FOURIER FEATURE PYRAMIDS FOR PHYSICS-INFORMED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an improved neural field architecture for solving partial differential equations (PDEs). Current physics-informed neural networks (PINNs) provide a flexible framework for solving PDEs, but they struggle to achieve highly accurate solutions and require computation that scales poorly with parameter count. Our model, which we call `beignet` (Bandlimited Embedding with Interpolated Grid Network), replaces the random Fourier feature embedding used by existing PINN models with a trainable multi-resolution Fourier feature pyramid. To query `beignet` at a continuous coordinate, we use Fourier interpolation at each level of the pyramid to return features at the input coordinate, and then decode this vector with a fully-connected neural network trunk. Our model provides multiple benefits: 1) Spatial derivatives can be computed efficiently by using the chain rule to compose derivatives of the neural network computed with automatic differentiation with derivatives of the feature grid computed spectrally by the Fast Fourier transform (FFT). 2) `beignet` can achieve higher accuracy in a compute-efficient manner by scaling the parameter count of this Fourier feature pyramid, instead of the less-efficient strategy of scaling the neural network architecture. 3) `beignet` can directly control the representation bandlimit, resulting in more stable optimization for difficult PDEs. We demonstrate that `beignet` finds significantly more accurate solutions on PDE benchmarks using fewer parameters than state-of-the-art PINN methods. We further evaluate `beignet` on the self-similar inviscid Burgers blowup problem and show that it can minimize residuals to near machine precision using Adam, an accuracy regime previously attained only by using computationally expensive higher-order optimizers.

1 INTRODUCTION

Physics-informed neural networks (PINNs) (Raissi et al., 2019) are a flexible approach for approximating partial differential equation (PDE) solutions by parameterizing them as coordinate-based neural networks (multilayer perceptrons, or MLPs, that take in coordinates and output values of the field). PINNs are optimized to minimize the PDE residual over batches of coordinates sampled from the solution domain. As noted in prior work (Basri et al., 2020; Rahaman et al., 2019; Xu et al., 2020; 2019), neural networks suffer from spectral bias as standard gradient descent optimization dynamics result in them learning low-frequency components much faster than high-frequency components. Fourier feature embeddings (Tancik et al., 2020) mitigate this problem by passing input coordinates through a set of sinusoidal mappings before applying the MLP. In PINNs, the frequencies for these sinusoids are typically sampled from Gaussian distributions, with a single scalar that controls the bandwidth of the induced kernel (Wang et al., 2021).

Existing approaches for scaling PINNs to more challenging problems rely on increasing either the Fourier feature bandwidth or the size of the underlying MLP, but both strategies introduce significant optimization and computational challenges. Because Gaussian Fourier features are not strictly bandlimited, increasing the Gaussian bandwidth can introduce higher-frequency components into the induced kernel that can make optimization brittle. Alternatively, increasing the size of the MLP substantially increases the computational cost of evaluating PDE derivatives through automatic differentiation. The computational burden of scaling MLP-based PINNs is exacerbated by the need for second-order optimization methods to achieve high numerical accuracy (Wang et al., 2026; 2025b), and therefore memory consumption becomes prohibitive for even moderately large architectures.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We introduce a PINN architecture, which we call `beignet`, that addresses these issues by replacing Gaussian random Fourier features with a learnable multi-resolution Fourier feature pyramid. In the spatial domain, this representation can be thought of as a set of periodic feature grids at coarse-to-fine spatial resolutions. To evaluate the field at any continuous coordinate, we simply use Fourier interpolation to compute a single feature for each level of the pyramid, stack these into a feature vector for the coordinate, and pass this vector to a small MLP decoder. This representation allows us to bandlimit the representation by limiting the maximum resolution in the pyramid and also benefit from the coarse-to-fine optimization with a multiscale representation. Furthermore, the equispaced grid structure also enables efficient computation: feature values and derivatives can be evaluated on structured grids using the Fast Fourier transform (FFT) and then propagated through an MLP via the chain rule.

`beignet`'s explicit control over spatial frequency content is particularly useful for PINNs because different PDEs have different spectral requirements. Nonlinear terms in a PDE can generate high harmonics or sharp spatial structure, favoring broader frequency support. At the same time, differential operators amplify high frequency errors, so indiscriminately increasing the bandwidth can result in unstable optimization. Our Fourier feature pyramid exposes these choices directly: the finest grid resolution determines the maximum spatial frequency represented in the feature embedding, while the number of levels and channels per level controls the representational capacity within that band.

We evaluate `beignet` in two regimes. First, using four time-dependent PDEs from the JAX-PI benchmark (Wang et al., 2023a), we compare against the JAX-PI and PirateNet PINN architectures using a common evaluation protocol. These benchmarks span both scalar 1D+time PDEs and two-component 2D+time reaction–diffusion systems with periodic spatial domains. Across all four tasks, `beignet` achieves substantially lower relative error. We also investigate how different PDEs favor different spectral parameterizations—some benefit from broader frequency support (e.g., Allen–Cahn) while others perform best with a more compact bandwidth and greater concentration of representational capacity in low- and mid-frequency bands (e.g., KdV)—and demonstrate how `beignet`'s design space is well-suited for navigating this tradeoff. Second, we study how `beignet` can support extreme-precision optimization on a self-similar inviscid Burgers blowup problem, an important stepping stone toward understanding singularity formation in PDEs. This broader class of questions is central to mathematical fluid dynamics: the unforced Navier–Stokes existence and smoothness problem, for example, remains among the most fundamental open problems in mathematical analysis (Fefferman, 2006). In this setting, coordinate-based MLPs trained with Adam plateau at much higher residuals, while `beignet` drives the fixed-grid residuals to near machine precision with the same first-order optimizer. These results suggest that the feature representation is a key ingredient for scaling PINNs to high-accuracy regimes.

2 Related Work

Spectral-bias remedies in neural representations. Standard MLPs tend to fit smooth low-frequency structure before fine high-frequency variation. This behavior has motivated neural field and PINN architectures that make high-frequency components easier to represent or optimize. Fourier feature embeddings (Tancik et al., 2020), reviewed in Section 3.3, address this by mapping input coordinates with sinusoids of varying frequency. SIREN (Sitzmann et al., 2020) uses sinusoidal activation functions to represent fine detail and obtain accurate derivatives, BACON (Lindell et al., 2022) constructs band-limited coordinate networks with a multiscale representation, and WIRE (Saragadam et al., 2023) uses wavelet activations to combine frequency localization with spatial localization. Another line of work modifies the network parameterization through learnable univariate functions: Kolmogorov–Arnold Networks (KANs) (Liu et al., 2025; 2024) have been studied as a way to improve expressivity and mitigate spectral bias (Wang et al., 2025a), with extensions to PINNs (Shukla et al., 2024; Toscano et al., 2025) and neural operators (Lee et al., 2025). Our work shares the goal of improving high-frequency representation in PINNs, but instead of modifying the MLP decoder, we focus on improving the input feature representation.

Fourier-basis PINN architectures. Several works have incorporated Fourier bases into PINNs beyond the random Fourier feature embeddings discussed in Section 3.3. Wang et al. (2024) discussed equispaced spectral features as a coordinate embedding in their ablation studies, reporting that this achieved higher training accuracy than Random Fourier Features (RFF) but exhibited overfitting.

They consequently adopted RFF in their method's architecture. Our work revisits this design choice: by organizing equispaced features into a learnable multi-resolution pyramid and using stochastic grid shifts as regularization, we make deterministic equispaced Fourier features effective in the PINN setting while retaining direct control over spatial bandwidth and frequency-band redundancy. In a complementary architectural direction, Fourier PINNs (Cooley et al., 2025) combine equispaced Fourier bases additively with an MLP output and train via alternating least squares with L^2-regularized basis pruning. Our architecture instead prepends the equispaced basis as an input representation, preserving compatibility with the dominant architecture paradigm (RFF followed by a neural network decoder) and enabling direct substitution into existing architectures. Spectral-Informed Neural Networks (Yu et al., 2025) operate entirely in the spectral domain, taking frequencies as inputs and outputting Fourier coefficients for PDEs with periodic boundary conditions, a different regime from coordinate-based PINNs.

Fourier neural operators. Fourier Neural Operators (FNOs) (Kovachki et al., 2023; Li et al., 2021) also use spectral representations for PDEs, but typically target a different learning setting: they learn an amortized solution operator across many PDE instances, while PINNs optimize a representation for a particular instance from residual, boundary, and initial-condition losses. Physics-Informed Neural Operators (PINO) (Li et al., 2024; Ganeshram et al., 2022) narrow this gap by adding PDE residual losses to operator learning. Architecturally, FNOs use repeated grid-to-grid spectral convolution layers, whereas `beignet` uses a multi-resolution latent feature grid that is interpolated to collocation points and decoded by a small MLP.

3 Background

3.1 Physics-informed neural networks for solving PDEs

Consider a general partial differential equation (PDE) parameterized by a differential operator $\mathcal{F}$ acting on an unknown function $u : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$:

$$\mathcal{F}(\boldsymbol{x}, u, \nabla_{\boldsymbol{x}} u, \nabla^2_{\boldsymbol{x}} u, ...) = 0, \tag{1}$$

with some boundary conditions.

Physics-informed neural networks (PINNs) provide a framework to numerically approximate the solution u using a neural network. The neural network is trained to approximate a PDE solution by minimizing the square of the residual on the left hand side of Equation 1, averaged over batches of collocation points in the domain.

3.2 Spectral Bias and the Neural Tangent Kernel

Coordinate-based neural networks trained with gradient descent learn low-frequency components of a target function before high-frequency ones (Basri et al., 2020; Rahaman et al., 2019). This *spectral bias* is a property of optimization rather than expressivity, and it is especially relevant for PINNs, whose solutions may contain sharp interfaces, localized waves, or other fine-scale structure.

The neural tangent kernel (NTK) formalizes this phenomenon (Jacot et al., 2018). For a network $f_{\boldsymbol{\theta}}$ with training inputs $\{\boldsymbol{x}_i\}_{i=1}^N$, the empirical NTK at initialization is $\mathbf{K}_{ij} = \langle \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}_i), \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}_j) \rangle$. In the infinite-width or linearized-training regime, $\mathbf{K}$ remains approximately constant, and each component of the training error along an eigenvector of $\mathbf{K}$ decays exponentially under gradient flow at a rate proportional to the corresponding eigenvalue. When the kernel is stationary and the inputs lie on a uniform periodic grid, the eigenvectors of $\mathbf{K}$ are Fourier modes and the eigenvalue $\lambda(\mathbf{k})$ governs the convergence rate of the frequency-$\mathbf{k}$ error component. Spectral bias then corresponds to $\lambda(\mathbf{k})$, and consequently the convergence rate in frequency $\mathbf{k}$, decaying rapidly with $|\mathbf{k}|$ in coordinate MLPs. Shaping the kernel spectrum $\lambda(\mathbf{k})$ therefore controls how quickly each frequency component of the solution is learned.

3.3 Fourier Feature Embeddings

Fourier feature embeddings address spectral bias by modifying the input representation so that the NTK allocates more eigenvalue mass to higher spatial frequencies, thereby accelerating their

optimization. A Fourier feature embedding maps coordinates to a collection of sinusoidal features before applying a neural network (Tancik et al., 2020). Given a frequency matrix $\mathbf{B}$ with M rows, the embedding is

$$\gamma(\boldsymbol{x}) = \begin{bmatrix} \cos(\mathbf{B}\boldsymbol{x}) \\ \sin(\mathbf{B}\boldsymbol{x}) \end{bmatrix} , \tag{2}$$

with the trigonometric functions applied element-wise. A Fourier feature network then takes the form $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{MLP}_{\boldsymbol{\theta}}(\gamma(\boldsymbol{x}))$. In the case of a linear Fourier feature model $f(\boldsymbol{x}) = \frac{1}{\sqrt{M}}\mathbf{W}\gamma(\boldsymbol{x})$, with fixed matrix $\mathbf{B}$ and trainable output weights $\mathbf{W}$, the NTK is exactly the feature kernel

$$\mathbf{K}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{M}\sum_{j=1}^{M} \cos\big(\boldsymbol{b}_j^\top(\boldsymbol{x} - \boldsymbol{x}')\big) , \tag{3}$$

where $\boldsymbol{b}_j$ is the jth row of $\mathbf{B}$. Thus, the frequencies in $\mathbf{B}$ determine the spectrum of the kernel. For a nonlinear MLP, the NTK composes the MLP kernel with this feature kernel; in the infinite-width limit it remains stationary, and its spectrum spreads around each embedded frequency with a smooth falloff (Tancik et al., 2020). As a result, one can approximately tune the convergence behavior of a Fourier feature network by adjusting the frequencies in $\mathbf{B}$.

Most prior work constructs $\mathbf{B}$ using random Fourier features (RFFs), typically sampling entries of $\mathbf{B}$ independently from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. This is motivated by the classical kernel setting, where Gaussian RFFs give a Monte Carlo approximation to a stationary RBF kernel (Rahimi & Recht, 2007; 2008), whose bandwidth is set by σ (Tancik et al., 2020). This simple bandwidth control has made RFFs a standard tool for PINNs, where they substantially improve on coordinate networks without a feature embedding (Tancik et al., 2020; Wang et al., 2021). Gaussian draws concentrate near low frequencies, so higher-frequency components of the solution are covered sparsely and converge slowly. Such components matter little at moderate accuracy but can be limiting in high-precision regimes. `beignet` instead places its frequencies on a multi-resolution grid, which covers equispaced frequencies up to a chosen cutoff and lets us set how densely each band is covered.

4 METHOD

Our goal is to design a parametric family of neural function approximators which can be used to numerically solve a wide range of PDEs. Our representation, `beignet`, is based on Fourier features but we use a fixed multi-resolution grid of frequencies instead of random sampling. Our design enables tight control of the spectral content of the resulting function while enabling efficient computation of the output and its derivatives.

Section 4.1 describes our multi-resolution Fourier pyramid, the fundamental component in our method. Section 4.2 describes the extension of our method to time-dependent PDEs: each pyramid level stores a small number of learned temporal feature slices, which are linearly interpolated and decoded by a single shared MLP. Section 4.3 details how our representation enables efficient computation of derivatives, and Section 4.4 describes how we use the same properties to reduce aliasing.

4.1 MULTI-RESOLUTION FOURIER FEATURE PYRAMID

Our architecture relies on a learnable multi-resolution pyramid of periodic spatial feature grids. Each level of the pyramid stores a feature field at a prescribed grid resolution. Evaluating the model at a point consists of two steps. First, we use Fourier interpolation to query the input coordinate at each of the model's levels. We concatenate the resulting features, then pass them through a decoder MLP.

Formally, we represent our function $u : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}$ as the composition of L Fourier interpolation operations $g_{\boldsymbol{\theta}^{(\ell)}} : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{h}}}$ with parameters $\boldsymbol{\theta}^{(\ell)}$, one at each level $\ell = 1, ..., L$, followed by an MLP $f_{\boldsymbol{\phi}} : \mathbb{R}^{L \cdot d_{\mathrm{h}}} \to \mathbb{R}^{d_{\mathrm{out}}}$ with parameters $\boldsymbol{\phi}$:

$$u(\boldsymbol{x}) = f_{\boldsymbol{\phi}}\left(g_{\boldsymbol{\theta}^{(1)}}(\boldsymbol{x}), g_{\boldsymbol{\theta}^{(2)}}(\boldsymbol{x}), ..., g_{\boldsymbol{\theta}^{(L)}}(\boldsymbol{x})\right) . \tag{4}$$

Let $\widehat{\boldsymbol{\theta}}^{(\ell)}$ denote the normalized discrete Fourier transform of the learned spatial feature grid $\boldsymbol{\theta}^{(\ell)}$. The Fourier interpolation operation at the ℓth level can then be written as

$$g_{\boldsymbol{\theta}^{(\ell)}}(\boldsymbol{x}) = \mathrm{Re}\Big(\sum_{\mathbf{k} \in \mathcal{K}^{(\ell)}} \widehat{\boldsymbol{\theta}}^{(\ell)}[\mathbf{k}] \exp(2\pi i\, \mathbf{k} \cdot \boldsymbol{x})\Big) , \tag{5}$$

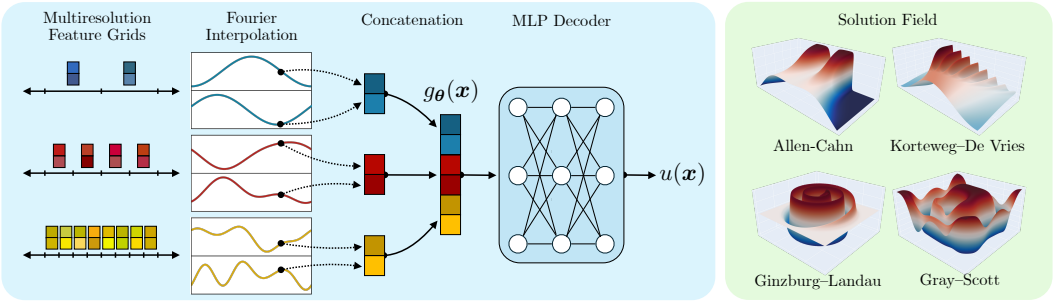


Figure 1: Architecture overview for `beignet`. A learnable multi-resolution Fourier feature pyramid stores feature grids at coarse-to-fine spatial resolutions, inducing a corresponding hierarchy of bandlimited feature fields. For each query point, every level is evaluated by Fourier interpolation to produce a per-level feature vector. These feature vectors are concatenated and decoded by a compact MLP to produce the solution value at a particular input coordinate; for time-dependent problems, the normalized time coordinate is also passed to the decoder. Our proposed architecture `beignet` is supervised on the initial conditions and PDE residual, and can be evaluated at arbitrary points inside the domain. `beignet` achieves SOTA performance on four time-dependent PDEs from the JAX-PI suite (Wang et al., 2023a), detailed in Sec. 5.

where $\widehat{\boldsymbol{\theta}}^{(\ell)}[\mathbf{k}]$ is the d_{h}-dimensional Fourier coefficient vector for mode $\mathbf{k}$. For a grid with resolution $N^{(\ell)}$ (assumed to be even) the signed FFT mode set is:

$$\mathcal{K}^{(\ell)} = \left\{ -\frac{N^{(\ell)}}{2}, -\frac{N^{(\ell)}}{2} + 1, ..., \frac{N^{(\ell)}}{2} - 1 \right\}^{d_{\mathrm{in}}}. \tag{6}$$

The resolution $N^{(\ell)}$ determines the spectral support available at that level, while the channel dimension d_{h} controls how much feature capacity is allocated within that support. By changing the number of levels L, their resolutions $N^{(1)}, ..., N^{(L)}$, and their channel counts d_{h}, the model can allocate more or less capacity to different bands. Unlike prior work which uses random Fourier features, this is done without relying on a random frequency draw. In `beignet`'s multi-resolution parameterization, the grid resolutions determine which feature-level frequencies are available, while the level structure and channel counts determine how much trainable feature capacity is placed at each scale. In practice, we choose the resolutions to scale dyadically, e.g., $N^{(\ell)} = 2^{\ell-1} N^{(1)}$, in the spirit of classical image-pyramid constructions such as the Laplacian pyramid (Burt & Adelson, 1983); however, the parameterization itself allows arbitrary choices of grid resolutions $N^{(1)}, \ldots, N^{(L)}$.

4.2 Temporal Pyramid Blending

For time-dependent PDEs, the solution can vary sharply in space while evolving non-periodically in time. Naïvely treating time as an additional coordinate and applying the same Fourier construction for it would impose an artificial temporal periodicity and couple the spatial FFT resolution to the number of time samples.

Our solution is to divide the time axis $t \in [0, 1]$ into T bins (in all of our time-dependent PDE experiments we use $T = 7$, i.e., 8 temporal slices), where for $i = 0, ..., T-1$, the ith bin spans $[i/T, i+1/T)$. We store $T+1$ pyramids, $\boldsymbol{\theta}_0, ..., \boldsymbol{\theta}_T$, each consisting of L levels, and in order to evaluate the function at a specific time value t, we linearly interpolate between the features:

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}_i + (tT - i)\,(\boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i), \tag{7}$$

where $i = \min(\lfloor tT \rfloor, T-1)$ is the index of the bin containing t, and the interpolation is done identically over all elements in all levels of the pyramids. The interpolated pyramid is then used for Fourier interpolation as in Equation 5, and fed into the shared MLP to predict the output value. In practice, we also input the time coordinate t directly to the MLP.

Our representation allows the spatial representation to remain bandlimited, while treating time as a non-periodic coordinate. The learned temporal slices provide local time-varying features, while the

raw time coordinate gives the MLP a direct continuous variable for smooth trends and for chaining time derivatives through the decoder. The number of time bins T controls the number of learned temporal anchors in the pyramid.

4.3 Spectral Feature Derivatives

The Fourier-interpolated feature grids also provide an efficient way to compute the spatial derivatives needed for computing the PDE residual (see Equation 1). We do this by differentiating the feature fields in Fourier space and then propagating through the decoder by the chain rule.

For a pyramid level ℓ, let $\boldsymbol{\theta}^{(\ell)}$ be the temporally interpolated spatial feature grid at time t (see Section 4.2 for details on this interpolation). Let $\widehat{\boldsymbol{\theta}}^{(\ell)}$ denote the normalized discrete Fourier transform of $\boldsymbol{\theta}^{(\ell)}$. The spatial derivatives of the feature field $g_{\boldsymbol{\theta}^{(\ell)}}$ can be computed in Fourier space by multiplying these coefficients by the wavenumbers. This can be seen by taking the derivative of Equation 5:

$$\nabla_{\boldsymbol{x}} g_{\boldsymbol{\theta}^{(\ell)}}(\boldsymbol{x}) = \mathrm{Re}\Big(\sum_{\mathbf{k}\in\mathcal{K}^{(\ell)}} \widehat{\boldsymbol{\theta}}^{(\ell)}[\mathbf{k}](2\pi i\mathbf{k}) \exp(2\pi i\, \mathbf{k}\cdot\boldsymbol{x})\Big), \tag{8}$$

and this generalizes to higher-order derivatives by multiplying by the corresponding higher-order wavenumber factors.

After multiplication by the appropriate wavenumber factors, we use the inverse Fast Fourier Transform (FFT) to obtain feature values and spatial derivative features on the structured collocation grid. When needed, spectra are zero-padded to match the residual evaluation grid before the inverse FFT. Raw step-time comparisons for this FFT-structured residual path are given in Appendix E.

This process allows us to compute all spatial derivatives of the Fourier interpolation g defined in Equation 5, but our goal is to compute the derivatives of the entire function u, i.e., the Fourier interpolation composed with the MLP. To do that, we use Faà di Bruno's formula, which generalizes the chain rule to higher order derivatives. For example, the first derivative of u is simply:

$$\nabla_{\boldsymbol{x}} u(\boldsymbol{x}) = \sum_{\ell=1}^{L} \frac{\partial f}{\partial g_{\boldsymbol{\theta}^{(\ell)}}} \nabla_{\boldsymbol{x}} g_{\boldsymbol{\theta}^{(\ell)}}(\boldsymbol{x})\,. \tag{9}$$

In implementation, these terms are evaluated with nested forward-mode automatic differentiation, using the spectrally computed feature derivatives as tangent directions. This avoids forming dense derivative tensors while still computing the pointwise derivatives required by the PDE residual.

The same construction applies componentwise in multiple spatial dimensions, with mixed derivatives obtained by multiplying Fourier coefficients by the corresponding products of wavenumbers. Thus, the PDE residual is evaluated by combining FFT-based feature derivatives with chain-rule derivatives through the decoder. This preserves the bandlimited structure of each pyramid level while allowing the nonlinear decoder to mix features across scales.

4.4 Stochastic grid shift

Fixed collocation grids create a potential issue with aliasing: the network can learn high-frequency functions that satisfy the PDE residual at grid points while remaining inaccurate between them. This is particularly relevant for Fourier feature grids, which make high-frequency feature fields directly available to the MLP. To reduce this effect, we randomly offset the resolution M structured residual grid during training by a random shift $\boldsymbol{\delta} \sim \mathrm{Uniform}([0, 1/M]^{d_{\mathrm{in}}})$ (assuming for simplicity that the inputs are in $[0,1]^{d_{\mathrm{in}}}$). The shift is implemented efficiently as a phase ramp in Fourier space, multiplying the component of the wavenumber $\mathbf{k}$ by $\exp(2\pi i\mathbf{k}\cdot\boldsymbol{\delta})$. This preserves the structured FFT evaluation while training on off-grid spatial locations; we ablate this choice in Appendix D.1.

5 Experiments

5.1 Time-dependent PDE benchmarks

We evaluate `beignet` on a benchmark containing four time-dependent PDEs from the JAX-PI suite (Wang et al., 2023a): Allen–Cahn, Korteweg–de Vries (KdV), complex Ginzburg–Landau,

Table 1: Time-dependent PDE comparison. We report relative L_2 error on the primary field and, for two-component systems, on the secondary field, as mean $\pm$ standard deviation over three random seeds (42, 43, 44). For Ginzburg–Landau and Gray–Scott, the table reports the per-window parameter count with the number of windows. JAX-PI and PirateNet results are reproduced from the JAX-PI codebase under the same evaluation protocol.

Problem	Method	L_2 error	L_2 error (sec.)	Params
Allen–Cahn	JAX-PI (Wang et al., 2023a)	$(6.98 \pm 5.39) \times 10^{-5}$	—	396,930
	PirateNet (Wang et al., 2024)	$(1.21 \pm 0.58) \times 10^{-5}$	—	1,916,041
	`beignet` (ours)	$\mathbf{(1.88 \pm 0.22) \times 10^{-6}}$	—	411,938
KdV	JAX-PI (Wang et al., 2023a)	$(4.06 \pm 1.05) \times 10^{-4}$	—	396,930
	PirateNet (Wang et al., 2024)	$(2.59 \pm 0.15) \times 10^{-4}$	—	1,916,041
	`beignet` (ours)	$\mathbf{(3.71 \pm 0.63) \times 10^{-5}}$	—	412,674
Ginzburg–Landau	JAX-PI (Wang et al., 2023a)	$(2.10 \pm 0.44) \times 10^{-2}$	$(2.04 \pm 0.96) \times 10^{-2}$	$331{,}396 \times 5$
	PirateNet (Wang et al., 2024)	$(1.15 \pm 0.43) \times 10^{-2}$	$(9.41 \pm 4.97) \times 10^{-3}$	$1{,}916{,}553 \times 5$
	`beignet` (ours)	$\mathbf{(1.08 \pm 0.16) \times 10^{-3}}$	$\mathbf{(1.31 \pm 0.55) \times 10^{-3}}$	$855{,}936 \times 5$
Gray–Scott	JAX-PI (Wang et al., 2023a)	$(7.28 \pm 6.98) \times 10^{1}$	$(8.08 \pm 12.95) \times 10^{2}$	$331{,}396 \times 10$
	PirateNet (Wang et al., 2024)	$(3.10 \pm 0.14) \times 10^{-3}$	$(8.00 \pm 0.41) \times 10^{-3}$	$1{,}916{,}553 \times 10$
	`beignet` (ours)	$\mathbf{(1.84 \pm 1.52) \times 10^{-4}}$	$\mathbf{(4.80 \pm 3.95) \times 10^{-4}}$	$855{,}936 \times 10$

and Gray–Scott. In all benchmark experiments, we replace the random Fourier feature embedding in JAX-PI with the `beignet` Fourier feature pyramid while keeping the ModifiedMLP backbone architecture (with width reduced to 128 for the two 2D systems); exact configurations are provided in the supplement. These problems span a wide range of behaviors and different dimensionalities of the domain and range of the function u being approximated, including mapping from 1-dimensional spatial coordinates and time to a scalar field, mapping from 2-dimensional spatial coordinates and time to a vector field, reaction–diffusion equations, and dispersive wave equations. While all four equations are periodic, which makes them natural for FFT-based feature derivatives, periodicity is not essential to our approach; we show results on non-periodic Dirichlet problems with irregular geometry in Appendix B.

The benchmarks test different aspects of the representation. Allen–Cahn requires resolving sharp transition layers generated by nonlinear reaction–diffusion dynamics. KdV contains a third-order dispersive derivative, which makes broad high-frequency spatial representations difficult to train. Complex Ginzburg–Landau and Gray–Scott extend the evaluation to coupled two-field 2D systems with fine-scale pattern formation. We generally match the training strategy of the JAX-PI benchmark suite. Benchmark details and `beignet` configurations are given in Appendix H, and solution visualizations in Appendix A.

As shown in Table 1, `beignet` finds significantly higher accuracy solutions than both JAX-PI and PirateNet across all four benchmarks. The improvement is largest on Ginzburg–Landau and Gray–Scott, where the solution dynamics place strong demands on the spatial representation. On Ginzburg–Landau, `beignet` reduces error by roughly an order of magnitude relative to PirateNet. On Gray–Scott, `beignet` reduces the error by more than an order of magnitude on both fields while using less than half the per-window parameter count of PirateNet. These results support the central premise of the method: allocating spectral capacity in the feature representation can be a more parameter-efficient scaling axis than increasing only the MLP backbone.

5.2 Self-similar blowup in 1D Burgers

A promising application of high-accuracy PINNs is the computation of self-similar singularity profiles. In singularity formation problems, the quantity of interest may become unbounded in finite time, making direct time-dependent simulation increasingly ill-conditioned near blowup. A common strategy is instead to introduce self-similar variables, which transform the search for a finite-time singularity into the search for a stationary profile (Hou & Liu, 2015). Accurate numerical profiles can then serve as candidates for subsequent stability analysis or computer-assisted proof. This direction has recently received renewed attention in fluid singularity problems, beginning with neural network computations of asymptotic self-similar profiles for the 3D axisymmetric Euler equations (Wang et al., 2023b), and continuing in 1D model equations, Boussinesq-type systems, and related fluid settings

Table 2: Self-similar 1D Burgers profile with fixed $\lambda = 0.5$. This controlled singularity-profile problem tests whether the Fourier feature pyramid can move first-order PINN training into the high-precision regime. Using only Adam, `beignet` reaches near-machine precision residual accuracy, while the Adam-trained coordinate MLP baseline remains many orders of magnitude less accurate. This suggests that representation design can reduce reliance on expensive second-order refinement for high-precision profile computations. Memory scaling refers to optimizer state as model size grows; Adam is linear in parameters, while full BFGS stores a dense inverse-Hessian approximation and is quadratic.

Method	Optimizer	Memory	PDE MSE	$\log_{10} \max \lvert F_U \rvert$
MLP (Wang et al., 2026)	Adam	$\mathcal{O}(P)$	6.21×10^{-7}	-2.41
MLP (Wang et al., 2026)	BFGS	$\mathcal{O}(P^2)$	7.15×10^{-20}	-9.10
`beignet` pyramid (12×4)	Adam	$\mathcal{O}(P)$	6.29×10^{-19}	-8.46

(Wang et al., 2026; 2025b). Developing methods capable of accurately resolving such singular structures is an important step toward the broader challenge of understanding singularity formation in fluid dynamics, including the questions underlying the unforced Navier–Stokes Millennium Prize Problem (Fefferman, 2006).

This setting is a demanding test of optimization accuracy. Recent high-precision PINN approaches (Wang et al., 2026; 2025b) can identify self-similar profiles, but rely on expensive second-order or Gauss–Newton-type refinement to reach residual levels suitable for downstream analysis. This may limit scalability in higher dimensions, where richer representations are needed and curvature-based methods can scale quadratically or worse with parameter count. Here, we ask whether the Fourier feature pyramid can reach comparable precision using first-order optimization alone.

Following Wang et al. (2026), we use the 1D inviscid Burgers equation as a model problem. Although simple, it retains the key ingredients needed for our investigation: a self-similar blowup ansatz, an unbounded profile domain, and a residual minimization problem whose attainable precision is highly sensitive to the choice of neural representation. We consider the 1D Burgers equation:

$$u_t + u u_x = 0, \tag{10}$$

and seek self-similar solutions of the form:

$$u(x,t) = \tau^{\lambda} U(y), \qquad y = x\tau^{-(1+\lambda)}, \qquad \tau = T - t, \tag{11}$$

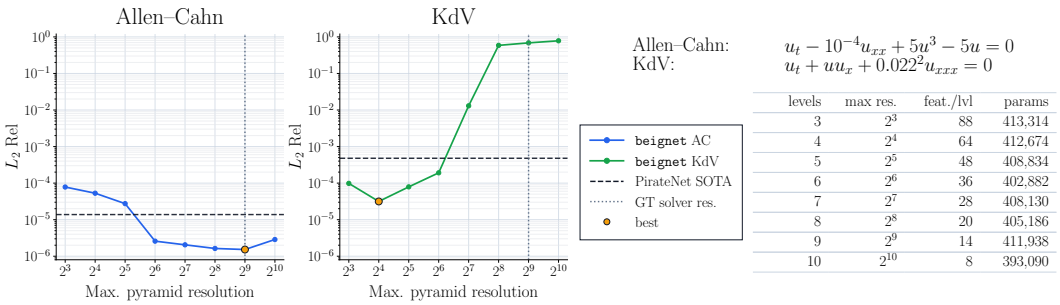
where λ denotes the similarity exponent, T is the blowup time, and $U(y)$ is a stationary profile that describes the rescaled solution shape near the singularity. Rather than learning the full time-dependent solution $u(x,t)$, the PINN is trained to represent the profile function $U(y)$. Substituting this ansatz into the PDE in Equation 10 gives the stationary profile equation:

$$-\lambda U + \big((1+\lambda)y + U\big)U_y = 0. \tag{12}$$

In the experiments below we fix $\lambda = 0.5$ and minimize the squared profile residual together with the derivative residual used by Wang et al. (2026). Implementation details for the profile representation, domain mapping, and residual terms are provided in the supplement.

Table 2 compares the residual accuracy obtained by a coordinate MLP and by the Fourier feature pyramid on the fixed diagnostic grid. With Adam alone, the coordinate MLP stalls at a PDE MSE of 6.21×10^{-7}, whereas the 12-level `beignet` pyramid reaches 6.29×10^{-19} after the same 20K-step schedule. The maximum pointwise profile residual shows a comparable gap, improving from $\log_{10} \max |F_U| = -2.41$ for the coordinate MLP to -8.46 for `beignet`. The learned profiles and pointwise residuals are shown in Figure 4.

This experiment complements the time-dependent PDE benchmarks above. There, `beignet` improves accuracy and parameter efficiency on standard PINN tasks; here, in a low-dimensional singularity-profile problem, the same representation substantially changes the residual floor reachable by first-order optimizers like Adam. This makes Fourier feature pyramids a promising direction for higher-dimensional self-similar profile computations, where second-order refinement with second-order optimizers like BFGS can become substantially more expensive.



levels	max res.	feat./lvl	params
3	2^3	88	413,314
4	2^4	64	412,674
5	2^5	48	408,834
6	2^6	36	402,882
7	2^7	28	408,130
8	2^8	20	405,186
9	2^9	14	411,938
10	2^{10}	8	393,090

Figure 2: Feature-pyramid size sweep for Allen–Cahn and KdV. Each point is a parameter-matched `beignet` model in which the number of pyramid levels sets the maximum represented grid resolution and the number of features per level is adjusted to keep the total parameter count approximately fixed. Dashed horizontal lines show PirateNet's error (single run) relative to the numerical reference solution, and the vertical dotted line indicates the spatial resolution of the numerical solver used to generate that reference solution. Allen–Cahn benefits from increasing pyramid bandwidth up to the resolution of the numerical solver, whereas KdV is most accurate with a lower-bandwidth pyramid.

5.3 PDE-DEPENDENT SPECTRAL ALLOCATION

The central design choice in `beignet` is not only the overall support of the Fourier features, but how capacity is distributed across resolutions. Unlike random Fourier features, whose frequency coverage is randomly drawn from a single-parameter distribution, `beignet` allocates features across all the chosen grid resolutions. This exposes two practical choices: the maximum resolution of the finest grid, which sets the representation's frequency support, and the allocation of feature channels across levels, which sets the decoder budget within that frequency support. We choose these quantities with a small parameter-matched sweep. In Figure 2, all models have roughly 4×10^5 parameters; moving right increases the maximum grid resolution while reducing the number of features per level.

Interestingly, the two one-dimensional benchmarks favor different allocations. Allen–Cahn continues to improve as the pyramid adds finer levels, reaching its best error near the 2^9 reference-grid resolution. This is expected because its solutions can contain sharp interfaces, so fine features help represent real structure in the solution; at the same time, the diffusive term damps small high-frequency errors. KdV behaves differently. It contains a third spatial derivative, which weights every Fourier mode by a factor proportional to its frequency cubed. This makes the weight of high-frequency components in the residual extremely high, which can make optimization harder, a standard source of stiffness in KdV time-stepping methods (Klein, 2008). Thus, adding very fine feature levels can hurt even when the parameter count is held fixed. The best KdV model is the shallow 4-level, 64-feature pyramid, which keeps more capacity in low and middle frequencies rather than allocating it to very fine scales. This sweep motivates treating the pyramid bandwidth as a PDE-dependent design parameter rather than a monotone scaling knob.

6 CONCLUSION

We have introduced `beignet`, a PINN architecture that replaces fixed Gaussian random Fourier features with a deterministic multi-resolution Fourier feature pyramid. The central idea is to treat the feature embedding as a spectral allocation: the pyramid controls both the spatial bandwidth of the representation and the redundancy assigned to each frequency band. This provides a simple way to adapt Fourier feature capacity to the PDE structure, retaining broad coverage when high frequencies are useful while avoiding excessive capacity in bands that cause training instability.

On four time-dependent JAX-PI benchmarks, `beignet` improves over the JAX-PI and PirateNet baselines. On a self-similar PDE blowup problem, our representation enables Adam to drive the PDE residual near machine precision, multiple orders of magnitude lower than what is achievable with Adam-trained versions of the standard coordinate-based MLPs used for this problem setting. These results suggest that carefully designing the feature representation itself is a key ingredient for pushing PINNs into new high-accuracy regimes, opening the door to neural PDE solvers capable of resolving fine-scale structure, extreme multiscale dynamics, and challenging phenomena such as singularities.

AI USE STATEMENT

Generative AI tools were used to assist with implementing the method, running experiments, and drafting paper text. Generative AI was not used for any other task requiring disclosure under the ICLR 2027 policy. The authors manually verified all AI-generated code and results and substantially revised all AI-drafted text. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Our code is available at `https://github.com/beignet-paper/beignet-release/`. It includes the `beignet` implementation and scripts for all experiments in this paper. Per-problem `beignet` configurations are given in Appendix H, the JAX-PI benchmark setup and evaluation protocol in Appendix I, and the self-similar Burgers blowup setup in Appendix J.

REFERENCES

Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. *ICML*, 2020.

Peter J Burt and Edward H Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 1983.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. In *IJCAI*, 2021.

Madison Cooley, Varun Shankar, Robert M Kirby, and Shandian Zhe. Fourier PINNs: From strong boundary conditions to adaptive Fourier bases. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=KqRnsEMYLx`.

Charles L Fefferman. Existence and smoothness of the Navier-Stokes equation. *The Millennium Prize Problems*, 2006.

Adarsh Ganeshram, Haydn Maust, Valentin Duruisseaux, Zongyi Li, Yixuan Wang, Daniel Leibovici, Oscar Bruno, Thomas Hou, and Anima Anandkumar. FC-PINO: High precision physics-informed neural operators via Fourier continuation. *arXiv:2211.15960*, 2022.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, and Jun Zhu. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 76721–76774, 2024.

Thomas Y. Hou and Pengfei Liu. Self-similar singularity of a 1D model for the 3D axisymmetric Euler equations. *Research in the Mathematical Sciences*, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Christian Klein. Fourth order time-stepping for low dispersion Korteweg-de Vries and nonlinear Schrödinger equation. *Electronic Transactions on Numerical Analysis*, 2008.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *JMLR*, 2023.

Jin Lee, Ziming Liu, Xinling Yu, Yixuan Wang, Haewon Jeong, Murphy Yuezhen Niu, and Zheng Zhang. KANO: Kolmogorov-Arnold neural operator. *arXiv:2509.16825*, 2025.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 2024. doi: 10.1145/3648506.

David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: Band-limited coordinate networks for multiscale scene representation. *CVPR*, 2022.

Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. KAN 2.0: Kolmogorov-Arnold networks meet science. *arXiv:2408.10205*, 2024.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. KAN: Kolmogorov-Arnold networks. *ICLR*, 2025.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *ICML*, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *NeurIPS*, 2007.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *NeurIPS*, 2008.

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

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. WIRE: Wavelet implicit neural representations. *CVPR*, 2023.

Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. *Computer Methods in Applied Mechanics and Engineering*, 2024.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020.

Matthew Tancik, Pratul 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. *NeurIPS*, 2020.

Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From PINNs to PIKANs: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 2025.

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*, 2021.

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

Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. PirateNets: Physics-informed deep learning with residual adaptive networks. *JMLR*, 2024.

Yixuan Wang, Jonathan W Siegel, Ziming Liu, and Thomas Y Hou. On the expressiveness and spectral bias of KANs. In *ICLR*, 2025a.

Yixuan Wang, Ziming Liu, Zongyi Li, Anima Anandkumar, and Thomas Y. Hou. High precision PINNs in unbounded domains: application to singularity formulation in PDEs. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=g34kZZAbT8.

Yongji Wang, Ching-Yao Lai, Javier Gómez-Serrano, and Tristan Buckmaster. Asymptotic self-similar blow-up profile for three-dimensional axisymmetric Euler equations using neural networks. *Physical Review Letters*, 2023b.

Yongji Wang, Mehdi Bennani, James Martens, Sébastien Racanière, Sam Blackwell, Alex Matthews, Stanislav Nikolov, Gonzalo Cao-Labora, Daniel S Park, Martin Arjovsky, et al. Discovery of unstable singularities. *arXiv:2509.14185*, 2025b.

Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *International Conference on Neural Information Processing*, 2019.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 2020.

Tianchi Yu, Yiming Qi, Ivan Oseledets, and Shiyi Chen. Spectral informed neural networks. *Journal of Computational and Applied Mathematics*, 2025.

A Solution Visualizations

We visualize the `beignet` solutions for Allen–Cahn and KdV in Figure 3, Ginzburg–Landau in Figure 6, and Gray–Scott in Figure 7. Figure 4 shows the self-similar Burgers profile and its residual, and Figure 5 shows the irregular-geometry Poisson problems of Appendix B. Absolute errors are plotted on a logarithmic color scale spanning three decades below the largest error shown.

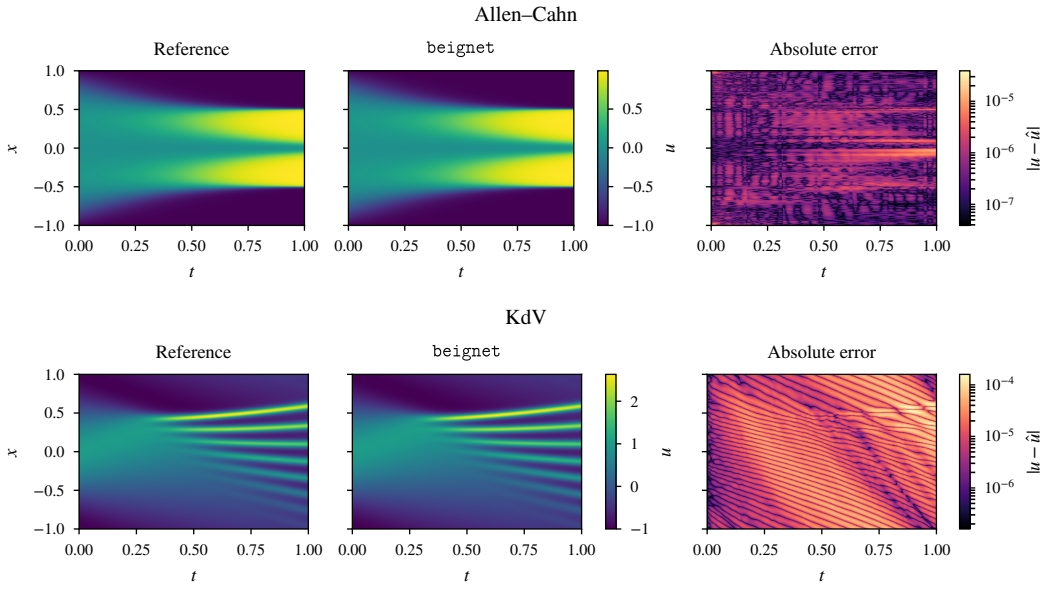


Figure 3: Allen–Cahn (top) and KdV (bottom) solutions over (t, x): reference solution, `beignet` prediction, and absolute error (log scale). The shown runs have relative L_2 errors of 1.62×10^{-6} (Allen–Cahn) and 3.40×10^{-5} (KdV).

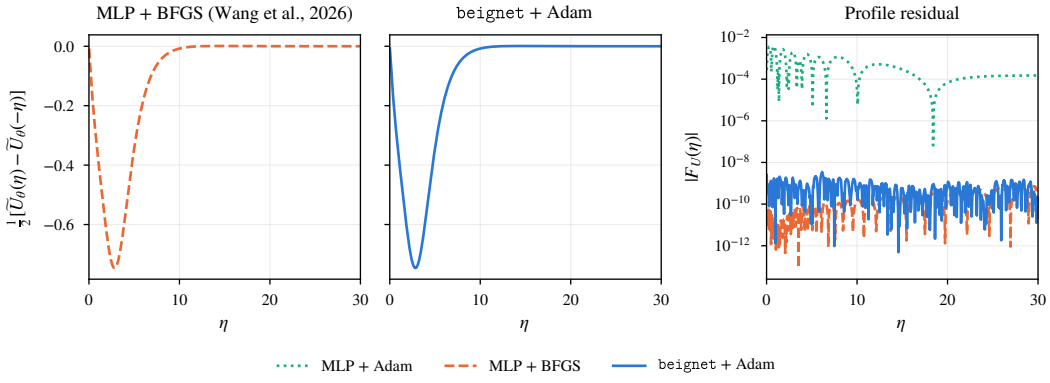


Figure 4: Self-similar Burgers profile with $\lambda = 0.5$ on the diagnostic grid of Appendix J. Left and middle: the learned odd component of Eq. (20) for the coordinate MLP trained with BFGS (Wang et al., 2026) and for `beignet` trained with Adam. Right: pointwise profile residual $|F_U|$ from Eq. (19) for all three runs of Table 2, including the coordinate MLP trained with Adam.

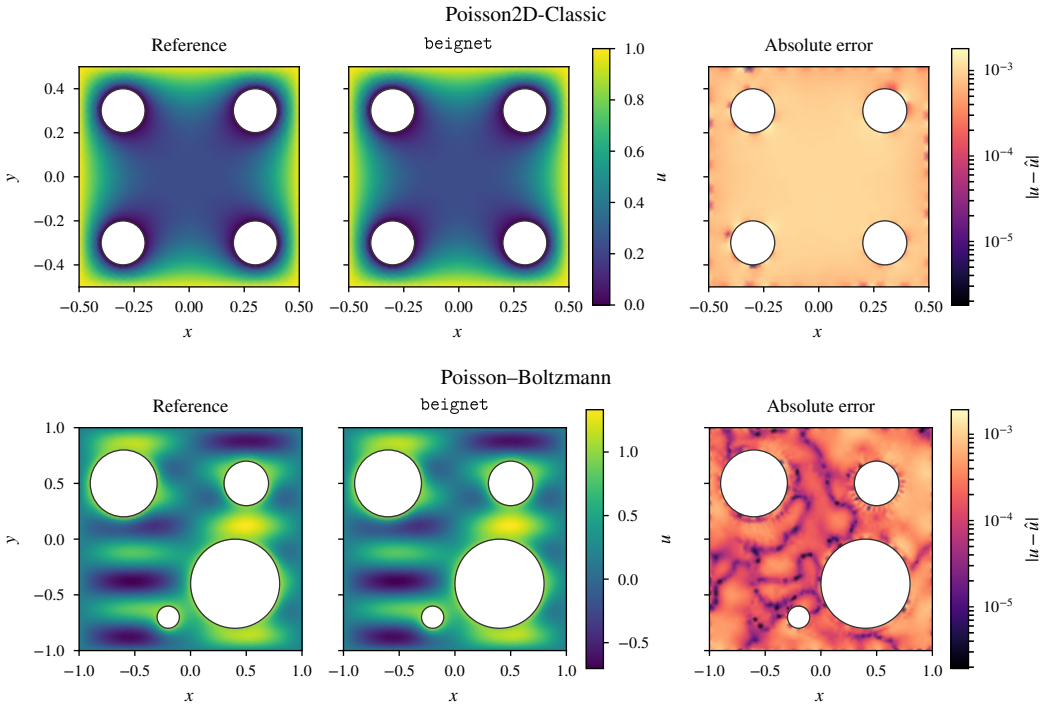


Figure 5: Poisson2D-Classic (top) and Poisson–Boltzmann (bottom) from the PINNacle benchmark (Hao et al., 2024) on their irregular domains: COMSOL reference solution, `beignet` prediction, and absolute error (log scale), shown on the COMSOL mesh nodes with linear interpolation between nodes. The displayed runs have relative L_2 errors of 1.81×10^{-3} (Poisson2D-Classic) and 6.25×10^{-4} (Poisson–Boltzmann).

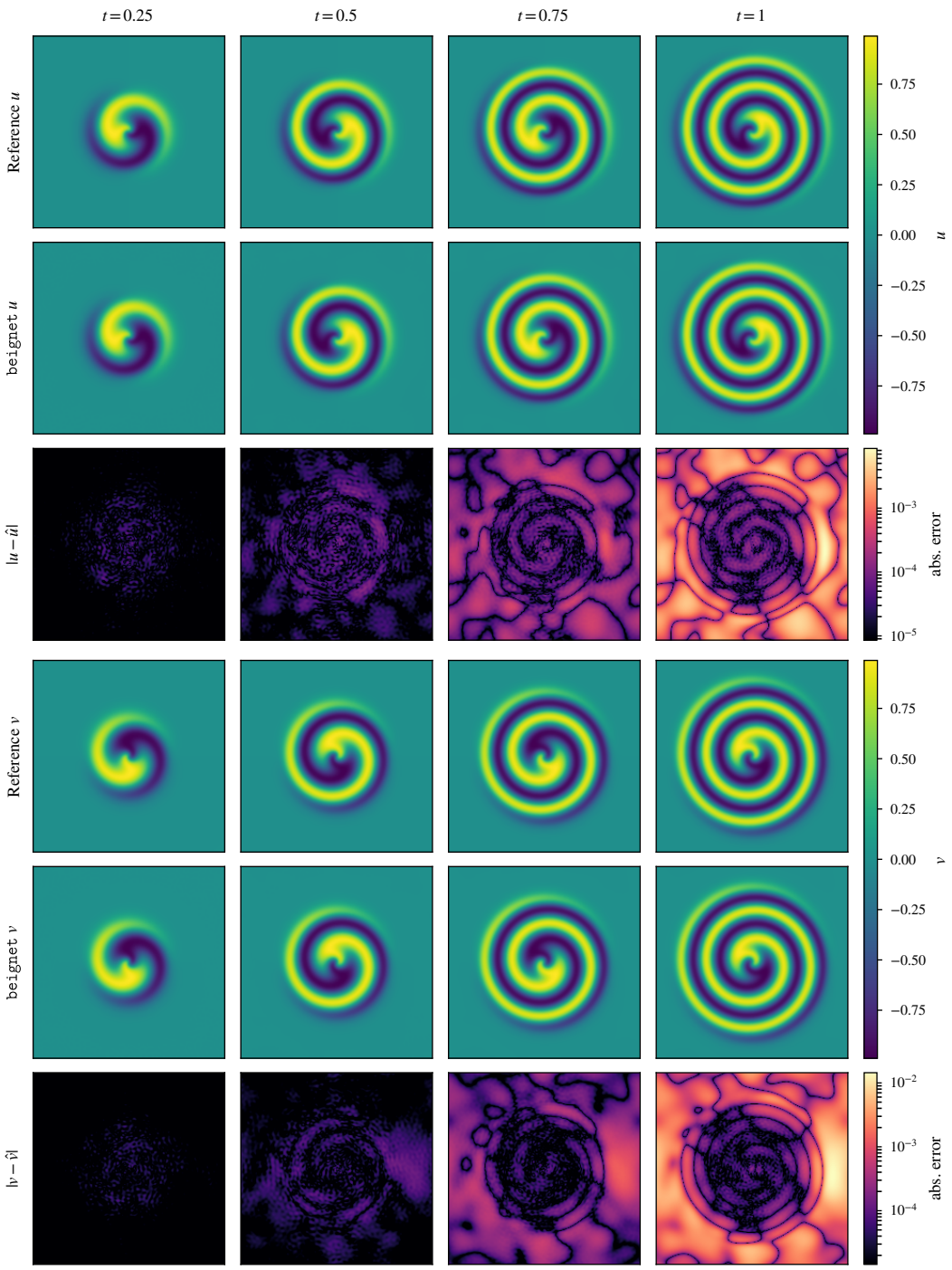


Figure 6: Complex Ginzburg–Landau snapshots on $[-1,1]^2$ at $t \in \{0.25, 0.5, 0.75, 1\}$. Rows show the reference solution, the `beignet` prediction, and the absolute error (log scale) for u (top three rows) and v (bottom three rows). The displayed run has full-trajectory relative L_2 errors of 1.04×10^{-3} (u) and 1.94×10^{-3} (v).

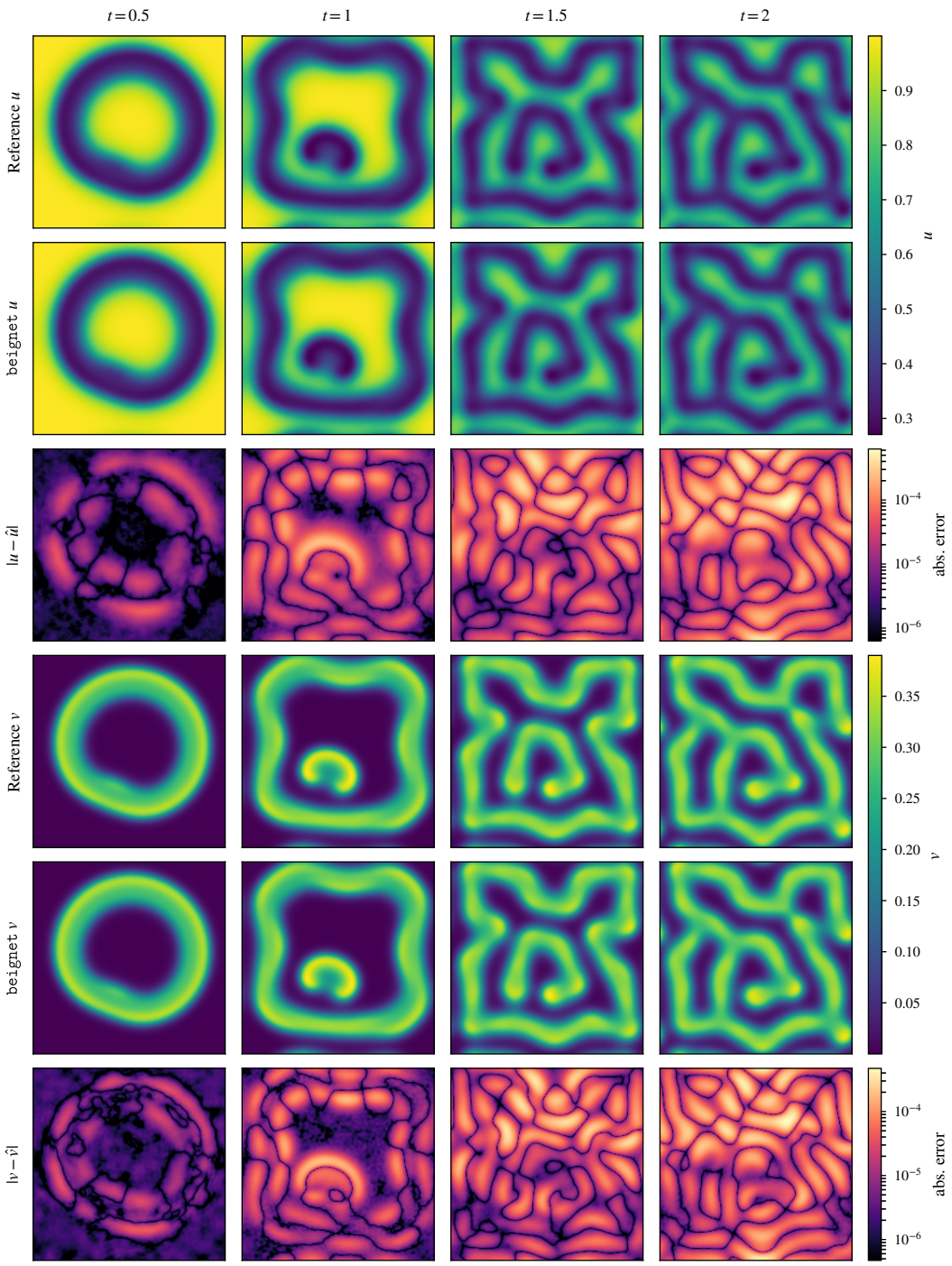


Figure 7: Gray–Scott snapshots on $[-1, 1]^2$ at $t \in \{0.5, 1, 1.5, 2\}$. Rows show the reference solution, the `beignet` prediction, and the absolute error (log scale) for u (top three rows) and v (bottom three rows). The displayed run has full-trajectory relative L_2 errors of 9.95×10^{-5} (u) and 2.60×10^{-4} (v).



Figure 8: Direct 256×256 RGB image fitting. The `beignet` representation fits the image nearly exactly under the same 2000-step optimization budget, while using a smaller decoder and fewer trainable parameters than the number of stored RGB values in the target image.

B Non-periodic domains with irregular geometry

We test `beignet` on two Dirichlet problems with irregular geometry from PINNacle (Hao et al., 2024): Poisson2D-Classic ($-\nabla^2 u = 0$) and Poisson–Boltzmann ($-\nabla^2 u + 64u = f$), each posed on a square with four circular holes and Dirichlet data on all eight boundary components. We embed the domain in a periodic box twice as wide in each direction, enforce the PDE residual only at stochastically shifted lattice points inside the domain, and impose the boundary conditions with penalty terms. We compare against an MLP and an MLP with Gaussian random Fourier features ($\sigma = 1$), both taking raw coordinates and parameter-matched to `beignet` (7 levels up to 128^2). All models share the same training settings and are evaluated by relative L_2 error on the COMSOL reference nodes provided with PINNacle. `beignet` attains the lowest error on both problems (Table 3), $2.4\times$ and $5.7\times$ lower than the best baseline, respectively; the solutions are visualized in Figure 5.

Table 3: Dirichlet problems with irregular geometry from PINNacle: relative L_2 error, mean $\pm$ standard deviation over three random seeds (42, 43, 44).

Problem	Method	Relative L_2 error	Params
Poisson2D-Classic	MLP	$(4.04 \pm 1.15) \times 10^{-3}$	85,261
	RFF + MLP	$(3.70 \pm 2.62) \times 10^{-3}$	85,667
	`beignet` (ours)	$\mathbf{(1.56 \pm 0.26) \times 10^{-3}}$	85,689
Poisson–Boltzmann	MLP	$(6.68 \pm 2.58) \times 10^{-3}$	131,041
	RFF + MLP	$(4.12 \pm 0.65) \times 10^{-3}$	131,023
	`beignet` (ours)	$\mathbf{(7.22 \pm 3.99) \times 10^{-4}}$	130,777

C Image Fitting

Although `beignet` is designed for PDE residual training, the underlying Fourier feature pyramid is a general coordinate-regression representation. To isolate this representation from the PDE-specific residual and derivative machinery, we also evaluate it on direct RGB image fitting, following the image regression setting used by Tancik et al. (2020) to study Fourier feature embeddings.

We fit a starfish image by training a coordinate network $(x, y) \mapsto (r, g, b)$ with a full-grid RGB mean-squared-error objective on a 256×256 starfish image. All models are trained for 2000 Adam steps with learning rate 10^{-3}. We compare a vanilla coordinate MLP, a Gaussian random Fourier feature model with $\sigma = 10$, and our learned Fourier feature pyramid. The vanilla MLP and RFF baselines use a larger 256-wide, 4-layer decoder, while `beignet` uses a smaller 64-wide, 3-layer decoder after its learned multi-resolution feature pyramid. Despite this smaller decoder, `beignet` achieves near-perfect reconstruction and uses fewer trainable parameters than the number of RGB values in the image itself.

Table 4: Starfish image-fitting parameter count and PSNR at resolution 256×256. The ground-truth row counts the number of stored RGB image values, $256 \times 256 \times 3$.

Method	Params	PSNR (dB)
Vanilla MLP	133,123	19.61
RFF, $\sigma = 10$	264,195	36.73
`beignet`	180,203	58.39
Ground truth image	196,608	—

D Ablation Studies

We include three ablations to isolate the contributions of stochastic grid shifts, the coarse-to-fine pyramid structure, and temporal pyramid blending. Unless otherwise stated, all ablations are trained with the same physics-informed objective used in the main experiments, including the PDE residual loss and the corresponding initial and boundary condition terms. The reported PDE residual is a training loss quantity evaluated on collocation points, whereas the relative L_2 error is a test loss measured against a reference numerical solution on the evaluation grid. This distinction is important for interpreting the ablations: a small residual on the training collocation grid does not necessarily imply small solution error between those points.

D.1 Stochastic Grid-Shift prevents fixed-grid overfitting

Table 5 reports experimental results with and without stochastic shifting. The effect depends on the pyramid shape: Allen–Cahn uses a wide pyramid reaching spatial resolution 512, so the unshifted grid can overfit fixed collocation locations while generalizing poorly between them (low PDE residual but high relative L_2 error). KdV uses a much shallower pyramid, with capacity concentrated in low and mid spatial bands, and is therefore less susceptible to this aliasing failure mode.

Table 5: Stochastic grid-shift ablation. For Allen–Cahn, removing the shifts produces a classic collocation overfitting failure: the model attains a small residual on the fixed training grid but fails between grid points. KdV is less affected because the selected pyramid has lower spatial bandwidth.

Problem	Setting	Relative L_2 error	PDE residual loss
Allen–Cahn	shifted	1.52×10^{-6}	1.81×10^{-9}
Allen–Cahn	no shift	9.50×10^{-1}	2.12×10^{-10}
KdV	shifted	3.15×10^{-5}	1.12×10^{-7}
KdV	no shift	3.73×10^{-5}	9.65×10^{-8}

D.2 Coarse-to-fine pyramid structure

To isolate the effect of the pyramid structure from the effect of high maximum resolution, we compare the Allen–Cahn pyramid used in the size sweep (see Figure 2) with a parameter-matched ablation in which every level is fixed at the finest spatial resolution, 512.

Table 6: Allen–Cahn pyramid versus a parameter-matched all-full-resolution ablation. Both models have comparable parameter counts and use the same finest spatial resolution, but distributing capacity across resolutions gives substantially lower error and residual than placing all feature grids at resolution 512.

Representation	Params	Relative L_2 error	PDE residual loss
Pyramid, 14 features/level	411,938	1.52×10^{-6}	1.81×10^{-9}
All levels at 512, 5 features/level	419,586	1.32×10^{-4}	2.60×10^{-8}

The all-full-resolution ablation is $86.8\times$ worse in relative L_2 error and has $14.4\times$ higher residual loss, despite a similar parameter count. This shows that the improvement is not simply due to exposing

high-resolution Fourier features. The coarse-to-fine structure is a central part of the embedding: it allocates capacity across spatial bands and optimizes substantially better than stacking parameter-matched feature grids at the finest resolution alone.

D.3 Temporal pyramid blending

We ablate the temporal blending described in Section 4.2 using the Allen–Cahn `beignet` model. In the ablated model, each pyramid level is collapsed to a single learned temporal slice. The raw time coordinate is still passed to the decoder, so the ablation removes only the learned temporal anchors and their linear blending.

Table 7: Allen–Cahn ablation with and without temporal pyramid blending. Both runs use the same spatial pyramid, decoder, and optimizer. Temporal blending improves both the final relative L_2 error and the PDE residual.

Setting	Temporal slices	Relative L_2 error	PDE residual loss
Without temporal blending	1	4.28×10^{-6}	3.38×10^{-9}
With temporal blending	8	1.52×10^{-6}	1.81×10^{-9}

Removing temporal blending increases the relative L_2 error by $2.81\times$ and the PDE residual by $1.86\times$. Thus, even though the decoder still receives the raw time coordinate, the learned temporal anchors provide a useful time-dependent feature representation for this Allen–Cahn configuration.

E Timing and efficiency comparison on Allen–Cahn

We compare the computational cost of Allen–Cahn residual training for three settings: the FFT-structured `beignet` residual path, a pointwise DFT/JVP evaluation of the same `beignet` model, and the PirateNet baseline (Wang et al., 2024). The goal of this experiment is to isolate per-step residual-evaluation throughput from convergence behavior. We therefore time raw optimizer updates after compilation at matched residual NFEs. Timings exclude logging, evaluator calls, checkpointing, sampler generation during the timed region, and periodic weight-recalculation steps. All timing tests were performed on a single NVIDIA A100 80GB GPU.

Table 8: Raw Allen–Cahn training-step timing at matched residual NFEs. Values are seconds per 100 optimizer steps, reported as mean $\pm$ standard deviation over five timed trials. The FFT-structured `beignet` residual path is fastest at every tested NFE.

Residual NFE	**`beignet` (FFT)**	`beignet` (DFT)	PirateNet (Wang et al., 2024)
8,192	$\mathbf{1.877 \pm 0.006}$	7.601 ± 0.012	12.749 ± 0.023
16,384	$\mathbf{2.996 \pm 0.016}$	14.582 ± 0.122	18.768 ± 0.051
32,768	$\mathbf{4.436 \pm 0.044}$	30.097 ± 0.096	31.388 ± 0.209
65,536	$\mathbf{6.797 \pm 0.128}$	55.561 ± 0.427	50.390 ± 0.187

The FFT path is substantially faster because spatial feature derivatives are computed spectrally on structured grids and then reused across collocation points. In contrast, the DFT/JVP variant and PirateNet baseline evaluate the residual more pointwise, so their cost grows more directly with the number of residual evaluations. At 65,536 residual NFEs, the FFT-structured path is $8.2\times$ faster than the DFT/JVP evaluation of the same `beignet` model and $7.4\times$ faster than PirateNet in raw training-step time.

Table 9 reports the corresponding accuracy comparison. We include the lowest-NFE PirateNet baseline and `beignet` runs obtained by reducing the structured residual grid size. This separates the efficiency claim from the accuracy claim: even at the matched 8,192 residual NFE setting, `beignet` achieves lower final relative L_2 error than PirateNet, while the larger structured-grid runs maintain similar accuracy at higher NFEs.

These results show two complementary benefits of the structured residual path. First, at a fixed residual NFE, FFT-based derivative evaluation gives much higher training-step throughput than

Table 9: Allen–Cahn final relative L_2 error for the lowest-NFE PirateNet baseline and FFT-structured `beignet` runs at different residual NFEs. The matched 8,192-NFE comparison shows that the accuracy gain is not only due to using more residual points.

Method	M_x	Residual NFE	Relative L_2 error
PirateNet (Wang et al., 2024)	—	8,192	1.38×10^{-5}
`beignet` (FFT)	32	8,192	$\mathbf{1.91 \times 10^{-6}}$
`beignet` (FFT)	64	16,384	$\mathbf{1.32 \times 10^{-6}}$
`beignet` (FFT)	128	32,768	$\mathbf{1.78 \times 10^{-6}}$
`beignet` (FFT)	256	65,536	$\mathbf{1.52 \times 10^{-6}}$

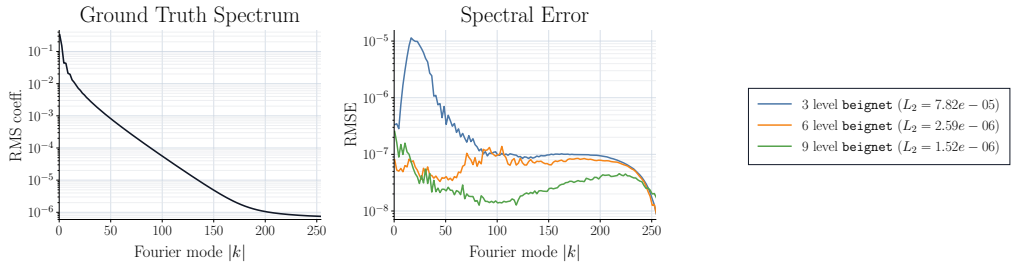


Figure 9: Frequency-resolved Allen–Cahn diagnostic. Left: root mean squared spatial Fourier coefficient magnitude of the ground-truth solution, averaged over time. Right: root mean squared spatial Fourier coefficient magnitude of the prediction error for 3-, 6-, and 9-level `beignet` models. The duplicated periodic endpoint in the reference grid is removed before applying the FFT, and adjacent Fourier modes are RMS-binned in pairs.

pointwise residual evaluation. Second, the memory efficiency of the structured path makes it practical to train with many more collocation points: the 65,536-NFE `beignet` run fits on a single NVIDIA A100 80GB GPU, whereas the PirateNet baseline requires two GPUs for its 8,192-NFE configuration with gradient-norm loss reweighting. Thus the structured `beignet` residual path improves both the matched-NFE comparison and the feasible high-NFE training regime.

F Spectral Error Analysis

Fig. 9 shows a frequency-resolved diagnostic for the Allen–Cahn size sweep. The left panel reports the spectrum of the reference solution, and the right panel reports the spectrum of the prediction error for 3-, 6-, and 9-level `beignet` models. For each selected checkpoint, we evaluate the learned solution on the Allen–Cahn reference grid and compute the spatial Fourier transform of the error field

$$e(t, x) = u_{\boldsymbol{\theta}}(t, x) - u_{\mathrm{ref}}(t, x).$$

The plotted value at mode $|k|$ is the root mean squared Fourier coefficient magnitude over time, binned over neighboring pairs of Fourier modes. The comparison isolates how increasing the Fourier feature pyramid bandwidth changes the error distribution across spatial modes. The 3-level model has limited spatial bandwidth and leaves substantially larger error in the low modes. Increasing to 6 levels reduces the error by roughly an order of magnitude across most of the spectrum, while the 9-level model further lowers the error, especially in the higher-frequency tail. This supports the observation in Fig. 2 that Allen–Cahn benefits from a wider pyramid: its sharp interface dynamics require resolving a broad range of spatial harmonics, and increasing the maximum represented feature resolution reduces the spectral error rather than only improving low-frequency fit.

G KdV is stiff at high frequencies

The KdV size sweep in Fig. 2 favors a compact pyramid rather than the widest representation. This is consistent with the structure of the KdV residual, whose spatial part contains a third derivative:

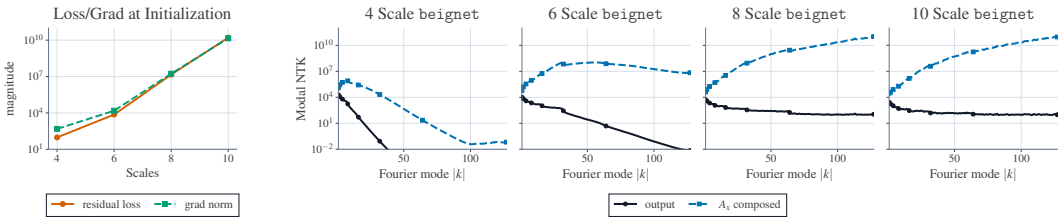


Figure 10: Initialization diagnostic for KdV across 4-, 6-, 8-, and 10-scale `beignet` models. The loss/gradient panel reports initial residual loss and pyramid-gradient norm, both of which grow rapidly as finer scales are added. The modal panels compare the model-output modal NTK with the same tangent response composed with the linearized KdV spatial residual operator A_x. Increasing the number of scales exposes higher represented frequencies whose operator-composed response grows rapidly, matching the rise in initialization loss and gradient magnitude; this results in optimization instability at higher frequency levels of the pyramid.

high-frequency Fourier modes are strongly amplified by the PDE operator. Thus, for KdV, adding finer pyramid levels does not merely increase representational bandwidth; it can also introduce trainable directions that are poorly conditioned for residual minimization.

Figure 10 first compares the initial residual loss and pyramid-gradient norm across KdV pyramids with increasing numbers of scales. Both quantities grow rapidly as finer levels are added, suggesting that the wider pyramids are already poorly conditioned at initialization.

To explain this effect, we additionally propose a rough modal tangent diagnostic. For a spatial Fourier probe $q_k(x)$, such as a sine or cosine mode at wavenumber k, define $a_k(\boldsymbol{\theta}) = \langle u_{\boldsymbol{\theta}}, q_k \rangle$. The squared norm $\|\nabla_{\boldsymbol{\theta}} a_k\|_2^2$ gives the tangent energy associated with that Fourier mode. This is a finite-width empirical analogue of the NTK spectral view used to study frequency-dependent learning in coordinate networks: when the kernel is approximately stationary, Fourier modes diagonalize it, and the corresponding NTK eigenvalues control the convergence rates of individual frequency components (Tancik et al., 2020; Cao et al., 2021).

We also compute an operator-composed variant by applying the KdV spatial residual operator, linearized around the initial condition u_0,

$$A_x[v] = u_0\, v_x + (\partial_x u_0)\, v + 0.022^2\, v_{xxx},$$

before taking the parameter tangent, replacing $a_k(\boldsymbol{\theta})$ with $\langle A_x[u_{\boldsymbol{\theta}}], q_k \rangle$. This curve measures the frequency-dependent sensitivity seen by the residual objective, rather than the sensitivity of the raw model output alone.

The modal tangent curves show that the rise in initialization loss and gradient norm is not explained by the raw output sensitivity alone. The model-output modal NTK remains comparatively moderate, while the operator-composed response grows by many orders of magnitude as finer pyramid levels expose higher frequencies. This supports the interpretation that broad KdV pyramids become optimization-stiff because high-frequency degrees of freedom are strongly amplified by the residual operator. The best compact model instead allocates capacity to low- and mid-frequency structure without exposing as much stiff high-frequency response.

H `BEIGNET` CONFIGURATIONS FOR EACH PROBLEM

Table 10 summarizes the configuration settings we used for the four JAX-PI benchmark PDEs.

I JAX-PI BENCHMARK DESCRIPTIONS

The four time-dependent benchmarks follow the JAX-PI[1] (Wang et al., 2023a) repository. Training minimizes an initial-condition loss and a PDE residual loss on uniformly sampled collocation points,

[1] `https://github.com/PredictiveIntelligenceLab/jaxpi`

with causal time-chunk weighting and periodic loss-weight updates. Evaluation reports relative L_2 error on the stored solver grid. Our `beignet` runs use the same entry points and data files, replacing the coordinate embedding and residual-evaluation path (and, for the 2D systems, the MLP width); the two 2D systems are trained in time windows, passing the predicted terminal state from one window as the next initial condition.

Allen–Cahn. The Allen–Cahn equation is

$$u_t - 10^{-4}u_{xx} + 5u^3 - 5u = 0, \qquad x \in [-1, 1], \quad t \in [0, 1], \tag{13}$$

with periodic boundary conditions and initial condition $u(x, 0) = x^2 \cos(\pi x)$. This benchmark contains sharp phase-interface dynamics and benefits from broad spatial spectral coverage. The reference file is generated with MATLAB Chebfun's `spin` solver on 201 time samples and a 512-point periodic spatial grid, including the repeated endpoint.

KdV. The Korteweg–de Vries equation is

$$u_t + uu_x + 0.022^2 u_{xxx} = 0, \qquad x \in [-1, 1], \quad t \in [0, 1], \tag{14}$$

with periodic boundary conditions and initial condition $u(x, 0) = \cos(\pi x)$. Unlike Allen–Cahn, KdV contains a third-order spatial derivative, so high spatial modes are strongly amplified in the residual. The reference file is generated with MATLAB Chebfun's `spin` solver on 201 time samples and the same 512-point periodic spatial grid.

Complex Ginzburg–Landau. The complex Ginzburg–Landau equation is a two-component reaction–diffusion system for $A = u + iv$ on the periodic domain $\mathbf{x} = (x, y) \in [-1, 1]^2$. In real variables, the system is

$$u_t = \varepsilon(u_{xx} + u_{yy}) + \kappa \left[u - u(u^2 + v^2) + \frac{3}{2} v(u^2 + v^2) \right], \tag{15}$$

$$v_t = \varepsilon(v_{xx} + v_{yy}) + \kappa \left[v - v(u^2 + v^2) - \frac{3}{2} u(u^2 + v^2) \right]. \tag{16}$$

The cubic reaction terms and cross-coupling generate coherent wave and defect dynamics, providing a 2D test of the multi-resolution representation. The reference uses $\kappa = 10$, $\varepsilon = \kappa/50^2$, 101 time samples, and a 200×200 spatial grid; training is split into five time windows.

Gray–Scott. The Gray–Scott system is a two-component autocatalytic reaction–diffusion model on the periodic domain $\mathbf{x} = (x, y) \in [-1, 1]^2$:

$$u_t = \varepsilon_u(u_{xx} + u_{yy}) + b_1(1 - u) - c_1 uv^2, \tag{17}$$

$$v_t = \varepsilon_v(v_{xx} + v_{yy}) - b_2 v + c_2 uv^2. \tag{18}$$

It produces fine-scale spot and labyrinthine patterns, making it a stringent test of spatial resolution and multi-scale feature capacity. The reference uses $(\varepsilon_u, \varepsilon_v, b_1, b_2, c_1, c_2) = (0.2, 0.1, 40, 100, 1000, 1000)$, 101 time samples over $t \in [0, 2]$, and a 200×200 spatial grid; training is split into ten time windows.

J Self-similar Burgers blowup details

We follow the 1D Burgers self-similar profile formulation of Wang et al. (2026). For fixed similarity exponent $\lambda = 0.5$, the profile U satisfies

$$F_U(y) := -\lambda U(y) + \big((1 + \lambda)y + U(y)\big)U_y(y) = 0. \tag{19}$$

Since the profile is defined on an unbounded domain, we train in the computational coordinate $\eta \in [0, c]$ with $c = 30$ and map to the profile coordinate by $y = \sinh(\eta)$. Derivatives in Eq. (19) are therefore evaluated with the chain-rule factor $\partial_y = (\cosh \eta)^{-1}\partial_\eta$. We use the same analytic structure as in Wang et al. (2026) to build the known symmetry and far-field behavior into the ansatz:

$$U(y(\eta)) = \widetilde{U}_{\boldsymbol{\theta}}(\eta) - \widetilde{U}_{\boldsymbol{\theta}}(-\eta) - \left(\frac{y}{1 + y} \right)^{15} y^{\lambda/(1+\lambda)}, \qquad y = \sinh(\eta), \tag{20}$$

where $\widetilde{U}_{\boldsymbol{\theta}}$ is the learnable component. This construction enforces odd symmetry through antisymmetrization and incorporates the prescribed leading-order tail through the final analytic term.

The Adam runs sample a fresh batch $\eta \sim \mathrm{Unif}(0, c)$ at each step and minimize the squared profile residual together with the squared derivative residual,

$$\mathcal{L} = \mathbb{E}_{\eta \sim \mathrm{Unif}(0,c)} \left[|F_U(y(\eta))|^2 + |\partial_\eta F_U(y(\eta))|^2 \right]. \tag{21}$$

Both Adam rows use JAX with 64-bit arithmetic, 20K steps, batch size 4096, Adam with learning rate 10^{-4}, cosine decay to zero, $\epsilon = 10^{-15}$, no weight decay, and seed 0. The coordinate MLP has width 20, depth 4, and tanh activations. The `beignet` model uses the same decoder MLP and maps η to $(\eta + c)/(2c)$ before querying a 12-level Fourier feature pyramid with 4 channels per level, grid sizes $2, 4, \ldots, 4096$, global preconditioner 1000, per-level preconditioner 0.30, and initialization noise 10^{-5}.

The BFGS row uses the same width-20, depth-4 coordinate MLP, trained in PyTorch float64 on CUDA with batch size 10000, seed 0, and 20000 BFGS iterations in blocks of 1000 using the SSBroyden1 inverse-Hessian update. Its training objective follows the high-precision reference implementation and includes $0.1\,\mathbb{E}|F_U|^2 + \mathcal{L}_b + 0.1\,\mathbb{E}|\partial_\eta F_U|^2$ with a boundary term $\mathcal{L}_b$. For Table 2, however, the saved BFGS checkpoint is re-evaluated with the same unweighted diagnostic definitions used for the JAX rows.

All table metrics are evaluated after training on the centered diagnostic grid $\eta_i = c(i + 1/2)/1000$, $i = 0, \ldots, 999$, with $y_i = \sinh(\eta_i)$. We report $1000^{-1} \sum_i F_U(y_i)^2$ and $\log_{10} \max_i |F_U(y_i)|$; this grid is used only for reporting and is not resampled during optimization.

Table 10: Configuration comparison for the `beignet` models used in the JAX-PI benchmark experiments.

Area	Field	Allen–Cahn	KdV	Ginzburg–Landau	Gray–Scott
Pyramid	`num_scales`	9	4	5	5
Pyramid	`num_features` per level	14	64	64	64
Pyramid	Min grid	`x=2, t=8`	`x=2, t=8`	`x=2, y=2, t=8`	`x=2, y=2, t=8`
MLP	Width x depth	`256 x 4`	`256 x 4`	`128 x 3`	`128 x 3`
MLP	Activation	`tanh`	`tanh`	`swish`	`swish`
MLP	Reparam	`weight_fact`, mean 1.0, std 0.1	`weight_fact`, mean 1.0, std 0.1	`weight_fact`, mean 0.5, std 0.1	`weight_fact`, mean 0.5, std 0.1
Training	`max_steps`	300,000	200,000	100,000 per window	100,000 per window
Training	Time windows	not configured	not configured	5	10
Optim	Optimizer	Adam	Adam	Adam	Adam
Optim	LR	`1e-3`	`1e-3`	`1e-3`	`1e-3`
Optim	Decay steps	3000	2000	2000	2000
Optim	Warmup steps	1200	0	5000	0