Numerical Methods For PDEs Over Manifolds Using Spectral Physics Informed Neural Networks

Yuval Zelig Shai Dekel

School of Mathematical Sciences, Tel Aviv University
February 13, 2023

Abstract

We introduce an approach for solving PDEs over manifolds using physics informed neural networks whose architecture aligns with spectral methods. The networks are trained to take in as input samples of an initial condition, a time stamp and point(s) on the manifold and then output the solution's value at the given time and point(s). We provide proofs of our method for the heat equation on the interval and examples of unique network architectures that are adapted to nonlinear equations on the sphere and the torus. We also show that our spectral-inspired neural network architectures outperform the standard physics informed architectures. Our extensive experimental results include generalization studies where the testing dataset of initial conditions is randomly sampled from a significantly larger space than the training set.

1 Introduction

Time dependent differential equations are a basic tool for understanding many processes in physics, chemistry, biology, economy and any field requires the analyze of time-dependent dynamical process. Therefore, solving those equations is an active area of research [1, 2]. For many of those equations, an analytical solution does not exist and a numerical method must be used.

Numerical methods such as finite differences and finite elements methods are applied successfully in many scenarios, however when the PDEs are given on manifolds, discretization processes and application of time steps can become challenging.

In recent years, there is an emergence of machine learning methods and most notably Physics Informed (PI) deep learning models [4],[20]. Physics Informed Neural Networks (PINN) are designed to solve partial differential equations and inverse problems by enforcing the networks to approximately obey the given governing equations by using corresponding loss functions during the training phase. This technique allows to obtain relatively high quality approximation with relatively small datasets. There are various neural network architectures that have been developed for this purpose, with different settings and strategies such as automation differentiation [18], numerical schemes [5], grid-free [3, 4] or grid-dependent approaches [5], and the ability to handle different geometries [19]. In this paper, we present a generalization of spectral based deep learning methods for PDEs [23],[24],[25],[26]:

(i) We provide a physics informed deep learning approach that can handle the general case of differential equations over compact Riemannian manifolds.

(ii) The design of the networks relies on the paradigm of spectral approximation, where on each manifold we use the corresponding eigenfunction basis of the Laplace-Beltrami operator. Previous works discussing the connection between harmonic analysis and deep learning over manifolds are described in [16, 21]. As we shall see, this allows to construct neural networks that provide higher accuracy using less parameters when benchmarked with standard PINN architectures.

(iii) Typically, PINNs need to be re-trained for each given initial condition. Our approach allows to train a network that can take in as input a subspace of initial conditions over the manifold.

The outline for the remainder of this paper is as follows. Section 2 reviews some preliminaries about PINNs and spectral approximation over manifolds. Section 3 describes the key aspects of our approach. In Section 4 we provide, as a pedagogical example, the theory and details of the method for the simple case of the heat equation over the unit interval. In Sections 5 and 6 we show how our approach is applied for nonlinear equations over the sphere and torus. Our extensive experimental results include generalization studies where the testing dataset is sampled from a significantly larger space than the training set. We also verify the stability of our models by injecting random noise to the input and validating the errors increase in controlled manner. Concluding remarks are found in Section 7.

2 Preliminaries

2.1 Physics informed neural networks

In this section, we describe the basic approach to PINNs presented in [4]. Generally, the goal is to approximate the solution for a differential equation over a domain Ω of the form:

$$u_t + \mathcal{N}[u] = 0, \quad t \in [0, T],$$

with some pre-defined initial and/or boundary conditions. Typically, a PINN $\tilde{u}(x, t)$ is realized using a Multi Layer Perception (MLP) architecture. This is a pass forward network where each j-th layer takes as input the vector v_{j-1} which is the output of the previous layer, applies to it an affine transformation $y = M_j v + b_j$ and then a coordinate-wise nonlinearity σ to produce the layer's output v_j

$$v_j = \sigma \circ (M_j v_{j-1} + b_j). \tag{1}$$

In some architectures either the bias vector b_j and/or the coordinate-wise nonlinearity σ are not applied in certain layers. In a standard PINN architecture, the input to the network $\tilde{u}$ is $v_0 = (x, t)$. The unknown parameters of the network are the collection of weights $\{M_j, b_j\}_j$ and the network is trained to minimize the following loss function:

$$MSE_B + MSE_D,$$

with the boundary/initial value loss

$$MSE_B = \frac{1}{N_b} \sum_{i=1}^{N_b} |\tilde{u}(x_i^b, t_i^b) - u(x_i^b, t_i^b)|^2,$$

and the differential loss

$$MSE_D = \frac{1}{N_d} \sum_{i=1}^{N_d} |(\tilde{u}_t + \mathcal{N}[\tilde{u}])(x_i^d, t_i^d)|^2.$$

In the above, $\{(x_i^b, t_i^b)\}_{i=1}^{N_b}$ is a discretized set of time and space points, where each $u(x_i^b, t_i^b)$ is the true given initial or boundary value at (x_i^b, t_i^b). The set $\{(x_i^d, t_i^d)\}_{i=1}^{N_d}$, typically contains randomly distributed internal domain collocation points. Since the architecture of the neural network is given analytically (as in (1) for the case of MLP), the value $(\tilde{u}_t + \mathcal{N}[\tilde{u}])|_{(x_i^d, t_i^d)}$ at a data-point (x_i^d, t_i^d) can be computed using the automatic differentiation feature of software packages such as TensorFlow and Pytorch [6, 7] (in our work we used TensorFlow). Thus, the aggregated loss function enforces the approximating function $\tilde{u}$ to satisfy required initial and boundary conditions as well as the differential equation.

2.2 Spectral decompositions over manifolds

We recall a fundamental result in the spectral theory over manifolds regarding the spectrum of the Laplace-Beltrami operator Δ [8, Theorem 10.13]

Theorem 1. *Let Ω be a non-empty compact relatively open subset of a Riemannian manifold $\mathcal{M}$ with metric g and measure μ. The spectrum of $\mathcal{L} := -\Delta$ on Ω is discrete and consists of an increasing sequence $\{\lambda_k\}_{k=1}^{\infty}$ of non-negative eigenvalues (with multiplicity) such that $\lim_{k\to\infty} \lambda_k = \infty$. There is an orthonormal basis $\{\phi_k\}_{k=1}^{\infty}$ in $L_2(\Omega)$ such that each function ϕ_k is an eigenfunction of $-\Delta$ with eigenvalue λ_k. Moreover, if we wish to solve the heat equation $u_t = \Delta u$ on Ω with initial condition $u(x,t) = f(x),\ f \in L_2(\Omega)$, the solution is given by:*

$$u(x,t) = \sum_{k=1}^{\infty} e^{-\lambda_k t} \langle f, \phi_k \rangle \phi_k(x).$$

This well established result motivates the following spectral paradigm. To solve the heat equation with some initial condition, one should first decompose the initial condition function to a linear combination of the eigenfunctions basis and then apply a time-dependent exponential decay on the initial value coefficients. An approximation entails working with the subspace spanned by $\{\phi_k\}_{k=1}^{K}$, for some sufficiently large K (see e.g. Theorem 4 below). For a general manifold $\mathcal{M}$, the eigenfunctions do not necessarily have an analytic form and need to be approximated numerically. As we will show, we also follow the spectral paradigm for more challenging cases of nonlinear equations over manifolds, where the time dependent processing of the initial value coefficients is not obvious. Nevertheless, a carefully crafted architecture can provide superior results over standard PINNs.

3 The architecture of spectral PINNs

Let $\mathcal{M} \subset \mathbb{R}^n$ be a Riemannian manifold, $\Omega \subset \mathcal{M}$ a non-empty compact relatively open subset and $\mathcal{N}$ a differential operator over this manifold, which can possibly be nonlinear. We assume our family of initial conditions comes from a finite space $W \subset L_2(\Omega)$, that can be selected to be sufficiently large. Given a vector of samples $\vec{f}$ of $f \in W$ over a fixed discrete subset of Ω, a point $x \in \mathcal{M}$ and $t \in [0,T]$, we would like to find an approximation $\tilde{u}(\vec{f}, x, t)$, given by a trained neural network, to the solution

$$u_t + \mathcal{N}[u] = 0,$$
$$u(x, t=0) = f(x),\ \forall x \in \Omega.$$

Recall that typically PI networks are trained to approximate a solution for a single specific initial condition (such as in [4]).However, we emphasize that our neural network model is trained only once for the family of initial conditions from the subspace W and that once trained, it can be used to solve the equation with any initial condition from W. Moreover, as we demonstrate in our experimental results, the trained network has the 'generalization' property, since it is able to approximate well the solutions when the initial value functions are randomly sampled from a larger space containing W.

Our method takes inspiration from spectral methods for solving PDEs. It is composed of 3 steps implemented by 3 blocks, as depicted in Figure 1:

1. **Transformation Block -** The role of this block is to compute from the samples $\vec{f}$ of the initial value condition a 'projection' onto $W_K = span\{\phi_k\}_{k=1}^{K}$, for some given K, where $\{\phi\}_{k=1}^{\infty}$ are the eigenfunctions of the Laplace-Beltrami operator on the manifold. We denote this block as $\tilde{\mathcal{C}} : \vec{W} \to \mathbb{R}^K$, where $\vec{W}$ is a subset of $\mathbb{R}^L$ which contains sampling vectors of functions from W over a fixed discrete subset of Ω. The desired output of the block is an estimation $\{\tilde{f}_k\}_{k=1}^{K}$ of the coefficients $\{\langle f, \phi_k \rangle\}_{k=1}^{K}$. However, in cases where it is difficult to

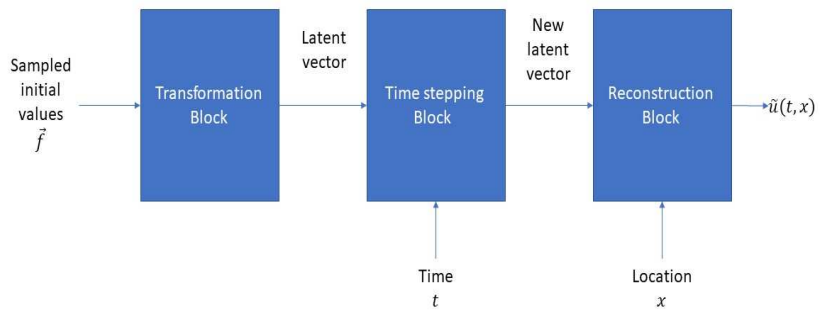

Figure 1: General description of our method

work with the spectral basis, one can train an encoder to transform the input samples to a compressed representation space of dimension K. Also, although the network is trained on point samples from W, it is able to receive as input a sample vector $\vec{f}$ of a function f which is from a larger space containing W and approximate the solution.

In most cases, it is advantageous to have the choice of the sampling set and the quantities L and K to be determined by 'Nyquist-Shannon'-type theorems for the given subspaces W, W_K and the manifold. In the scenario where $W = W_K$ and the sampling set of size L is selected to provide perfect 'Shanon'-type reconstruction, the transformation block may take the form of a simple linear transformation. In complex cases, where we have no prior knowledge about the required sampling rate or we do not have perfect reconstruction from the samples, we train a transformation block $\tilde{\mathcal{C}}$ that is optimized to perform a nonlinear 'projection' based on a carefully selected training set.

2. **Time Stepping Block -** In this block we apply a neural network that takes as input the output of the transformation block $\tilde{\mathcal{C}}(\vec{f})$, which may be the approximation of the spectral basis coefficients $\{\tilde{f}_k\}_{k=1}^K$, and a time stamp t, to compute a time dependent representation. We denote this block as $\tilde{\mathcal{D}} : \mathbb{R}^K \times [0,T] \to \mathbb{R}^K$.

3. **Reconstruction Block -** In this block we apply an additional neural network on the output of the time stepping block $\tilde{\mathcal{D}}$, together with the given input point $x \in \Omega$, to provide an estimate $\tilde{u}(x,t)$ of the solution $u(x,t)$. We denote this block as $\tilde{\mathcal{R}} : \mathbb{R}^K \times \Omega \to \mathbb{R}$.

Thus, our method is in fact a composition of the 3 blocks $\tilde{u} : \vec{W} \times \Omega \times [0,T] \to \mathbb{R}$

$$\tilde{u}(\vec{f}, x, t) = \tilde{\mathcal{R}}(\tilde{\mathcal{D}}(t, \tilde{\mathcal{C}}(\vec{f})), x).$$

Observe that in scenarios where one requires multiple evaluations at different locations $\{\tilde{u}(\vec{f}, x_i, t)\}_i$, $x_i \in \Omega$, at a given time step $t \in [0,T]$, one may compute once the output of the time stepping block $\tilde{\mathcal{D}}$ and use it multiple times for all $\{x_i\}_i$, and by that reducing the total computation time.

4 Introduction of the spectral PINN for the heat equation over $\Omega = [0, 1]$

We first review the prototype case of the heat equation on the unit interval where we can provide rigorous proofs for our method as well as showcase simple realization versions of our spectral network construction. Recall the heat equation:

$$u_t = \alpha u_{xx}, \quad x \in [0, 1], t \in [0, 0.5],$$

with initial time condition:

$$u(x, t = 0) = f(x), \ x \in [0, 1].$$

4.1 Architecture and theory for the heat equation over $\Omega = [0, 1]$

The analytic solution to this equation can be computed in 3 steps that are aligned with the 3 blocks of our architecture. Assume the initial condition $f : [0, 1] \to \mathbb{R}$ has the following spectral representation

$$f(x) = \sum_{k=1}^{\infty} c_k \sin(2\pi k x).$$

Next, apply the following transformation on the coefficients for a given time step t

$$\mathcal{D}(t, c_1, c_2, ...) := (e^{-4\pi^2 \alpha t} c_1, e^{-4\pi^2 \cdot 2^2 \alpha t} c_2, ...).$$

Finally, evaluate the time dependent representation at the point x:

$$u(x, t) = \mathcal{R}(e^{-4\pi^2 \alpha t} c_1, e^{-4\pi^2 \cdot 2^2 \alpha t} c_2, ..., x) := \sum_{k=1}^{\infty} e^{-4\pi^2 k^2 \alpha t} c_k \sin(2\pi k x).$$

We now proceed to provide the details of the numerical spectral PINN approach in this scenario. First, we select as an example $K = 20$ and $W = W_{20}$, where

$$W_{20} := \left\{ \sum_{k=1}^{20} c_k \sin(2\pi k x), \quad c_1, ..., c_{20} \in [-1, 1], \sqrt{c_1^2 + ... + c_{20}^2} = 1 \right\}.$$

We sample each $f \in W_{20}$ using $L = 101$ equip-spaced points in the segment $[0, 1]$ to compute a vector $\vec{f}$. For the training of the networks we use a loss function which is a sum of two loss terms $L_0 + L_D$. The loss L_0 enforces the solution to satisfy the initial time condition

$$L_0(\theta) = \frac{1}{101N} \sum_{i=1}^{N} \sum_{j=0}^{100} |\tilde{u}_\theta(\vec{f_i}, X_j, 0) - (\vec{f_i})_j|^2 \tag{2}$$

where $\tilde{u}_\theta$ is the model with weights θ and $(\vec{f_i})_j$ is the value of f_i at $X_j = \frac{1}{100} j$. For the second loss term we randomly generate $N = 5,000$ triples $(\vec{f_i}, x_i, t_i)_{i=1}^N$ and enforce the model to obey the differential condition

$$L_D(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\partial \tilde{u}_\theta(\vec{f_i}, x_i, t_i)}{\partial t} - \alpha \frac{\partial^2 \tilde{u}_\theta(\vec{f_i}, x_i, t_i)}{\partial x^2} \right|^2. \tag{3}$$

The derivatives of the given neural network approximation in (3) are calculated using the automatic differentiation capabilities of deep learning frameworks. In this work we use TensorFlow [6].

We compare two PINN architectures that provide an approximation to the solution u:

(i) **The naive model -** We benchmark our method with a deep learning model which is a standard MLP neural network, that takes in as input $(\vec{f}, t, x) \in \mathbb{R}^{103}$ and outputs an approximation. This model is trained to be PI using the loss function $L_0 + L_D$, where the two terms are defined in (2) and (3). The network is composed of 5 dense layers $\mathbb{R}^{103} \to \mathbb{R}^{103}$ and finally a dense layer $\mathbb{R}^{103} \to \mathbb{R}$. Each of the first five dense layers is followed by a non-linear activation function. Typically, a Rectifier Linear Unit (ReLU) $\sigma(x) = (x)_+$, is a popular choice as the nonlinear activation for MLP networks [12]. However, it is not suitable in this case, since its second derivative is almost everywhere zero. Therefore we use tanh as the nonlinear activation function.

(ii) **The spectral model -** In some sense, our spectral model $\tilde{u}$ is 'strongly' physics informed. Exactly as the naive model, it is also trained using the loss functions (2) and (3), to provide solutions to the heat equation. However, its architecture is different from the naive architecture, in that it is modeled to match the spectral method. The spectral model $\tilde{u}$ approximates u using the 3 blocks of the spectral paradigm approximation presented in the previous section. We now provide the details of the architecture and support our choice of design with rigorous proofs

1. **Sine transformation block** This block receives as input a sampling vector $\vec{f}$ and returns the sine transformation coefficients. Due to the high sampling rate $L = 101$, compared with the frequency used $K = 20$, the sampled function f can be fully reconstructed from $\vec{f}$ and this operation can be realized perfectly using the Nyquist-Shannon sampling formula. However, so as to simulate a scenario on a manifold where the sampling formula cannot be applied, we train a network to apply the transformation. To this end, we created $1,000$ initial value conditions using trigonometric polynomials of degree 20, and trained this block to extract the coefficients of those polynomials. In other words, we pre-trained $\tilde{\mathcal{C}} : \mathbb{R}^{101} \to \mathbb{R}^{20}$ for the following task:
$$\tilde{\mathcal{C}}(\vec{f}) = (c_1, ..., c_{20}),$$
where $\vec{f}$ is the sampling vector of the function
$$f(x) = \sum_{k=1}^{20} c_k \sin(2\pi k x).$$
In this simple case where $\Omega = [0, 1]$, the network can simply be composed of one dense layer with no nonlinear activation, which essentially implies computing a transformation matrix from samples to coefficients. In more difficult cases, the architecture of the network will be more complex.

2. **Time stepping block** The time stepping block should approximate the function:
$$\mathcal{D}(t, c_1, ..., c_{20}) = (e^{-4\pi^2 \alpha t} c_1, e^{-4\pi^2 \cdot 2^2 \alpha t} c_2, ..., e^{-4\pi^2 20^2 \alpha t} c_{20}). \tag{4}$$
We consider 2 architectures for this block:

Realization time stepping block:
In the case of the heat equation we know exactly how the time stepping block should operate and so we can design a true realization. The first layer computes
$$t \to (-4\pi^2 \alpha t, -4\pi^2 2^2 \alpha t, ..., -4\pi^2 20^2 \alpha t).$$
The second layer applies the exponential nonlinearity
$$(-4\pi^2 \alpha t, -4\pi^2 \cdot 2^2 \alpha t, ..., -4\pi^2 20^2 \alpha t) \to (e^{-4\pi^2 \alpha t}, e^{-4\pi^2 \cdot 2^2 \alpha t}, ..., e^{-4\pi^2 20^2 \alpha t}).$$
Finally, we element-wise multiply the output of the second layer with $(c_1, ..., c_{20})$ to output the time dependent spectral representation (4).

Approximate time stepping block:
In the case of general manifolds we may not be able to fully realize the time stepping block. Therefore, we examine what are the consequences of using an MLP network that approximates for given $K \geq 1$

$$\mathcal{D}(t, c_1, ..., c_K) := (e^{-4\pi^2 \alpha t} c_1, ..., e^{-4\pi^2 K^2 \alpha t} c_K).$$

We prove the following theorem (see the appedix for proofs):

Theorem 2. *For any $\epsilon > 0$ and $K \geq 1$ there exists a MLP network $\tilde{\mathcal{D}}$, consisting of dense layers and* tanh *as an activation function, with $O(K^3 + K \log^2(\epsilon^{-1}))$ weights such that*

$$\|\tilde{\mathcal{D}}(t, c_1, ..., c_K) - \mathcal{D}(t, c_1, ..., c_K)\|_\infty \leq \epsilon,$$

for all inputs $c_1, ..., c_K \in [-1, 1], t \in [0, 1]$.

We remark that it is possible to approximate $\mathcal{D}$ using ReLU as the nonlinear activation as it shown in [13]. However, recall the ReLU is not suitable for our second order differential loss function (3). In the experiments below, the approximating MLP time stepping block is composed of 5 layers.

3. **Reconstruction Block** The reconstruction block should operate as follow:

$$\mathcal{R}(a_1, ..., a_K, x) = \sum_{k=1}^{K} a_k \sin(2\pi k x).$$

In the case of the heat equation, for given $t \in [0, 1]$, the coefficients $\{a_k\}_{k=1}^K$ are $\{e^{-4\pi^2 k^2 t} c_k\}_{k=1}^K$ or an approximation to these coefficients. Here, also one can design a realization block which uses the sine function as a nonlinearity. To support the general case we have the following result

Theorem 3. *For fixed $A > 0$, $K \geq 1$ and any $\epsilon > 0$, there exists a MLP network $\tilde{\mathcal{R}}$, consisting of dense layers and* tanh *as an activation function, with $O(K^2 + K \log^2(K\epsilon^{-1}))$ weights for which*

$$|\tilde{\mathcal{R}}(a_1, ..., a_K, x) - \mathcal{R}(a_1, ..., a_K, x)| \leq \epsilon,$$

where $a_1, ..., a_K \in [-A, A], x \in [0, 1]$.

In the experiments below, the approximating MLP reconstruction block is composed of 5 layers. Using theorem 2 and 3, we can prove a general theorem that provides an estimate for the approximation of a MLP network. We first give the definition of Sobolev spaces [22]:

Definition 1. Let $\Omega \subset \mathbb{R}^n$ and $C_0^r(\Omega)$ be the space of continuously r-differentiable with compact support functions. For $1 \leq p < \infty$, the Sobolev space $W_p^r(\Omega)$ is the completion of $C_0^r(\Omega)$ with respect to the norm

$$\|f\|_{W_p^r(\Omega)} = \sum_{|\alpha| \leq r} \|\partial^\alpha f\|_{L_p(\Omega)},$$

where $\partial^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} ... \partial x_n^{\alpha_n}}$, $|\alpha| = \sum_{i=1}^n \alpha_i$.

With this definition at hand we are ready to state a result on the approximation capabilities of our spectral architecture when MLP networks are used to approximate the spectral realization

Theorem 4. *Let $r \in \mathbb{N}$. For any $\epsilon > 0$ there exists a MLP neural network $\tilde{u}$, with* tanh *nonlinearities and $O(\epsilon^{-3/r} + \epsilon^{-1/r} \log^2(\epsilon^{-(1+1/r)}))$ weights (the constant depends on r) for which the following holds: For any $f \in W_2^r([0, 1])$, $f = \sum_{k=1}^\infty c_k \sin(2\pi k x)$, $\|f^{(r)}\|_2 \leq 1$ and u, the solution to the heat equation on $\Omega = [0, 1]$ with the initial condition f, the network $\tilde{u}$ takes the input $\{c_k\}_{k=1}^K$, $K \leq c\epsilon^{-1/r}$ and provides the estimate*

$$\|u(f, \cdot, t) - \tilde{u}(f, \cdot, t)\|_{L_2[0,1]} \leq \epsilon, \qquad \forall t \in [0, 1].$$

Model number in plots	Model Architecture	#Model weights	Testing MSE: 5,000 training samples	Testing MSE: 25,000 training samples
1	Naive Model	53,664	1.3e-4	1.19e-4
2	Spectral model - full realization (time stepping and reconstruction blocks)	**2,960**	**9.0e-6**	**8.3e-6**
3	Spectral model - MLP approximation of time stepping block, realization of reconstruction block	11,980	5.7e-5	4.9e-5
4	Spectral model - realization of time stepping block, MLP approximation of reconstruction block	10,401	2.9e-5	2.87e-5

Table 1: Heat equation over $\Omega = [0, 1]$ - Comparison of standard naive PINN model with 3 variants of our spherical PINN model

4.2 Experimental Results

We benchmark 4 models: the naive PINN model with vanilla MLP architecture consisting of 6 layers and 3 variations of the spectral model with the various blocks realized or approximated. The training was performed using $5,000$ and $25,000$ samples of the form $(\vec{f}, x, t)$, where $\vec{f}$ is a sampling vector of trigonometric polynomial of degree 20 on 101 equip-spaced points in the segment $[0, 1]$ with $t \in [0, 0.5]$. To guarantee slow vanishing of the solution over time we used $\alpha = 0.01$. The comparison of the averaged Mean Squared Error (MSE) of the 4 models over 20 randomly sampled test initial conditions is presented in Table 1.

In Figure 2 we plot the norm of the error at different time steps

$$Error(t) = \sum_{i=1}^{20} \frac{1}{101} \sqrt{\sum_{k=0}^{100} \left| \tilde{u}_{model}(\vec{f}_i, \frac{1}{100}k, t) - u(f_i, \frac{1}{100}k, t) \right|^2}.$$

We show some examples of the exact solution u and the output of the different neural network solution at different times and with several initial condition in figure 3.

In addition, we performed generalization and stability analysis for the different architectures. To evaluate the ability of our networks to generalize beyond the training space of polynomials of degree 20, we tested the different networks using initial conditions from a space of polynomials of degree 30. Namely,

$$W_{30} = \left\{ \sum_{k=1}^{30} c_k \sin(2\pi k x), \quad c_1, ..., c_{20} \in [-1, 1], \sqrt{c_1^2 + ... + c_{20}^2} = 1 \right\}.$$

To evaluate the stability of our networks, we add normal random noise with mean 0 and variance 0.001 to the initial condition sample vectors and evaluate in different time stamps using the following normalized metric

$$\frac{\|\tilde{u}_{model}(\vec{f} + \vec{\delta}, \cdot, t) - \tilde{u}_{model}(\vec{f}, \cdot, t)\|_2}{\|\delta\|_2} \tag{5}$$

where $\delta_i \sim N(0, 0.001)$. The results of the generalization test can be found in Table 2, and the results (averaged over 20 random initial conditions) for the stability test can be found in Table 3.

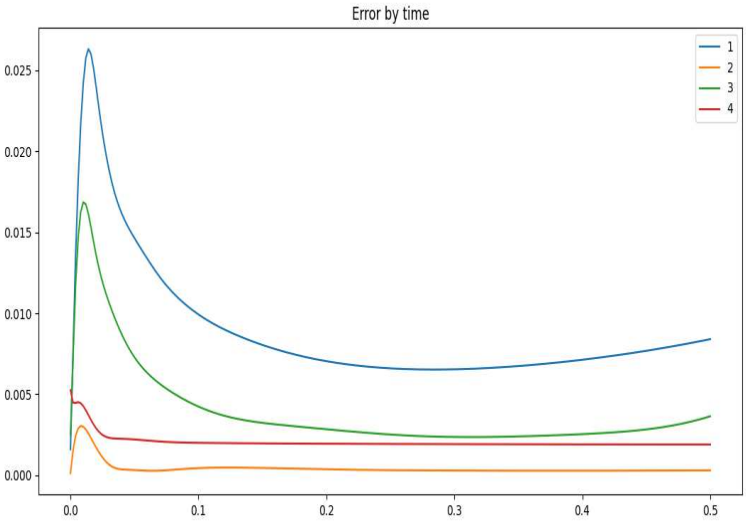

Figure 2: Heat equation on $[0, 1]$ - Error versus time

Model number in plots	Model Architecture	MSE
1	Naive Model	1.0e-3
2	Spectral model - full realization (time stepping and reconstruction blocks)	**7.1e-4**
3	Spectral model - MLP approximation of time stepping block, realization of reconstruction block	8.1e-4
4	Spectral model - realization of time stepping block, MLP approximation of reconstruction block	7.4e-4

Table 2: Heat equation on $[0, 1]$ - generalization results

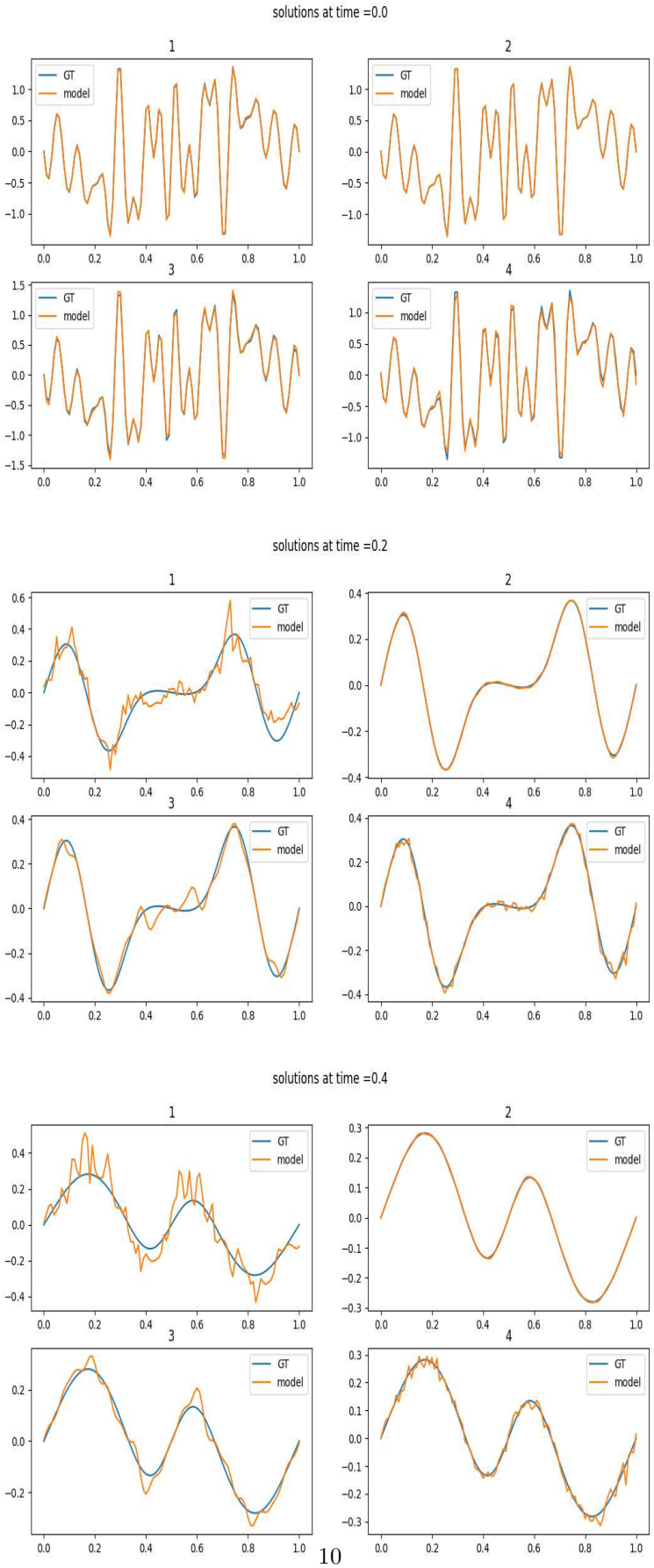

Figure 3: Heat equation over [0,1] - comparisons of the ground truth solution and the different neural network solutions with different initial conditions and at different times

Model number in plots	Model Architecture	$T = 0.2$	$T = 0.4$	$T = 0.5$
1	Naive Model	3.53	3.55	3.66
2	Spectral model - full realization (time stepping and reconstruction blocks)	0.95	0.73	0.67
3	Spectral model - MLP approximation of time stepping block, realization of reconstruction block	0.97	0.79	0.75
4	Spectral model - realization of time stepping block, MLP approximation of reconstruction block	0.95	0.75	0.68

Table 3: Heat equation on $[0, 1]$ - stability test results using the normalized metric (5)

In both tests, we can observe that all spectral model variants outperform the naive model.

The theoretical and empirical results for the simple case of the heat equation over $\Omega = [0, 1]$ motivate us to establish guidelines for designing spectral PINN networks in much more complicated scenarios. Namely, we should try to realize the various blocks, approximate them or at the least design their sub-components to have elements from the realization such as nonlinear activations relating to the spectral basis.

5 The sphere $\mathbb{S}^2$

In this section, we demonstrate our method in a more challenging setup, a nonlinear equation on a curved manifold. The Allen-Cahn equation over the sphere $\mathbb{S}^2$ is defined as follow [14]:

$$u_t = 0.1\Delta u + u - u^3, \ t > 0, \tag{6}$$

where the Laplace-Beltrami operator is

$$\Delta = \frac{\partial^2}{\partial\theta^2} + \frac{\cos\theta}{\sin\theta}\frac{\partial}{\partial\theta} + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2},$$

with ϕ is the azimuth angle and θ is the polar angle.

5.1 Theory and spectral PINN architecture for the Allen-Cahn equation on $\mathbb{S}^2$

On $\mathbb{S}^2 \subset \mathbb{R}^3$ the spectral basis is the spherical harmonic functions [9]:

Definition 2. The spherical harmonic function of degree l and order m is given by:

$$Y_l^m(\theta, \phi) = (-1)^m \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi},$$

where $\theta \in [0, \pi]$ is the polar angle, $\phi \in [0, 2\pi)$ is the azimuth angle and $P_l^m : [-1, 1] \to \mathbb{R}$ is the associated Legendre polynomial.

Each spherical harmonic function is an eigenfunction of the Laplace-Beltrami operator satisfying

$$\Delta Y_l^m = -l(l+1)Y_l^m.$$

In our work, for simplicity, we use the real version of the spherical harmonics, defined by:

$$Y_{lm} = \begin{cases} \sqrt{2}(-1)^m Im(Y_l^{|m|}), & -l \le m < 0, \\ Y_l^0, & m = 0, \\ \sqrt{2}(-1)^m Re(Y_l^m), & 0 < m \le l. \end{cases}$$

The input to our networks are of type $(F, (\theta, \phi), t)$, where $F \in \mathbb{R}^{20\times 20}$ is a sampling matrix of the initial condition on equip-spaced azimuth-polar grid of a spherical function, $\theta \in [0, \pi], \phi \in [0, 2\pi)$ are the coordinates of a point on the sphere and $t \in [0, 1]$. The loss functions are similar to the loss functions used in section 4, with the required modifications, such as for the differential loss term

$$L_D(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\partial \tilde{u}_\theta(F_i, x_i, t_i)}{\partial t} - (0.1\Delta\tilde{u}_\theta + \tilde{u}_\theta - \tilde{u}_\theta^3)(F_i, x_i, t_i)\right|^2. \tag{7}$$

Our goal is to construct a spectral PINN architecture that will outperform the naive PINN architecture. Here are the details of the 3 blocks of the spectral model:

1. **Transformation Block**
   This block receives as input a flatten sampling matrix $\vec{F} \in \mathbb{R}^{400}$ of the initial condition and returns the 100 spherical harmonic coefficients of degree 9. By [17, Theorem 3] under these conditions, spherical harmonics of degree 9 can be perfectly reconstructed. Thus, training one dense linear layer with sufficient samples, recovers the perfect reconstruction formula. In other words, we pre-trained $\tilde{\mathcal{C}} : \mathbb{R}^{400} \to \mathbb{R}^{100}$ for the following task:

   $$\tilde{\mathcal{C}}(\vec{F}) = (c_{0,0}, c_{1,-1}, c_{1,0}, c_{1,1}, ..., c_{9,-9}, ..., c_{9,0}, ..., c_{9,9}),$$

   where $\vec{F}$ is a flatten sampling matrix of the function

   $$\sum_{l=0}^{9}\sum_{m=-l}^{l} c_{l,m}Y_{lm}(\theta, \phi).$$

2. **Time Stepping Block**
   Unlike the heat equation on the unit interval, the Allen-Cahn equation (6) on the sphere, does not admit an analytic spectral solution. Nevertheless, we design an architecture that follows the spectral paradigm and compare it with a standard PINN MLP architecture. We test our hypothesis by conducting an ablation study using three optional architectures for the time stepping block:

   (a) **Input of Allen-Cahn Nonlinear Part**
   In this architecture, we further adapt the architecture to the nature of the equation, specifically to the non-linear part of the Allen-Cahn equation. Thus, in this variant, the input to the time stepping block is composed of: the transformation of the initial condition, the transformation of the nonlinear part of the initial condition and the time variable $(\tilde{\mathcal{C}}(\vec{F}), \tilde{\mathcal{C}}(\vec{F} - \vec{F}^3), t)$. Therefore the time stepping block is defined as

   $$\tilde{\mathcal{D}} : \mathbb{R}^{100} \times \mathbb{R}^{100} \times [0, T] \to \mathbb{R}^{100},$$

   where

   $$\tilde{\mathcal{D}}(\tilde{\mathcal{C}}(\vec{F}), \tilde{\mathcal{C}}(\vec{F} - \vec{F}^3), t) = (c_{0,0}(t), c_{1,-1}(t), c_{1,0}(t), c_{1,1}(t), ..., c_{9,-9}(t), ..., c_{9,0}(t), ..., c_{9,9}(t)).$$

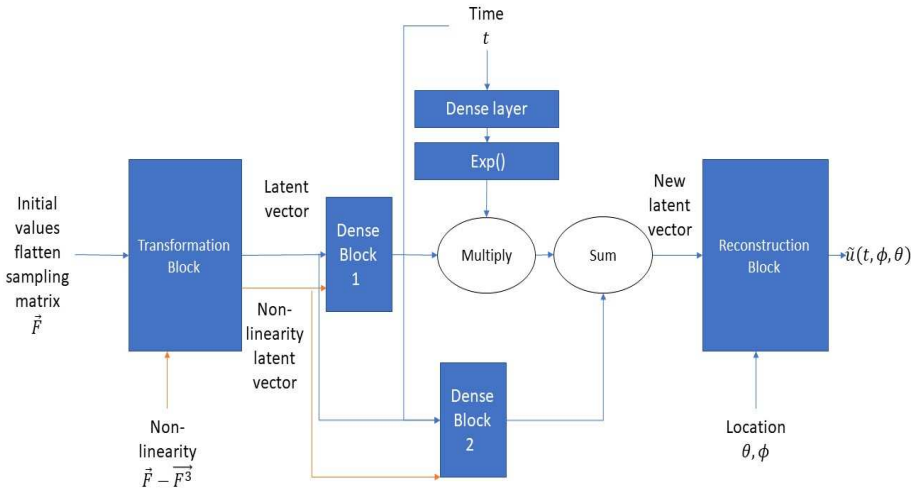

Figure 4: Full architecture for spherical setting - the red arrows are used only in variant (a) for time stepping block

With the additional input of the non-linear part, this variant of the time stepping block is a sum of two sub-blocks $\tilde{\mathcal{D}} = \tilde{\mathcal{D}}_1 + \tilde{\mathcal{D}}_2$. The component $\tilde{\mathcal{D}}_1$ is a sub-block designed to capture an exponential dynamic of the solution across time. The sub-block $\tilde{\mathcal{D}}_2$ is a standard PINN sub-block. The exponential sub-block $\tilde{\mathcal{D}}_1$ is defined by

$$\tilde{\mathcal{D}}_1(\tilde{\mathcal{C}}(\vec{F}), \tilde{\mathcal{C}}(\vec{F} - \vec{F}^3), t) = e^{\tilde{\mathcal{D}}_{1,1}(t)} \odot \tilde{\mathcal{D}}_{1,2}(\tilde{\mathcal{C}}(\vec{F}), \tilde{\mathcal{C}}(\vec{F} - \vec{F}^3)),$$

where $\odot$ is element-wise vector multiplication. The component $\tilde{\mathcal{D}}_{1,1} : \mathbb{R} \to \mathbb{R}^{100}$ is a simple dense layer with no bias, i.e. $\tilde{\mathcal{D}}_{1,1}(t) = V \cdot t$ where $V \in \mathbb{R}^{100}$ is a learnable vector. The component $\tilde{\mathcal{D}}_{1,2} : \mathbb{R}^{100} \times \mathbb{R}^{100} \to \mathbb{R}^{100}$ is an MLP subnetwork with 6 layers with tanh activations. Finally, the sub-block $\tilde{\mathcal{D}}_2$ is also an MLP subnetwork with 6 layers and tanh activations. The full architecture with this time stepping variant is depicted in Figure 4.

(b) **Standard Exponential Block**
This variant of the time-stepping block is similar to the one described in (a), but without the non-linear input $\tilde{\mathcal{C}}(\vec{F} - \vec{F}^3)$. Thus, the subnetwork capturing the exponential behavior takes the form

$$\tilde{\mathcal{D}}_1(\tilde{\mathcal{C}}(\vec{F}), t) = e^{\tilde{\mathcal{D}}_{1,1}(t)} \odot \tilde{\mathcal{D}}_{1,2}(\tilde{\mathcal{C}}(\vec{F})).$$

In this architecture we add 5 more dense layers to this block, as each layer requires less weights.

(c) **Naive MLP Time Stepping Block**
In this variant of the time stepping block, the input is $(\tilde{\mathcal{C}}(\vec{F}), t)$ and the architecture is a simple MLP block of 12 layers with tanh activation functions.

3. **Reconstruction Block**
The heuristics of our spectral approach is that the output of the time stepping block should be (once trained) a representation space resembling the coefficients of the spectral basis at the given time. Therefore, we design the reconstruction block to be composed of dense layers, but we use activation functions of the form $\sin^l, \cos^l$, $0 \leq l \leq 9$, on the input data point (θ, ϕ),

since these activation functions are the building blocks of the spherical harmonics functions. To this end, we first apply two subnetworks on the data point (θ, ϕ)

$$\mathcal{R}_{l,\sin,0}(\theta,\phi), \mathcal{R}_{l,\cos,0}(\theta,\phi) : \mathbb{R}^2 \to \mathbb{R}^2.$$

We then apply on their output, component wise, the spectral activation functions

$$\sin^l \circ \mathcal{R}_{l,\sin,0}(\theta,\phi), \quad \cos^l \circ \mathcal{D}_{l,\cos,0}(\theta,\phi), \quad 0 \le l \le 9.$$

Next we apply dense layers on the output of the activation functions

$$\mathcal{R}_{l,\sin,1}, \mathcal{R}_{l,\cos,1} : \mathbb{R}^2 \to \mathbb{R}^{100}, \quad 0 \le l \le 9.$$

We assemble these pieces to produce a subnetwork $\mathcal{R}_{loc} : \mathbb{R}^2 \to \mathbb{R}^{100}$

$$\mathcal{R}_{loc}(\theta,\phi) = \sum_{l=0}^{9} \mathcal{R}_{l,\sin,1}(\sin^l \circ \mathcal{R}_{l,\sin,0}(\theta,\phi)) \odot \mathcal{R}_{l,\cos,1}(\cos^l \circ \mathcal{R}_{l,\cos,0}(\theta,\phi)),$$

where $\odot$ is element-wise vector multiplication.

We apply separately, on the output of the time stepping block a subnetwork $\mathcal{R}_d : \mathbb{R}^{100} \to \mathbb{R}^{100}$. Finally, our reconstruction network $\tilde{\mathcal{R}}$ is a dot-product between the outputs of $\mathcal{R}_d$ and $\mathcal{R}_{loc}$

$$\begin{aligned}&\tilde{\mathcal{R}}(c_{0,0}(t), c_{1,-1}(t), c_{1,0}(t), c_{1,1}(t), ..., c_{9,-9}(t), ..., c_{9,0}(t), ..., c_{9,9}(t), \theta, \phi)\\ &\quad = \langle \mathcal{R}_d(c_{0,0}(t), c_{1,-1}(t), c_{1,0}(t), c_{1,1}(t), ..., c_{9,-9}(t), ..., c_{9,0}(t), ..., c_{9,9}(t)), \mathcal{R}_{loc}(\theta,\phi)\rangle.\end{aligned}$$

5.2 Experimental Results

We generated training data consisting of $N = 5,000$ randomly chosen samples of the form $(\vec{F}, (\theta,\phi), t)$, where $\vec{F}$ is a flattened sampling matrix of initial conditions randomly sampled from

$$W = \left\{ \sum_{l=0}^{9} \sum_{m=-l}^{l} c_{lm} Y_{lm}(\theta,\phi), \qquad c_{lm} \in [-1,1], \sqrt{\sum_{l=0}^{9} \sum_{m=-l}^{l} c_{lm}^2} = 1 \right\},$$

on the equip-spaced grid

$$\theta_j = \frac{\pi}{19} j, \ j \in \{0, ..., 19\}, \qquad \phi_k = \frac{2\pi}{20} k, \ k \in \{0, ..., 19\}.$$

During the training of the spectral model we used some manipulations to improve the results:

1. Pre-training the transformation block and the reconstruction block separately before training the full model, using the PI loss function

$$\frac{1}{N} \sum_{i=1}^{N} \left| F(\theta_i, \phi_i) - \tilde{\mathcal{R}}(\tilde{\mathcal{C}}(\vec{F_i}), (\theta_i, \phi_i)) \right|^2.$$

2. When training the full model, we started the first 20 epochs by freezing the weights of the transformation and reconstruction blocks that were pre-trained separately in (1) and training only the time stepping block. We observed that this technique where the transformation block and the reconstruction are pre-trained and then kept constant for the first epochs provides better initialization of the time-stepping block and overall better results.

   In this stage of the training, we used a loss function containing three terms. In addition to the standard initial condition loss and the differential loss we added new loss to enforce that the time stepping block does not change the spherical harmonics coefficients at time zero. Formally, the new loss term over the training set is

$$\frac{1}{100N} \sum_{i=1}^{N} \|\tilde{\mathcal{D}}(\tilde{\mathcal{C}}(\vec{F_i}), 0) - \tilde{\mathcal{C}}(\vec{F_i})\|_2^2. \tag{8}$$

Model number in plots	Model Architecture	#weights	MSE
1	Naive Model	4,070,704	1.1e-4
2	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	**391,186**	**4.8e-5**
3	Spectral model, time stepping variant (b) - Standard time stepping exponential block	452,590	6.7e-5
4	Spectral model, time stepping variant (c) - Naive time stepping dense block	490,682	9.7e-5

Table 4: Allen-Cahn equation over $\mathbb{S}^2$ - Comparison of standard naive PINN model with 3 variants of our spherical PINN model

3. Finally, we trained the full model with all 3 loss terms for 25 more epochs.

Since there is no analytical solution for the Allen-Cahn equation over $\mathbb{S}^2$, we used the numerical scheme IMEX-BDF4 [14] as ground truth for testing our models. Unlike [14], we used the spherical harmonic functions basis and not the double spherical Fourier method which was used in [14] due to performance considerations. We tested our models using 20 random initial conditions and predicted the solutions for all grid points:

$$\theta_j = \frac{\pi}{19}j,\ j \in \{0,...,19\}, \quad \phi_k = \frac{2\pi}{20}k,\ k \in \{0,...,19\}, \quad t_n = \frac{1}{500}n,\ n \in \{0,...,500\}.$$

We benchmarked 3 spectral PINN variants of the with the naive PINN model that has MLP architecture consisting of 26 layers with tanh activations. The comparison of the 4 models is described in Table 4. We can see that our model achieves better accuracy than the naive model, with significantly less parameters. We can also see that there is a benefit to the special processing of the non-linear part of Allen Cahn equation by feeding the time stepping block with the non-linear part of the initial condition. In Figure 5 we show the norm of the error in different time steps. As in the previous example, we performed generalization and stability tests. For the generalization test we used random initial conditions from the larger set of spherical harmonics of degree 14:

$$W_{gen} = \left\{ \sum_{l=0}^{14} \sum_{m=-l}^{l} c_{lm} Y_{lm}(\theta,\phi), \qquad c_{lm} \in [-1,1], \sqrt{\sum_{l=0}^{14} \sum_{m=-l}^{l} c_{lm}^2} = 1 \right\}.$$

For the stability test we used the technique as in the previous section with noise $\delta \sim N(0, 0.001)$. The results of generalization and stability tests can be found in tables 5 and 6 respectively (averaged over 20 random initial conditions). Again, we can see that all spectral model variants outperform the naive model.

6 The embedded torus $\mathbb{T} \subset \mathbb{R}^3$

In this section, we demonstrate our method on the embedded torus

$$\mathbb{T} = \{((R + r\cos\theta)\cos\phi, (R + r\cos\theta)\sin\phi, r\sin\theta) | \theta, \phi \in [0, 2\pi)\} \subset \mathbb{R}^3.$$

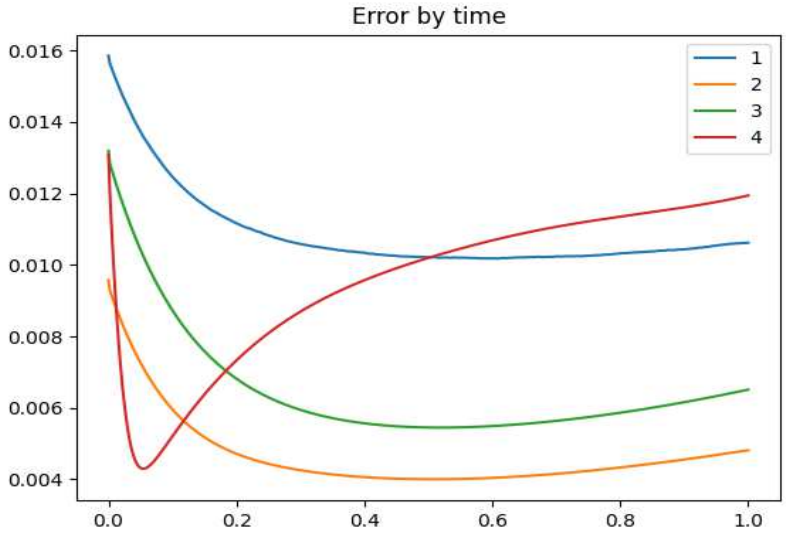

Figure 5: Allen-Cahn equation over $\mathbb{S}^2$ - Error over time of the naive and spectral variant PINN models on testing dataset

Model number in plots	Model Architecture	MSE
1	Naive Model	3.6e-4
2	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	**1.1e-4**
3	Spectral model, time stepping variant (b) - Standard time stepping exponential block	1.3e-4
4	Spectral model, time stepping variant (c) - Naive time stepping dense block	1.2e-4

Table 5: Allen-Cahn equation over $\mathbb{S}^2$ - generalization test results

Model number in plots	Model Architecture	$T = 0.4$	$T = 0.7$	$T = 1.0$
1	Naive Model	3.83	3.19	2.8
2	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	0.88	0.81	0.81
3	Spectral model, time stepping variant (b) - Standard time stepping exponential block	0.70	0.66	0.65
4	Spectral model, time stepping variant (c) - Naive time stepping dense block	2.85	2.86	2.86

Table 6: Allen-Cahn equation over $\mathbb{S}^2$ - stability test results using the normalized metric (5)

In this setting, the Laplace-Beltrami operator is [10]

$$\Delta_{\mathbb{T}} = \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2} - \frac{\sin\theta}{r(R + r\cos\theta)}\frac{\partial^2}{\partial\theta} + \frac{1}{(R + r\cos\theta)^2}\frac{\partial^2}{\partial\phi^2}.$$

On this manifold we demonstrate our spectral PINN method again using the Allen-Cahn equation (6). As the class of initial conditions we use the set

$$W_5 = \left\{ \sum_{k=1}^{5}\sum_{l=1}^{5} c_{k,l}\sin(k\theta)\sin(l\phi), \quad \sqrt{\sum_{k.l=1}^{5} c_{k,l}^2} = 1 \right\}.$$

We sample functions from this set on an equip-spaced grid with $N_\theta = N_\phi = 15$. On the embedded torus one is required to use a numeric approximation of the spectral basis and we used the finite-elements method implemented in the python package SPHARAPY [11]. The choice of spectral basis implementation impacts the design of the architecture of the transformation and reconstruction blocks. We test several options for each block. For the transformation and reconstruction blocks we consider two options:

1. **Numerical Spectral basis blocks -** In this option, we first create a dataset of 5,000 triples, each composed of a sampling matrix of a function $f(\theta,\phi) = \sum_{k=1}^{5}\sum_{l=1}^{5} c_{k,l}\sin(k\theta)\sin(l\phi)$ on the equip-spaced grid with $N_\theta = N_\phi = 15$ and two random coordinates $(\theta,\phi) \in [0, 2\pi)^2$ of a point on the torus. We then train the transformation and reconstruction blocks separately as follows.

   For the training of the transformation block $\tilde{\mathcal{C}}$ we further compute for each function in the training set, using its sampling matrix, the (numerical) spectral transformation using SPHARAPY. We use the coefficients computed by SPHARAPY as ground truth to train our transformation block. The block's architecture is composed of 3 convolution layers followed by one dense layer.

   The reconstruction block $\tilde{\mathcal{R}}$ in this variant is trained to take as input the coefficients of the spectral representation and the coordinate $(\theta,\phi) \in [0, 2\pi)^2$ and approximate the ground truth function value at this coordinate. The block architecture is a MLP subnet with 15 layers.

2. **Auto-Encoder-Decoder blocks -** In this variant, we train the transformation block $\tilde{\mathcal{C}}$ as the encoder together with the reconstruction block $\tilde{\mathcal{R}}$ as a decoder, without using explicitly the spectral representation on the torus. However, this approach is certainly inspired by the

Model number in plots	Transformation and Reconstruction blocks	Time stepping block	#Weights	MSE
1	Numerical spectral basis blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	**2,564,105**	**2.5e-5**
2	Auto-encoder-decoder blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	3,800,555	8.3e-5
3	Numerical spectral basis blocks	Spectral model, time stepping variant (b)	3,129,976	1.7e-4
4	Naive Model		4,130,001	2.7e-4

Table 7: Allen-Cahn equation over $\mathbb{T} \subset \mathbb{R}^3$ - Comparison of standard naive PINN model with 3 variants of our spherical PINN model

spectral method as we are ultimately optimizing some compressed representation space. The transformation encoder block simply learns to create a compressed latent representation of dimension 150 from the 225 function samples in a representation space. Its architecture is 5 convolution layers followed by one dense layer. Then the decoder takes the compressed representation together with the coordinate $(\theta, \phi) \in [0, 2\pi)^2$ and tries to recover the ground truth function value as this coordinate. Its architecture is 17 dense layers. The loss function is then

$$\frac{1}{N}\sum_{i=1}^{N}\left|\tilde{\mathcal{R}}(\tilde{\mathcal{C}}(\vec{F}_i),(\theta_i,\phi_i)) - f_i(\theta_i,\phi_i)\right|^2.$$

For the time stepping block we test two options

(a) A custom made time stepping block that receives as input the coefficients of the initial condition as well as the coefficients of the nonlinear part and a time step

$$(\tilde{\mathcal{C}}(\vec{F}), \tilde{\mathcal{C}}(\vec{F} - \vec{F}^3), t),$$

similarly to variant (a) of the time stepping block in the spherical case from previous section. Recall that such an architecture aims to be 'more' physics aware and adapted to the nature of the equation. For this variant of the time stepping block we use 9 dense layers.

(b) A network that takes as input

$$(\tilde{\mathcal{C}}(\vec{F}), t),$$

without the nonlinear part. Here we used 15 dense layers.

We denote this block as earlier with $\tilde{\mathcal{D}}$. For validation of our models, we used the IMEX-BDF4 numeric solver [14] to obtain approximations of solutions to the equations that we considered as ground truth. In table 7 we summarize the benchmarks of the various architectures and also compare them to a naive PINN architecture, with 26 layers, that simply takes in the samples of the initial condition as well as the time step and location on the torus and outputs an approximation of the value of the solution.

We can observe that the best result, in terms of accuracy and smaller size of the network, can be obtained using both numerical spectral basis blocks as transformation and reconstruction blocks, combined with the non-linear input time stepping block. Also, even the encoder-decoder variant that 'follows' the spectral paradigm to some extent without actually using the numerical spectral basis, provides a better result than the naive PINN model. In Figure 6 we show time plots of errors of the different PINN models averaged over 20 random initial conditions. For the generalization

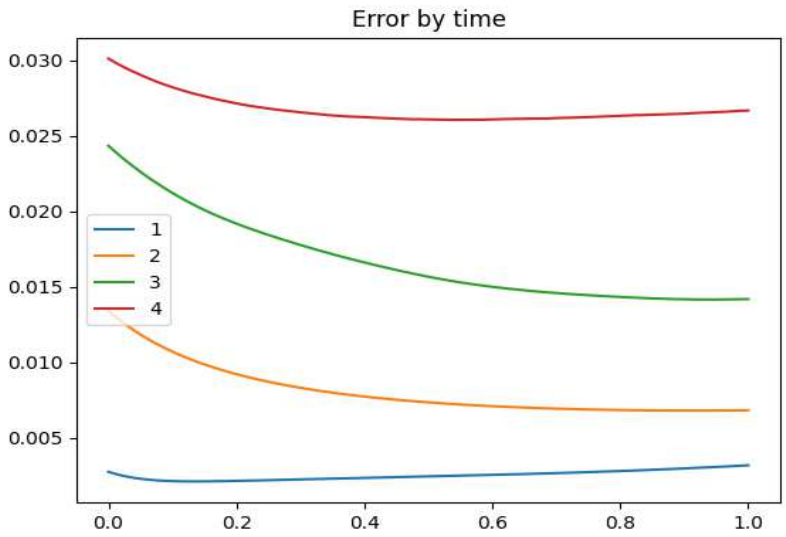

Figure 6: Allen-Cahn equation over $\mathbb{T} \subset \mathbb{R}^3$ - Error over time of the naive and spectral variant PINN models on testing dataset

test presented in Table 8, the network that was trained on samples from W_5 was tested on random initial conditions from the larger set

$$W_{10} = \left\{ \sum_{k=1}^{10} \sum_{l=1}^{10} c_{k,l} \sin(k\theta) \sin(l\phi), \quad \sqrt{\sum_{k.l=1}^{10} c_{k,l}^2} = 1 \right\}.$$

The stability tests listed in Table 9 are averaged over 20 random initial conditions.

7 Conclusion

In this work we presented a physics informed deep learning strategy for building PDE solvers over manifolds which is aligned with the method of spectral approximation. Our method allows to train

Model number in plots	Transformation and Reconstruction blocks	Time stepping block	MSE
1	Numerical spectral basis blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	**9.7e-5**
2	Auto-encoder-decoder blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	1.1e-4
3	Numerical spectral basis blocks	Spectral model, time stepping variant (b)	2.0e-4
4	Naive Model		2.1e-4

Table 8: Allen-Cahn equation over $\mathbb{T} \subset \mathbb{R}^3$ - generalization test results

Model number in plots	Transformation and Reconstruction blocks	Time stepping block	$T = 0.4$	$T = 0.7$	$T = 1.0$
1	Numerical spectral basis blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	0.73	0.70	0.74
2	Auto-encoder-decoder blocks	Spectral model, time stepping variant (a) - Input of Allen-Cahn nonlinear part	0.20	0.20	0,21
3	Numerical spectral basis blocks	Spectral model, time stepping variant (b)	1.5	1.46	1.46
4	Naive Model		2.4	2.9	2.3

Table 9: Allen-Cahn equation over $\mathbb{T} \subset \mathbb{R}^3$ - stability test results using the normalized metric (5)

a model that can take as input initial conditions from a pre-determined subset or subspace and is grid free. We showed empirically that our approach provides better approximation with much less weights compared with standard PINN architectures. For the case of the heat equation over the unit interval we provided a rigorous proof for the degree of approximation of a spectral PINN based on MLP components.

References

[1] B. Gustafsson, H. Kreiss & J. Oliger, Time dependent problems and difference methods, John Wiley & Sons, 1995.

[2] A. Tveito & R. Winther, Introduction to partial differential equations: a computational approach, Springer Science & Business Media, 2004.

[3] L. Bar, & N. Sochen, Unsupervised deep learning algorithm for PDE-based forward and inverse problems, arXiv preprint, 2019.

[4] M. Raissi, P. Perdikaris & G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics 378 (2019), 686-707.

[5] O. Ovadia, A. Kahana, E. Turkel & S. Dekel, Beyond the Courant-Friedrichs-Lewy condition: Numerical methods for the wave problem using deep learning, Journal of Computational Physics 442 (2021), 110493.

[6] https://www.tensorflow.org/guide

[7] https://pytorch.org/tutorials/

[8] A. Grigoryan, Heat kernel and analysis on manifolds, American Mathematical Soc. 47, 2009.

[9] K. Atkinson & W. Han, Spherical harmonics and approximations on the unit sphere: an introduction, Springer Science & Business Media, 2012.

[10] H. Volkmer, The Laplace-Beltrami operator on the embedded torus, Journal of Differential Equations 271 (2021), 821-848.

[11] https://spharapy.readthedocs.io/en/latest/

[12] X. Glorot, A. Bordes, Y. & Bengio, Deep sparse rectifier neural networks, In Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, 315-323.

[13] H. Montanelli, H. Yang & Q. Du, Deep ReLU networks overcome the curse of dimensionality for bandlimited functions, Journal of computational mathematics 39 (2021), 801-815.

[14] H. Montanelli & Y. Nakatsukasa, Fourth-order time-stepping for stiff PDEs on the sphere, SIAM Journal on Scientific Computing 40 (2018), A421-A451.

[15] H. Mhaskar, Neural networks for optimal approximation of smooth and analytic functions, Neural computation 8 (1996), 164-177.

[16] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam & P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (2017), 18-42.

[17] J. Driscoll & D. Healy, Computing Fourier transforms and convolutions on the 2-sphere, Advances in applied mathematics 15 (1994), 202-250.

[18] A. Baydin, B. Pearlmutter, A. Radul & J. Siskind, Automatic differentiation in machine learning: a survey. Journal of Marchine Learning Research 18 (2018), 1-43.

[19] A. Kashefi & T. Mukerji, T. (2022), Physics-Informed PointNet: A Deep Learning Solver for Steady-State Incompressible Flows and Thermal Fields on Multiple Sets of Irregular Geometries, arXiv preprint, 2022.

[20] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang & L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (2021), 422-440.

[21] S. Zafeiriou, M. Bronstein, T. Cohen, O. Vinyals, L. Song, J. Leskovec & M. Gori, Non-Euclidean Machine Learning, IEEE Transactions on Pattern Analysis & Machine Intelligence 44 (2022), 723-726.

[22] R. DeVore & G. Lorentz, Constructive approximation, Springer Science & Business Media, 1993.

[23] Z. Li, N. Kovachki, K. Azizzadenesheli, A. Stuart & A. Anandkumar, Fourier neural operator for parametric partial differential equations, in Proc. ICLR (2020), 1-16.

[24] M. Xia, L. Böttcher and T. Chou, Spectrally Adapted Physics-Informed Neural Networks for Solving Unbounded Domain Problems, arXiv preprint, 2022.

[25] M. Rafio, G. Rafio & G. Sang Choi, DSFA-PINN: Deep Spectral Feature Aggregation Physics Informed Neural Network, IEEE Access 10 (2022), 22247-22259.

[26] B. Lütjens, C. Crawford, M. Veillette & D. Newman, Spectral PINNs: fast uncertainty propagation with physics-informed neural networks, DLDE Workshop, NeurIPS 2021.

A Appendix

Proof of theorem 2. To approximate $\mathcal{D}$, we first build two types of MLP sub-networks. The first one M, approximates the multiplication

$$(x_1, x_2) \to x_1 x_2, \quad x_1, x_2 \in [-1, 1].$$

We also assume we have MLP sub-networks E_k, $1 \leq k \leq K$, that approximate

$$e^{-4\pi^2 k^2 \alpha t}, \quad t \in [0, 1].$$

Our MLP network $\tilde{\mathcal{D}}$ can then be implemented as the following feed forward composition

$$\begin{pmatrix} t \\ c_1 \\ c_2 \\ \vdots \\ c_K \end{pmatrix} \to \begin{pmatrix} c_1 \\ \vdots \\ c_K \\ E_1(t) \\ E_2(t) \\ \vdots \\ E_K(t) \end{pmatrix} \to \begin{pmatrix} M(c_1, E_1(t)) \\ M(c_2, E_2(t)) \\ \vdots \\ M(c_{K-1}, E_{K-1}(t)) \\ M(c_K, E_K(t)) \end{pmatrix}.$$

Let us assume that our sub-networks satisfy

$$|M(x_1, x_2) - x_1 x_2| \leq \frac{\epsilon}{2}, \quad \forall x_1, x_2 \in [-1, 1], \tag{9}$$

$$|E_k(t) - e^{-2\pi^2 k^2 t}| \leq \frac{\epsilon}{2}, \quad 1 \leq k \leq K, , t \in [0, 1]. \tag{10}$$

Then,

$$\begin{aligned} |M(c_k, E_k(t)) - c_k \cdot e^{-4\pi^2 k^2 \alpha t}| &\leq |M(c_k, E_k(t)) - c_k \cdot E_k(t)| + |c_k \cdot E_k(t) - c_k \cdot e^{-4\pi^2 k^2 \alpha t}| \\ &\leq \frac{\epsilon}{2} + |c_k| \frac{\epsilon}{2} \\ &\leq \epsilon. \end{aligned}$$

This immediately implies that

$$\|\tilde{\mathcal{D}}(t, c_0, ..., c_K) - \mathcal{D}(t, c_0, ..., c_K)\|_\infty \leq \epsilon.$$

It remains to construct the MLP sub-networks M and E_k, $1 \leq k \leq K$. Our main tool is Theorem 2.3 in [15] which provides the following special case. Assume $f : \mathbb{C}^d \to \mathbb{C}$ is analytic in the poly-ellipse defined for $\rho \geq 1$

$$E_\rho := \left\{ (z_1, \ldots, z_d) \in \mathbb{C}^d : \quad \left| z_j + \sqrt{|z_j|^2 - 1} \right| \leq \rho, \quad 1 \leq j \leq d \right\}.$$

Then, for any $\rho_1 < \rho$ there exists a constant $c(\rho_1, \rho) > 0$, such that for any $n \geq 1$ there exist coefficients $\{a_j\}_{j=1}^n$, vectors in $\mathbb{R}^d$, $\{v_j\}_{j=1}^n$ and a bias $b \in \mathbb{R}$, such that

$$\left\| f - \sum_{j=1}^{n} a_j \tanh(v_j \cdot + b) \right\|_{L_\infty[-1,1]^d} \leq c \rho_1^{-n^{1/d}} \max_{z \in E_\rho} |f(z)|. \tag{11}$$

The first application of this result, for the case $f(x_1, x_2) = x_1 x_2$, implies that for any $n \geq 1$, there exists a subnetwork of two layers M_n, with $O(n)$ parameters, which satisfies

$$\max_{x_1, x_2 \in [-1,1]} |x_1 x_2 - M_n(x_1, x_2)| \leq c e^{-n^{1/2}}.$$

Setting

$$\frac{c}{e^{n^{1/2}}} = \frac{\epsilon}{2},$$

implies we should choose

$$n = \log^2 \frac{2c}{\epsilon}.$$

We conclude there exists a subnetwork M, with $O(\log^2(\epsilon^{-1}))$ weights that provides the approximation (9).

Similarly, for any $1 \le k \le K$, we can apply (11) for $f(t) = e^{-2\pi^2 k^2 \alpha t}$, to obtain the estimate for subnetworks with n parameters $E_{k,n}$

$$\max_{t\in[0,1]} |e^{-2\pi^2 k^2 \alpha t} - E_{k,n}(t)| \le c e^{2\pi^2 k^2 \alpha \rho - n}, \quad 1 \le k \le K.$$

Thus, we may construct subnetworks $\{E_k\}_{k=1}^K$ with $O(K^2 + \log(\epsilon^{-1}))$ weights which provide the approximation (10).

We conclude that by assembling the subnetworks, we can construct the network $\tilde{\mathcal{D}}$ with $O(K^3 + K\log^2(\epsilon^{-1}))$ weights that provides the required approximation. □

Proof of theorem 3. The technique of the proof is similar to the method of proof of Theorem 2. We use (11) to obtain an estimate for subnetworks $S_{k,n}$, $1 \le k \le K$ each of $O(n)$ weights, satisfying

$$\max_{x\in[0,1]} |\sin(2\pi k x) - S_{k,n}(x)| \le c e^{2\pi k - n}.$$

So, it is possible to construct subnetworks S_k, $1 \le k \le K$, each with $O(K + \log K + \epsilon^{-1}) = O(K + \epsilon^{-1})$ weights such that

$$\max_{x\in[0,1]} |\sin(2\pi k x) - S_k(x)| \le \frac{\epsilon}{2K}.$$

We require a multiplication subnetwork M as in the proof of Theorem 2. However, this time, assuming that for sufficiently small $\epsilon > 0$, the outputs of the subnets $\{S_k\}_{k=1}^K$ are in $[-2, 2]$. Since these are inputs to the multiplication network, we construct a subnetwork with $O(\log^2(K\epsilon^{-1}))$ weights that satisfies

$$|M(x_1, x_2) - x_1 x_2| \le \frac{\epsilon}{2K}, \quad \forall x_1, x_2 \in [-2, 2], . \tag{12}$$

We assemble the subnetworks to construct an approximating MLP network with $O(K^2 + K\log^2(K\epsilon^{-1}))$ weights

$$\tilde{\mathcal{R}}(a_1, \dots, a_K, x) := \sum_{k=1}^K M(a_k, S_k(x)).$$

Finally, we obtain the required estimate by

$$\begin{aligned}
\left|\sum_{k=1}^K a_k \sin(2\pi k x) - \tilde{\mathcal{R}}(a_1, \dots, a_K, x)\right| &= \left|\sum_{k=1}^K a_k \sin(2\pi k x) - \sum_{k=1}^K M(a_k, S_k(x))\right| \\
&\le \left|\sum_{k=1}^K a_k \sin(2\pi k x) - \sum_{k=1}^K a_k S_k(x)\right| + \left|\sum_{k=1}^K a_k S_k(x) - \sum_{k=1}^K M(a_k, S_k(x))\right| \\
&\le K\frac{\epsilon}{2K} + K\frac{\epsilon}{2K} \le \epsilon.
\end{aligned}$$

□

Proof of theorem 4. It is well known that the Fourier series has the 'spectral approximation' property. Namely, for a Sobolev function $g \in W_2^r[0, 1]$, with the Fourier expansion $g = \sum_{k=-\infty}^{\infty} \hat{g}(k) e^{2\pi i k x}$,

we can estimate the error of the truncated Fourier expansion:

$$\begin{aligned}\left\| g - \sum_{k=-K}^{K} \hat{g}(k) e^{2\pi i k x} \right\|_2^2 &= \left\| \sum_{k=-\infty}^{\infty} \hat{g}(k) e^{2\pi i k x} - \sum_{k=-K}^{K} \hat{g}(k) e^{2\pi i k x} \right\|_2^2 \\ &= \left\| \sum_{|k|>K} \hat{g}(k) e^{2\pi i k x} \right\|_2^2 \\ &= \sum_{|k|>K} |\hat{g}(k)|^2 = \\ &\leq \sum_{|k|>K} \left(\frac{|2\pi k|}{K} \right)^{2r} |\hat{g}(k)|^2 \\ &\leq K^{-2r} \sum_{k=-\infty}^{\infty} |2\pi k|^{2r} |\hat{g}(k)|^2 \\ &= K^{-2r} \sum_{k=-\infty}^{\infty} |\widehat{g^{(r)}}(k)|^2 = K^{-2r} \|g^{(r)}\|_2^2.\end{aligned}$$

Thus, for $g = \sum_{k=1}^{\infty} g_k \sin(2\pi k x)$, $g \in W^r_{2,[0,1]}$, $\|g^{(r)}\|_2 \leq 1$, we obtain

$$\|g - \sum_{k=-K}^{K} \hat{g}(k) e^{2\pi i k x}\|_2 \leq K^{-r}.$$

This implies that for any initial condition function $f = \sum_{k=1}^{\infty} c_k \sin(2\pi k x)$, $f \in W^r_{2,[0,1]}$, $\|f^{(r)}\|_2 \leq 1$ and $t \in [0,1]$, we may approximate the solution $u(f,x,t)$ to the heat equation by

$$\|u(f,\cdot,t) - u_K(\cdot,t)\|_2 \leq K^{-r}, \qquad u_K(x,t) := \sum_{k=1}^{K} c_k e^{-4\pi^2 k^2 t} \sin(2\pi k x).$$

For the given ϵ and $r \geq 1$, we select

$$K := \left(\frac{3}{\epsilon} \right)^{1/r},$$

which gives

$$\|u(f,\cdot,t) - u_K(\cdot,t)\|_2 \leq \frac{\epsilon}{3}, \tag{13}$$

uniformly for all initial conditions from our Sobolev ball and all times $t \in [0,1]$. With this choice of K, we construct the following two networks

1. Using Theorem 2, we may construct for $\tilde{\epsilon} := \epsilon/(3K)$ a block $\tilde{\mathcal{D}}$ containing

   $$O(K^3 + K \log^2(\tilde{\epsilon}^{-1})) = O(K^3 + K \log^2(K\epsilon^{-1})) = O(\epsilon^{-3/r} + \epsilon^{-1/r} \log^2(\epsilon^{-(1+1/r)}))$$

   weights that satisfies

   $$\|\tilde{\mathcal{D}}(t, c_1, ..., c_K) - \mathcal{D}(t, c_1, ..., c_K)\|_\infty \leq \frac{\epsilon}{3K}. \tag{14}$$

2. Based on (14), for sufficiently small $\epsilon > 0$, the output of $\tilde{\mathcal{D}}$ is a vector in $[-2,2]^K$, since it approximates the output of $\mathcal{D}$ which is a vector in $[-1,1]^K$. Using Theorem 3 we may construct a block $\tilde{\mathcal{R}}$ containing

   $$O(K^2 + K \log^2(K\epsilon^{-1})) = O(\epsilon^{-2/r} + \epsilon^{-1/r} \log^2(\epsilon^{-(1+1/r)}))$$

   weights that satisfies

   $$|\tilde{\mathcal{R}}(a_1, ..., a_K, x) - \mathcal{R}(a_1, ..., a_K, x)| \leq \frac{\epsilon}{3}, \qquad \forall a_k \in [-2,2], 1 \leq k \leq K, \quad x \in [0,1].$$

Our approximating solution is then defined by $\tilde{u}(f,x,t) := \tilde{\mathcal{R}}(\tilde{\mathcal{D}}(t,c_1,\ldots,c_K),x)$, where the network contains a total of $O(\epsilon^{-3/r}+\epsilon^{-1/r}\log^2(\epsilon^{-(1+1/r)}))$ weights. We can estimate the approximation by

$$\|u(f,\cdot,t)-\tilde{u}(f,\cdot,t)\|_2 \leq \|u(f,\cdot,t)-u_K(\cdot,t)\|_2 + \|u_K(\cdot,t)-\tilde{u}(f,\cdot,t)\|_2. \tag{15}$$

Applying (13) provides the bound $\epsilon/3$ for the first right hand side term in (15). We now proceed to bound the second term by $2\epsilon/3$ using the the simple inequality $\|g\|_{L_2[0,1]} \leq \|g\|_{L_\infty[0,1]}$. To this end for any $x \in [0,1]$

$$\begin{aligned}
|u_K(x,t)-\tilde{u}(f,x,t)| &= \left|\mathcal{R}(\mathcal{D}(t,c_1,...,c_K),x) - \tilde{\mathcal{R}}(\tilde{\mathcal{D}}(t,c_1,...,c_K),x)\right| \\
&\leq \left|\mathcal{R}(\mathcal{D}(t,c_1,...,c_K),x) - \mathcal{R}(\tilde{\mathcal{D}}(t,c_1,...,c_K),x)\right| + \left|\mathcal{R}(\tilde{\mathcal{D}}(t,c_1,...,c_K),x) - \tilde{\mathcal{R}}(\tilde{\mathcal{D}}(t,c_1,...,c_K),x)\right| \\
&\leq K\frac{\epsilon}{3K} + \frac{\epsilon}{3} = \frac{2\epsilon}{3}.
\end{aligned}$$

□