Physics-Informed Neural Networks with Fourier Features and Attention-Driven Decoding

Rohan Arni *
High Technology High School
Lincroft, NJ 07738
roarni@ctemc.org

Carlos Blanco
Institute for Gravitation and the Cosmos
The Pennsylvania State University
University Park, PA 16802
&
Department of Physics
Princeton University
Princeton, NJ, 08544
carlosblanco@psu.edu

Abstract

Physics-Informed Neural Networks (PINNs) are a useful framework for approximating partial differential equation solutions using deep learning methods. In this paper, we propose a principled redesign of the PINNsformer, a Transformer-based PINN architecture. We present the Spectral PINNSformer (S-Pformer), a refinement of encoder-decoder PINNSformers that addresses two key issues; 1. the redundancy (i.e. increased parameter count) of the encoder, and 2. the mitigation of spectral bias. We find that the encoder is unnecessary for capturing spatiotemporal correlations when relying solely on self-attention, thereby reducing parameter count. Further, we integrate Fourier feature embeddings to explicitly mitigate spectral bias, enabling adaptive encoding of multiscale behaviors in the frequency domain. Our model outperforms encoder-decoder PINNSformer architectures across all benchmarks, achieving or outperforming MLP performance while reducing parameter count significantly.

1 Introduction

Numerically computing solutions to partial differential equations has been a key area of research in science and engineering. Typical computational methods such as the finite difference method [Nakayama, 2018] or the spectral method suffer [Fornberg, 1996] from computational overhead due to fine spatial-temporal discretization. These grid-based approaches also struggle with adaptability to complex geometries and require careful treatment of boundary conditions, which can increase the computational cost and implementation complexity [Murugesh et al., 2025].

In recent years, Physics-Informed Neural Networks (PINNs) have emerged as a promising alternative, as they utilize deep learning methods to approximate the solution of PDEs by embedding the governing physical laws directly into the loss function [Raissi et al., 2019]. Rather than explicitly discretizing the domain, PINNs use neural networks that accept continuous space-time coordinates as input, enabling mesh-free solution approximations.

The majority of PINNs rely on multilayer perceptrons (MLPs), which suffer from spectral bias: a limitation where the networks have difficulty learning high-frequency components in differential equation solutions [Wang et al., 2021]. In addition, MLP-based PINNs can suffer from limited generalization when applied to more challenging or nonlinear problems due to simplicity bias [Xu et al., 2025].

*Alternate email: rohan.arni@gmail.com

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Science.

To address these limitations, attention-based architectures such as the PINNsformer have been developed [Zhao et al., 2024]. These models use the self-attention mechanism from Transformers, allowing the network to attend to relevant spatial and temporal relationships in the input. The attention mechanism allows for dramatic improvements in performance.

In this work, we propose a Spectral PINNsformer (S-Pformer) architecture that streamlines the architecture by replacing the encoder layer with Fourier feature embeddings and applying self-attention directly within the decoder. The Fourier feature embeddings enabling the S-Pformer to better capture multiscale behaviors by adaptively projecting input coordinates into a spectral representation in order to better understand different frequencies in the PDE solution [Tancik et al., 2020]. In addition, replacing the encoder helps reduce parameter count while still maintaining high performance on PDE benchmarks.

2 Methodology

2.1 Background

Let d be the number of input spatial dimensions. Let Ω be an open set in $\mathbb{R}^d$ bounded by $\partial\Omega$. A PDE with spatiotemporal input $(\mathbf{x}, t)$ where $\mathbf{x} \in \mathbb{R}^d$ and $t \in \mathbb{R}$ follows the abstraction:

$$\mathcal{F}(u(\mathbf{x}, t)) = 0, \qquad \forall(\mathbf{x}, t) \in \Omega \times [0, T], \tag{1}$$

$$\mathcal{I}(u(\mathbf{x}, 0)) = 0, \qquad \forall \mathbf{x} \in \Omega, \tag{2}$$

$$\mathcal{B}(u(\mathbf{x}, t)) = 0, \qquad \forall(\mathbf{x}, t) \in \partial\Omega \times [0, T] \tag{3}$$

where $u : \mathbb{R}^{d+1} \to \mathbb{R}^m$ is the solution to the PDE. The operator $\mathcal{F}$ encodes the physical law (the PDE in residual form), $\mathcal{I}$ represents initial conditions, and $\mathcal{B}$ encodes boundary conditions. Physics-Informed Neural Networks (PINNs) approximate $u(\mathbf{x}, t)$ by enforcing these constraints during training through the loss function:

$$\begin{aligned}\mathcal{L}(u_\theta) =& \frac{\lambda_1}{N_\mathcal{F}} \sum_{i=1}^{N_\mathcal{F}} \|\mathcal{F}(u_\theta(\mathbf{x}_i, t_i))\|^2 \\ &+ \frac{\lambda_2}{N_\mathcal{I}} \sum_{i=1}^{N_\mathcal{I}} \|\mathcal{I}(u_\theta(\mathbf{x}_i, 0))\|^2 \\ &+ \frac{\lambda_3}{N_\mathcal{B}} \sum_{i=1}^{N_\mathcal{B}} \|\mathcal{B}(u_\theta(\mathbf{x}_i, t_i))\|^2\end{aligned} \tag{4}$$

where u_θ is the neural network approximation of the PDE solution, $N_\mathcal{F}, N_\mathcal{I}, N_\mathcal{B}$ are the number of training points used for PDE, initial, and boundary regions, and $\lambda_1, \lambda_2, \lambda_3$ are respective weighting coefficients for each loss term.

Traditional PINNs focus on point-wise predictions without incorporating temporal dependencies in PDE solutions. This approach is suitable primarily for elliptic PDEs, which lack explicit time derivatives. However, hyperbolic and parabolic PDEs involve time derivatives, which the architecture of MLP-based PINNs does not inherently account for [Zhao et al., 2024].

2.2 Previous Architecture

To address the lack of spatiotemporal relationships in MLP-based PINNs, the PINNsformer architecture was proposed. PINNsformers, which are based on encoder-decoder transformer architectures, rely on attention mechanisms to better capture spatio-temporal relationships compared to a traditional MLP PINN. The PINNsformer architecture is built on four main components: a pseudo-sequence generator, a spatio-temporal mixer, an encoder-decoder with multi-head attention, and an output layer [Zhao et al., 2024].

Pseudo-Sequence Generator Transformer-based models are trained on sequential data that are not compatible with the singular points used to train MLP-based PINNs. To account for this, the PINNsformer paper proposed the pseudo-sequence generator [Zhao et al., 2024], which performs the following operation given a single space-time coordinate:

$$(\mathbf{x}, t) \xRightarrow{\text{Gen}} \{(\mathbf{x}, t), (\mathbf{x}, t + \Delta t), \ldots, (\mathbf{x}, t + (k-1)\Delta t)\} \tag{5}$$

where Δt is a very small constant, and k is the number of timesteps. This creates a temporal sequence while preserving the underlying physical relationships [Zhao et al., 2024]. In practice, this means taking a single spatiotemporal coordinate and generating a temporal sequence by holding the spatial location $\mathbf{x}$ constant while creating k time points separated by intervals of Δt.

Encoder-Decoder Architecture The Encoder-Decoder, inspired by Transformers, includes multiple layers combining self-attention and feedforward operations in the encoder, while the decoder excludes self-attention and reuses the encoder's embeddings [Vaswani et al., 2023]. This design enables effective spatio-temporal dependency learning in differential equation solutions. [Zhao et al., 2024].

Wavelet Activation Function The Wavelet Activation function is introduced in place of traditional non-linear functions like ReLU and LayerNorm, which can be suboptimal for Physics-Informed Neural Networks (PINNs) due to issues like discontinuous derivatives [Zhao et al., 2024]. Inspired by the Real Fourier Transform, the Wavelet function captures periodic behavior effectively without requiring prior knowledge of the solution. This is formulated as a weighted sum of sine and cosine:

$$\text{Wavelet}(z) = \omega_1 \sin(z) + \omega_2 \cos(z) \tag{6}$$

where ω_1 and ω_2 are learnable parameters. The Wavelet Activation function is used in the PINNsformer architecture as an activation function in the encoder and decoder [Zhao et al., 2024].

2.3 Spectral Architecture

The encoder in the PINNsformer potentially introduces unnecessary computational overhead and parameter redundancy. Traditional encoder-decoder architectures were designed for sequence-to-sequence tasks where input and output have different structures (e.g., translation) [Gao et al., 2022]. The encoder creates representations that the decoder must then reinterpret, creating an potentially unnecessary computational bottleneck. This two-stage processing adds complexity without a potentially corresponding benefit. Furthermore, the encoder does not directly address spectral bias, a fundamental limitation in neural PDE solvers [Wang et al., 2021].

Therefore, we propose a Spectral PINNsformer (S-Pformer), which is a decoder-only transformer architecture for efficiency and simplicity. This new architecture (shown in Figure 1)is smaller than the full PINNsformer architecture, inherently prevents spectral bias with fourier features, and offers improved performance compared to previous architectures. The S-Pformer consists of three parts: Input embeddings with Fourier features, a decoder with multi-head attention, and an output linear layer.

2.3.1 Embeddings

The input, given as a sequence of space-time points from the pseudo-sequence generator, is encoded using the modified embeddings module. It combines two components: a Fourier feature mapping and a learnable positional embedding. The Fourier mapping transforms the low-dimensional input $[\mathbf{x}, t]$ into a higher-dimensional periodic space using sine and cosine functions applied to randomized frequency projections [Tancik et al., 2020]. The Fourier feature mapping allows the model to inherently resolve spectral bias [Wang et al., 2021]. In parallel, the positional embedding applies a linear transformation directly to the input coordinates, which preserves spatial and temporal locality. This feature mapping directly replaces the spatio-temporal mixer and the encoder from the original architecture. Fourier features encode global periodic patterns essential for capturing oscillatory PDE solutions across multiple frequency scales, while positional embeddings preserve local spatial-temporal relationships.

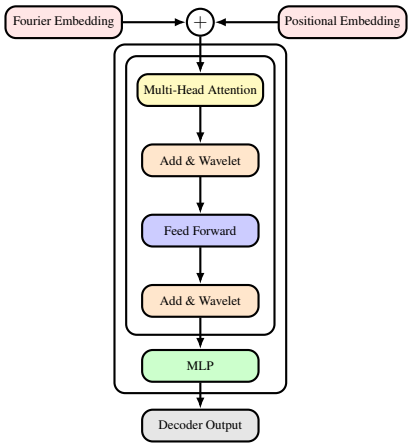


Figure 1: Spectral PINNsformer Architecture

Let $\mathbf{z} = (\mathbf{x}, t) \in \mathbb{R}^{d_{\text{in}}}$ denote the input coordinate where $d_{\text{in}} = d + 1$ (spatial dimensions plus time). The model first normalizes the coordinate to $\tilde{\mathbf{z}} = (\tilde{\mathbf{x}}, \tilde{t}) \in [0, 1]^{d_{\text{in}}}$ such that all components fall within the range $[0, 1]$.

This normalized input is then fed into our embedding module with output dimension d_{emb}. It consists of two parts: a Fourier feature embedding, and a positional embedding. The Fourier embedding depends on d_{mapping}, which determines the number of frequency bands that the input is projected into. We use a random projection matrix $\mathbf{B} \in \mathbb{R}^{d_{\text{mapping}} \times d_{\text{in}}}$ sampled from $\mathcal{N}(0, \sigma^2 \mathbf{I})$. Since the input coordinates are normalized to [0,1], we set $\sigma^2 = 1$ to ensure the random frequency projections have unit variance. Given a linear transformation $\theta_f : \mathbb{R}^{2d_{\text{mapping}}} \to \mathbb{R}^{d_{\text{emb}}}$, the Fourier feature embedding $E_f(\tilde{\mathbf{z}})$ is:

$$E_f(\tilde{\mathbf{z}}) = \theta_f \left(\begin{bmatrix} \sin(2\pi \mathbf{B} \tilde{\mathbf{z}}) \\ \cos(2\pi \mathbf{B} \tilde{\mathbf{z}}) \end{bmatrix} \right) \tag{7}$$

The positional embedding applies a linear transformation $\theta_p : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{emb}}}$ to preserve spatial-temporal locality:

$$E_p(\tilde{\mathbf{z}}) = \theta_p(\tilde{\mathbf{z}}) \tag{8}$$

Finally, the combined input embedding is written as

$$E(\tilde{\mathbf{z}}) = E_f(\tilde{\mathbf{z}}) + E_p(\tilde{\mathbf{z}}) \tag{9}$$

2.3.2 Attention-Driven Decoder

The decoder processes the embedded input through N transformer layers, where each layer consists of multi-head attention followed by a feed-forward network, with residual connections and wavelet activations.

Let $\phi(z)$ denote the Wavelet activation function. Let $\mathrm{MH}(Q, K, V)$ denote a multi-head attention operation with query Q, key K, and value V matrices. The feed-forward component $\mathrm{FF}(x)$ is defined as a 3-layer MLP with hidden dimension d_{ff}.

The decoder processes N layers sequentially. We initialize with the embedding output:

$$H^{(0)} = E(\tilde{\mathbf{z}}) \tag{10}$$

Each decoder layer applies the following transformations:

$$U^{(l)} = \phi(H^{(l-1)}) \quad \text{(Pre-attention normalization)} \tag{11}$$

$$A^{(l)} = \mathrm{MH}(U^{(l)}, U^{(l)}, U^{(l)}) \quad \text{(Self-attention)} \tag{12}$$

$$S^{(l)} = H^{(l-1)} + A^{(l)} \quad \text{(Residual connection)} \tag{13}$$

$$V^{(l)} = \phi(S^{(l)}) \quad \text{(Pre-FFN normalization)} \tag{14}$$

$$F^{(l)} = \mathrm{FF}(V^{(l)}) \quad \text{(Feed-forward)} \tag{15}$$

$$H^{(l)} = S^{(l)} + F^{(l)} \quad \text{(Residual connection)} \tag{16}$$

2.3.3 Linear Output Network

We are given our output dimension d_{out}. Given an a 3-layer output MLP θ_{out} with hidden dimension d_{hidden}, we can write our final output of the model with decoder input D as:

$$\mathrm{Out}(D) = \theta_{\text{out}}(D) \tag{17}$$

2.4 NTK Learning Scheme

We use Neural Tangent Kernel methods to balance the loss components as in Equation 4. PINNs typically struggle with convergence due to the imbalanced contributions of residual, boundary, and initial losses. To address this, the model computes the NTK trace over different sets of loss terms [Wang et al., 2020].

First, the training loop computes the Jacobians (gradients) of predictions with respect to model parameters for PDE residuals, initial conditions, and boundary conditions.

For each Jacobian, we calculate the NTK trace as:

$$K_i = \mathrm{Tr}(J_i J_i^\top) \tag{18}$$

where J_i is the Jacobian matrix of the model outputs (corresponding to the i-th loss component) with respect to the model parameters. This measures the sensitivity of each loss component on the model's parameters during training [Zhao et al., 2024]. A higher K_i indicates the corresponding loss component has a greater influence on the parameter updates. As a result, each loss weight is inversely proportional to the K_i value. Each weight is calculated as:

$$\lambda_i = \frac{\sum K}{K_i} \tag{19}$$

This dynamic weighting process leads to more stable convergence during the training process. We re-compute each weight every 50 iterations in the training loop.

3 Experiments

3.1 Setup

3.1.1 Model Ablation

To examine the effects of the Fourier Features on the performance of the model, we created a new model called the Decoder-Only PINNsformer. This replaces the Fourier Feature embeding in the Spectral PINNsformer architecture with a single linear layer. This model serves as an ablation to examine the effects of the Fourier Features on model performance.

3.1.2 Data Generation

Training and test datasets are generated via uniform sampling of collocation points over the spatial and temporal domains for each PDE. For the 1D-reaction, convection and wave equations on $x \in [x_{\text{min}}, x_{\text{max}}]$, $t \in [t_{\text{min}}, t_{\text{max}}]$ as defined in A.3, we generated the following data sets:

- Initial condition points $\{(x_i, 0)\}_{i=1}^{N_{ic}}$ with x_i drawn uniformly from $[x_{\text{min}}, x_{\text{max}}]$.
- Boundary condition points $\{(0, t_j)\}_{j=1}^{N_{bc}}$ and $\{(2\pi, t_j)\}_{j=1}^{N_{bc}}$ with t_j drawn uniformly from $[t_{\text{min}}, t_{\text{max}}]$.
- Residual collocation points $\{(x_i, t_j)\}_{i=1,j=1}^{N_x,N_t}$ on a Cartesian grid of size $N_x \times N_t$.

For the MLP-based PINNs baseline we set $N_{ic} = N_{bc} = 101$ and $N_x = N_t = 101$. For the transformer-based models (Pformer, S-Pformer, DO-Pformer) we use $N_{ic} = N_{bc} = 51$ and $N_x = N_t = 51$. Exact analytical solutions $u_{\text{true}}(x, t)$ are computed in closed form and evaluated on a test grid of 101×101 points for error metrics. The network will learn the solution to the PDE by evaluating on these collocation points and enforcing the PDE via automatic differentiation.

Navier–Stokes Data Generation We use the 2D cylinder wake dataset from Raissi et al. [2019], which provides velocity fields $U_* \in \mathbb{R}^{N\times 2\times T}$ and pressure $p_* \in \mathbb{R}^{N\times T}$ at spatial locations $X_* \in \mathbb{R}^{N\times 2}$ over time $t \in \mathbb{R}^T$. We form a full spatio-temporal grid by combining all spatial points with all time steps, yielding $N \times T$ total points. From these, we randomly sample 2500 training points with coordinates (x, y, t) as model inputs. The velocity components (u, v) serve as ground truth for the loss function. For evaluation, we use the pressure field at $t = 20.0$.

Because the incompressible Navier-Stokes equations determine pressure only up to an additive constant, we align predictions with ground truth by computing the optimal offset:

$$C = \frac{1}{N} \sum_{i=1}^{N} (p_{\text{true},i} - p_{\text{pred},i}) \tag{20}$$

The corrected prediction $p'_{\text{pred}} = p_{\text{pred}} + C$ is used for evaluation.

3.1.3 Network Benchmarking

Our empirical evaluations rely on four types of PDEs: convection, 1D-reaction, 1D-wave, and Navier-Stokes, as defined in A.3. For the MLP-based architecture, we uniformly sampled N_{ic} = N_{bc} = 101 initial and boundary points, as well as a uniform grid of 101×101 mesh points for the residual domain. In the case of training Pformer, S-Pformer and DO-Pformer, we reduce the collocation points to N_{ic} = N_{bc} = 51 initial and boundary points, and a uniform grid of 51×51 mesh points. For the Navier-Stokes PDE, we sample 2500 points from the residual domain for training. For all models, $d_{\text{hidden}} = 512$, $d_{\text{emb}} = 32$, $N = 1$ (the number of decoder layers), $n_{\text{heads}} = 2$ (the number of attention heads). For the S-Pformer, we had a baseline of $d_{\text{mapping}} = 64$. The experimental setup detailed above closely matches the analysis put forth in the original PINNsformer paper for equal benchmarking [Zhao et al., 2024].

3.1.4 Evaluation

All models were trained using the L-BFGS optimizer with Strong-Wolfe linear search for 1000 iterations. We use the L-BFGS optimizer as opposed to the more widespread Adam optimizer because of its enhanced performance in PINN optimization tasks [Urbán et al., 2025].

3.1.5 Reproducibility

All models are implemented in PyTorch [Paszke et al., 2019], and are trained on single NVIDIA Tesla A10G GPU.

All code is included and reproducible at this URL.

3.2 Results

We first compare parameter counts across the PINNsformer (Pformer), Spectral PINNsformer (S-Pformer), and Decoder-Only PINNsformer (DO-Pformer) using the parameters outlined in 3.1.3.

The S-Pformer achieves an 18.6% reduction in parameter count with respect to the Pformer, making it a more lightweight and efficient model by comparison.

Table 1: Parameter comparison between models

PFormer	DO-Pformer	S-Pformer
453,561	366,959	369,039

Next, we evaluate all transformer-based models using the un-optimized parameters outlined in experimental setup across four benchmark PDEs: Convection, 1D-Reaction, 1D-Wave, and 2D Navier-Stokes. We report three metrics for each PDE: the relative Mean Absolute Error (rMAE), the relative Root Mean Squared Error (rMSE), and the training time.

Table 2: Comparison of transformer-based models on different PDE types, models as described in Table 1

Model	PDE Type	rMAE	rMSE	Training Time (H:MM:SS)
Pformer	Convection	0.018	0.020	0:17:53
	1D-Reaction	7.38e-3	0.163	0:03:59
	1D-Wave	0.083	0.091	1:11:45
	Navier-Stokes	0.091	0.085	2:17:09
DO-Pformer	Convection	0.025	0.029	0:11:41
	1D-Reaction	9.12e-3	0.020	0:03:40
	1D-Wave	0.015	0.017	0:37:48
	Navier-Stokes	0.095	0.110	1:37:22
S-Pformer	Convection	**0.016**	**0.018**	0:14:29
	1D-Reaction	**1.15e-3**	**2.98e-3**	0:03:48
	1D-Wave	**6.94e-3**	**7.01e-3**	0:42:40
	Navier-Stokes	**0.079**	**0.071**	1:03:55

To further analyze the effect of Fourier features on the effectiveness of the model at handling spectral bias, we can show the error across different frequency bands of the solution of the convection PDE using a Fourier transform. We take a Fourier transform of each convection PDE solution along the spatial domain. We define the Nyquist sampling frequency f_n, and separate frequencies into frequency bands based on relative positions to the Nyquist frequency. We then compute the frequency error for each spectral band using mean absolute error (MAE).

Table 3: Error (MAE) across FFT frequency bands for different transformer-based models on convection problem, models as described in Table 1.

	Error (MAE)		
FFT Frequency Band	S-Pformer	DO-Pformer	Pformer
Very Low Frequency ($f < 0.3\, f_n$)	0.1401	0.1940	**0.1400**
Low Frequency ($0.3\, f_n \leq f < 0.5\, f_n$)	**0.0904**	0.1683	0.1764
Mid Frequency ($0.5\, f_n \leq f < 0.7\, f_n$)	**0.0302**	0.0354	0.0363
High Frequency ($0.7\, f_n \leq f < 0.9\, f_n$)	**0.0110**	0.0157	0.0155
Very High Frequency ($f \geq 0.9\, f_n$)	**0.0093**	0.0136	0.0133

In addition, we optimized the hyperparameters d_{hidden}, d_{emb}, d_{mapping} of the S-Pformer to show the full capability of the architecture. We implemented a Bayesian optimization algorithm using Optuna [Akiba et al., 2019] for 100 trials for each problem. The model yielding the lowest rMAE was chosen over the 100 trials. For comparison, we also optimized the d_{hidden} and n_{layers} of an MLP-based PINN for comparison for 100 trials using Optuna. Optimized hyperpaameters are shown in A.2.

Table 4: Comparison of optimized S-Pformer vs. optimized MLP-PINN performance

Problem	Model	rMAE	rMSE	Num. Params
Convection	MLP-PINN	0.663	0.745	**66,561**
	S-Pformer	**0.015**	**0.018**	305,551
1D-Reaction	MLP-PINN	0.014	0.028	1,052,673
	S-Pformer	**1.09e-3**	**2.15e-3**	**167,471**
1D-Wave	MLP-PINN	0.023	0.023	2,365,441
	S-Pformer	**2.89e-3**	**2.94e-3**	**247,823**
2D Navier-Stokes	MLP-PINN	**0.045**	**0.046**	264,706
	S-Pformer	0.057	0.062	**149,680**

4 Discussion

Our results demonstrate that architectural simplification yields superior performance in physics-informed neural networks. The S-Pformer achieves consistent improvements compared to transformer-based architectures across all benchmark PDEs while reducing parameter count by 18.6%, challenging the conventional "bigger is better" paradigm in physics-based deep learning. In addition, the optimized S-Pformer shows improved or comparable performance compared to an optimized MLP-based PINN, while using a fraction of the parameter count.

The frequency band analysis (Table 3) provides direct evidence that Fourier features effectively address spectral bias - a persistent limitation of both traditional PINNs and the original PINNsformer. The 30% error reduction in high-frequency regimes ($f > 0.7f_n$) compared to the decoder-only baseline confirms that explicit frequency encoding is important for capturing multiscale PDE behaviors. This improvement is significant for problems like the convection equation where high-frequency dynamics dominate the solution.

The decoder-only design proves that the encoder in the original PINNsformer introduced unnecessary computational overhead without corresponding performance gains. By applying self-attention directly to embedded coordinates, we maintain the temporal dependency modeling that makes transformers effective for time-dependent PDEs while eliminating redundant computation. This efficiency gain becomes more pronounced with increasing problem complexity, as evidenced by consistently shorter training times.

The S-Pformer demonstrates consistent versatility across elliptic, parabolic, and hyperbolic PDEs. This robustness stems from the adaptive nature of the Fourier feature mapping, which learns optimal frequency representations for each problem type rather than relying on fixed spectral assumptions.

While our optimized S-Pformer variants (Table 6) show the architecture's full potential, the hyperparameter sensitivity suggests room for more principled hyperparameter choices. While we optimized key architectural parameters (d_{hidden}, d_{emb}, d_{mapping}), future work should explore the sensitivity to attention head count, though preliminary experiments suggest $n_{\text{heads}} = 2$ provides a good efficiency-performance trade-off.

The Navier-Stokes results reveal a constraint: while the S-Pformer excels on physics-informed problems with purely automatic-differentiation-based losses, it shows marginal underperformance compared to MLPs on data-driven components such as the Navier-Stokes benchmark.

Future work should investigate adaptive frequency selection mechanisms, extend evaluation to more complex geometries and coupled systems, and explore approaches that combine the strengths of both transformer and MLP architectures for different physics-based and data-driven components of the PINN loss function.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL `https://arxiv.org/abs/1907.10902`.

Bengt Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1996.

Yingbo Gao, Christian Herold, Zijian Yang, and Hermann Ney. Is encoder-decoder redundant for neural machine translation?, 2022. URL https://arxiv.org/abs/2210.11807.

V Murugesh, M Priyadharshini, T R Mahesh, and Esmael Adem Esleman. A thermodynamic inspired ai based search algorithm for solving ordinary differential equations. *Scientific Reports*, 15, 05 2025. doi: 10.1038/s41598-025-03093-6.

Yasuki Nakayama. Chapter 15 - computational fluid dynamics. In Yasuki Nakayama, editor, *Introduction to Fluid Mechanics (Second Edition)*, pages 293–327. Butterworth-Heinemann, second edition edition, 2018. ISBN 978-0-08-102437-9. doi: https://doi.org/10.1016/B978-0-08-102437-9.00015-2. URL https://www.sciencedirect.com/science/article/pii/B9780081024379000152.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

M. Raissi, P. Perdikaris, and 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*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. URL https://arxiv.org/abs/2006.10739.

Jorge F. Urbán, Petros Stefanou, and José A. Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can pinns be? *Journal of Computational Physics*, 523:113656, 2025. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2024.113656. URL https://www.sciencedirect.com/science/article/pii/S0021999124009045.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *CoRR*, abs/2007.14527, 2020. URL https://arxiv.org/abs/2007.14527.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.113938. URL https://www.sciencedirect.com/science/article/pii/S0045782521002759.

Chenhui Xu, Dancheng Liu, Yuting Hu, Jiajie Li, Ruiyang Qin, Qingxiao Zheng, and Jinjun Xiong. Sub-sequential physics-informed learning with state space model, 2025. URL https://arxiv.org/abs/2502.00318.

Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks, 2024. URL https://arxiv.org/abs/2307.11833.

A Appendix

A.1 PINN Architecture Computational Performance Comparison

We took every model type and computed metrics to evaluate computational performance, in this case on the 1D-Reaction PDE.

Table 5: Computational Performance Comparison of PINN Architectures

Model	Avg. Step Time (s)	Avg. GPU Mem. (MB)	Params	MFLOPs
MLP-based PINN	0.39	430.5	527,361	5.28
Pformer	0.54	457.7	453,561	4.54
DO-Pformer	0.48	363.5	366,959	3.68
S-Pformer	0.50	372.5	370,991	3.72

A.2 Hyperparameter Optimization

We optimized the hyperparameters d_{hidden}, d_{emb}, d_{mapping} of the S-Pformer in Optuna for 100 trials for each problem. The model yielding the lowest rMAE was chosen over the 100 trials.

Table 6: Optimized S-Pformer Performance

Problem	rMAE	rMSE	d_{hidden}	d_{emb}	d_{mapping}	Num. Params
Convection	0.015	0.018	256	128	32	305,551
1D-Reaction	1.09e-3	2.15e-3	256	32	96	167,471
1D-Wave	2.89e-3	2.94e-3	128	128	64	247,823
2D Navier-Stokes	0.057	0.062	256	16	112	149,680

We also optimized the d_{hidden} and n_{layers} of an MLP-based PINN for comparison for 100 trials using Optuna.

Table 7: Optimized MLP-PINN Performance

Problem	rMAE	rMSE	d_{hidden}	n_{layers}	Num. Params
Convection	0.663	0.745	128	6	66,561
1D-Reaction	0.014	0.028	512	6	1,052,673
1D-Wave	0.023	0.023	768	6	2,365,441
2D Navier-Stokes	0.045	0.046	256	6	264,706

A.3 PDE Equations

Convection The one-dimensional convection problem is a hyperbolic PDE used to model transfer processes.

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad \forall x \in [0, 2\pi],\ t \in [0, 1] \tag{21}$$

$$\text{IC:} \quad u(x, 0) = \sin(x) \qquad \text{BC:} \quad u(0, t) = u(2\pi, t) \tag{22}$$

where β is the convection coefficient. In this case, $\beta = 50$. This is a high-frequency PDE, which makes it difficult for conventional PINNs to approximate.

1D-Reaction The one-dimensional reaction problem is a hyperbolic PDE used to model chemical reactions.

$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0, \quad \forall x \in [0, 2\pi],\ t \in [0, 1] \tag{23}$$

$$\text{IC:} \quad u(x, 0) = \exp\left(-\frac{(x-\pi)^2}{2(\pi/4)^2}\right), \qquad \text{BC:} \quad u(0, t) = u(2\pi, t) \tag{24}$$

where ρ is the reaction coefficient, where $\rho = 5$. The equation has an analytical solution:

$$u_{\text{analytical}} = \frac{h(x)\exp(\rho t)}{h(x)\exp(\rho t) + 1 - h(x)} \tag{25}$$

where $h(x)$ is the function of the initial condition.

1D-Wave PDE The 1D-Wave equation is a hyperbolic PDE that is used to describe the propagation of waves in one spatial dimension.

$$\frac{\partial^2 u}{\partial t^2} - \beta \frac{\partial^2 u}{\partial x^2} = 0 \quad \forall x \in [0, 1],\ t \in [0, 1] \tag{26}$$

$$\text{IC:} \quad u(x, 0) = \sin(\pi x) + \frac{1}{2}\sin(\beta \pi x), \quad \frac{\partial u(x, 0)}{\partial t} = 0 \tag{27}$$

$$\text{BC:} \quad u(0, t) = u(1, t) = 0 \tag{28}$$

where β is the wave speed, where $\beta = 3$. The equation has an analytical solution:

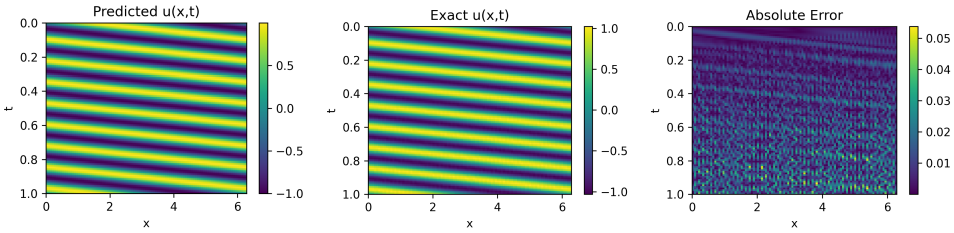
$$u(x, t) = \sin(\pi x)\cos(2\pi t) + \frac{1}{2}\sin(3\pi x)\cos(6\pi t) \tag{29}$$

2D Navier-Stokes PDE. The 2D Navier-Stokes equation is a parabolic PDE that consists of a pair of partial differential equations that describe the behavior of incompressible fluid flow in two-dimensional space.

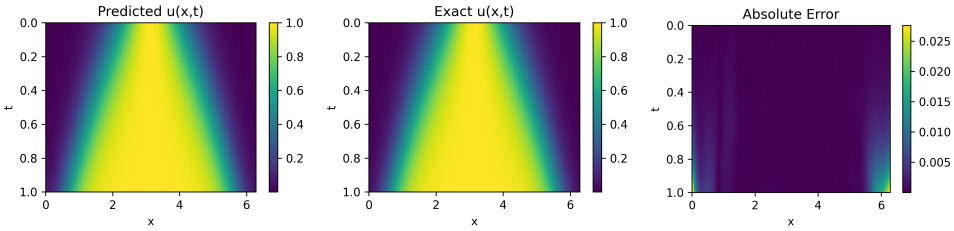
$$\frac{\partial u}{\partial t} + \lambda_1 \left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \lambda_2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$
$$\frac{\partial v}{\partial t} + \lambda_1 \left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \lambda_2 \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

where $u(t, x, y)$ and $v(t, x, y)$ are the x-component and y-component of the velocity field separately, and $p(t, x, y)$ is the pressure. Here, $\lambda_1 = 1$ and $\lambda_2 = 0.01$. The simulated solution is given by [Raissi et al., 2019].

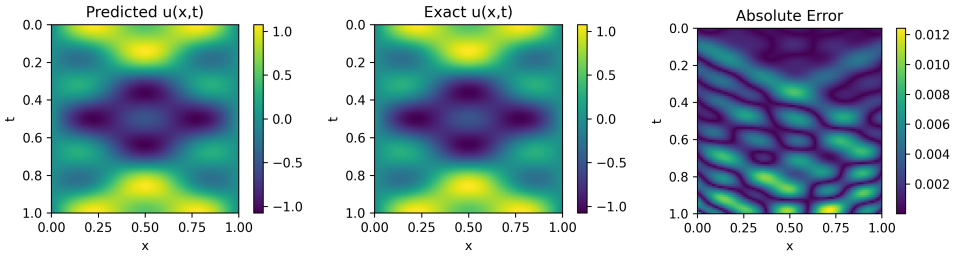
A.4 Visualization of Spectral PINNsformer



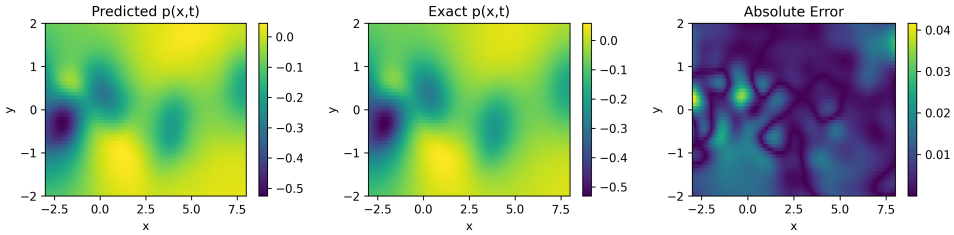
(a) Pred - Convection (b) Exact - Convection (c) Error - Convection



(d) Pred - 1D Reaction (e) Exact - 1D Reaction (f) Error - 1D Reaction



(g) Pred - 1D Wave (h) Exact - 1D Wave (i) Error - 1D Wave



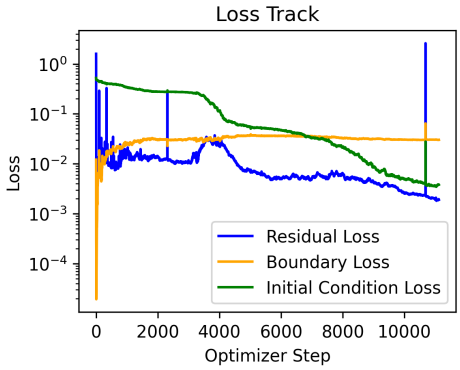
(j) Pred - 2D N-S ($t = 20$) (k) Exact - 2D N-S ($t = 20$) (l) Error - 2D N-S ($t = 20$)

Figure 2: In the figure above, the first column shows the S-Pformer prediction, the middle column shows the ground truth, and the last column shows the prediction error.

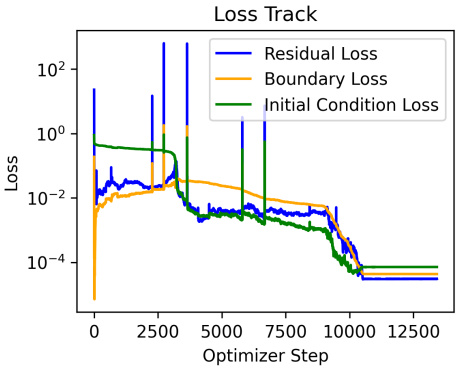
A.5 Loss Convergence

To examine the stability and convergence of the loss as all models trained, we plotted the loss for each optimizer step during the training process.

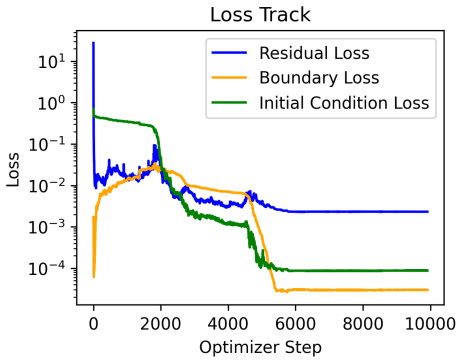
A.5.1 Convection Loss



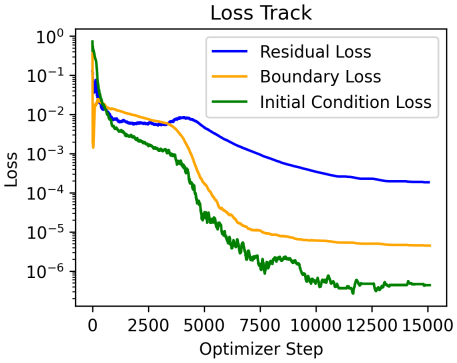
(a) Loss - PINN



(b) Loss - Pformer

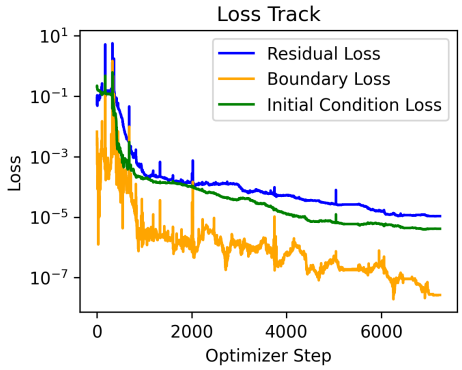


(a) Loss - DO-Pformer

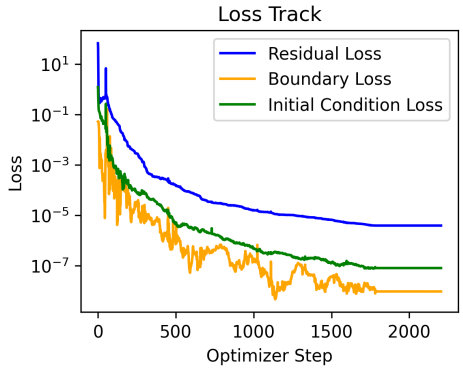


(b) Loss - S-Pformer

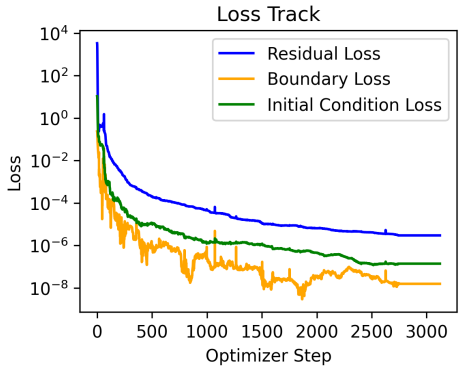
A.5.2 1D Reaction Loss



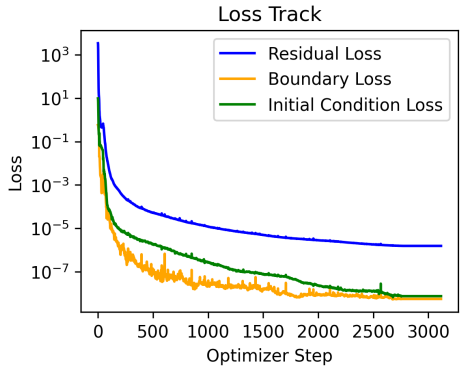
(a) Loss - PINN



(b) Loss - Pformer

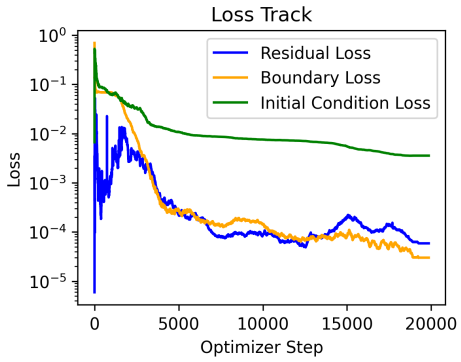


(a) Loss - DO-Pformer

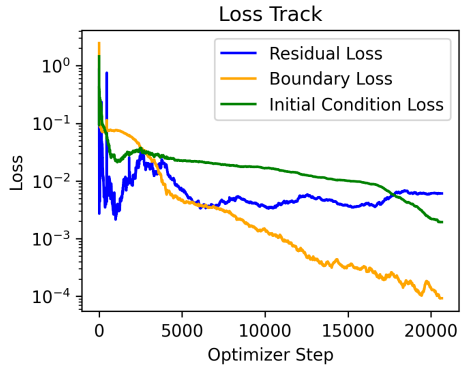


(b) Loss - S-Pformer

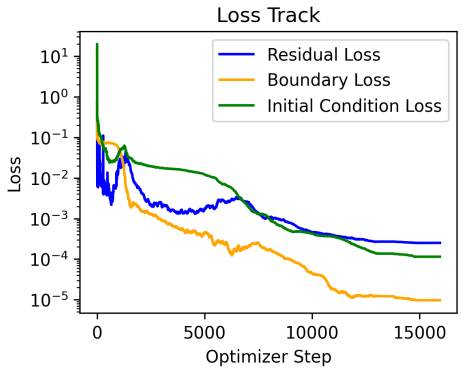
A.5.3 1D Wave Loss



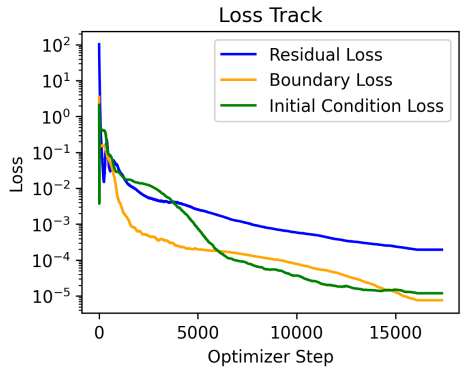
(a) Loss - PINN



(b) Loss - Pformer



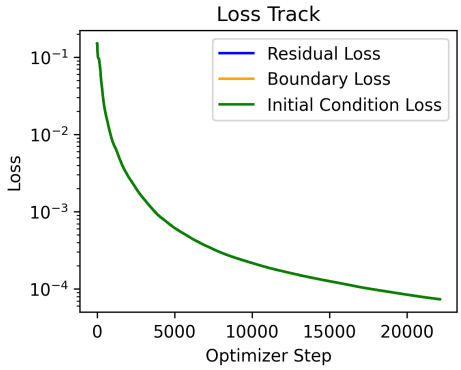
(a) Loss - DO-Pformer



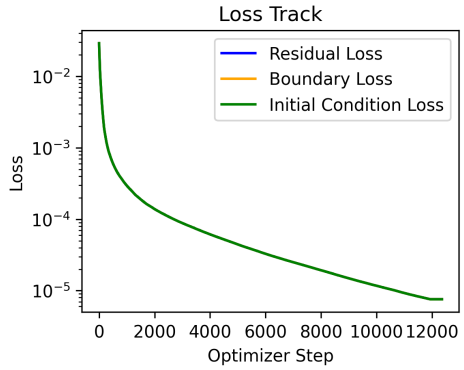
(b) Loss - S-Pformer

A.5.4 2D Navier-Stokes Loss

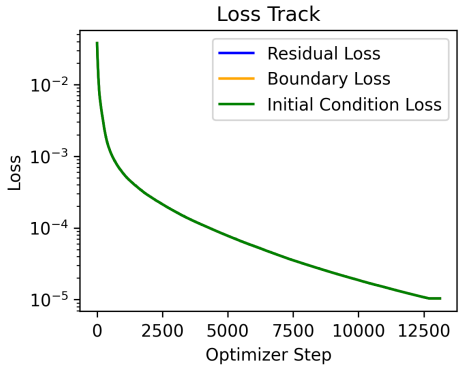
Note: There were no initial or boundary conditions on this problem, therefore only the residual loss is plotted.



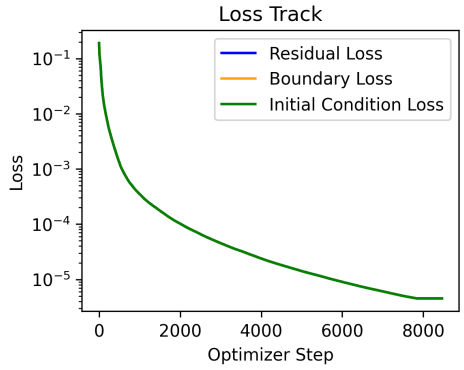
(a) Loss - PINN



(b) Loss - Pformer



(a) Loss - DO-Pformer



(b) Loss - S-Pformer