geophysical Research: Solid Earth 127 (5) (2022) e2021JB023120.
[34] J. Zou, C. Liu, Y. Wang, C. Song, U. b. Waheed, P. Zhao, Accelerating the convergence of physics-informed neural networks for seismic wave simulation, Geophysics 90 (2) (2025) T23–T32.
[35] J. Zou, C. Liu, P. Zhao, C. Song, Seismic wavefields modeling with variable horizontally-layered velocity models via velocity-encoded pinn, IEEE Transactions on Geoscience and Remote Sensing (2024).
[36] A. D. Jagtap, Z. Mao, N. Adams, G. E. Karniadakis, Physics-informed neural networks for inverse problems in supersonic flows, Journal of Computational Physics 466 (2022) 111402.
[37] W. Wu, M. Daneker, M. A. Jolley, K. T. Turner, L. Lu, Effective data sampling strategies and boundary condition constraints of physics-informed neural networks for identifying material properties in solid mechanics, Applied mathematics and mechanics 44 (7) (2023) 1039–1068.
[38] H. Hu, L. Qi, X. Chao, Physics-informed neural networks (pinn) for computational solid mechanics: Numerical frameworks and applications, Thin-Walled Structures (2024) 112495.
[39] R. d. O. Teloli, R. Tittarelli, M. Bigot, L. Coelho, E. Ramasso, P. Le Moal, M. Ouisse, A physics-informed neural networks framework for model parameter identification of beam-like structures, Mechanical Systems and Signal Processing 224 (2025) 112189.
[40] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating pdes, IMA Journal of Numerical Analysis 43 (1) (2023) 1–43.
[41] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in neural information processing systems 34 (2021) 26548–26560.
[42] S. Wang, H. Wang, P. 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 (2021) 113938.
[43] J. Yu, L. Lu, X. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse pde problems, Computer Methods in Applied Mechanics and Engineering 393 (2022) 114823.
[44] X. Li, Y. Liu, Z. Liu, Physics-informed neural network based on a new adaptive gradient descent algorithm for solving partial differential equations of flow problems, Physics of Fluids 35 (6) (2023).
[45] S. Basir, I. Senocak, Critical investigation of failure modes in physics-informed neural networks, in: AiAA SCITECH 2022 Forum, 2022, p. 2353.
[46] Y. Shi, M. Beer, Physics-informed neural network classification framework for reliability analysis, Expert Systems with Applications 258 (2024) 125207.
[47] A. N. Kolmogorov, On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition, in: Dokl. Akad. Nauk USSR, Vol. 114, 1957, pp. 953–956.
[48] V. Arnold, On the representation of functions of several variables by superpositions of functions of fewer variables, Mat. Prosvesh. 3 (1958) 41–61.
[49] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, M. Tegmark, Kan: Kolmogorov-arnold networks, arXiv preprint arXiv:2404.19756 (2024).
[50] A. N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, Doklady Akademii Nauk SSSR 114 (1957) 953–956.
[51] V. I. Arnold, On functions of three variables, Doklady Akademii Nauk SSSR 114 (1957) 679–681.
[52] K. Shukla, J. D. Toscano, Z. Wang, Z. Zou, G. E. Karniadakis, A comprehensive and fair comparison between mlp and kan

representations for differential equations and operator networks, Computer Methods in Applied Mechanics and Engineering 431 (2024) 117290.
[53] F. Mostajeran, S. A. Faroughi, Epi-ckans: Elasto-plasticity informed kolmogorov-arnold networks using chebyshev polynomials, arXiv preprint arXiv:2410.10897 (2024).
[54] F. Mostajeran, S. A. Faroughi, Scaled-cpikans: Spatial variable and residual scaling in chebyshev-based physics-informed kolmogorov-arnold networks, Journal of Computational Physics 537 (2025) 114116.
[55] S. SS, K. AR, A. KP, et al., Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation, arXiv preprint arXiv:2405.07200 (2024).
[56] C. Guo, L. Sun, S. Li, Z. Yuan, C. Wang, Physics-informed kolmogorov-arnold network with chebyshev polynomials for fluid mechanics, arXiv preprint arXiv:2411.04516 (2024).
[57] A. Jacot, F. Gabriel, C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, Advances in neural information processing systems 31 (2018).
[58] M. Seleznova, G. Kutyniok, Analyzing finite neural networks: Can we trust neural tangent kernel theory?, in: Mathematical and Scientific Machine Learning, PMLR, 2022, pp. 868–895.
[59] M. Seleznova, G. Kutyniok, Neural tangent kernel beyond the infinite-width limit: Effects of depth and initialization, in: International Conference on Machine Learning, PMLR, 2022, pp. 19522–19560.
[60] S. Wang, X. Yu, P. Perdikaris, When and why pinns fail to train: A neural tangent kernel perspective, Journal of Computational Physics 449 (2022) 110768.
[61] M. H. Saadat, B. Gjorgiev, L. Das, G. Sansavini, Neural tangent kernel analysis of pinn for advection-diffusion equation, arXiv preprint arXiv:2211.11716 (2022).
[62] V. Arnold, On the representation of continuous functions of three variables by superpositions of continuous functions of two variables, Math. Sb.(NS) 48 (90) (1959) 3–74.
[63] V. I. Arnold, On functions of three variables, Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965 (2009) 5–8.
[64] G. Lorentz, Approximation of functions.-holt, rinehart and wilson, Inc., New York (1966).
[65] G. G. Lorentz, M. von Golitschek, Y. Makovoz, Constructive approximation: advanced problems, Vol. 304, Citeseer, 1996.
[66] D. A. Sprecher, On the structure of continuous functions of several variables, Transactions of the American Mathematical Society 115 (1965) 340–355.
[67] D. A. Sprecher, An improvement in the superposition theorem of kolmogorov, Journal of Mathematical Analysis and Applications 38 (1) (1972) 208–213.
[68] B. L. Fridman, An improvement in the smoothness of the functions in an kolmogorov's theorem on superpositions, in: Doklady Akademii Nauk, Vol. 177, Russian Academy of Sciences, 1967, pp. 1019–1022.
[69] R. Hecht-Nielsen, Kolmogorov's mapping neural network existence theorem, in: Proceedings of the international conference on Neural Networks, Vol. 3, IEEE press New York, NY, USA, 1987, pp. 11–14.
[70] V. Kůrková, Kolmogorov's theorem is relevant, Neural computation 3 (4) (1991) 617–622.
[71] V. Kůrková, Kolmogorov's theorem and multilayer neural networks, Neural networks 5 (3) (1992) 501–506.
[72] D. A. Sprecher, A numerical implementation of kolmogorov's superpositions, Neural networks 9 (5) (1996) 765–772.
[73] M. Köppen, On the training of a kolmogorov network, in: Artificial Neural Networks—ICANN 2002: International Conference Madrid, Spain, August 28–30, 2002 Proceedings 12, Springer, 2002, pp. 474–479.
[74] J. Braun, M. Griebel, On a constructive proof of kolmogorov's superposition theorem, Constructive approximation 30 (2009) 653–675.
[75] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (4) (1989) 303–314.
[76] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (2) (1991) 251–257.
[77] A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Transactions on Information theory 39 (3) (1993) 930–945.
[78] M. Telgarsky, Benefits of depth in neural networks, in: Conference on learning theory, PMLR, 2016, pp. 1517–1539.
[79] Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, The expressive power of neural networks: A view from the width, Advances in neural information processing systems 30 (2017).
[80] J. Schmidt-Hieber, Nonparametric regression using deep neural networks with relu activation function (2020).
[81] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review, International Journal of Automation and Computing 14 (5) (2017) 503–519.
[82] D. A. Sprecher, S. Draghici, Space-filling curves and kolmogorov superposition-based neural networks, Neural Networks 15 (1) (2002) 57–67.
[83] D. Fakhoury, E. Fakhoury, H. Speleers, Exsplinet: An interpretable and expressive spline-based neural network, Neural Networks 152 (2022) 332–346.
[84] J. He, On the optimal expressive power of relu dnns and its application in approximation with kolmogorov superposition theorem, arXiv preprint arXiv:2308.05509 (2023).
[85] Y. Sun, L. Xiao, H. Zhao, On the performance and generalization of kolmogorov-arnold networks for high-dimensional function approximation, Neural Networks 138 (2021) 23–38.
[86] B. Chang, G. Li, X. Bai, Bridging Kolmogorov-Arnold representation and deep architectures: A survey and analysis, IEEE Transactions on Neural Networks and Learning Systems 33 (10) (2022) 5170–5183.
[87] Z. Bozorgasl, H. Chen, Wav-kan: Wavelet kolmogorov-arnold networks, 2024, arXiv preprint arXiv:2405.12832.
[88] Z. Li, Kolmogorov-arnold networks are radial basis function networks, arXiv preprint arXiv:2405.06721 (2024).
[89] T. J. Rivlin, Chebyshev polynomials, Courier Dover Publications, 2020.
[90] J. Schmidt-Hieber, The kolmogorov–arnold representation theorem revisited, Neural networks 137 (2021) 119–126.
[91] Z. Hu, K. Shukla, G. E. Karniadakis, K. Kawaguchi, Tackling the curse of dimensionality with physics-informed neural networks, Neural Networks 176 (2024) 106369.
[92] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, R. Wang, On exact computation with an infinitely wide neural net, Advances in neural information processing systems 32 (2019).
[93] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, J. Pennington, Wide neural networks of any depth evolve as linear models under gradient descent, Advances in neural information processing systems 32 (2019).
[94] A. M. Mood, Introduction to the theory of statistics. (1950).
[95] S. Ross, Probability and statistics for engineers and scientists, Elsevier, New Delhi 16 (2009) 32–33.
[96] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, Vol. 4, Springer, 2006.
[97] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM review 60 (2) (2018) 223–311.
[98] C. E. Shannon, A mathematical theory of communication, The Bell system technical journal 27 (3) (1948) 379–423.
[99] T. M. Cover, Elements of information theory, John Wiley & Sons, 1999.
[100] D. S. Ornstein, B. Weiss, Entropy and data compression schemes, IEEE Transactions on information theory 39 (1) (1993)

78–83.
[101] K. J. Balakrishnan, N. A. Touba, Relationship between entropy and test data compression, IEEE Transactions on computer-aided design of integrated circuits and systems 26 (2) (2007) 386–395.
[102] K. P. Murphy, Machine learning: a probabilistic perspective, MIT press, 2012.
[103] S. A. Sepúlveda-Fontaine, J. M. Amigó, Applications of entropy in data analysis and machine learning: A review, Entropy 26 (12) (2024) 1126.
[104] I. Goodfellow, Deep learning (2016).

Appendix A. Proof of Theorem 1

The Neural Tangent Kernel between two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ is defined as the inner product of the gradients of the network output with respect to the parameters,

$$\begin{aligned}\boldsymbol{K}_{\text{ntk}}(\boldsymbol{x},\boldsymbol{x}') &= \left\langle \frac{\partial f(\boldsymbol{x};\boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{x}';\boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}} \right\rangle \\ &= \sum_{i,j,n} \frac{\partial f(\boldsymbol{x};\boldsymbol{\theta}(0))}{\partial w^{(1)}_{i,j,n}} \cdot \frac{\partial f(\boldsymbol{x}';\boldsymbol{\theta}(0))}{\partial w^{(1)}_{i,j,n}} + \sum_{j,n} \frac{\partial f(\boldsymbol{x};\boldsymbol{\theta}(0))}{\partial w^{(2)}_{j,1,n}} \cdot \frac{\partial f(\boldsymbol{x}';\boldsymbol{\theta}(0))}{\partial w^{(2)}_{j,1,n}}.\end{aligned} \tag{A.1}$$

Now, we simplify the NTK expression statistically, using the fact that the coefficients are i.i.d. standard normal,

$$w^{(1)}_{i,j,n},\ w^{(2)}_{j,1,n} \sim \mathcal{N}(0,1). \tag{A.2}$$

We simplify the expected NTK,

$$\mathbb{E}[\boldsymbol{K}_{\text{ntk}}(\boldsymbol{x},\boldsymbol{x}')] = \mathbb{E}\left[K^{(1)}(\boldsymbol{x},\boldsymbol{x}')\right] + \mathbb{E}\left[K^{(2)}(\boldsymbol{x},\boldsymbol{x}')\right]. \tag{A.3}$$

where,

$$K^{(1)}(\boldsymbol{x},\boldsymbol{x}') = \sum_{i,j,n} \frac{\partial f(\boldsymbol{x};\boldsymbol{\theta}(0))}{\partial w^{(1)}_{i,j,n}} \cdot \frac{\partial f(\boldsymbol{x}';\boldsymbol{\theta}(0))}{\partial w^{(1)}_{i,j,n}}, \quad K^{(2)}(\boldsymbol{x},\boldsymbol{x}') = \sum_{j,n} \frac{\partial f(\boldsymbol{x};\boldsymbol{\theta}(0))}{\partial w^{(2)}_{j,1,n}} \cdot \frac{\partial f(\boldsymbol{x}';\boldsymbol{\theta}(0))}{\partial w^{(2)}_{j,1,n}}. \tag{A.4}$$

This helps us understand the average behavior of the kernel at initialization.
Step I: In this step, we consider the expectation of the second term,

$$\mathbb{E}[K^{(2)}(\boldsymbol{x},\boldsymbol{x}')] = N \cdot \sum_{n=0}^{k} \mathbb{E}\left[T_n(z_j(\boldsymbol{x})) \cdot T_n(z_j(\boldsymbol{x}'))\right], \tag{A.5}$$

where $z_j(\boldsymbol{x}) = \tanh(h_j(\boldsymbol{x}))$ and $z_j(\boldsymbol{x}') = \tanh(h_j(\boldsymbol{x}'))$ are jointly distributed through the bivariate Gaussian pair $(h_j(\boldsymbol{x}), h_j(\boldsymbol{x}'))$ with zero mean and variances $\sigma_j^2(\boldsymbol{x})$ and $\sigma_j^2(\boldsymbol{x}')$, as defined previously in Eq. (25). The covariance between $h_j(\boldsymbol{x})$ and $h_j(\boldsymbol{x}')$ is given by,

$$\text{Cov}[h_j(\boldsymbol{x}), h_j(\boldsymbol{x}')] = \sum_{i=1}^{d} \sum_{n=0}^{k} T_n(\tanh(x_i)) \cdot T_n(\tanh(x_i')) =: \rho_j(\boldsymbol{x},\boldsymbol{x}'). \tag{A.6}$$

Defining,

$$C_n(\boldsymbol{x},\boldsymbol{x}') := \mathbb{E}\left[T_n(\tanh(h)) \cdot T_n(\tanh(h'))\right], \tag{A.7}$$

for $(h,h') \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma^2(\boldsymbol{x}) & \rho(\boldsymbol{x},\boldsymbol{x}') \\ \rho(\boldsymbol{x},\boldsymbol{x}') & \sigma^2(\boldsymbol{x}') \end{bmatrix}\right)$, we obtain the compact expression,

$$\mathbb{E}[K^{(2)}(\boldsymbol{x},\boldsymbol{x}')] = N \cdot \sum_{n=0}^{k} C_n(\boldsymbol{x},\boldsymbol{x}'). \tag{A.8}$$

Since Chebyshev polynomials are bounded on the interval $(-1,1)$, and the tanh function maps Gaussian variables into this range, each term $C_n(\boldsymbol{x},\boldsymbol{x}')$ remains finite.
Step II: In the second step, we compute the expectation of the first-term kernel $\mathbb{E}[K^{(1)}(\boldsymbol{x},\boldsymbol{x}')]$. Using Eq. (21), the first-term kernel can be written as,

$$K^{(1)}(\boldsymbol{x},\boldsymbol{x}') = \sum_{i,j,n} A_{i,j,n}(\boldsymbol{x}) \cdot A_{i,j,n}(\boldsymbol{x}'), \tag{A.9}$$

where,

$$A_{i,j,n}(\boldsymbol{x}) = T_n(\tilde{x}_i) \cdot \psi(\boldsymbol{x}) \cdot \sum_{m=0}^{k} w^{(2)}_{j,1,m} \cdot T_m'(\tanh(h_j(\boldsymbol{x}))), \tag{A.10}$$

and $\psi(\boldsymbol{x}) = (1 - \tanh^2(h_j(\boldsymbol{x})))$. Taking expectation with respect to the second-layer coefficients $w^{(2)}_{j,1,m} \sim \mathcal{N}(0,1)$, and noting that for fixed $h_j(\boldsymbol{x})$, the sum becomes a linear combination of independent Gaussian variables, we obtain a zero mean and the following variance,

$$\sum_{m=0}^{k} \left(T_m'(\tanh(h_j(\boldsymbol{x})))\right)^2. \tag{A.11}$$

This leads to the following expectation,

$$\mathbb{E}_{w^{(2)}}[A_{i,j,n}(\boldsymbol{x}) \cdot A_{i,j,n}(\boldsymbol{x}')] = \\ T_n(\tilde{x}_i) \cdot T_n(\tilde{x}_i') \cdot \mathbb{E}_{h_j(\boldsymbol{x}),h_j(\boldsymbol{x}')}\left[\psi(\boldsymbol{x}) \cdot \psi(\boldsymbol{x}') \cdot \textstyle\sum_{m=0}^{k} T_m'(\tanh(h_j(\boldsymbol{x}))) \cdot T_m'(\tanh(h_j(\boldsymbol{x}')))\right]. \tag{A.12}$$

We denote the inner expectation as $D(\boldsymbol{x}, \boldsymbol{x}')$, defined by,

$$D(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}_{h,h'}\left[(1 - \tanh^2(h)) \cdot (1 - \tanh^2(h')) \cdot \psi(\boldsymbol{x}') \cdot \sum_{m=0}^{k} T_m'(\tanh(h)) \cdot T_m'(\tanh(h'))\right], \tag{A.13}$$

where $(h, h') \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma^2(\boldsymbol{x}) & \rho(\boldsymbol{x}, \boldsymbol{x}') \\ \rho(\boldsymbol{x}, \boldsymbol{x}') & \sigma^2(\boldsymbol{x}') \end{bmatrix}\right)$. The variance and covariance terms are defined as (see Equations (25) and (A.6)),

$$\sigma^2(\boldsymbol{x}) := \mathrm{Var}[h_j(\boldsymbol{x})], \quad \rho(\boldsymbol{x}, \boldsymbol{x}') := \mathrm{Cov}[h_j(\boldsymbol{x}), h_j(\boldsymbol{x}')]. \tag{A.14}$$

The derivatives $T_m'(\cdot)$ are evaluated elementwise inside the expectation, as they depend on the random variable $\tanh(h)$, a nonlinear transformation of the Gaussian variable h. Since $h_j(\boldsymbol{x})$ are i.i.d. across neurons, the expectation becomes independent of the index j, yielding the simplified form,

$$\mathbb{E}[K^{(1)}(\boldsymbol{x}, \boldsymbol{x}')] = N \cdot \sum_{i=1}^{d} \sum_{n=0}^{k} T_n(\tilde{x}_i) \cdot T_n(\tilde{x}_i') \cdot D(\boldsymbol{x}, \boldsymbol{x}'). \tag{A.15}$$

Using Equations (A.8) and (A.15), we have,

$$\mathbb{E}[\boldsymbol{K}_{\mathrm{ntk}}(\boldsymbol{x}, \boldsymbol{x}')] = N \cdot \left[\sum_{n=0}^{k} C_n(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i=1}^{d} \sum_{n=0}^{k} T_n(\tilde{x}_i) \cdot T_n(\tilde{x}_i') \cdot D(\boldsymbol{x}, \boldsymbol{x}')\right], \tag{A.16}$$

where $C_n(\boldsymbol{x}, \boldsymbol{x}')$ is correlation of Chebyshev polynomials evaluated at $\tanh(h_j(\boldsymbol{x}))$ and $\tanh(h_j(\boldsymbol{x}'))$, and $D(\boldsymbol{x}, \boldsymbol{x}')$ is expectation involving ψ terms and Chebyshev polynomial derivatives.

Appendix B. Proof of Theorem 2

Let $f(\boldsymbol{x};\boldsymbol{\theta})$ be the output of a cKAN with parameters $\boldsymbol{\theta}$, and let $\boldsymbol{K}_{\text{ntk}}^{(\tau)}(\boldsymbol{x},\boldsymbol{x}')$ denote the NTK between two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ at training time τ. By definition,

$$\boldsymbol{K}_{\text{ntk}}^{(\tau)}(\boldsymbol{x},\boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}(\tau)), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}';\boldsymbol{\theta}(\tau)) \rangle \,. \tag{B.1}$$

We now differentiate this quantity with respect to the training time τ using the product rule,

$$\frac{d}{d\tau}\boldsymbol{K}_{\text{ntk}}^{(\tau)}(\boldsymbol{x},\boldsymbol{x}') = \left\langle \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}(\tau)) \cdot \dot{\boldsymbol{\theta}}(\tau), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}';\boldsymbol{\theta}(\tau)) \right\rangle + \left\langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}(\tau)), \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{x}';\boldsymbol{\theta}(\tau)) \cdot \dot{\boldsymbol{\theta}}(\tau) \right\rangle . \tag{B.2}$$

Taking the absolute value and applying the Cauchy–Schwarz inequality and the uniform bounds B_1, B_2, we obtain the bound,

$$\left| \frac{d}{d\tau}\boldsymbol{K}_{\text{ntk}}^{(\tau)}(\boldsymbol{x},\boldsymbol{x}') \right| \leq 2B_1B_2 \cdot \left\| \dot{\boldsymbol{\theta}}(\tau) \right\| . \tag{B.3}$$

Integrating from 0 to τ, we get,

$$\left| \boldsymbol{K}_{\text{ntk}}^{(\tau)}(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{K}_{\text{ntk}}^{(0)}(\boldsymbol{x},\boldsymbol{x}') \right| \leq 2B_1B_2 \cdot \int_0^\tau \left\| \dot{\boldsymbol{\theta}}(s) \right\| ds = 2B_1B_2 \cdot \|\boldsymbol{\theta}(\tau) - \boldsymbol{\theta}(0)\| \,. \tag{B.4}$$

Therefore, the matrix norm satisfies,

$$\|\boldsymbol{K}_{\text{ntk}}(\tau) - \boldsymbol{K}_{\text{ntk}}(0)\| \leq C \cdot \|\boldsymbol{\theta}(\tau) - \boldsymbol{\theta}(0)\| \,, \tag{B.5}$$

for some constant $C = 2B_1B_2$. In the infinite-width limit, parameter drift satisfies,

$$\|\boldsymbol{\theta}(\tau) - \boldsymbol{\theta}(0)\| \to 0 \quad \text{as } N \to \infty, \tag{B.6}$$

which implies,

$$\|\boldsymbol{K}_{\text{ntk}}(\tau) - \boldsymbol{K}_{\text{ntk}}(0)\| \to 0 \quad \text{as } N \to \infty. \tag{B.7}$$