Training dynamics in Physics-Informed Neural Networks with feature mapping

Chengxi Zeng [1] **Tilo Burghardt** [1] **Alberto M Gambaruto** [1]

Abstract

Physics-Informed Neural Networks (PINNs) have emerged as an iconic machine learning approach for solving Partial Differential Equations (PDEs). Although its variants have achieved significant progress, the empirical success of utilising feature mapping from the wider Implicit Neural Representations studies has been substantially neglected. We investigate the training dynamics of PINNs with a feature mapping layer via the limiting Conjugate Kernel and Neural Tangent Kernel, which sheds light on the convergence and generalisation of the model. We also show the inadequacy of commonly used Fourier-based feature mapping in some scenarios and propose the conditional positive definite Radial Basis Function as a better alternative. The empirical results reveal the efficacy of our method in diverse forward and inverse problem sets. This simple technique can be easily implemented in coordinate input networks and benefits the broad PINNs research.

1. Introduction

Our observed world is described by the laws of physics, and many phenomena can be defined by sets of Differential Equations (DEs). The physics knowledge is then used as prior in modern machine learning algorithms backed by the Universal Approximation Theorem (Hornik et al., 1989). The learning paradigm that enforces the mathematical rules and makes use of the available data is called Physics-Informed Machine Learning (PIML) (Karniadakis et al., 2021). Physics-Driven approaches have recently achieved significant success in a wide range of leading scientific research, from Electronics (Smith et al., 2022; Hu et al., 2023; Nicoli et al., 2023) and Medical Image (Goyeneche et al., 2023; Salehi & Giannacopoulos, 2022; Pokkunuru et al., 2023) to Dynamical System (Thangamuthu et al., 2022; Ni & Qureshi, 2023) and Meteorology (Kashinath et al., 2021; Giladi et al., 2021). Among these, one of the most prominent methods is termed Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019). It leverages the expressivity and differentiability of deep Neural Networks (NN) and integrates the DEs in the NN as a regulariser to introduce strong inductive biases during training. Conforming to traditional solvers, the formulation of PINN requires initial/boundary conditions (IC/BC) in a bounded spatial-temporal domain. The sampled points and their conditions (e.g. real value Dirichlet boundary) at the bounds are trained along with the collocation points that evaluate the residuals of the DEs. The goal is to optimise the overparameterised NN by minimising the residuals. Such converged parameter space can hence constitute a surrogate model that represents the solution space of the DEs.

PINNs share many common challenges, faltering at accurate convergence that is referred to as 'failure modes' of PINNs. Wang et al. (2020) leverage the 'Neural Tangent Kernel' theory that reveals PINNs suffer from 'Spectral Bias' due to the 'lazy-training' regime (Geiger et al., 2019); Krishnapriyan et al. (2021) demonstrate that PINNs are inherently difficult to optimise in harder Partial Differential Equations (PDEs) and multi-dimension cases; Lack of symmetry in the distribution of PDE and imbalanced residuals resulting in solutions from IC/BC cannot effectively alleviate the trivial solution in the PDE, and this is described as 'propagation failure' in (Daw et al., 2022). These analyses are principled in the design of PINNs variants, such as loss-reweighting (Wang et al., 2021b;c; 2022b; Psaros et al., 2021), domain decomposition (Jagtap & Karniadakis, 2020; Kharazmi et al., 2020; Moseley et al., 2021; Li et al., 2022), stronger regularisation (Yu et al., 2022; Wang et al., 2022a; Akhound-Sadegh et al., 2023) amongst others. We refer the readers to comprehensive reviews (Cuomo et al., 2022; Hao et al., 2022) on PINNs.

Whilst notable progress has been made in previous work, feature mapping has not been thoroughly studied, with only few works (Wang et al., 2021c; Wong et al., 2022) finding its potential in PINNs. Feature mapping was initially proposed in Natural Language Processing(NLP) with the goal of mapping the input to a higher-dimensional feature space and was later found to be effective at tackling spectral bias (Tancik et al., 2020) in visual representations.

In this paper, we investigate the training dynamics of PINNs with a feature mapping layer through the lens of linking two kernels, the Conjugate Kernel(CK) and Neural Tangent

[1]University of Bristol, UK. Correspondence to: Chengxi Zeng <cz15306@bristol.ac.uk>.

Preprint, Under Review

Kernel(NTK), where the former kernel is largely overlooked in the PINNs community. There are mainly two characteristics imposed by the CK and NTK in the infinite-width limit. Firstly, in this limit, the neural network behaves as a linear model that can be analysed in a conventional regression setting. Moreover, gradient descent is able to find the global minimum with unchanged parameters. This is confirmed in two-layer PINNs by (Gao et al., 2023). With appropriate initialisation and loss functions, the training loss converges to zero. Secondly, the CK and NTK drive the training dynamics of the neural network, and they govern the generalisation property in an overparameterised model. The limiting CK and NTK are sensitive to the inputs and network parameters. More specifically, they depend on the input gradient of the model, the variance of each layer and the non-linear activations that they pass through. Hence the training dynamics of PINNs are strongly influenced by what is fed to the model before the parameterised layers. We show that the coordinate-based input after a feature mapping layer positively impacts the CK and NTK. As a result, it improves the overall convergence of the model training.
We propose a framework for the design of the feature map layer that helps CK and NTK propagate in a practical setting. This one embedding layer is easy to implement and can seamlessly work with many PINNs variants and various training strategies such as curriculum learning and causal training. Our contribution can be summarised as follows:

- We provide theoretical work on the training dynamics of PINNs with a feature mapping layer in the limiting Conjugate Kernel and Neural Tangent Kernel scope. It reveals that the initial distribution of the feature mapping layer impacts the core of the PINNs.
- We extensively benchmark various feature mapping methods including those which have not been previously used in PINNs and show the limitation and failure of the common Fourier-based feature mapping in some Partial Differential Equations.
- We demonstrate a general framework for the design of feature mapping and propose conditional positive definite Radial Basis Function, which outperforms Fourier-based feature mapping in a range of forward and inverse tasks.

2. Background and Prior theory

2.1. Physics-Informed Neural Network

Following Raissi et al. (2019)'s formulation, the Physics-Informed Neural Network that solves both forward and inverse problems in PDEs is reviewed in a general form:

$$\begin{cases} \mathcal{D}[u(x,t;\alpha_i)] = F(x,t) & t \in \mathcal{T}[0,T], \forall x \in \Omega \\ u(x,0) = G(x) & x \in \Omega \\ \mathcal{B}[u(x,t)] = H(x,t) & t \in \mathcal{T}[0,T], x \in \partial\Omega \end{cases} \quad (1)$$

where $\mathcal{D}[\cdot]$ is the differential operator, x and t are the independent variables in spatial and temporal domains Ω and $\mathcal{T}$, respectively. The α_i are coefficients of the DE system and remain wholly or partially unknown in inverse problems. The DE confines to the initial condition when $t = 0$ and the boundary operator $\mathcal{B}$ at the boundary $\partial\Omega$. F, G and H are arbitrary functions.
PINN is parameterised by θ that can solve $\hat{u}_\theta$ at any x and t in the domain, hence the training loss functions are defined as follows:

$$\begin{aligned} \mathcal{L}(\theta; \mathcal{X}_{(x,t)}) = &\frac{\lambda_r}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{D}[\hat{u}_\theta(x_r^i)] - F(x_r^i) \right|^2 + \\ &\frac{\lambda_{ic}}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| \hat{u}_\theta(x_{ic}^i) - G(x_{ic}^i) \right|^2 + \\ &\frac{\lambda_{bc}}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}[\hat{u}_\theta(x_{bc}^i)] - H(x_{bc}^i) \right|^2 \end{aligned} \quad (2)$$

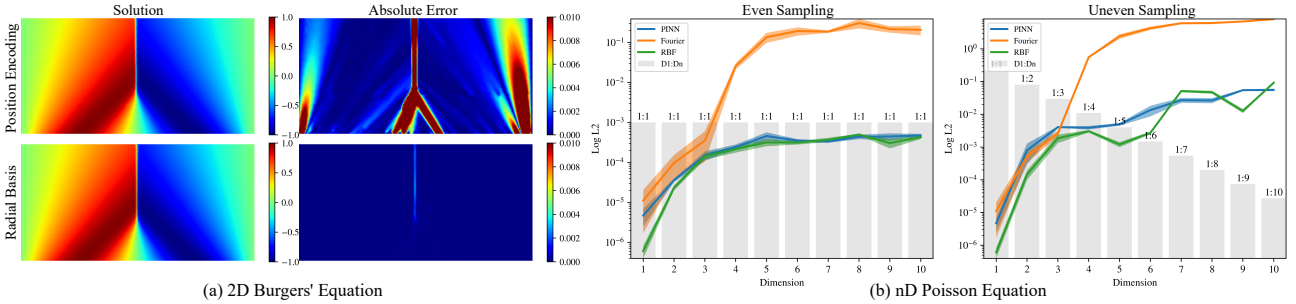
where $\{x_r^i\}_{i=1}^{N_r}$, $\{x_{ic}^i\}_{i=1}^{N_{ic}}$ and $\{x_{bc}^i\}_{i=1}^{N_{bc}}$ are collocation points, initial condition points and boundary condition points sampled from the bounded domain and commonly evaluated by computing the mean squared error. λ_r, λ_{ic} and λ_{bc} are the corresponding weights of each term. In this paper, we consider solving DEs in an unsupervised learning setting, though additional experimental/data points can be simply added to form the loss term $\mathcal{L}_{data}(\theta; \mathbf{x}_{data})$ as a strong regulariser.

2.2. Feature Mapping in PINNs

In the original form of PINNs, standard multi-layer perceptions (MLPs) have been adopted as the implicit neural presentation of the DEs. The MLPs can be mathematically expressed in the following recursive formulation, the model is parameterised by $\theta = \{w_j^l, b_j^l\}_{j=0}^{L}$, w is the weight matrix of the l-th layer, and b is the trainable bias after the non-linear activation function a. At initialisation, w and b are sampled independently from $\mathcal{N}(0,1)$:

$$\begin{aligned} f^l(x_i;\theta) &= \left(\frac{1}{\sqrt{d^l}} \sum_j w_{ij}^l h^{l-1}(x_i;\theta) + b_j^l \right), \\ h^l(x_i;\theta) &= a\left(f^l(x_i;\theta) \right). \end{aligned} \quad (3)$$

The normalisation $\frac{1}{\sqrt{d^l}}$ of weights by width d^l is placed so that we can take the width of the layers to infinity in the



(a) 2D Burgers' Equation (b) nD Poisson Equation

Figure 1. (a) PINNs solve Diffusion Equation with Positional Encoding (Top) and Our RBF (Bottom) feature mappings; (b) Error on nD Poisson equation from 1 to 10 dimensions cases (left), and a more realistic setting with uneven sampling on each dimension (right).

wide neural network regime. Feature mapping in the first layer is defined as:

$$f^1 = \frac{1}{\sqrt{d^1}} \sum_j w_{ij}^1 \varphi(x_i; \theta) \,, \tag{4}$$

where x is the coordinate-based input, φ is a feature mapping operator that projects input to a higher dimension feature space, $\Phi : x \in \mathbb{R}^n \to \mathbb{R}^m$, and typically $n \ll m$. Feature mapping is a broader term for positional encoding that can involve either fixed encoding or trainable embedding. Examples of Fourier feature mappings and other methods are given in Appendix F. Following this, we introduce two theoretical works regarding feature mapping in PINNs. One major work which theoretically supports feature mapping in PINNs is done by (Wang et al., 2020; 2021c), which formulates the training dynamics of PINNs following the seminal work done in general MLPs (Jacot et al., 2018) and proves the PINNs model converges to a deterministic kernel when the width of the NN tends to infinity. As a result, the training of PINNs is dominated by the leading eigenvalues in the Neural Tangent Kernel (NTK), this is termed Spectral Bias (more details about Spectral Bias in PINNs can be found in Appendix B). Hence a tunable bandwidth kernel is desirable to mitigate the Spectral Bias phenomenon. Bochner's theorem is employed to approximate any stationary kernel such that:

$$\begin{aligned} \boldsymbol{K}_\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \begin{bmatrix} \cos(2\pi \boldsymbol{b_m} \boldsymbol{x_i}) \\ \sin(2\pi \boldsymbol{b_m} \boldsymbol{x_i}) \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \cos(2\pi \boldsymbol{b_m} \boldsymbol{x_i}) \\ \sin(2\pi \boldsymbol{b_m} \boldsymbol{x_j}) \end{bmatrix} \\ &= \sum_{k=1}^{m} \cos\left(2\pi b_k^T \boldsymbol{x}_i\right) \cos\left(2\pi b_k^T \boldsymbol{x}_j\right) \\ &+ \sin\left(2\pi b_k^T \boldsymbol{x}_i\right) \sin\left(2\pi b_k^T \boldsymbol{x}_j\right) \\ &= \sum_{k=1}^{m} A_k^2 \cos\left(2\pi b_k^T \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right) . \end{aligned} \tag{5}$$

where A is the Fourier Series coefficients, $\boldsymbol{b}$ is randomly sampled from $\mathcal{N}(0, \sigma^2)$ and σ is an arbitrary hyperparameter that controls the bandwidth. Thereafter, the feature space becomes the input of the NTK which gives the identities: $\boldsymbol{K}_{NTK}(x_i^T x_j) = \boldsymbol{K}_{NTK}(\varphi(x_i)^T \varphi(x_j)) = \boldsymbol{K}_{NTK}(\boldsymbol{K}_\Phi(x_i - x_j))$. The final NTK is composed of a shift-invariant kernel with a controllable kernel width. It tends to be very useful to have computed Fourier features in multi-scale or high-frequency physical cases, and the optimal value of σ in each case can be decided by line search. Another proposition by (Wong et al., 2022) suggests that PINN suffers from limited input gradient variability under certain weight initialisation, this prevents the optimisation of parameter space from reaching joint PDE and BC optimal solutions. The keynote is that they reveal it is the enhanced input gradient distribution that improves the performance, not the features themselves. They employed the learnable Sinusoidal feature from concurrent work (Sitzmann et al., 2020) which can increase input gradient variability and help gradient descent escape the local minimum at initalisation. None of the prior theories addressed how feature mapping is particularly controlling the training dynamics of PINNs with a feature mapping layer, nor showing the limitation of Fourie-based features. In the next subsection, we show two examples that Fourier-based features exemplify poor functionality.

2.3. Motivating examples

Simple Fourier features can cause undesirable artefacts, shown in Figure 1 (a) (detailed equations are in Appendix J). When using Positional Encoding to solve Burgers equations, there appears to be a high prediction error in the region approaching a discontinuous solution. This effect is analogous to the Gibbs phenomenon, that is the approximated function value by a finite number of terms in its Fourier series tends to be overshooting and oscillating around a discontinuity. Another surprising experimental result which exhibits poor performance is using Random Fourier Features (Tancik et al., 2020) in high-dimension problems, as demonstrated in Figure 1 (b). An nD Poisson equation is tested from 1 to 10 dimensions with a Dirichlet boundary condition. First, ten test cases with increasing dimensions are set up with a fixed number of collocation points and they are evenly sampled for each dimension (i.e. the ratio between each dimension is

equal in each case, we denote the ratio between the first dimension to the last dimension by the bar chart). The total ℓ^2 error increases when the dimension of each case increases, as it becomes harder to solve. An evident result is that the random Fourier feature mapping does not generalise well in high dimensions($D > 3$). A more realistic case is set up when the number of sampling points is not the same in different dimensions(Figure 1 (b) right). In this setting, we set the number of sampling points $x_r = \frac{1}{D}$ for each case, meaning that as the dimension increases, there are fewer sampled points. This resembles the training setting in the unsteady Naverier-Stokes equations for fluids dynamics, which has fine sampling density in the spatial domain, but potentially rather sparse sampling in the temporal dimension. Although all methods have shown an increase in error, the one with Random Fourier Features in particular has been significantly underperforming in higher dimension cases. We tuned a few hyperparameters in the Fourier feature including the arbitrary scale σ and the number of Fourier features, yet none of the trials reduced the high error in high dimension cases. The two experiments are repeated 3 times with different random seeds, the variances are shown in the highlighted area. We show in the following proof that the Fourier function in the feature mapping settings is likely not to be a one-to-one function.

Lemma 2.1. *Consider a randomly sampled and normalised coordinate-based input* $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$, $x \in [0,1]^d$, *and its corresponding features in* $\Phi : \mathbb{R}^d \to \mathbb{R}^m = [\varphi(x_1), \varphi(x_2), \cdots, \varphi(x_n)]^T$, *let the feature mapping function* $\varphi(x) = sin(2\pi\mathcal{B}\mathbf{x}) \in [-1,1]$, *where* $\mathcal{B}$ *is sampled from a Gaussian distribution* $\mathcal{N}(0,\sigma)$, *the mapping function* $sin(\cdot)$ *is not bijective w.h.p.*

Proof. see Appendix C. □

A non-bijective function denotes that the inputs are only surjectively mapped to the feature space from the domain. This indicates that an overlapping image is likely to be formed in the projected codomain, and it can also partially explain the Gibss phenomenon in discontinuous regions with overshoot function values. As it can be easily inferred, in higher dimensions the probability of the Fourier features is even lower to be bijective.

3. Kernel matrices in PINNs

Give the training dataset $X = \{x_r^i\}_{i=1}^{N_r} \cup \{x_{ic}^i\}_{i=1}^{N_{ic}} \cup \{x_{bc}^i\}_{i=1}^{N_{bc}}$. We denote the matrices after the feature mapping and post-activation by $\mathcal{X}_\Phi = \varphi(X) \in \mathcal{R}^{N\times n}$ and $\mathcal{X}_h^l = h^l(X,\theta) = \frac{1}{d^l}a(W^l X^{l-1}) \in \mathcal{R}^{N\times d^l}$, $\mathcal{X} = \{\mathcal{X}_\Phi\} \bigvee \{\mathcal{X}_h\}$. In the infinite-width limits, the non-linear neural network evolves similarly to the kernel regression models (Lee et al., 2019). Hence we can leverage two types of kernels, the **Conjugate Kernel(CK)** and the **Neural Tangent Kernel(NTK)** to analyse the initial distribution of the model and the training dynamics of the PINNs to the infinite-width limit.
The CK in each layer is defined as:

$$K_{CK}^l = {\mathcal{X}^l}^T \mathcal{X}^l \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}} = \Sigma^l(x,x') \tag{6}$$

and we formulate the general (in contrast to the formulation in (Wang et al., 2020)) NTK in PINNs as:

$$K_{NTK}^l = \nabla_\theta f^l(\mathcal{X};\theta)^T \nabla_\theta f^l(\mathcal{X};\theta) \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}} = \Theta^l(x,x') \tag{7}$$

We now derive the training dynamics of the PINNs in these two limiting kernels taking account of the feature mapping.

Proposition 3.1 (Propagation of the Conjugate Kernel)**.** *Let input* $x \in \mathcal{R}^{N\times n}$, *and each layer of the Neural Network is parameterised with independent and identically distributed (i.i.d.) weights and biases from standard Gaussian distribution. Hence* $f^l(x;\theta_0) \sim \mathcal{GP}(0, \Sigma^l(x,x'))$, *and the Conjugate Kernels propagate through the Neural Network in the following recursive form:*

$$\begin{aligned} \Sigma^1(x,x') &= < x, x' > + 1 \\ \Sigma^2(x,x') &= \mathbf{E}[\varphi(x)^T \varphi(x')] + 1 \\ \Sigma^l(x,x') &= \mathbf{E}[a(Z)^T a(Z')] + 1, \quad 2 < l \le L \end{aligned} \tag{8}$$

where φ *is the feature mapping function at* $l = 2$ *and* Z, Z' *are the hidden layer state from previous layer and* $\begin{bmatrix} Z \\ Z' \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma^{\ell-1}(x,x) & \Sigma^{\ell-1}(x',x) \\ \Sigma^{\ell-1}(x,x') & \Sigma^{\ell-1}(x',x') \end{bmatrix})$.

Proof. see Appendix D. □

Here we derive the initial distribution of each layer in the network through the feature mapping layer and the non-linear activated layers by computing the explicit Conjugate Kernels. This indicates the Gaussian distribution can propagate layers from the very first feature layer. One important note is that $\mathbf{E}(f^l(x;\theta_0)) = 0$ holds true for all layers after the feature mapping layer, however, it is only true for the feature mapping layer if the features are randomly sampled. This gives us an insight into the design of feature mapping functions, i.e. φ needs to incorporate randomly sampled initialisation.
We now investigate the training dynamics of the PINNs by linking the CK and NTK.

Theorem 3.2 (Evolution of the NTK with CK)**.** *Let input* $x \in \mathcal{R}^{N\times n}$, $\phi(x) = \varphi(x) \bigvee a(x)$ *; Recall* $\Sigma^2(x,x') = \mathbf{E}[\varphi(x)^T\varphi(x')] + 1$, $\Sigma^l(x,x') = \mathbf{E}[\phi(x)\phi(x')] + 1$ *and its derivative is* $\dot{\Sigma}^l(x,x') = \mathbf{E}[\dot{\phi}(x)\dot{\phi}(x')], \in R^{N\times N}$. *Assuming the infinity width limit, the gradient* ∇f^l *satisfies:*

$$\nabla_\theta f^l(x;\theta_0)^T \nabla_\theta f^l(x';\theta_0) \to \Theta^l(x,x') \tag{9}$$

The evolution of the kernels follows:

$$\Theta^2(x,x') = \Theta^1(x,x')\dot{\Sigma}^2(x,x') + \Sigma^2(x,x')$$
$$\Theta^l(x,x') = \Theta^{l-1}(x,x')\dot{\Sigma}^l(x,x') + \Sigma^l(x,x'), \quad 2 < l \leq L \tag{10}$$

Proof. see Appendix E. □

The second key theoretical results reveal the NTK in each layer depended on the NTK of the last layer and the CK and its derivative from the current layer. More importantly, the distribution of the feature mapping layer will propagate through the network from layer 2 and onwards. The CK derivative of the feature mapping layer plays a key role in the evolution. We want the feature mapping to form a kernel which is at least 1st-order differentiable.

4. Proposed feature mapping method

Recall that the MLP function is approximated by the convolution of stationary composed NTK function $K_{COMP} = K_{NTK} \circ K_\Phi$ with weighted Dirac delta over the input $\mathbf{x}$, we can formulate the K_{COMP} by:

$$\begin{aligned} K_{COMP}(\mathbf{x}) &= (K_{COMP} * \delta_x)(\mathbf{x}) \\ &= \int K_{COMP}(x')\delta(x-x')dx \\ &\approx \int K_{COMP}(x')K_\Phi(x-x')dx \end{aligned} \tag{11}$$

The accuracy of the continuous approximation can be analysed by Taylor series expansion:

$$\begin{aligned} K_{COMP}(\mathbf{x}) = \int & (K_{COMP}(\mathbf{x}) + \nabla_x K_{COMP}(x-x') \\ & + \frac{1}{2}(x-x')\nabla^2 K_{COMP}(x-x') \\ & + \mathcal{O}((x-x')^3))K_\Phi(x-x')dx \\ = & K_{COMP}(\mathbf{x}) \int K_\Phi(x-x')dx \\ & + \nabla_x K_{COMP}(x-x') \int (x-x')K_\Phi(x-x')dx \\ & + \mathcal{O}((x-x')^2) \end{aligned} \tag{12}$$

To make sure the composing kernel is 1st-order accurate, we will need the term $\int K_\Phi(x-x')dx = 1$ and the second term in Equation 12 to be 0. This can be achieved simply by normalising the feature mapping kernel and making it meet a symmetry condition. We propose a positive definite Radial Basis Function (RBF) for such kernel, and the feature mapping function is given by:

$$\Phi(\mathbf{x}) = \frac{\sum_i^m w_i \varphi(|\mathbf{x}-\mathbf{c}_i|)}{\sum_i^m \varphi(|\mathbf{x}-\mathbf{c}_i|)} \tag{13}$$

where $\mathbf{x} \in R^n$ is the input data, $\mathbf{C} \in R^{n\times m}$ are the centres of the RBFs and are trainable parameters. A natural choice for the RBF can be the Gaussian function, $\varphi(x) = e^{-\frac{|\mathbf{x}-\mathbf{c}|^2}{\sigma^2}}$. If we choose the same number of features as the input size (i.e. $n = m$), this is the same as the mesh-free RBF interpolation method which gives approximate computation of desired function value by kernel regression. Unfortunately, the training input size is normally very large in PINNs, so it does not scale well in this setting. In our empirical study, we show a few hundred RBFs are sufficient to outperform other types of feature mapping functions. At initialisation, $\mathbf{c}$ is sampled from a standard Gaussian to follow the propagation in Proposition 3.1.

4.1. Conditionally positive definite RBF

Each layer of the NN is considered a linear system in the infinite-width limit. To ensure a unique solution, one way is to introduce conditionally positive definite radial functions by adding polynomial terms. The weights work as Lagrange multipliers that can constrain the RBF coefficients in the parameter space. The feature mapping function is modified as:

$$\Phi(\mathbf{x}) = \frac{\sum_i^m w_i^m \varphi(|\mathbf{x}-\mathbf{c}_i|)}{\sum_i^m \varphi(|\mathbf{x}-\mathbf{c}_i|)} + \sum_j^k w_j^k p_j(\mathbf{x}) \tag{14}$$

In the feature mapping layer, it can be represented as:

$$\begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} W_m \\ - \\ W_k \end{bmatrix} \left[\begin{array}{ccc|ccc} \varphi(r_1^1) & \dots & \varphi(r_1^m) & 1 & x_1 & x_k^P \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \varphi(r_N^1) & \dots & \varphi(r_N^m) & 1 & x_N & x_N^P \end{array}\right] \tag{15}$$

where $r = x - c$ and P is the order of the polynomial term.

We now give the weak definition of a conditionally positive definite kernel.

Definition 4.1. Given a dataset $X = \{x_1, \cdots, x_N\} \in R^{N\times n}$, the continuous Radial Basis Function with centres $\mathbf{c} = \{c_1, \cdots, c_m\} \in R^m$ and polynomial terms $P(x) = \{x^0, x^1, \cdots, x^k\}$ is defined as conditionally positive definite if:

$$\mathbf{W}\Phi = \sum_{j=1}^{m+k}\sum_{i=1}^{N}(W_m\varphi(|x_i - c_j|) + W_k P(|x|)) > 0 \tag{16}$$

We empirically find the polynomial term extremely useful in non-linear function approximation such as the Burgers Equation and Naiver-Stokes Equation without adding too much computational overhead.

By this principle, we can use many other types of RBF without too many restrictions. Other types of RBF are shown in Appendix H Table 7.

4.2. Compact support RBF

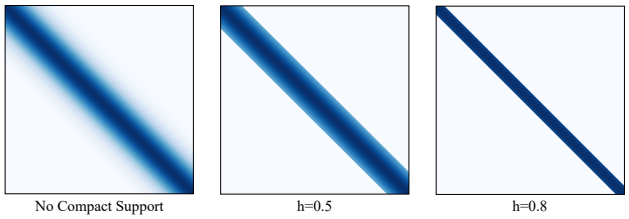


Figure 2. Feature map with different RBF support.

Here we introduce compact support RBFs. RBFs such as Gaussian tend to be very close to 0 at infinity, but they never reach 0. We use a cut-off distance that makes distant points produce values of 0. This is formulated by:

$$\Phi(x,h) = \begin{cases} \varphi(|x-c|,h), & h \le \xi \\ 0, & h > \xi \end{cases}. \tag{17}$$

where ξ is an arbitrary cutting-off distance. This guarantees the points with high Euclidean distance do not yield features (in a way, it reduces subjectiveness) and make the resulting feature matrix a sparse matrix that can be potentially more efficient for computation.

5. Related Work

Coordinate Sampling. As a mesh-free method, PINNs are normally evaluated on scattered collocation points both on the interior domain and IC/BC. Therefore, the sampling strategy is crucial to PINNs' performance and efficiency. A poorly distributed initial sampling can lead to the PDE system being ill-conditioned and NN training instability. The whole design of experiments on the fixed input sampling is reviewed by (Das & Tesfamariam, 2022). Based on the study of uniform sampling, Wu et al. (2023) proposed an adaptive sampling scheme that refines high residual area during training. Similarly, Importance Sampling inspired by Monte Carlo approximation is investigated by (Nabian et al., 2021; Yang et al., 2023). Daw et al. (2022) proposed a novel sampling strategy that mitigates the 'propagation failure' of solutions from IC/BC to the PDE residual field.

Novel Activation. The activation function in the MLP has been found to play an important role in the convergence of the PINNs. Popular activation ReLU is deficient for high-order PDEs since its second-order derivative is 0. Apart from the standard Tanh activation (Raissi et al., 2019), layer-wise and neuron-wise adaptive activation are proven to be useful to accelerate the training (Jagtap & Karniadakis, 2019; Jagtap et al., 2020). Another line of seminal work, SIREN (Sitzmann et al., 2020), which uses periodic activation function, has achieved remarkable results in Neural Representation and tested on solving the Poisson equation. Gaussian (Ramasinghe & Lucey, 2022) and Gabor Wavelet activations (Saragadam et al., 2023) are proven to be effective alternatives.

Positional Embedding. Broadly speaking, PINNs can also be considered as a special type of Neural Fields (Xie et al., 2021) in visual computing, which specifically feed coordinate-based input to MLPs that represent continuous field quantity (e.g. velocity field in fluid mechanics) over arbitrary spatial and temporal resolution. However, the PINNs community often ignores the fact both perspectives function the same way as Implicit Neural Representations. In the Neural Field, images and 3D shapes are naturally high-frequency signals, whereas deep networks are inherently learning towards the low-frequency components (Rahaman et al., 2018). Feature mapping hence has become a standard process in practice that maps the low-dimension coordinates to high-dimension space. The pioneering work was conducted by (Rahimi & Recht, 2007), who used Fourier features to approximate any stationary kernel principled by Bochner's theorem. the derivative works are done such as Positional Encoding (Mildenhall et al., 2020), Random Feature (Tancik et al., 2020) and Sinusoidal Feature (Sitzmann et al., 2020). Another concurrent work discusses non-periodic feature mapping (Zheng et al., 2022; Ramasinghe & Lucey, 2021; Wang et al., 2021a). To the best of our knowledge, feature mapping in PINNs has been largely uninvestigated. Only a few work preliminarily adopted Fourier-feature-based methods in PINN (Wang et al., 2021c; 2023; Wong et al., 2022).

6. Empirical Results

6.1. Experimental Setup

Compared methods. We compared our methods with other feature mapping methods for coordinate-based input networks. This includes Fourier-based methods such as Basic Encoding (BE), Positional Encoding (PE), Random Fourier Feature (FF) and Sinusoidal Feature (SF) and Non-Fourier-based ones such as Complex Triangle (CT) and Complex Gaussian (CG). The exact function and related literature can be found in Appendix F. RBF-INT is **our** standard feature mapping function in the formulation of RBF interpolants. RBF-POL and RBF-COM stand for RBF-INT with polynomials and RBF-INT with compact support throughout the paper. We use Gaussian RBF for all experiments unless otherwise stated.

Benchmarked PDEs. We conducted benchmarks from existing literature (Lu et al., 2021; Hao et al., 2023) on various PDEs in both forward and inverse problems. The forward problems demonstrated are the Wave equation (Hy-

Table 1. PDEs benchmark results comparing different feature mapping methods in $\ell 2$ error. The best results are in Blue. Complete experimental results with standard deviations are shown in Appendix G.

	PINN	BE	PE	FF	SF	CT	CG	**RBF-INT**	**RBF-POL**
Wave	3.731e-1	1.035e0	1.014e0	2.375e-3	7.932e-3	1.114e0	1.036e0	2.814e-2	2.361e-2
Diffusion	1.426e-4	1.575e-1	1.595e-1	2.334e-3	3.474e-4	1.860e0	2.721e-2	3.066e-4	3.498e-5
Heat	4.732e-3	6.491e-3	7.574e-3	2.190e-3	3.961e-3	4.524e-1	2.626e-1	1.157e-3	4.098e-4
Poisson	3.618e-3	4.964e-1	4.910e-1	7.586e-4	9.078e-4	6.348e-1	2.334e-1	5.259e-4	8.942e-4
Burgers	1.864e-3	5.585e-1	5.363e-1	7.496e-2	1.299e-3	9.935e-1	7.521e-1	2.945e-3	3.159e-4
Steady NS	5.264e-1	7.143e-1	6.332e-1	6.939e-1	3.769e-1	5.460e-1	4.867e-1	2.991e-1	2.567e-1

perbolic), Diffusion&Heat equation (Parabolic), Poisson equation (Elliptic) and Burgers&Navier-Stokes (NS) equation (Non-linear). The inverse problems are the Inverse Burgers equation and Inverse Lorenz equations. The full equations and their boundary conditions are specified in Appendix J.

Implementation details. All feature mapping methods are implemented in the same NN architecture that consists of 5 fully connected layers with 100 neurons each for the forward problems and 50 neurons each for the inverse problems unless otherwise specified. The numbers of features from each feature function tested are 64, 128 and 256. The numbers of polynomial terms tested are 5, 10, 15 and 20. The non-linear activation function chosen is Tanh. The NN parameters are initialised with Xavier initialisation (Glorot & Bengio, 2010). The NN is trained with the Adam optimiser with an initial learning rate of $1e-3$ for 20k epochs and L-BFGS for another 20k epochs. All codes are implemented in Pytorch 2.0.0 and can be found in this [Anonymous link].

Evaluation. We employed the standard mean square error (MSE) as the loss function for the PDE loss term and IC/BC loss terms, they generally have good behaviour during training. The prediction results are evaluated by a relative $\ell 2$ error.

$$\text{L2RE} = \sqrt{\frac{\sum_{i=1}^{n} \left(\mathbf{u_i} - \mathbf{u_i}'\right)^2}{\sum_{i=1}^{n} \mathbf{u_i}'^2}} \tag{18}$$

where $\mathbf{u_i}$ is the prediction results and $\mathbf{u_i}'$ is the ground truth from either analytical solution or high-fidelity numerical methods. For the inverse problems, the $\ell 2$ is computed between the predicted coefficients and true coefficients that are used to generate the data.

6.2. Forward Problems

Time-dependent PDEs. Some benchmarked Time-dependent PDEs only have Dirichlet initial conditions (e.g. the Diffusion equation and the Heat equation in Table 1), then their initial condition can be treated as a special type of boundary condition during loss optimisation. This offers us the advantage of homogeneously sampling IC/BC points across spatial and temporal domains. A higher penalty on the IC/BC terms is adopted in the experiments, that is set $\lambda_r = 1$, $\lambda_{ic} = 100$ and $\lambda_{bc} = 100$ from Equation 2. By doing so, we found it is easier for the solutions from the IC to propagate to the domain, and comply with the hard BC constraints.

Our solution in the Diffusion equation shows superior performance over other methods by some order of magnitude. The boundary errors are visibly more in Fourier-based methods shown in Appendix K Figure 11.

The RBFs are also better at handling multiscale problems demonstrated by the Heat equation J.3. The Heat equation case has a huge contrast in coefficients, $\frac{1}{500\pi^2}$ and $\frac{1}{\pi^2}$ for the x and y directions. Figure 12 has shown the RBF-INT method preserves the details of the solution in each time step.

Non-linear PDEs. We evaluate the methods on two classic non-linear PDEs, the Burgers equation and the Navier-Stokes equation. It has been shown in Figure 14 that the RBFs with polynomial terms are more capable of solving the discontinuity at $x = 0$. The steady N-S equation has no time derivative term. However, the back step flow geometry makes the model harder to generalise, hence we again penalise the BC loss term with a magnitude of 100.

All cases are tested with 2k sampling points at each boundary and 20k collocation points in the domain. The Wave equation has an addition Neumann boundary condition that is treated by the differential operation like the PDEs but added to the IC loss term.

6.3. Inverse Problems

Table 2. Benchmark on the Inverse problems in $\ell 2$ error. * indicates problems with noises added to the data. Full results with standard deviations in Appendix G.

	FF	SF	**RBF-INT**	**RBF-POL**
I-Burgers	2.391e-2	2.436e-2	1.741e-2	1.575e-2
I-Lorenz	6.516e-3	6.390e-3	6.080e-3	5.991e-3
I-Burgers*	2.509e-2	2.913e-2	1.993e-2	1.753e-2
I-Lorenz*	7.934e-3	6.856e-3	6.699e-3	6.342e-3

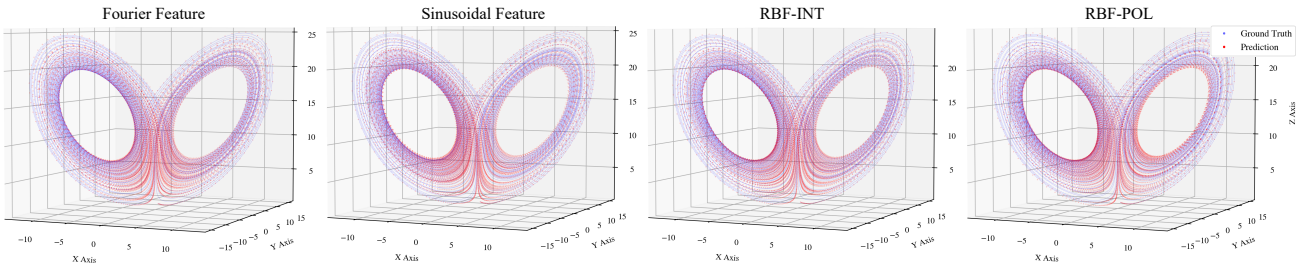


Figure 3. Visualisation of the Lorenz system with coefficients predicted by different feature mapping. A slight change of any of the coefficients will result visible deviations. (Zoom in for better trajectory visualisation.)

A major application of the PINNs is able to solve inverse problems. The unknown coefficients in the differential equations can be discovered by a small amount of data points, take the Lorenz system J.9 as an example; The α, ρ and β are three unknown coefficients during training. We can explicitly attach the coefficients to the neural network as learnable external parameters. The coefficients along with the PDEs are to construct the PDEs loss. With a small amount of data loss, the model will be able to converge and the learnable external parameters are optimised to the ideal coefficients. Unlike forward problems in unsupervised settings require a large amount of collocation points and IC/BC points, inverse problems are sometimes less demanding. The MLP naturally excels in data interpolation. There are 5000 data points used in the inverse Burgers equation problem and the same points are used to compute the PDE loss. For the inverse Lorenz system, only 40 data points are used and 400 collocations are sampled in $t \in [0, 3]$. Lorenz system is sensitive to coefficients and initial position changes. The initial positions $x_0 = 0$, $y_0 = 1$, $z_0 = 1.05$ are not given to the model.

Another experiment conducted is to test if the feature mapping functions are prone to noise. 1% Gaussian noises are added to the inverse Burgers problem and 0.5% to the Lorenze system data. The results shown in Table 2 indicate the 4 feature mapping methods tested are robust to noises to some degree. Among all, RBF-POL is the most resilient feature mapping function to noises.

6.4. Ablation studies

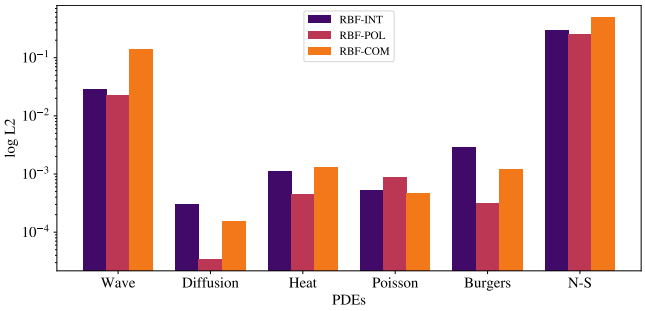


Figure 4. Performance of RBF with compact support.

First, we demonstrate the performance of RBFs with compact support proposed in Section 4.2. The RBFs with compact support perform similarly to the RBFs alone, but they have a slightly better convergence rate, shown in the next subsection. Other studies that show the results with different numbers of RBFs, numbers of polynomials and performance of different types of RBFs can be found in Appendix H.

6.5. Convergence, Complexity and Scalability

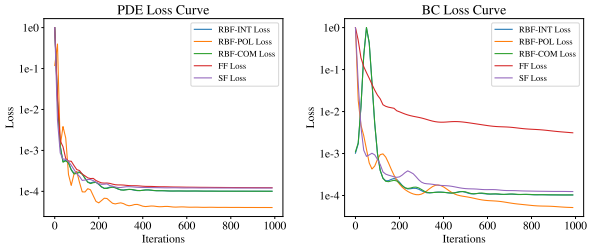


Figure 5. Loss curves of PDE and BC loss on Diffusion equation.

Convergence analysis on the Diffusion equation is shown in Figure 5. Our methods not only show better convergences overall, but also show a better adjustment with the boundary conditions. The comparison of complexity and scalability of feature mapping methods are followed in Appendix I.

7. Conclusion and Future Work

In conclusion, we provided theoretical proof that feature mapping in PINNs influences the Conjugate Kernel and Neural Tanget Kernel which dominate the training dynamic of PINNs. We introduce a framework to design an effective feature mapping function in PINNs and propose Radial Basis Function based feature mapping approaches. Our method not only improves the generalisation in a range of forward and inverse physics problems but also outperforms other feature mapping methods by a significant margin. RBF feature mapping can potentially work with many other PINNs techniques such as some novel activation functions and different types of loss or training strategies such as curriculum training. While this work focuses on solving PDEs, RBF feature mapping continues to explore its application in other coordinates-based input neural networks for different tasks.

Impact Statement

The aim of this research is to contribute to the development of Machine Learning. Our work may have various implications for society, but we do not think any of them need special attention here.

References

Akhound-Sadegh, T., Perreault-Levasseur, L., Brandstetter, J., Welling, M., and Ravanbakhsh, S. Lie point symmetry and physics informed networks. In *Neural Information Processing Systems*, 2023.

Cuomo, S., Cola, V. S. D., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92, 2022.

Das, S. and Tesfamariam, S. State-of-the-art review of design of experiments for physics-informed deep learning. *ArXiv*, abs/2202.06416, 2022.

Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. In *International Conference on Machine Learning*, 2022.

Gao, Y., Gu, Y., and Ng, M. K. Gradient descent finds the global optima of two-layer physics-informed neural networks. In *International Conference on Machine Learning*, 2023.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020, 2019.

Giladi, N., Ben-Haim, Z., Nevo, S., Matias, Y., and Soudry, D. Physics-aware downsampling with deep learning for scalable flood modeling. In *Neural Information Processing Systems*, 2021.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Goyeneche, A. D., Ramachandran, S., Wang, K., Karasan, E., Cheng, J. Y., Stella, X. Y., and Lustig, M. Resonet: Noise-trained physics-informed mri off-resonance correction. In *Neural Information Processing Systems*, 2023.

Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., and Zhu, J. Physics-informed machine learning: A survey on problems, methods and applications. *ArXiv*, abs/2211.08064, 2022.

Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., and Zhu, J. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *ArXiv*, abs/2306.08827, 2023.

Hornik, K., Stinchcombe, M. B., and White, H. L. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

Hu, Y., Li, J., Klemme, F., Nam, G.-J., Ma, T., Amrouch, H., and Xiong, J. Synctree: Fast timing analysis for integrated circuit design through a physics-informed tree-based graph neural network. In *Neural Information Processing Systems*, 2023.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Neural Information Processing Systems*, 2018.

Jagtap, A. D. and Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.*, 404, 2019.

Jagtap, A. D. and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 2020.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476, 2020.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3:422 – 440, 2021.

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

Kharazmi, E., Zhang, Z., and Karniadakis, G. E. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *ArXiv*, abs/2003.05385, 2020.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Neural Information Processing Systems*, 2021.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J. N., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Neural Information Processing Systems*, 2019.

Li, S., Penwarden, M., Kirby, R. M., and Zhe, S. Meta learning of interface conditions for multi-domain physics-informed neural networks. In *International Conference on Machine Learning*, 2022.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021. doi: 10.1137/19M1274067.

Matthews, A., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. 2018.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65:99–106, 2020.

Moseley, B., Markham, A., and Nissen-Meyer, T. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49, 2021.

Nabian, M. A., Gladstone, R. J., and Meidani, H. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36:962 – 977, 2021.

Ni, R. and Qureshi, A. H. Ntfields: Neural time fields for physics-informed robot motion planning. In *International Conference on Learning Representations*, 2023.

Nicoli, K. A., Anders, C. J., Funcke, L., Hartung, T., Jansen, K., Kuhn, S., Muller, K. R., Stornati, P., Kessel, P., and Nakajima, S. Physics-informed bayesian optimization of variational quantum circuits. In *Neural Information Processing Systems*, 2023.

Pokkunuru, A., Rooshenas, A., Strauss, T., Abhishek, A., and Khan, T. Improved training of physics-informed neural networks using energy-based priors: a study on electrical impedance tomography. In *International Conference on Learning Representations*, 2023.

Psaros, A. F., Kawaguchi, K., and Karniadakis, G. E. Meta-learning pinn loss functions. *J. Comput. Phys.*, 458: 111121, 2021.

Rahaman, N., Baratin, A., Arpit, D., Dräxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. C. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

Ramasinghe, S. and Lucey, S. A learnable radial basis positional embedding for coordinate-mlps. In *AAAI Conference on Artificial Intelligence*, 2021.

Ramasinghe, S. and Lucey, S. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *ECCV 2022: 17th European Conference Proceedings, Part XXXIII*, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19826-7.

Salehi, Y. and Giannacopoulos, D. D. Physgnn: A physics-driven graph neural network based model for predicting soft tissue deformation in image-guided neurosurgery. In *Neural Information Processing Systems*, 2022.

Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., and Baraniuk, R. Wire: Wavelet implicit neural representations. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18507–18516, 2023.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Neural Information Processing Systems*, 2020.

Smith, K. D., Seccamonte, F., Swami, A., and Bullo, F. Physics-informed implicit representations of equilibrium network flows. In *Neural Information Processing Systems*, 2022.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *Neural Information Processing Systems*, 2020.

Thangamuthu, A., Kumar, G., Bishnoi, S., Bhattoo, R., Krishnan, N. M. A., and Ranu, S. Unravelling the performance of physics-informed graph neural networks for dynamical systems. In *Neural Information Processing Systems*, 2022.

Wang, C., Li, S., He, D., and Wang, L. Is l2 physics informed loss always suitable for training physics informed neural network? In *Neural Information Processing Systems*, 2022a.

Wang, P.-S., Liu, Y., Yang, Y.-Q., and Tong, X. Spline positional encoding for learning 3d implicit signed distance fields. In *International Joint Conference on Artificial Intelligence*, 2021a.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *J. Comput. Phys.*, 449:110768, 2020.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.*, 43:A3055–A3081, 2021b.

Wang, S., Wang, H., and Perdikaris, P. 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, 2021c. ISSN 0045-7825.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *ArXiv*, abs/2203.07404, 2022b.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An expert's guide to training physics-informed neural networks. *ArXiv*, abs/2308.08468, 2023.

Wong, J., Ooi, C., Gupta, A., and Ong, Y. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, PP:1–5, 01 2022. doi: 10.1109/TAI.2022.3192362.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023. ISSN 0045-7825.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 41, 2021.

Yang, Z., Qiu, Z., and Fu, D. Dmis: Dynamic mesh-based importance sampling for training physics-informed neural networks. AAAI Press, 2023.

Yu, J., Lu, L., Meng, X., and Karniadakis, G. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 03 2022.

Zheng, J., Ramasinghe, S., Li, X., and Lucey, S. Trading positional complexity vs. deepness in coordinate networks. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

A. Abbreviations and Notations

Table 3. Long forms for the abbreviations used in the paper

Abbreviations	Long forms
BC	Boundary Condition
BE	Basic Encoding
CG	Complex Gaussian
CK	Conjugate Kernel
CT	Complex Triangle
DEs	Differential Equations
FF	Fourier Feature
IC	Initial Condition
i.i.d.	independent and identically distributed
L2RE	Relative $\ell 2$ error
MLP	Multi-Layer Perception
NLP	Natural Language Processing
NN	Neural Network
NTK	Neural Tangent Kernel
PDEs	Partial Differential Equations
PE	Positional Encoding
PIML	Physics-Informed Machine Learning
PINNs	Physics-Informed Neural Networks
RBF	Radial Basis Function
RBF-COM	RBF with Compact Support
RBF-INT	RBF with Interpolants
RBF-POL	RBF with Polynomials
SF	Sinusoidal Feature
w.h.p.	with high probability

Table 4. Symbols and their definitions in the paper

Symbols	Definition	Symbols	Definition
a	Activation Function	t	Temporal Coordinate
A	Fourier Series Coefficients	T	Temporal Range
b	Biases	$\mathcal{T}$	Temporal Domain
$\mathbf{b}$	Random sample	u	Differential Functions
$\mathcal{B}$	Boundary Operator	$\hat{u}_\theta$	Implicit Function
d	Number of Neurons	w	Weights
$\mathcal{D}$	Differential Operator	x	Spatial Coordinate or Input
f	Layer Function	Z	Hidden Layer Kernel
F	Arbitrary Function	$\forall$	For All
G	Arbitrary Function	φ	Feature Mapping Function
h	Hidden Layer Function	Φ	Feature Space
H	Arbitrary Function	λ	Loss Weighting
$\mathbf{K}$	Kernel	θ	Model Parameters
l	Layer	Θ	Conjugate Kernel
$\mathcal{L}$	Loss	Σ	Neural Tangent Kernel
N	Number of sample points	Ω	Spatial Domain

B. Spectral Bias in PINNs

Normally PINNs are setup as a standard MLP model $f(X;\theta)$, and θ is optimized on the loss function $L(\theta) = \frac{1}{2}|f(X;\theta) - Y|^2 = \frac{1}{2}\sum_i^N (f(x_i;\theta) - y_i)^2$, where X, Y and θ are training input, training ground truth and model parameters. For an easier formulation, we replace the conventional gradient descent formulation $\theta_{t+1} = \theta_t - \alpha\nabla_\theta L(\theta_t)$ to a gradient flow equation:

$$\frac{d\theta}{dt} = -\alpha\nabla_\theta L(\theta_t) \tag{19}$$

where α should be an infinitesimally small learning rate in the NTK setting.
Given PDE collocation data points$\{x_r^i, \mathcal{D}(\hat{u}_\theta(x_r^i))\}_{i=1}^{N_r}$, and boundary training points$\{x_{bc}^i, \mathcal{B}(\hat{u}_\theta(x_{bc}^i))\}_{i=1}^{N_{bc}}$. The gradient flow can be formulated as (Wang et al., 2020):

$$\begin{bmatrix} \frac{du(x_b,\theta_t)}{dt} \\ \frac{d\mathcal{L}u(x_r,\theta_t)}{dt} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{K}_{uu}^t & \boldsymbol{K}_{ur}^t \\ \boldsymbol{K}_{ru}^t & \boldsymbol{K}_{rr}^t \end{bmatrix} \cdot \begin{bmatrix} u(x_b,\theta_t) - \mathcal{B}(\hat{u}_\theta(x_{bc})) \\ \mathcal{L}u(x_r,\theta_t) - \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix}, \tag{20}$$

where the Kernels $\boldsymbol{K}$ are:

$$\begin{aligned} \left(\boldsymbol{K}_{uu}^t\right)_{ij} &= \left\langle \frac{du\left(x_b^i,\theta_t\right)}{d\theta}, \frac{du\left(x_b^j,\theta_t\right)}{d\boldsymbol{\theta}} \right\rangle \\ \left(\boldsymbol{K}_{rr}^t\right)_{ij} &= \left\langle \frac{d\mathcal{L}\left(x_r^i,\theta_t\right)}{d\theta}, \frac{d\mathcal{L}\left(x_r^j,\theta_t\right)}{d\theta} \right\rangle \\ \left(\boldsymbol{K}_{ur}^t\right)_{ij} &= \left(\boldsymbol{K}_{ru}^t\right)_{ij} = \left\langle \frac{du\left(x_b^i,\theta_t\right)}{d\theta}, \frac{d\mathcal{L}u\left(x_r^j,\theta_t\right)}{d\theta} \right\rangle \end{aligned} \tag{21}$$

Since $\boldsymbol{K}$ remains stationary, then $\boldsymbol{K}^t \approx \boldsymbol{K}^0$ as NN width tends to infinity, Equation 20 is rewritten as:

$$\begin{aligned} \begin{bmatrix} \frac{du(x_b,\theta_t)}{dt} \\ \frac{d\mathcal{L}u(x_r,\theta_t)}{dt} \end{bmatrix} &\approx -\boldsymbol{K}^0 \begin{bmatrix} u(x_b,\theta_t) - \mathcal{B}(\hat{u}_\theta(x_{bc})) \\ \mathcal{L}u(x_r,\theta_t) - \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} \\ &\approx (I - e^{-\boldsymbol{K}^0 t}) \cdot \begin{bmatrix} \mathcal{B}(\hat{u}_\theta(x_{bc}) \\ \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} \end{aligned} \tag{22}$$

By Schur product theorem, $\boldsymbol{K}^0$ is always Positive Semi-definite, hence it can be Eigen-decomposed to $\boldsymbol{Q}^T\Lambda\boldsymbol{Q}$, where $\boldsymbol{Q}$ is an orthogonal matrix and Λ is a diagonal matrix with eigenvalues λ_i in the entries. We can rearrange the training error in the form of:

$$\begin{aligned} \begin{bmatrix} \frac{du(x_b,\theta_t)}{dt} \\ \frac{d\mathcal{L}u(x_r,\theta_t)}{dt} \end{bmatrix} - \begin{bmatrix} \mathcal{B}(\hat{u}_\theta(x_{bc}) \\ \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} &\approx (I - e^{-\boldsymbol{K}^0 t}) \cdot \begin{bmatrix} \mathcal{B}(\hat{u}_\theta(x_{bc}) \\ \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} - \begin{bmatrix} \mathcal{B}(\hat{u}_\theta(x_{bc}) \\ \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} \\ &\approx -\boldsymbol{Q}^T e^{-\Lambda t}\boldsymbol{Q} \cdot \begin{bmatrix} \mathcal{B}(\hat{u}_\theta(x_{bc}) \\ \mathcal{D}(\hat{u}_\theta(x_r)) \end{bmatrix} \end{aligned} \tag{23}$$

where $e^{-\Lambda t} = \begin{bmatrix} e^{-\lambda_1 t} & & \\ & \ddots & \\ & & e^{-\lambda_N t} \end{bmatrix}$. This indicates the decrease of training error in each component is exponentially proportional to the eigenvalues of the deterministic NTK, and the NN is inherently biased to learn along larger eigenvalues entries of the $\boldsymbol{K}(0)$.

C. Proof of Lemma 2.1

Lemma 2.1. Consider a randomly sampled and normalised coordinate-based input $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$, $x \in [0,1]^d$, and its corresponding features in $\Phi : \mathbb{R}^d \to \mathbb{R}^m = [\varphi(x_1), \varphi(x_2), \cdots, \varphi(x_n)]^T$, let the feature mapping function $\varphi(x) = sin(2\pi\mathcal{B}\mathbf{x}) \in [-1,1]$, where $\mathcal{B}$ is sampled from a Gaussian distribution $\mathcal{N}(0,\sigma)$, the mapping function $sin(\cdot)$ is not bijective w.h.p.

Proof. Since $x \in [0,1]$, then $\Phi(x) \in [sin(0), sin(2\pi\mathcal{B})]$. Noting that $sin(\cdot)$ is only bijective on $(-\pi,\pi)$ or $sin(\cdot)$ is

bijective on $[0, 2\pi)$ only if $x \neq \pi$ with the domain limit. Hence we can derive the probability $\mathbb{P}$ of $sin(\cdot)$ is a bijection in following inequality form:

$$
\begin{aligned}
\mathbb{P} &< P(0 \leq \mathcal{B} < 1) \\
&< P(\mathcal{B} < 1) - P(\mathcal{B} < 0) \\
&< \int_{-\infty}^{x=1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \; - \int_{-\infty}^{x=0} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \\
&< \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x=1} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \; - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x=0} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \\
&\quad \text{Substitution} \quad \boxed{z = \frac{x-\mu}{\sqrt{2}\sigma}} \\
&< \frac{1}{\sqrt{2\pi\sigma^2}} \sqrt{2}\sigma \int_{-\infty}^{x=1} e^{-z^2}\, dz \; - \frac{1}{\sqrt{2\pi\sigma^2}} \sqrt{2}\sigma \int_{-\infty}^{x=0} e^{-z^2}\, dz \\
&< \frac{1}{\sqrt{2\pi\sigma^2}} \sqrt{2}\sigma \frac{\sqrt{\pi}}{2} erf(\frac{1}{\sqrt{2}\sigma}) - \frac{1}{\sqrt{2\pi\sigma^2}} \sqrt{2}\sigma \frac{\sqrt{\pi}}{2} erf(0) \\
&< \frac{1}{2} erf(\frac{1}{\sqrt{2}\sigma})
\end{aligned}
\tag{24}
$$

We calculated the upper bound of $\mathbb{P}$ for the $sin(\cdot)$ to be bijective is less than 0.5 as $\sigma \to 0$, and decreases as σ increases. Hence $sin(\cdot)$ is not bijective w.h.p is proved by contrapositive.

D. Proof of Proposition 3.1

Proposition 3.1 (Propagation of the Conjugate Kernel). Let input $x \in \mathcal{R}^{N \times n}$, and each layer of the Neural Network is parameterised with independent and identically distributed (i.i.d.) weights and biases from standard Gaussian distribution. Hence $f^l(x; \theta_0) \sim \mathcal{GP}(0, \Sigma^l(x, x'))$, and the Conjugate Kernels propagate through the Neural Network in the following recursive form:

$$
\begin{aligned}
\Sigma^1(x, x') &= < x, x' > + 1 \\
\Sigma^2(x, x') &= \mathbf{E}[\varphi(x)^T \varphi(x')] + 1 \\
\Sigma^l(x, x') &= \mathbf{E}[a(Z)^T a(Z')] + 1, \quad 2 < l \leq L
\end{aligned}
\tag{25}
$$

where φ is the feature mapping function at $l = 2$ and Z, Z' are the hidden layer state from previous layer and $\begin{bmatrix} Z \\ Z' \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma^{\ell-1}(x,x) & \Sigma^{\ell-1}(x',x) \\ \Sigma^{\ell-1}(x,x') & \Sigma^{\ell-1}(x',x') \end{bmatrix})$.

Proof.

Remark D.1. $X \sim \mathcal{N}(\mu_x, \Sigma_x)$ is equivalent to $X \sim \mu_x + \Sigma_x \mathcal{N}(0, 1)$. Hence if $Y = a + bX$, then $Y \sim \mathcal{N}(a + b\mu_x, b\Sigma_x b^T)$.

Recall Equation 3 and 4, the values of $f^{l=2}$ depend on the post feature mapping layer and the values of $f^{2<l<=L}$ depend on the previous layer. We treat each $f(x; \theta) = \frac{1}{n^l} \phi(x)\theta$, where $\phi(x)$ can represent the feature mapping function $\varphi(x)$ or activation function $a(x)$. Since $\theta \sim \mathcal{N}(0, 1)$, then $\mathbf{Var}[f(x; \theta)] = \phi(x)^T \phi(x) = K(x, x')$, as $w\phi(x) + b$ is a linear transformation.

Then the layers can be described as a Gaussian Process with mean 0 and covariance $K(x, x') = \phi(x)^T \phi(x) = \begin{cases} \frac{1}{n^{l-1}} \varphi(f^{l-1}(x; \theta_0))^T \varphi(f^{l-1}(x; \theta_0)), & l = 2 \\ \frac{1}{n^{l-1}} a(f^{l-1}(x; \theta_0))^T a(f^{l-1}(x; \theta_0)), & 2 < l <= L \end{cases}$.

The vector form f is a summation of its each components: $\frac{1}{n^{l-1}} \Phi(f(x; \theta_0))^T \Phi(f(x'; \theta_0)) = \frac{1}{n^{l-1}} \sum_{i=1}^{n^{l-1}} \Phi(f_i(x; \theta_0))^T \Phi(f_i(x'; \theta_0))$. Since the fs are independent, by applying the Law of Large Numbers, similarly in (Matthews et al., 2018), we can get $\frac{1}{n^{l-1}} \phi(f(x; \theta_0))^T \phi(f(x'; \theta_0)) = \mathbf{E}[\phi(f(x; \theta_0))^T \phi(f(x'; \theta_0))] = \mathbf{E}[\phi(Z)^T \phi(Z')]$

where $\mathbf{Cov}(Z, Z') = \begin{bmatrix} \Sigma^{\ell-1}(x,x) & \Sigma^{\ell-1}(x',x) \\ \Sigma^{\ell-1}(x,x') & \Sigma^{\ell-1}(x',x') \end{bmatrix}$.

E. Proof of Theorem 3.2

Theorem 3.2 (Evolution of the NTK with CK). Let input $x \in \mathcal{R}^{N\times n}$, $\phi(x) = \varphi(x) \bigvee a(x)$; Recall $\Sigma^2(x,x') = \mathbf{E}[\varphi(x)^T\varphi(x')] + 1$, $\Sigma^l(x,x') = \mathbf{E}[\phi(x)\phi(x')] + 1$ and its derivative is $\dot{\Sigma}^l(x,x') = \mathbf{E}[\dot{\phi}(x)\dot{\phi}(x')], \in R^{N\times N}$. Assuming the infinity width limit, the gradient ∇f^l satisfies:

$$\nabla_\theta f^l(x;\theta_0)^T \nabla_\theta f^l(x';\theta_0) \to \Theta^l(x,x') \tag{26}$$

The evolution of the kernels follows:

$$\begin{aligned} \Theta^2(x,x') &= \Theta^1(x,x')\dot{\Sigma}^2(x,x') + \Sigma^2(x,x') \\ \Theta^l(x,x') &= \Theta^{l-1}(x,x')\dot{\Sigma}^l(x,x') + \Sigma^l(x,x'), \quad 2 < l \leq L \end{aligned} \tag{27}$$

Proof. Since the NTK involves derivative with the θ, we need to consider the θs in both the previous layer and the current layer, Thus we formulate $\theta^l = \theta^{l-1} \cup \theta^{l*} = \theta^{l-1} \cup \{w^l, b^l\}$, which gives $f^l(x;\theta^l) = \frac{1}{\sqrt{n^{l-1}}} w^l \phi(f^{l-1}(x;\theta^{l-1})) + b^l$. With the new notation, we can split the derivatives by partial differention rules:

$$\begin{aligned} \nabla_{\theta^l} f^l(x;\theta^l)^T \nabla_{\theta^l} f^l(x;\theta^l) &= \nabla_{\theta^{l*}} f^l(x;\theta^l)^T \nabla_{\theta^{l*}} f^l(x;\theta^l) + \nabla_{\theta^{l-1}} f^l(x;\theta^l)^T \nabla_{\theta^{l-1}} f^l(x;\theta^l) \\ &= \frac{1}{n^{l-1}} \phi(f^{l-1}(x;\theta^{l-1}))^T \phi(f^{l-1}(x;\theta^{l-1})) + \nabla_{\theta^{l-1}} f^l(x;\theta^l)^T \nabla_{\theta^{l-1}} f^l(x;\theta^l) \\ &= \begin{cases} \Sigma^2(x,x') \\ \Sigma^l(x,x'), \quad 2 < l <= L \end{cases} + \nabla_{\theta^{l-1}} f^l(x;\theta^l)^T \nabla_{\theta^{l-1}} f^l(x;\theta^l) \end{aligned} \tag{28}$$

The first part of the partial differention becomes precisely the Conjugate Kernel that is derived from Proposition 3.1. The remaining part can be solved by chain rule.

$$\begin{aligned} \nabla_{\theta^{l-1}} f^l(x;\theta^l)^T \nabla_{\theta^{l-1}} f^l(x;\theta^l) &= \frac{1}{\sqrt{n^{l-1}}} w^l \nabla_{\theta^{l-1}} \phi(f^{l-1}(x;\theta^{l-1}))^T \frac{1}{\sqrt{n^{l-1}}} w^l \nabla_{\theta^{l-1}} \phi(f^{l-1}(x;\theta^{l-1})) \\ &= \frac{1}{\sqrt{n^{l-1}}} w^l diag[\phi'(f^{l-1}(x;\theta^{l-1}))] \nabla_{\theta^{l-1}} (f^{l-1}(x;\theta^{l-1}))^T \\ &\quad \frac{1}{\sqrt{n^{l-1}}} w^l diag[\phi'(f^{l-1}(x;\theta^{l-1}))] \nabla_{\theta^{l-1}} (f^{l-1}(x;\theta^{l-1})) \\ &= \frac{1}{n^{l-1}} w^l diag[\phi'(f^{l-1}(x;\theta^{l-1}))] \underbrace{(\nabla_{\theta^{l-1}} (f^{l-1}(x;\theta^{l-1}))^T \nabla_{\theta^{l-1}} (f^{l-1}(x;\theta^{l-1})))}_{\text{NTK}} \\ &\quad diag[\phi'(f^{l-1}(x;\theta^{l-1}))] w^{l^T} \\ &= \frac{1}{n^{l-1}} \sum_i^{n^{l-1}} w_i^l \phi'(f^{l-1}(x;\theta^{l-1})) \Theta^{l-1} \phi'(f^{l-1}(x;\theta^{l-1})) w_i^{l^T} \\ &= \Theta^{l-1} \frac{1}{n^{l-1}} \sum_i^{n^{l-1}} w_i^l \underbrace{\phi'(f^{l-1}(x;\theta^{l-1}))\phi'(f^{l-1}(x;\theta^{l-1}))}_{\text{derivative of CK}} w_i^{l^T} \\ &= \begin{cases} \Theta^2(x,x') = \Theta^1(x,x')\dot{\Sigma}^2(x,x') \\ \Theta^l(x,x') = \Theta^{l-1}(x,x')\dot{\Sigma}^l(x,x'), 2 < l <= L \end{cases} \end{aligned} \tag{29}$$

F. Different feature mapping methods in MLP

Basic Encoding: (Mildenhall et al., 2020) $\varphi(x) = [cos(2\pi\sigma^{j/m}x, sin(2\pi\sigma^{j/m}x]^T$ for $j = 0, .., m-1$.
Positional Encoding: (Mildenhall et al., 2020) $\varphi(x) = [cos(2\pi\sigma^{j/m}x, sin(2\pi\sigma^{j/m}x]^T$ for $j = 0, .., m-1$.
Random Fourier: (Tancik et al., 2020) $\varphi(x) = [cos(2\pi\sigma\mathcal{B}x), sin(2\pi\sigma\mathcal{B}x)]^T$, where $\mathcal{B} \in \mathbb{R}^{m\times d}$ is sampled from $\mathcal{N}(0, 1)$ and σ is an arbitrary scaling factor varies case to case.
Sinusoidal Feature: (Sitzmann et al., 2020) $\varphi(x) = [sin(2\pi\mathbf{W}x + \mathbf{b})]^T$, where $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters.
Complex Triangle: (Zheng et al., 2022) $\varphi(x) = [max(1 - \frac{|x_1-t|}{0.5d}, 0), max(1 - \frac{|x_2-t|}{0.5d}, 0), \cdots, max(1 - \frac{|x_i-t|}{0.5d}, 0)]^T$, where t is uniformly sampled from 0 to 1.
Complex Gaussian: (Zheng et al., 2022) $\varphi(x) = [e^{-0.5(x_1-\tau/d)^2/\sigma^2} \bigotimes \cdots \bigotimes e^{-0.5(x_d-\tau/d)^2/\sigma^2}]^T$, where τ is uniformly sampled from $[0, 1]$, and $\bigotimes$ is the Kronecker product.

G. Complete experimental results for Table 1&2

G.1. Complete results for Table 1

Table 5. Full PDEs benchmark results comparing different feature mapping methods in $\ell 2$ error. The best results are in Blue. Standard deviations are shown after $\pm$.

	PINN	BE	PE	FF	SF
Wave	3.731e-1±2.369e-2	1.035e0±3.548e-1	1.014e0±4.019e-1	2.375e-3±3.751e-4	7.932e-3±9.321e-4
Diffusion	1.426e-4±4.841e-5	1.575e-1±6.128e-2	1.595e-1±1.204e-2	2.334e-3±7.514e-4	3.474e-4±6.107e-5
Heat	4.732e-3±6.140e-5	6.491e-3±6.365e-4	7.574e-3±1.025e-4	2.190e-3±3.125e-4	3.961e-3±2.568e-4
Poisson	3.618e-3±1.236e-4	4.964e-1±2.146e-2	4.910e-1±1.084e-2	7.586e-4±9.013e-5	9.078e-4±1.024e-5
Burgers	1.864e-3±1.204e-4	5.585e-1±2.578e-2	5.363e-1±3.698e-2	7.496e-2±5.147e-3	1.299e-3±6.210e-4
Steady NS	5.264e-1±1.013e-2	7.143e-1±1.325e-2	6.332e-1±2.345e-2	6.939e-1±1.064e-3	3.769e-1±2.367e-2

	CT	CG	**RBF-INT**	**RBF-POL**
Wave	1.114e0±3.214e-2	1.036e0±1.054e-2	2.814e-2±3.647e-3	2.361e-2±1.598e-2
Diffusion	1.860e0±2.312e-2	2.721e-2±1.027e-1	3.066e-4±9.517e-6	3.498e-5±6.547e-6
Heat	4.524e-1±6.514e-2	2.626e-1±2.367e-2	1.157e-3±1.020e-4	4.098e-4±9.621e-6
Poisson	6.348e-1±3.049e-1	2.334e-1±5.471e-2	5.259e-4±6.243e-5	8.942e-4± 6.514e-5
Burgers	9.935e-1±4.512e-2	7.521e-1±3.249e-2	2.945e-3±2.354e-4	3.159e-4±2.146e-5
Steady NS	5.460e-1±2.357e-2	4.867e-1±3.654e-2	2.991e-1±6.514e-2	2.567e-1±6.217e-2

G.2. Complete results for Table 2

Table 6. Full Benchmark results on the Inverse problems in $\ell 2$ error. * indicates problems with noises added to the data.

	FF	SF	**RBF-INT**	**RBF-POL**
I-Burgers	2.391e-2±9.647e-4	2.436e-2±4.678e-3	1.741e-2±6.571e-3	1.575e-2±9.369e-4
I-Lorenz	6.516e-3±7.651e-4	6.390e-3±6.214e-4	6.080e-3±3.697e-4	5.991e-3±2.312e-4
I-Burgers*	2.509e-2±6.324e-3	2.913e-2±2.698e-3	1.993e-2±3.621e-3	1.753e-2±5.632e-3
I-Lorenz*	7.934e-3±8.651e-4	6.856e-3±6.363e-4	6.699e-3±5.201e-4	6.342e-3±8.614e-4

H. Ablation Study

In this section, we show some additional experiments on our RBF feature mapping including investigations on the Number of RBFs, Number of Polynomials and different RBF types.

H.1. Number of RBFs

Figure 6 has shown generally more RBFs (256) yield better results. It however does demand a higher memory and can be slow in some cases. It shows in the Diffusion equation, with 256 RBFs, the error reduces quite significantly. Otherwise, it only has limited improvements because the error is already very low. We use 128 RBFs in general case for a better performance-speed tradeoff.

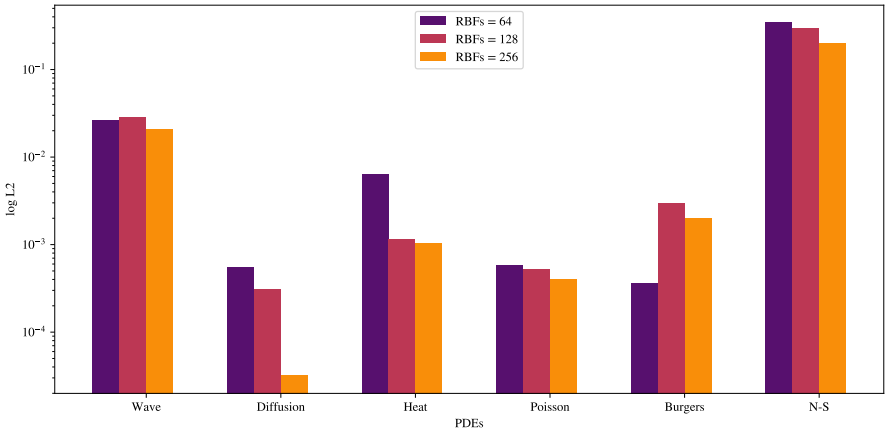


Figure 6. Ablation study on different number of RBFs

H.2. Number of Polynomials

Figure 9 shows an ablation study of how the number of polynomials in feature mappings influences performance in PDEs. It has shown RBF feature mapping with 20 polynomials has achieved best results in the Diffusion equation, Poisson equation and N-S equation. And 10 polynomial terms are better in Heat equation and Burgers equation, thought its performance is matching with only 5 polynomials.

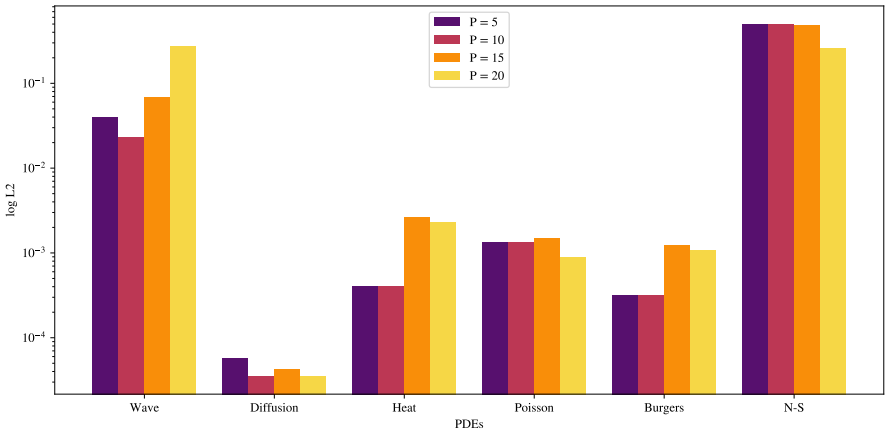


Figure 7. Ablation study on different number of polynomials

H.3. Different Types of RBFs

Following Table 7 are common positive definite Radial Basis Functions.

Table 7. Types of Radial Basis function and their formulation. $\mathbf{x} - \mathbf{c}$ is shorten as r.

Type	Radial function
Cubic	r^3
TPS(Thin Plate Spline)	$r^2 log(r)$
GA(Gaussian)	e^{-r^2/σ^2}
MQ(Multiquadric)	$\sqrt{1+r^2}$
IMQ(Inverse MQ)	$1/\sqrt{1+r^2}$

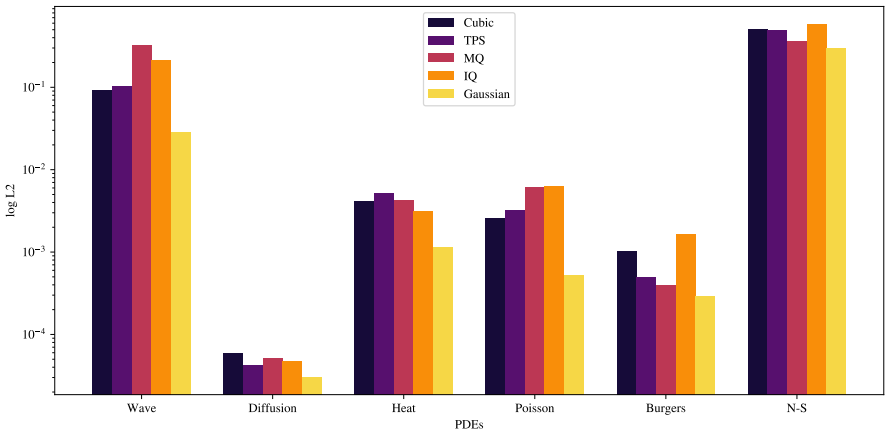


Figure 8. Ablation study on different types of RBFs

The Figure 8 has shown Gaussian RBF is dominating all types of PDEs. However other types of RBF are in similar performance. We generally prefer Gaussian RBF in all cases due to its nice properties.

I. Complexity and Scalability analysis

Although all feature mapping methods are similar in computational complexity, for completeness, we include the complexity of the feature layers that map 128 features and 4 fully connected layers with 50 neurons each.

Table 8. Computational complexity

	FF	SF	RBF-INT	RBF-POL-5	RBF-POL-10	RBF-POL-15	RBF-POL-20
FLOPs	139.5M	142.1M	139.5M	142.5M	145.0M	147.5M	150.0M
Params	14.2k	14.3k	14.2k	14.5k	14.7k	14.9k	15.2k

Due to software optimisation and package compatibility, the feature mapping methods can have very different computational efficiency in training. To demonstrate, we run the above models on different numbers of sample points on Diffusion equation for 3 times in different random seeds.RBF-COM stands for compact support RBF, and RBF-POL uses 20 polynomials. The time consumed by Fourier Features is noticeably higher than other methods. All methods have similar runtime for sample points less than $1e4$, that is because all sample points computed are within a GPU parallelisation capacity.

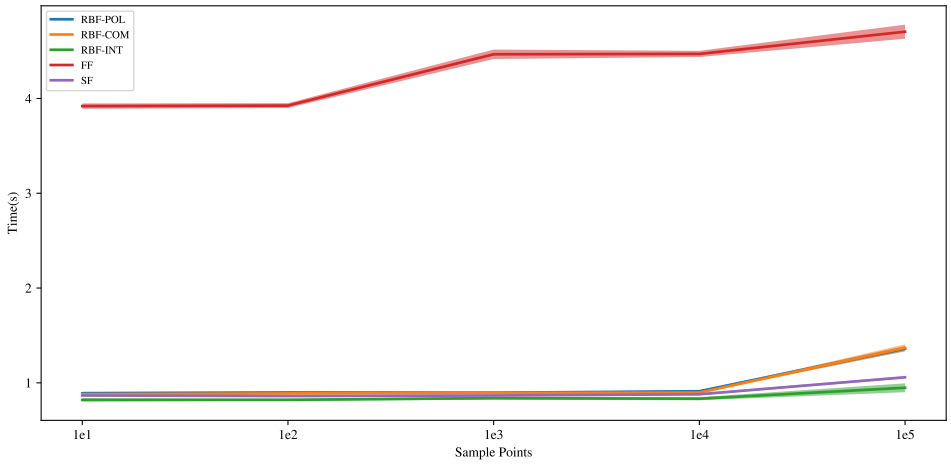


Figure 9. Time consumption on different numbers of sample points with different feature mapping methods

J. Benchmark PDEs and Boundary conditions

J.1. Wave Equation

The one-dimensional Wave Equation is given by:

$$u_{tt} - 4u_{xx} = 0. \tag{30}$$

In the domain of:

$$(x,t) \in \Omega \times T = [-1,1] \times [0,1]. \tag{31}$$

Boundary condition:

$$u(0,t) = u(1,t) = 0. \tag{32}$$

Initial condition:

$$u(x,0) = \sin(\pi x) + \frac{1}{2}\sin(4\pi x) \tag{33}$$

$$u_t = 0 \tag{34}$$

$$\tag{35}$$

The analytical solution of the equation is:

$$u(x,t) = \sin(\pi x)cos(2\pi t) + \frac{1}{2}\sin(4\pi x)cos(8\pi t). \tag{36}$$

J.2. Diffusion Equation

The one-dimensional Diffusion Equation is given by:

$$u_t - u_{xx} + e^{-t}(sin(\pi x) + \pi^2 sin(\pi x)) = 0 \tag{37}$$

In the domain of:

$$(x,t) \in \Omega \times T = [-1,1] \times [0,1]. \tag{38}$$

Boundary condition:

$$u(-1,t) = u(1,t) = 0 \tag{39}$$

Initial condition:

$$u(x,0) = sin(\pi x) \tag{40}$$

The analytical solution of the equation is:

$$u(x,t) = e^t sin(\pi x) \tag{41}$$

where $\alpha = 0.4, L = 1, n = 1$

J.3. Heat Equation

The two-dimensional Heat Equation is given by:

$$u_t - \frac{1}{(500\pi)^2} u_{xx} - \frac{1}{\pi^2} u_{yy} = 0. \tag{42}$$

In the domain of:

$$(\mathbf{x}, t) \in \Omega \times T = [0, 1]^2 \times [0, 5]. \tag{43}$$

Boundary condition:

$$u(x, y, t) = 0. \tag{44}$$

Initial condition:

$$u(x, y, 0) = \sin(20\pi x) \sin(\pi y). \tag{45}$$

J.4. Poisson Equation

The two-dimensional Poisson Equation is given by:

$$-\Delta u = 0 \tag{46}$$

In the domain of:

$$\mathbf{x} \in \Omega = \Omega_{rec} \backslash R_i. \tag{47}$$

where

$$\Omega_{rec} = [-0.5, 0.5]^2, \tag{48}$$
$$R_1 = [(x, y) : (x - 0.3)^2 + (y - 0.3)^2 \leq 0.1^2], \tag{49}$$
$$R_2 = [(x, y) : (x + 0.3)^2 + (y - 0.3)^2 \leq 0.1^2], \tag{50}$$
$$R_3 = [(x, y) : (x - 0.3)^2 + (y + 0.3)^2 \leq 0.1^2], \tag{51}$$
$$R_4 = [(x, y) : (x + 0.3)^2 + (y + 0.3)^2 \leq 0.1^2]. \tag{52}$$

Boundary condition:

$$u = 0, x \in \partial R_i, \tag{53}$$
$$u = 1, x \in \partial \Omega_{rec}. \tag{54}$$

J.5. Burgers Equation

The one-dimensional Burgers Equation is given by:

$$u_t + u u_x = \nu u_{xx} \tag{55}$$

In the domain of:

$$(x, t) \in \Omega = [-1, 1] \times [0, 1]. \tag{56}$$

Boundary condition:

$$u(-1, t) = u(1, t) = 0. \tag{57}$$

Initial condition:

$$u(x, 0) = -\sin \pi x \tag{58}$$

where $\nu = \frac{0.01}{\pi}$

J.6. Steady NS

The steady incompressible Navier Stokes Equation is given by:

$$\nabla \cdot \mathbf{u} = 0, \tag{59}$$

$$\mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}} \Delta \mathbf{u} = 0. \tag{60}$$

$$\tag{61}$$

In the domain(back step flow) of:

$$\mathbf{x} \in \Omega = [0, 4] \times [0, 2] \backslash ([0, 2] \times [1, 2] \cup R_i) \tag{62}$$

Boundary condition:

$$\text{no-slip condition:} \quad \mathbf{u} = 0. \tag{63}$$

$$\text{inlet:} \quad u_x = 4y(1 - y), u_y = 0. \tag{64}$$

$$\text{outlet:} \quad p = 0. \tag{65}$$

where $Re = 100$

J.7. nD Poisson Equation

The nth-dimensional Poisson Equation is given by:

$$-\Delta u = \frac{\pi^2}{4} \sum_{i=1}^{n} \sin\left(\frac{\pi}{2} x_i\right) \tag{66}$$

In the domain of:

$$x \in \Omega = [0, 1]^n \tag{67}$$

Boundary condition:

$$u = 0 \tag{68}$$

The analytical solution of the equation is:

$$u = \sum_{i=1}^{n} \sin\left(\frac{\pi}{2} x_i\right) \tag{69}$$

J.8. Inverse Burgers Equation

The one-dimensional Inverse Burgers Equation is given by:

$$u_t + \mu_1 u u_x = \mu_2 u_{xx} \tag{70}$$

In the domain of:

$$(x, t) \in \Omega = [-1, 1] \times [0, 1]. \tag{71}$$

Boundary condition:

$$u(-1, t) = u(1, t) = 0. \tag{72}$$

Initial condition:

$$u(x, 0) = -\sin \pi x \tag{73}$$

where $\mu_1 = 1$ and $\mu_2 = \frac{0.01}{\pi}$

J.9. Inverse Lorenz Equation

The 1st-order three-dimensional Lorenz Equation is given by:

$$\begin{aligned} \frac{dx}{dt} &= \alpha(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z, \end{aligned} \tag{74}$$

where $\alpha = 10$, $\beta = \frac{8}{3}$, $\rho = 15$ and the initial points are $x_0 = 0$, $y_0 = 1$, $z_0 = 1.05$.

K. Visualisations of PDEs solution

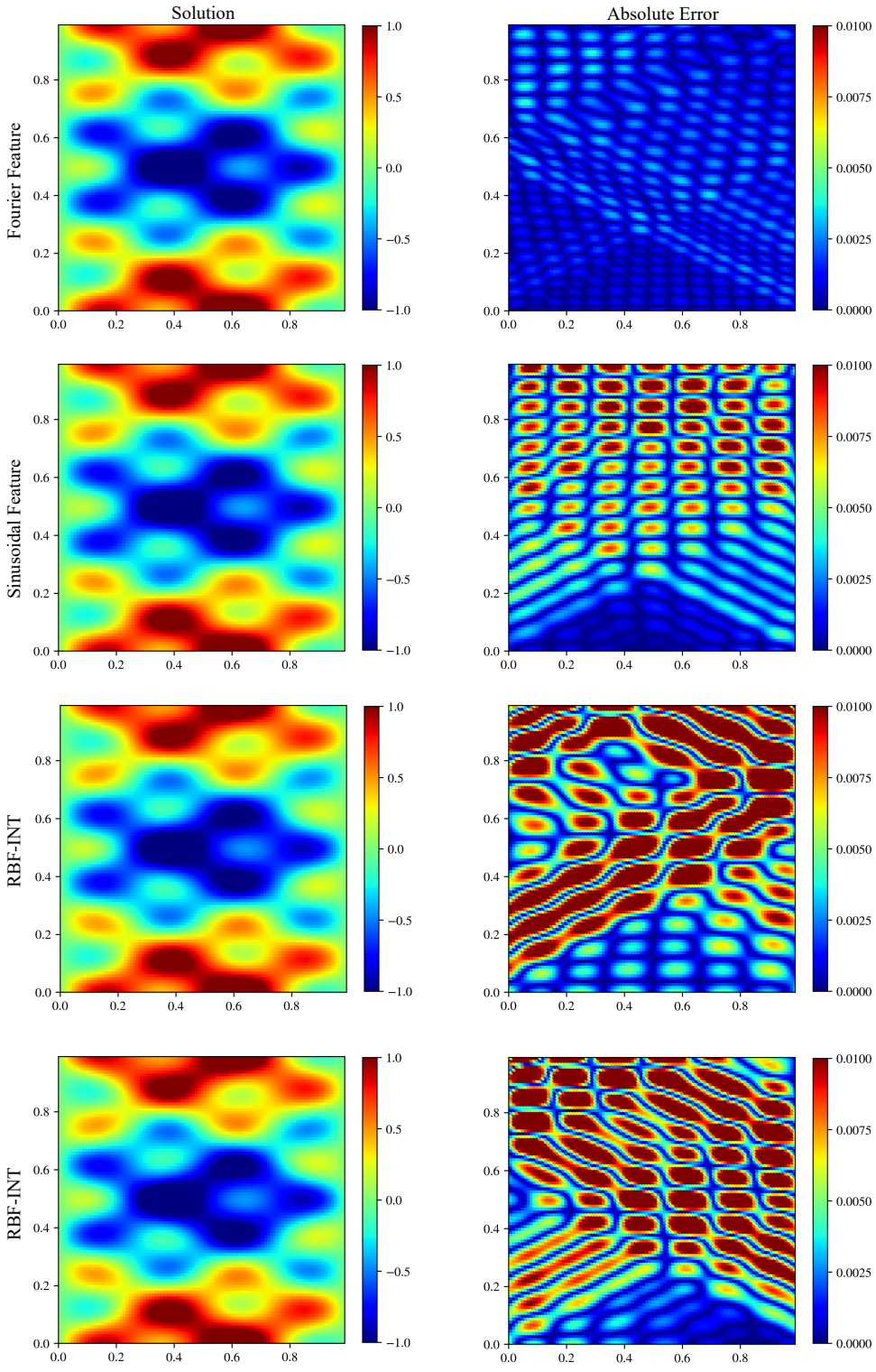


Figure 10. Wave equation

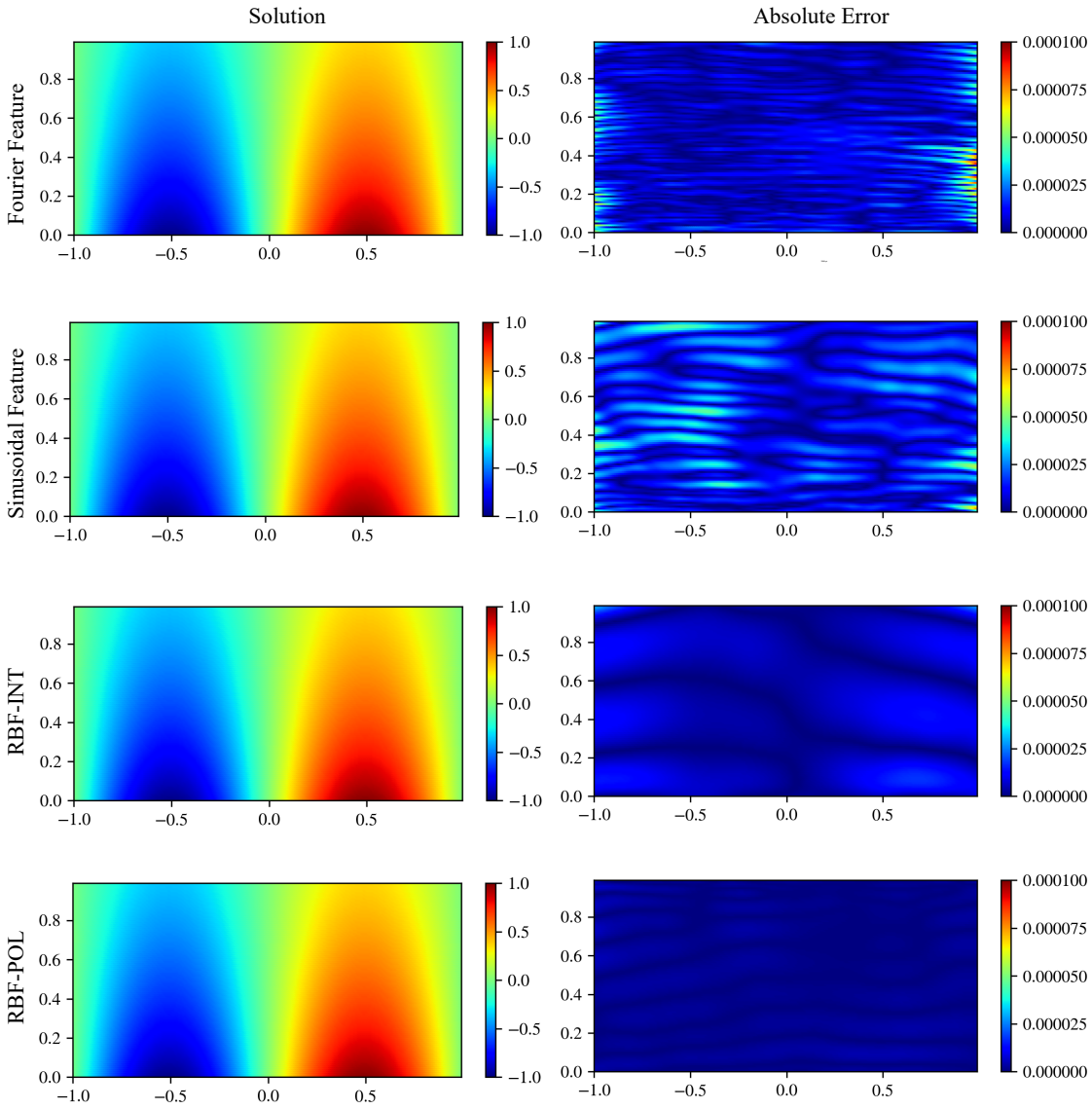


Figure 11. Diffusion equation

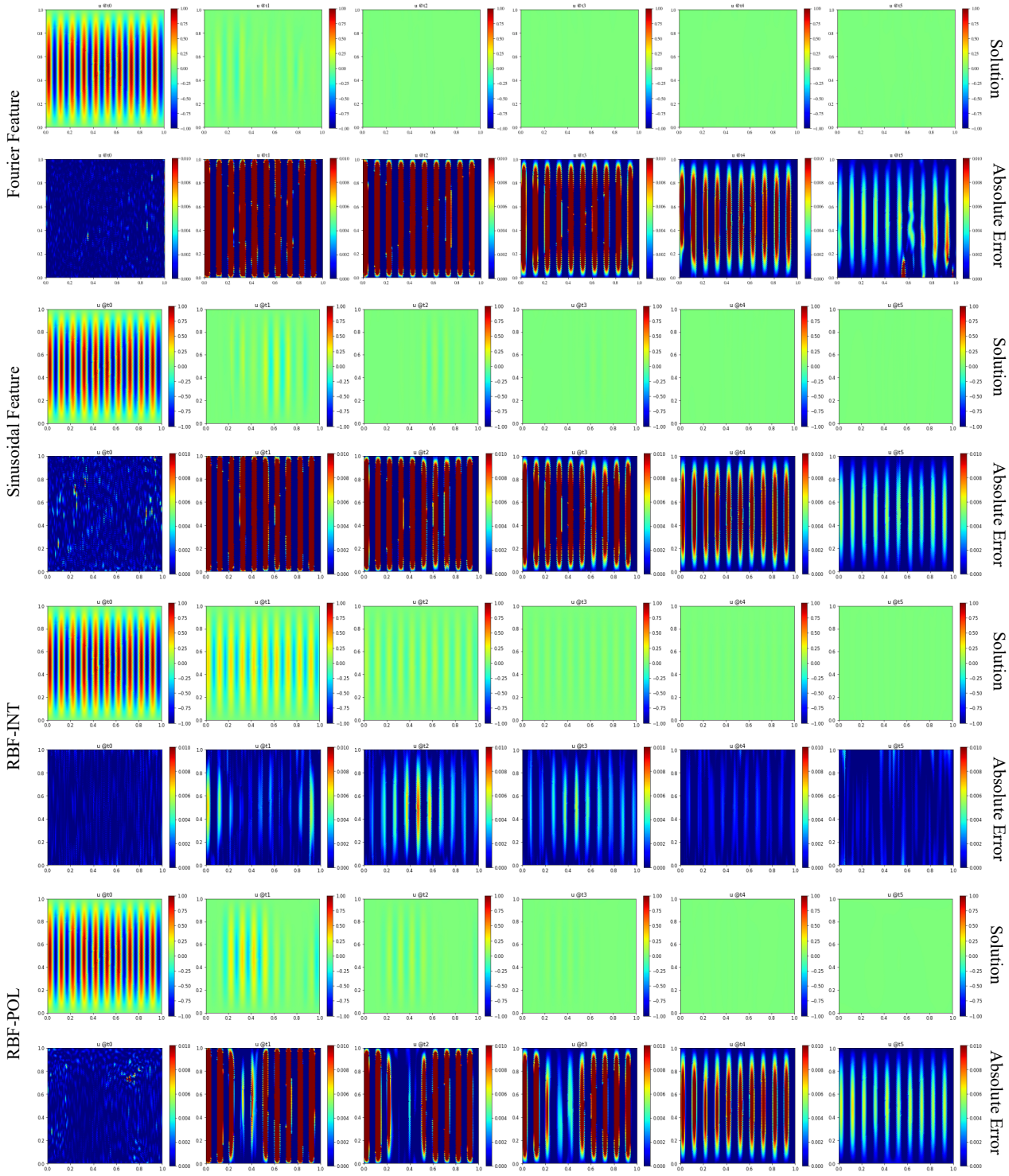


Figure 12. Heat equation

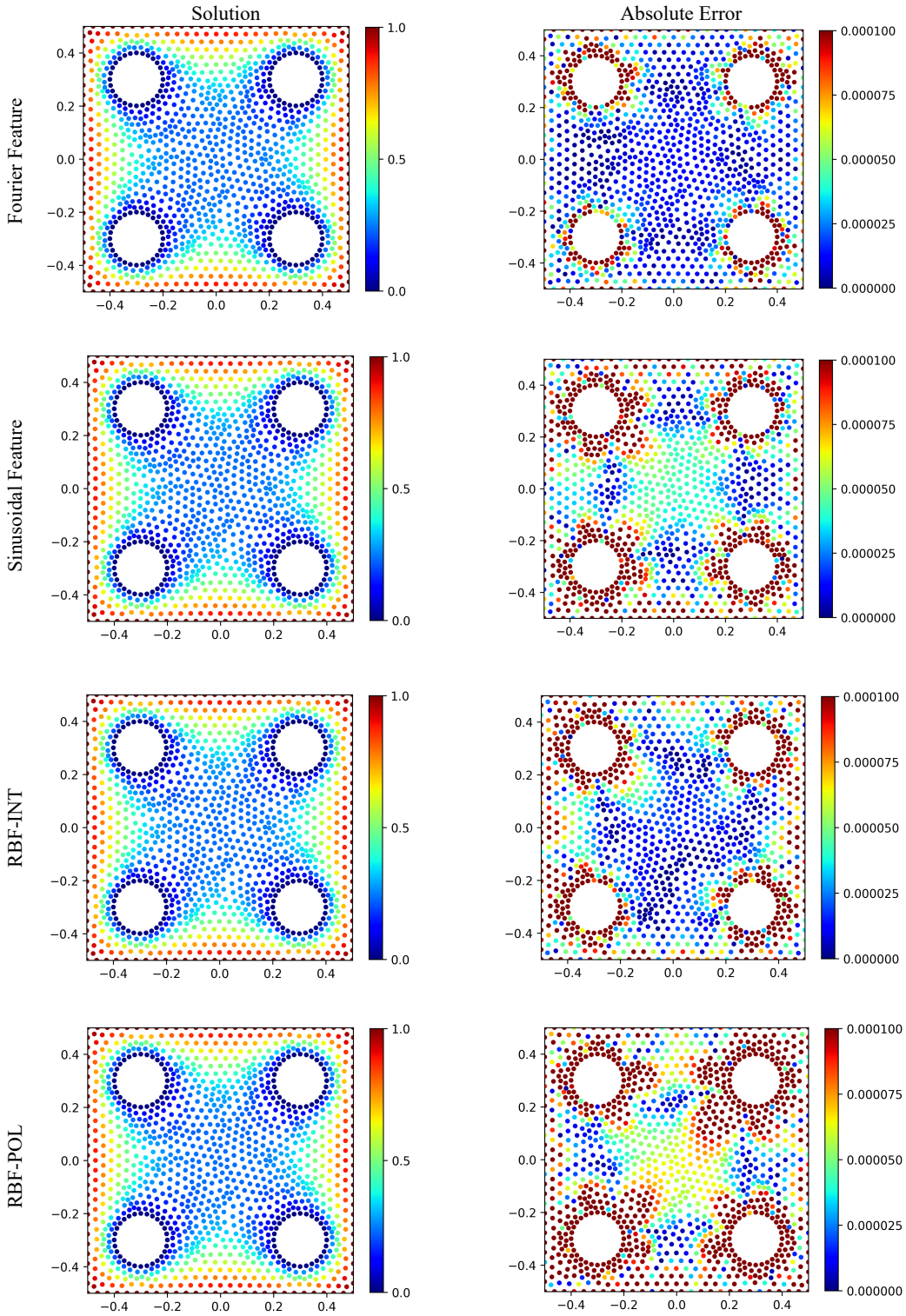


Figure 13. Poisson equation

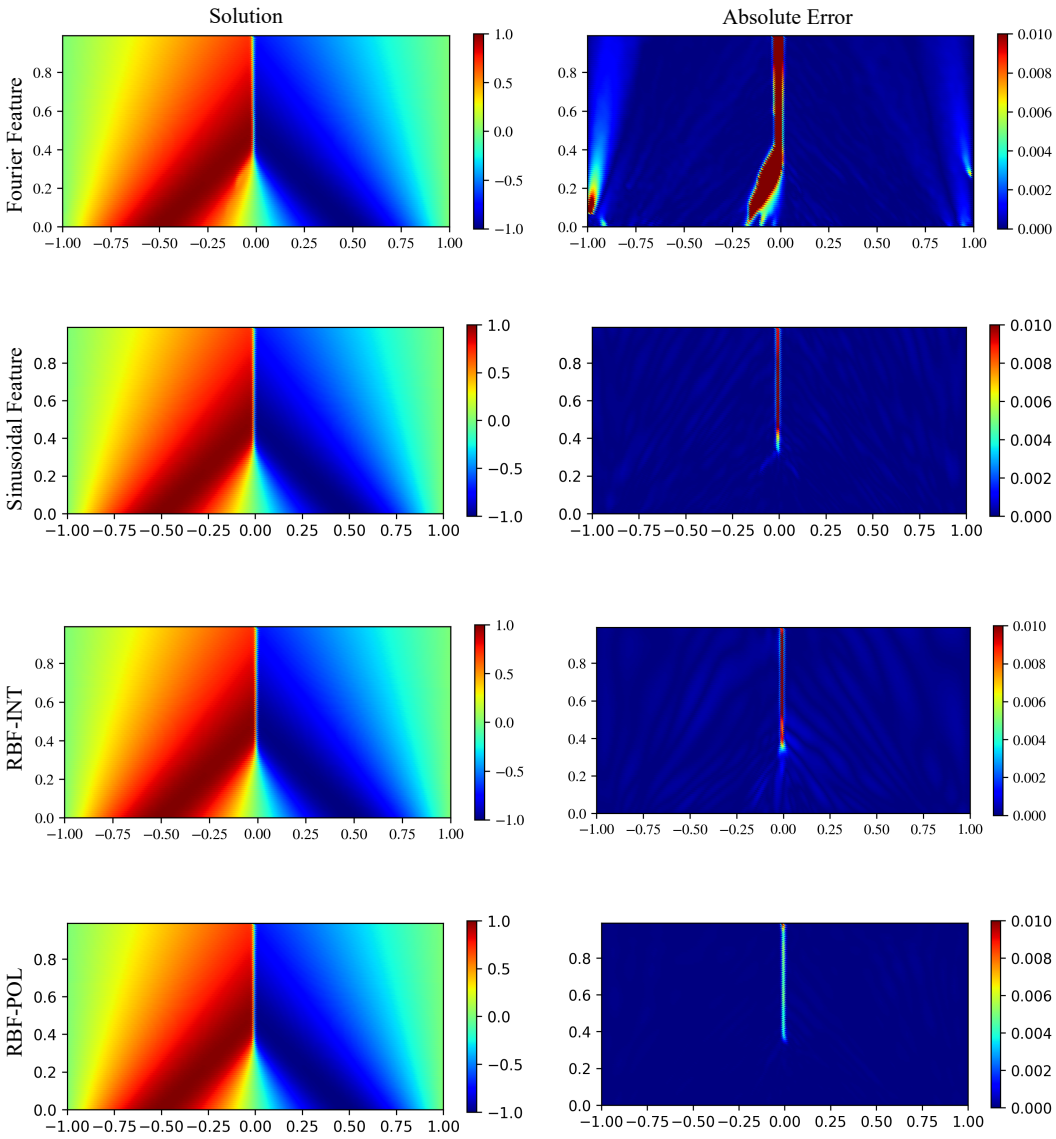


Figure 14. Burgers equation

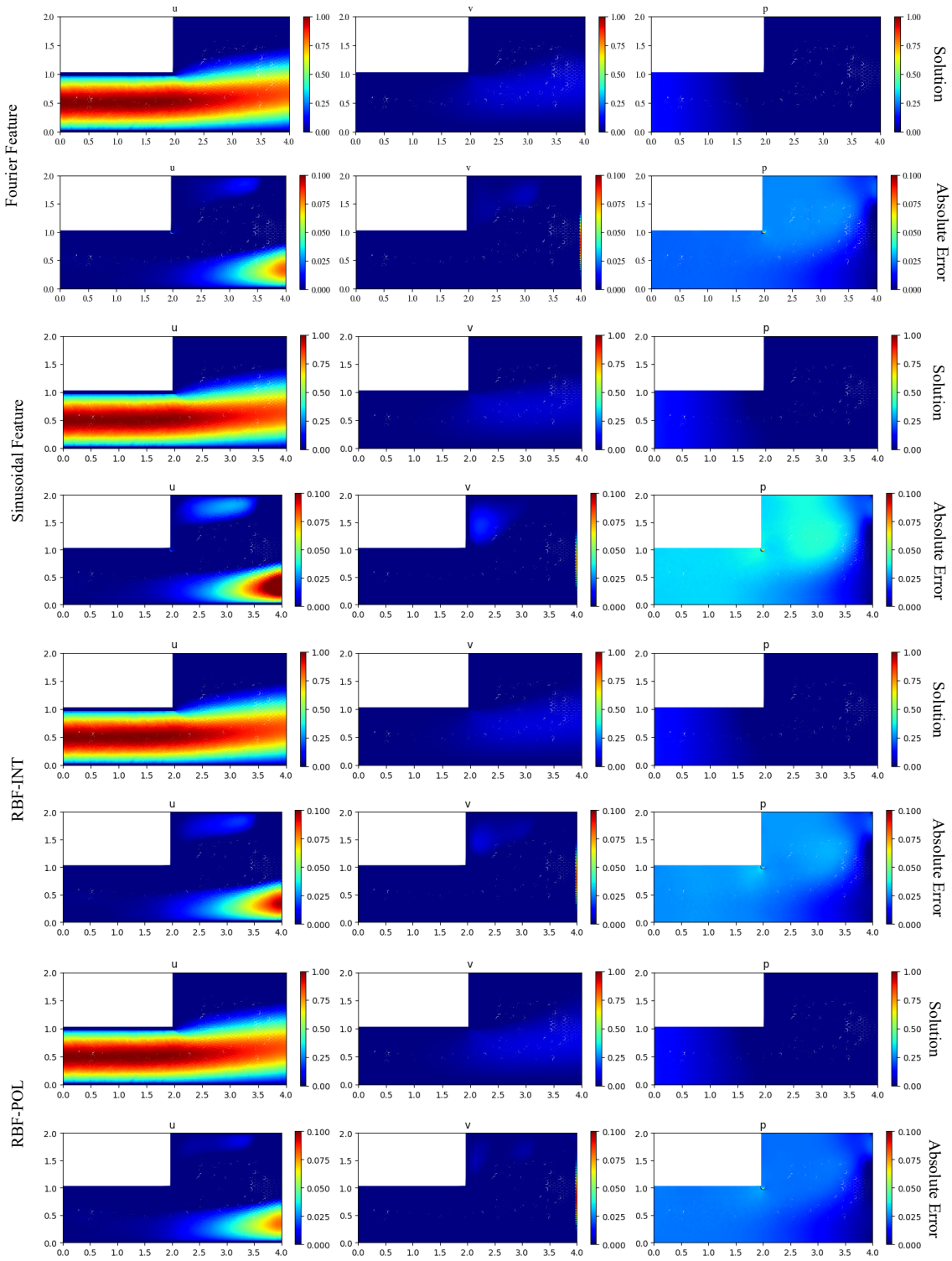


Figure 15. Navier-Stokes equation

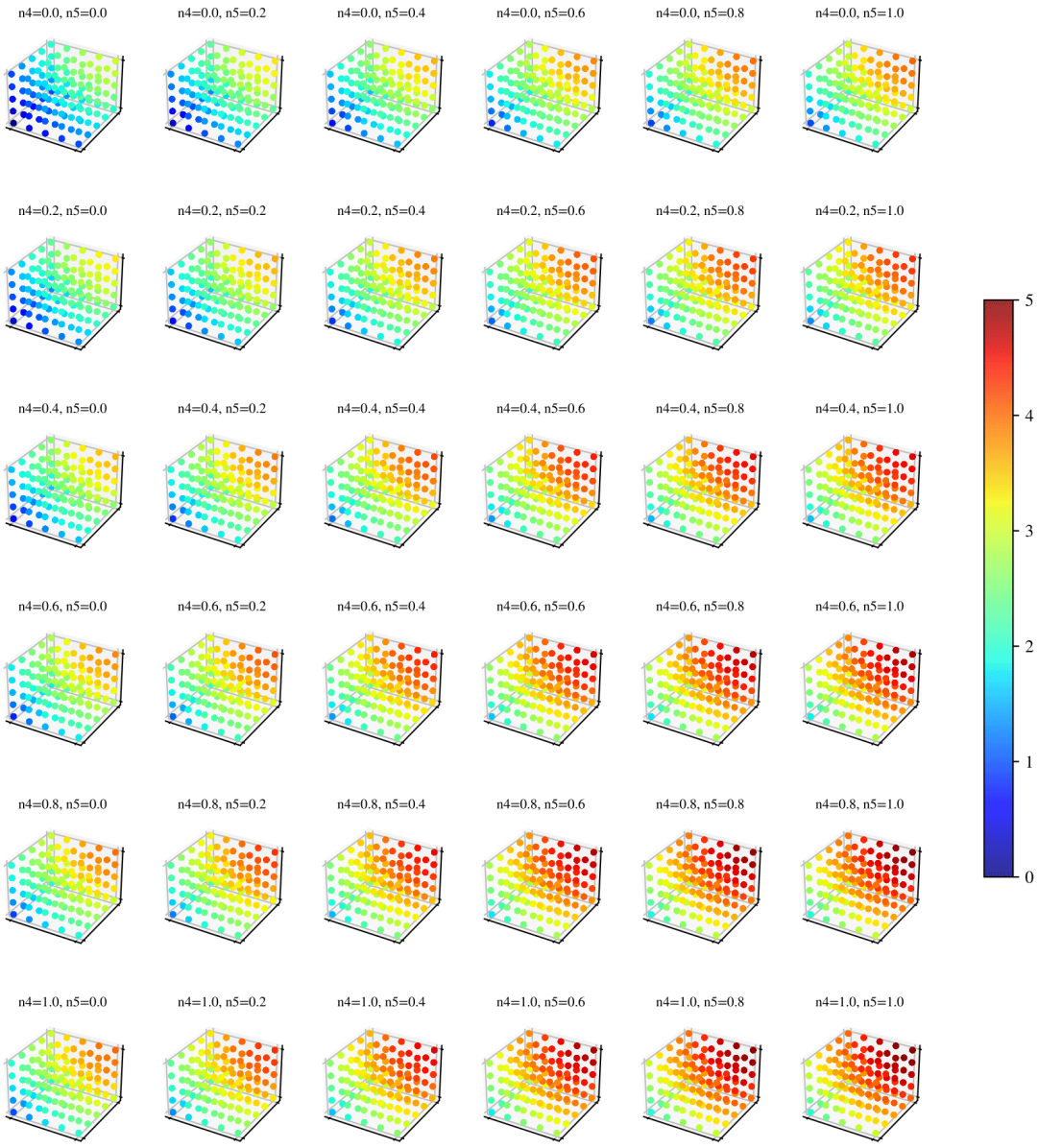


Figure 16. nD Poisson equation Ground Truth, when n = 5, the x,y,z direction of the cube is the 1st, 2nd and 3rd dimension, respectively, the rows direction of the image is the 4th dimension and the column direction of the image is the 5th dimension.

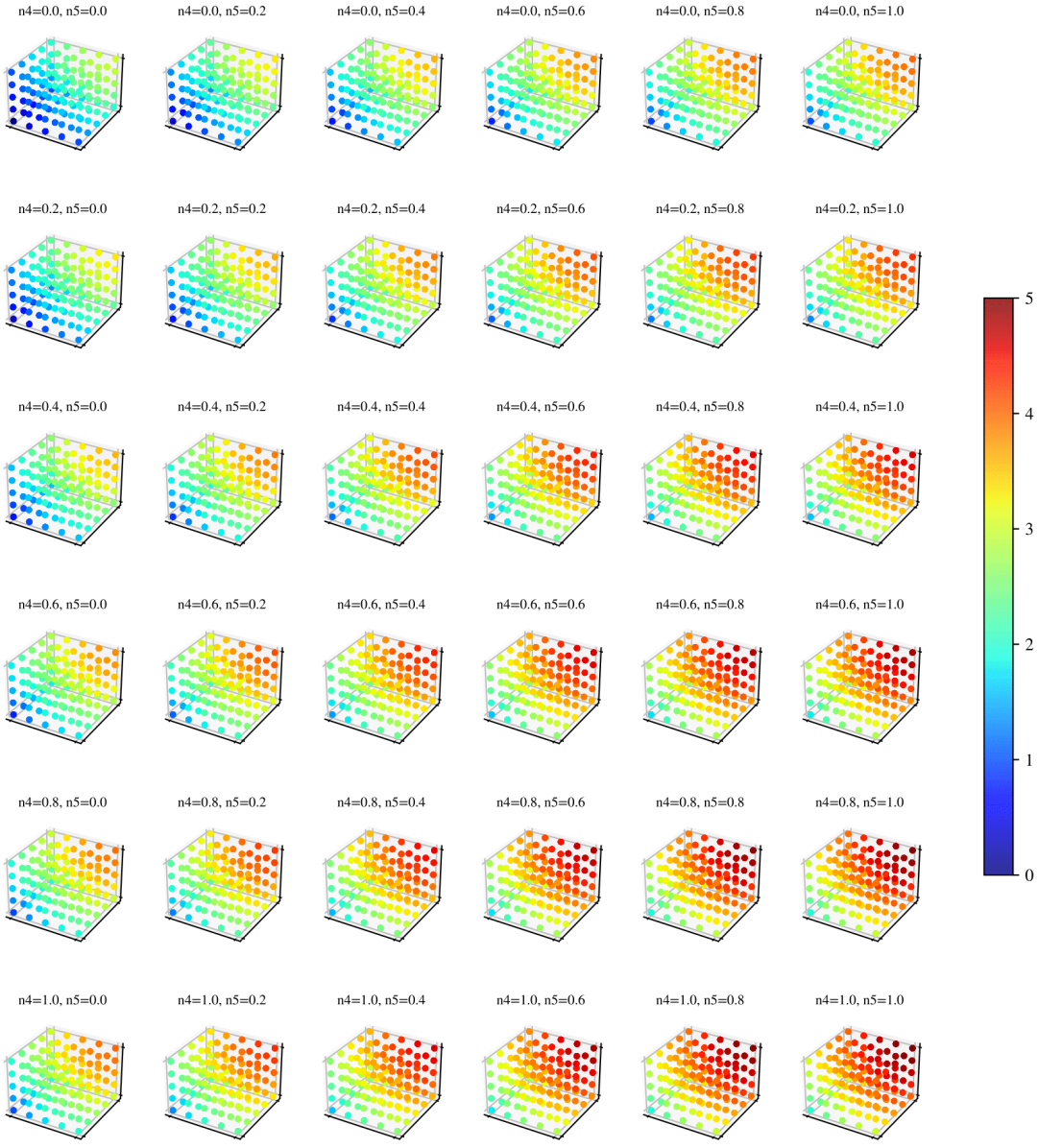


Figure 17. 5D Poisson Equation solved by RBF feature mapping.

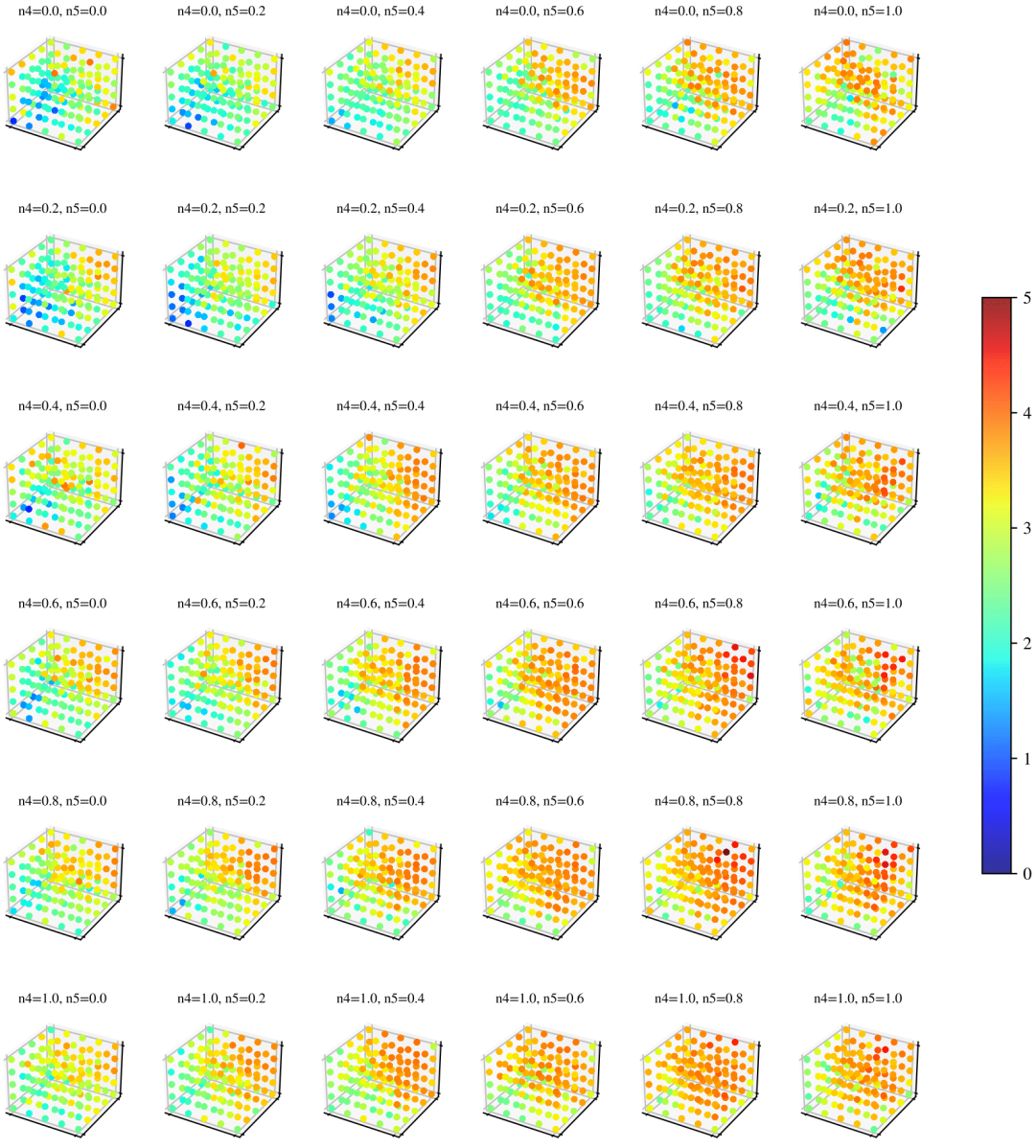


Figure 18. 5D Poisson Equation solved by Fourier feature mapping.