GrINd: Grid Interpolation Network for Scattered Observations

Andrzej Dulny ✉, Paul Heinisch, Andreas Hotho, and Anna Krause

CAIDAS, University Würzburg, Germany
andrzej.dulny@uni-wuerzburg.de
paul.heinisch@stud-mail.uni-wuerzburg.de
hotho@informatik.uni-wuerzburg.de
anna.krause@uni-wuerzburg.de

Abstract. Predicting the evolution of spatiotemporal physical systems from sparse and scattered observational data poses a significant challenge in various scientific domains. Traditional methods rely on dense grid-structured data, limiting their applicability in scenarios with sparse observations. To address this challenge, we introduce GrINd (Grid Interpolation Network for Scattered Observations), a novel network architecture that leverages the high-performance of grid-based models by mapping scattered observations onto a high-resolution grid using a Fourier Interpolation Layer. In the high-resolution space, a NeuralPDE-class model predicts the system's state at future timepoints using differentiable ODE solvers and fully convolutional neural networks parametrizing the system's dynamics. We empirically evaluate GrINd on the DynaBench benchmark dataset, comprising six different physical systems observed at scattered locations, demonstrating its state-of-the-art performance compared to existing models. GrINd offers a promising approach for forecasting physical systems from sparse, scattered observational data, extending the applicability of deep learning methods to real-world scenarios with limited data availability.

Keywords: Physics · Dynamical Systems · Fourier · NeuralPDE · DynaBench

1 Introduction

Understanding and accurately predicting the evolution of spatiotemporal physical systems is an important challenge in various scientific domains, encompassing fields such as climate modeling, weather forecasting, computational fluid simulation, biology, and many more [23,15,7,10]. These systems are typically guided by physical laws summarized as partial differential equations, which describe the rate of change of the systems' quantities as a function of their partial derivatives in space [13]. Traditionally, numerical simulations have served as the cornerstone of predictive modeling for these systems, with various methods such as

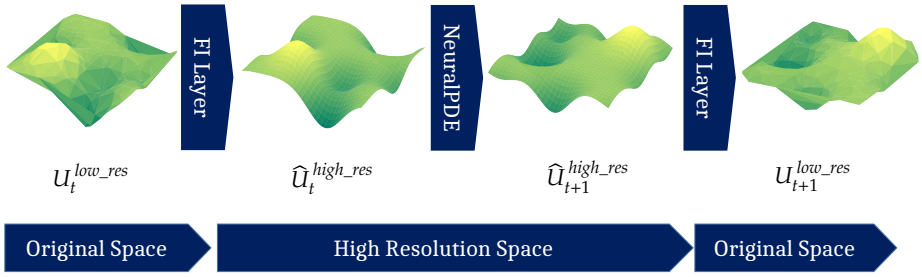


Fig. 1: Summary of our approach. The low-resolution observations are first mapped onto a high-resolution grid using a Fourier Interpolation Layer. In this high resolution space a predictive model (NeuralPDE) forecasts the evolution of the system which is then mapped back to the original observation space.

the method of lines, spectral methods, finite volume methods, and the finite element method designed to tackle specific challenges and specific classes of equations [27,46].

In recent years, a notable paradigm shift has emerged, with a range of new deep learning-based methods being proposed to replace or augment the classical numerical methods. Models like Physics Informed Neural Networks [41,42] can be used to solve differential equations by employing neural networks as solution approximators. Other approaches like SINDy [11] or PDE-Net [35] solve the task of reconstructing the equation that describes the evolution of the system by using symbolic regression and sparsity enforcement. Yet another range of methods focus solely on forecasting the evolution of the system. Models like PanguWeather [9], FourCastNet [25] or GraphCast [26] leverage large-scale neural architectures to learn the behavior of the system directly from data without any physical information.

Crucially, the performance of these types of models depends on the availability of large amounts of high-quality data that in most cases is assumed to be distributed across a spatial grid. In fact most methods proposed for forecasting physical systems work only on grid structured data [17] and a persistent challenge arises in scenarios where observational data is inherently sparse and spatially scattered, a common occurrence in many real-world applications. As Dulny et al. [17] pointed out, the task of predicting the evolution of a physical system based only on low-resolution scattered observations remains a challenging and unsolved task.

To tackle this challenge, we introduce GrINd (**Gr**id **I**nterpolation **N**etwork for Scattere**d** Observations), an abstract network architecture which leverages the high performance of grid-based models, by mapping the scattered observations onto a high-resolution grid using a Fourier Interpolation Layer. In the high-resolution space a NeuralPDE-class model [16] predicts the state of the system at future timepoints using a combination of differentiable ODE solvers [14] and

the method of lines [46] with fully convolutional neural networks parametrizing the dynamics of the system. Our approach is summarized in Figure 1.

We empirically evaluate the performance of our model on the DynaBench benchmark dataset [17] encompassing six different physical systems observed on scattered locations and show its state-of-the-art performance compared to other non-grid based models.

Overall, our contributions can be summarized as follows:

1. We introduce GrINd - a novel network architecture for forecasting physical systems from sparse, scattered observational data.
2. We combine a Fourier Interpolation Layer and a NeuralPDE model for efficient predictions in high-resolution space
3. We empirically evaluate our model on the DynaBench dataset, demonstrating state-of-the-art performance compared to existing models.

We make the source code containing our model as well as all experiments in this paper publicly available. [1]

2 Related Work

Learning the behavior of real physical systems from empirical data using machine learning techniques has become of increasing interest in recent years. The work in this area can generally be divided into two overlapping areas: data-driven solution and data-driven discovery of partial differential equations. Examples of the former include data-driven models for predicting the evolution of various physical systems such as weather with transformer models [9,25,26], fluid simulation with graph neural networks [45] or simply solving specific PDEs using Physics Informed Neural Networks [41]. Examples of the latter include SINDy [11] which uses sparse regression to find the underlying governing equations of a system and PDE Net [35] which uses a set of learnable filters and symbolic regression to reconstruct the equation. The relevance of the area also emerges from the increasing number of benchmark datasets available for physical systems [17,43,36].

Most works use high-resolution grid data which offers many advantages, allowing the use of well-developed models and architectures such as a modified vision transformer [9,25,26,1,32,29]. Another approach by Ayed et al. [4] uses a hidden-state neural-based model to forecast dynamical systems using a ResNet. The Finite Volume Neural Network (FINN) [39] predicts the evolution of diffusion-type systems by explicitly modeling the flow between grid points utilizing the Finite Volume Method. NeuralPDE [16] combines CNNs and the method of lines solver to find a solution for the underlying PDE.

Only a limited number of approaches have been proposed to tackle the challenging task of forecasting a physical system from scattered data. Iakovlev et al. [20] introduce continuous-time representation and learning of the dynamics of PDE-driven systems using a GNN. The MGKN (multipole graph kernel

[1] `https://professor-x.de/grind`

network) [33] proposed by Li et al. unifies GNNs with multiresolution matrix factorization to capture long-range correlations. PhyGNNet [21] divides the computational domain into meshes and uses the message-passing mechanism, as well as the discrete difference method, based on Taylor expansion and least squares regression, to predict future states. Graph networks, however, can become computationally expensive when reaching a certain size, common in real-world tasks.

To the best of our knowledge, our approach is the first model to take advantage of Fourier Analysis to interpolate observations onto a higher resolution grid. Several deep learning approaches have been proposed to perform training entirely in the Fourier domain such as Fourier Convolutional Neural Networks [40] or the Fourier Neural Operator [34]. However, all works taking advantage of the Fast Fourier Transform (FFT) still require the input data to be grid-structured.

3 Method

In this section, we describe the technical details of our proposed approach. The overall architecture of our approach consisting of the Fourier interpolation layer and the predictive NeuralPDE model can be seen schematically in Figure 1.

3.1 Fourier Interpolation Layer

One of the core components of our approach is the Fourier interpolation layer, which takes as input the values of a function u at several non-uniform locations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_H\}$ and outputs the interpolated values of the function $\tilde{u}$ at grid locations X_{grid}. To calculate the interpolation in a differentiable way, we leverage the approximation properties of the Fourier series summarized in Theorem 1.

While in principle any functional basis of the L^2 function space can be used for approximation, we opt for the Fourier basis as it has been previously studied in related work in the context of partial differential equations [24,2,34] and several algorithms could potentially be used for faster computations [49,5,6].

Theorem 1. *Given a periodic complex function $u\colon \Omega \to \mathbb{C}$ on the interval box $\Omega = [0,1]^M$, such that $\int_\Omega ||u||^2 d\mathbf{x} < \infty$, the following series:*

$$\hat{u}_N(\mathbf{x}) = \sum_{\mathbf{k} \in K} c_{\mathbf{k}}(u) e^{2i\pi \mathbf{k}\cdot\mathbf{x}} \tag{1}$$

with

$$\begin{aligned} c_{\mathbf{k}}(u) &= \int_\Omega u(\mathbf{x}) e^{2i\pi \mathbf{k}\cdot\mathbf{x}} d\mathbf{x} \\ N &= (N_1, \ldots, N_M) \\ K &= K_{N_1,\ldots,N_M} := K_{N_1} \times \ldots \times K_{N_M} \\ K_{N_i} &= \begin{cases} \{-\frac{N_i}{2}, \ldots, \frac{N_i}{2} - 1\} \text{ for } N_i \text{ even} \\ \{-\frac{N_i-1}{2}, \ldots, \frac{N_i-1}{2}\} \text{ for } N_i \text{ odd} \end{cases} \end{aligned}$$

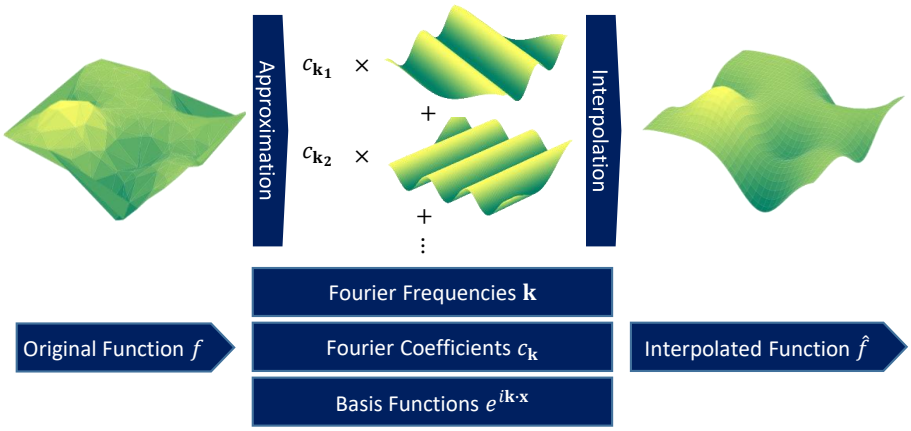


Fig. 2: Fourier Interpolation Layer. We use a Fourier series approximation of the original function u by fitting the Fourier coefficients $c_{\mathbf{k}}$ using LLS. The Fourier coefficients can be used to evaluate the approximation at any arbitrary collection of points $\mathbf{x}$ thus making it possible to interpolate the function onto a high-resolution grid.

converges pointwise to u *as* $N_i \to \infty$*, i.e.*

$$\textit{for each } \mathbf{x} \in [0,1]^M\textit{:} \lim_{\min N_i \to \infty} \hat{u}_N(\mathbf{x}) = u(\mathbf{x})$$

Proof. The proof of this theorem can be found in [47] or [38].

This theorem underpins various numerical methods and techniques, including the Fast Fourier Transform (FFT), which plays an important role in signal processing [8], data compression [44], and solving differential equations efficiently [24]. However, applying FFTs directly to non-uniformly sampled data presents challenges, as the traditional approach assumes data points are uniformly distributed along the domain [49,5]. To address this limitation, we employ an estimation technique to compute the Fourier coefficients $c_{\mathbf{k}}$ for non-uniformly sampled points as summarized in Figure 2.

Specifically, given the Fourier series expressed as Equation (1) and the function u measured at several non-uniform locations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_H\}$ we are trying to estimate the coefficients $c_{\mathbf{k}}, \mathbf{k} \in K$ such that the approximation error

$$\sum_{\mathbf{x} \in X} ||u(\mathbf{x}) - \hat{u}_N(\mathbf{x})||^2 \tag{2}$$

is minimized. $N = (N_1, \ldots, N_M)$ represents the number of frequencies in each dimension. We note that while the number of frequencies can be chosen separately for each dimension $1, \ldots, M$ in practice we keep this parameter constant as $N_1 = \ldots = N_M =: N_{freq}$.

We formulate the estimation process as a linear least squares (LLS) problem where the coefficients $c_{\mathbf{k}}$ are the unknowns. The LLS problem can be expressed as follows:

$$\min_{\mathbf{c}\in\mathbb{R}^{N_1\dots N_M}} ||\mathbf{A}\mathbf{c}-\mathbf{b}||^2 \tag{3}$$

where, $\mathbf{A}$ is a matrix representing the basis functions $e^{2i\pi\mathbf{k}\cdot\mathbf{x}}$ evaluated at the non-uniform sample points $\mathbf{x}$, i.e. $A=[a_{jl}]=[e^{2i\pi\mathbf{k}_l\cdot\mathbf{x}_j}]_{\mathbf{k}_l\in K,\mathbf{x}_j\in X}$, $\mathbf{c}$ is a vector containing the unknown Fourier coefficients: $\mathbf{c}=[c_{\mathbf{k}_l}]_{\mathbf{k}_l\in K}$ and $\mathbf{b}$ is a vector containing the values of the function u evaluated at the non-uniform sample points $\mathbf{x}$: $\mathbf{b}=[u(\mathbf{x}_j)]_{\mathbf{x}_j\in X}$

Solving this linear least squares system allows us to estimate the Fourier coefficients efficiently, enabling the interpolation $\tilde{u}$ of the function u onto a uniform grid of high resolution in Fourier space. We achieve this by evaluating the Fourier series approximation given by Equation (1) at new high-resolution grid location points $\mathbf{x}\in X_{grid}$ as follows:

Assuming $\mathbf{c}^*$ is the solution to the LLS problem given in Equation (3), we calculate the interpolated values at $\mathbf{x}\in X_{grid}$ as

$$\tilde{u}(\mathbf{x})=\Re(\sum_{\mathbf{k}\in K} c^*_{\mathbf{k}} e^{2i\pi\mathbf{k}\cdot\mathbf{x}}) \tag{4}$$

where $\Re(z)=a$ is the real part of a complex number $z=a+bi\in\mathbb{C}$.

This procedure originally only applies to the interpolation of periodic functions. In Appendix A we show how our approach can be extended to non-periodic functions.

Overall we call this interpolation procedure the FI (Fourier Interpolation) Layer which in summary takes as input a set of arbitrary locations $X\subset\mathbb{R}^M$ together with the corresponding values $u(X)$ of a function u and a set of target locations $X_{\text{target}}\subset\mathbb{R}^M$, transforms them into the Fourier coefficients $[c_{\mathbf{k}}]_{\mathbf{k}\in K}$. These are then used to evaluate the Fourier approximation at the set of target locations X_{target}. Overall this can be summarized as:

$$\text{FI}(X,u(X),X_{\text{target}}) \xrightarrow{\min_{\mathbf{c}\in\mathbb{R}^{|K|}}||\mathbf{A}\mathbf{c}-\mathbf{b}||^2} \mathbf{c}^* \xrightarrow{\Re(\sum_{\mathbf{k}\in K} c^*_{\mathbf{k}} e^{2i\pi\mathbf{k}\cdot\mathbf{x}})} \tilde{u}(X_{\text{target}}) \tag{5}$$

3.2 NeuralPDE

NeuralPDE were originally proposed by Dulny et al. [16] for modeling dynamical systems, particularly those governed by partial differential equations (PDEs). The core idea lies in representing the underlying PDE using the Method of Lines, which discretizes the spatial dimensions and transforms the PDE into a system of ordinary differential equations (ODEs). This discretization allows spatial derivatives to be represented as convolutional operations, making CNNs a natural choice for parametrization, which are then trained using a differentiable ODE solver [14].

Specifically, given measurements at grid locations $\mathbf{U}(t) = [U_{ij}(t)]$ of an underlying physical system u which is assumed to be governed by some partial differential equation:

$$\frac{\partial u}{\partial t} = F(u, \nabla u, \nabla^2 u, \ldots) \tag{6}$$

the task is to learn the function F from data and use it to forecast the state of the system at future timepoints $t+1, \ldots, t+T$.

The method of lines treats the system $\mathbf{U}$ as a collection of ordinary differential equations (ODEs), with the value of the system at each of the grid locations as one variable in the system. The governing Equation (6) then becomes an ODE

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \tilde{F}(\mathbf{U}) \tag{7}$$

with all partial derivatives $\nabla u, \nabla^2 u, \ldots$ replaced by finite difference approximations, at grid locations, e.g. $\frac{\partial U_{ij}}{\partial x} \sim U_{ij+1} - U_{ij-1}$.

The function $\tilde{F}$ can now be parametrized using a convolutional network θ, as every finite difference approximation can be learned using a convolution operator [16], e.g. $\frac{\partial U_{ij}}{\partial x} \sim U_{ij+1} - U_{ij-1}$ corresponds to the convolution $\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$.

The NeuralPDE model N_θ is trained using a differentiable ODE solver [14] to forecast the physical system u.

3.3 GrINd

The overall architecture of our model is displayed in Figure 1. We combine two Fourier Interpolation Layers with a spatiotemporal prediction model on gridded data to form the **Gr**id **I**nterpolation **N**etwork for Scattere**d** Observations (GrINd). We select NeuralPDE [16] as the spatiotemporal prediction model, as it showed best performance on the DynaBench Benchmark [17].

We first use a Fourier Interpolation Layer FI_1 to transform the scattered, low-resolution observations U_t^{low} at locations X_{low} at time t into a high-resolution grid interpolation $\hat{U}_t^{\text{high}}$ at grid locations X_{high}. In the high-resolution space we use the NeuralPDE model N_θ to forecast the state of the system at time $t+1$ as $\hat{U}_{t+1}^{\text{high}}$. Finally we transform the high resolution predictions back into the original space using another Fourier Interpolation Layer FI_1 to obtain the final predictions U_{t+1}^{low}. Overall this can be written as:

$$\mathrm{GrINd}(U_t^{\text{low}}) = \mathrm{FI}_2(X_{\text{high}}, \mathrm{N}_\theta(\mathrm{FI}_1(X_{\text{low}}, U_t^{\text{low}}, X_{\text{high}})), X_{\text{low}}) \tag{8}$$

4 Experiments

In this section we describe the experimental setup used for evaluating the performance of our proposed model.

4.1 Data

For our experiments we use the DynaBench [17] dataset proposed by Dulny et al., which serves as a standardized benchmark for evaluating machine learning models on dynamical systems with sparse observations. The dataset consists of six physical systems described by partial differential equations: advection, Burgers', Gas Dynamics, Kuramoto-Sivashinsky, Reaction-Diffusion and the Wave equation. The equations have been simulated on a high-resolution grid with periodic boundary conditions using the finite differences method [30] and recorded at spatially scattered locations. This mimics a real-world setting where measurement stations are typically also sparse and not located on a grid.

For our experiment we use the same version of the dataset on which the original benchmark experiments were performed [17]. It contains 7000 simulations of each physical system with varying initial conditions (splitted into 5000 train, 1000 validation and 1000 test simulations), each recorded over 201 time steps at 900 measurement locations.

For evaluating the performance of our proposed Fourier Interpolation layer, we use the raw high-resolution version of the dataset as ground truth for comparison with the output of the interpolation layer. Similar to Dulny et al., we also compare our results to models trained and evaluated on a version of the dataset containing observations on a 30×30 grid.

4.2 Baseline Models

We compare the performance of our proposed approach to the original models evaluated in the DynaBench benchmark [17]. These include four graph- and point-cloud based models: Point Transformer [51], Point GNN [48], Graph Kernel Network [3] and Graph PDE [20]. In our results we omit three models included in the original benchmark as they are strictly outperformed by all other models.

The baselines also include two models designed to work on grid data: NeuralPDE [16] and a Residual Network [19]. We omit the CNN baseline included in the original benchmark as it showed subpar performance.

Additionally we include the *persistence* baseline from the original benchmark which corresponds to predicting no change of the system at all.

4.3 Model Configuration

We use $N_{freq} = 18$ Fourier Frequencies in each Fourier Interpolation Layer, as this values shows the lowest interpolation error on most physical systems from the DynaBench dataset. The target interpolation points are distributed along a 64×64 grid, which is the original resolution of the simulations in the DynaBench dataset.

The NeuralPDE model architecture is taken from [17].

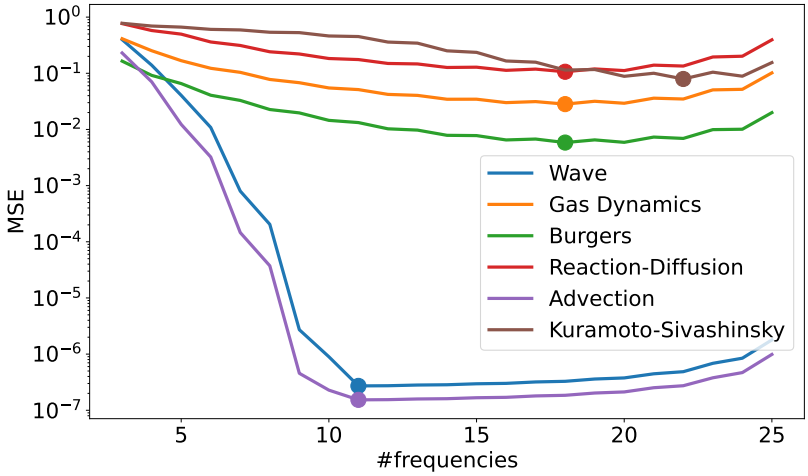


Fig. 3: Results of the interpolation experiment. The plot shows the Mean Squared Error between the interpolation given by our Fourier Interpolation Layer and the full resolution dataset as a function of the number of fourier frequencies used. The lowest interpolation error for each dataset has been marked with a dot.

4.4 Training

We replicate the original training setting from the DynaBench benchmark as closely as possible. The models are trained to predict the next simulation step and evaluated in a closed-loop rollout on 16 simulation steps. We train the models with the Mean Squared Error loss using the Adam Optimizer [22] with a learning rate of 0.0001.

All models are trained for a maximum of 100 000 optimization steps while monitoring the 16 step rollout MSE on a separate validation dataset every 1000 steps with early stopping if the metric did not improve after 5 consecutive checks. Afterwards the best checkpoint with respect to the validation metric is used for testing.

We implemented our models using PyTorch [37] and Pytorch Lightning [18]. The source code of our models and the performed experiments is available under `https://professor-x.de/grind`.

5 Results and Discussion

In this section we present the experiments we performed and discuss the results.

5.1 Interpolation accuracy

Firstly, to showcase the feasibility of our proposed approach, we evaluate the performance of our proposed Fourier Interpolation Layer on the task of directly interpolating a set of scattered observations onto a high resolution grid. To this end, we use the DynaBench dataset with 900 measurement locations as input to a single Fourier Interpolation Layer with interpolation target points distributed on a 64×64 grid over the simulation domain. We compare the output of the Fourier Interpolation Layer to the *full* resolution version of the DynaBench dataset which was used to sample the scattered observations. We quantify the interpolation error using the *Mean Squared Error* (MSE) metric. The experiment is repeated for different selection of frequencies N_{freq}, varying between 3 and 25 (cf. Section 3.1).

Figure 3 illustrates the outcomes of the interpolation experiment conducted across six distinct physical systems within the DynaBench dataset. Across all systems, a consistent trend emerges, characterized by a distinctive "U" shape in the performance curves. This trend is intuitively comprehensible: when a limited number of Fourier frequencies are chosen, the Fourier approximation fails to capture the intricate nature of the function adequately. Conversely, employing a high number of Fourier frequencies leads to overfitting of the Fourier Interpolation Layer, resulting in elevated reconstruction errors.

Furthermore, it becomes apparent that the optimal number of frequencies varies depending on the complexity of the system, suggesting that individual selection for each dataset is ideal. However, the experiments also show a relative flat minimum ranging from 10 to 20 frequencies per dimension, suggesting that our model is relatively robust with respect to this hyperparameter. Our results for the Dynabench datasets with a lower number of observations show similar results (cf. Appendix B). Therefore, we suggest setting the total number of Fourier frequencies in the range between $\frac{1}{3}$ and $\frac{2}{3}$ of the number of observations.

For our experiments on the DynaBench datasets we use 18 frequencies per dimension which achieves the empirical minimal reconstruction error.

5.2 DynaBench

In this section, we present the results of our proposed GrINd model on the DynaBench dataset, comparing its performance against the baseline models described in Section 4.2.

Figure 4 shows the rollout results of our model over 16 steps compared to the baselines from the DynaBench dataset. Overall, with the exception of the *Reaction-Diffusion* and *Kuramoto-Sivashinsky* equations, our model outperforms all models that work on scattered data (PointGNN, Point Transformer, KernelNN and GraphPDE) by a large margin. This is especially true over longer prediction horizons.

In case of the *Advection*, *Gas Dynamics* and *Wave* equation, our model performs on par with grid models, which get access to a grid structured array of observation locations. While the grid models still get access to the same amount

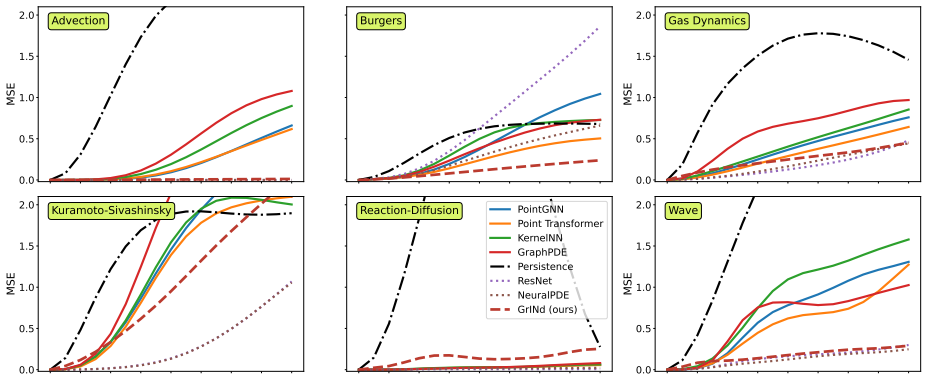


Fig. 4: Rollout results on the DynaBench dataset for 16 steps. The graph and point cloud baselines are plotted with solid lines, the grid baselines with dotted lines, the persistence baseline with a dash-dotted line and our model with a dashed line. Results of our model have been averaged over 10 runs. Baseline results taken from [17]. For better readability MSE scores over 2.0 are not displayed.

of measurement location (30×30 grid vs 900 scattered observations), this additional structure has been shown to be beneficial for predicting the evolution of a physical system [17].

Finally, in case of the *Burgers'* equation our model even outperforms the best grid model from the DynaBench benchmark. This is possible because our model interpolates the system onto a higher-resolution grid than was used to train the baseline models. We hypothesize that in this case the performance gain from using a higher grid outweighs the performance loss from the interpolation error (cf. Section 5.1).

Single Prediction Steps We additionally numerically evaluate the performance of our model both on single and multi-step predictions. Table 1 displays the Mean Squared Error (MSE) after a single prediction step for each physical system in the DynaBench dataset. While in this case our GrINd model still achieves competitive performance, it is however outperformed by other grid and non-grid based models with the exception of the *Burgers'* equation. We observe that in this case the numerical interpolation error, as examined in Section 5.1 is larger than the numerical forecasting error for the baseline models. We hypothesize that the gain from mapping the observations onto a high-resolution grid is not sufficient to outweigh the interpolation error. This is also supported by the conclusion from the original DynaBench benchmark, which suggests that for shorter prediction horizons grid models perform on par with non-grid models.

Multiple Prediction Steps We further assess the predictive capabilities of GrINd by evaluating its performance after 16 prediction steps. As shown in Table 2, for longer prediction horizons GrINd consistently outperforms non-grid based models in terms of MSE. For the *Burgers'* equation, our model even out-

Table 1: MSE after 1 prediction step. The best perfoming model for each equation has been underlined. Additionally, the best non-grid model has been underwaved. A = Advection, B = Burgers', GD = Gas Dynamics, KS = Kuramoto-Sivashinsky, RD = Reaction-Diffusion, W = Wave. Results of our model have been averaged over 10 runs. Baseline results taken from [17].

model	A	B	GD	KS	RD	W
GrINd(ours)	$1.92 \cdot 10^{-4}$	$6.56 \cdot 10^{-3}$	$4.93 \cdot 10^{-2}$	$4.54 \cdot 10^{-2}$	$2.33 \cdot 10^{-2}$	$3.61 \cdot 10^{-2}$
GraphPDE	$1.37 \cdot 10^{-4}$	$1.07 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	$7.20 \cdot 10^{-3}$	$1.42 \cdot 10^{-4}$	$2.07 \cdot 10^{-3}$
KernelNN	$6.31 \cdot 10^{-5}$	$1.06 \cdot 10^{-2}$	$1.34 \cdot 10^{-2}$	$6.69 \cdot 10^{-3}$	$1.87 \cdot 10^{-4}$	$5.43 \cdot 10^{-3}$
Point TF	$4.42 \cdot 10^{-5}$	$1.03 \cdot 10^{-2}$	$7.25 \cdot 10^{-3}$	$4.90 \cdot 10^{-3}$	$1.41 \cdot 10^{-4}$	$2.38 \cdot 10^{-3}$
PointGNN	$2.82 \cdot 10^{-5}$	$8.83 \cdot 10^{-3}$	$9.02 \cdot 10^{-3}$	$6.73 \cdot 10^{-3}$	$1.36 \cdot 10^{-4}$	$1.39 \cdot 10^{-3}$
NeuralPDE	$8.24 \cdot 10^{-7}$	$1.12 \cdot 10^{-2}$	$3.73 \cdot 10^{-3}$	$5.37 \cdot 10^{-4}$	$3.03 \cdot 10^{-4}$	$1.70 \cdot 10^{-3}$
ResNet	$2.16 \cdot 10^{-6}$	$1.48 \cdot 10^{-2}$	$3.21 \cdot 10^{-3}$	$4.90 \cdot 10^{-4}$	$1.57 \cdot 10^{-4}$	$1.46 \cdot 10^{-3}$
Persistence	$8.12 \cdot 10^{-2}$	$3.68 \cdot 10^{-2}$	$1.87 \cdot 10^{-1}$	$1.42 \cdot 10^{-1}$	$1.47 \cdot 10^{-1}$	$1.14 \cdot 10^{-1}$

performs the best grid baseline. As opposed to single step predictions, longer prediction horizons require more numerical stability which is not guaranteed by models working on unstructured data. In this case the benefit of using a grid representation outweighs the error associated with using the Fourier Interpolation Layer. The results for 16 prediction steps highlight our model's robustness in capturing the dynamics of physical systems over longer forecasting horizons.

Overall, the results demonstrate that GrINd is a promising approach for forecasting physical systems from sparse, scattered observational data. This is especially the case for longer prediction horizons, where our model shows more numerical stability than other models able to work with scattered data, similar to grid models like NeuralPDE [16] or ResNet [19].

5.3 Limitations

One of the limitations of our method is its computational cost. For a given Fourier interpolation layer with K^2 total Fourier coefficients, N_1 input points and N_2 target points, the complexity of the interpolation algorithm consists of $O(K^4 N_1)$ operations to calculate the Fourier coefficients using a least squares algorithm [31] and $O(N_2 K^2)$ operations to calculate the interpolated values in the target space. Considering the optimal number of frequencies is linearly dependent on the number of input points (cf. Section 5.1) the algorithm has a cubic complexity in terms of number of input points.

However, our model is designed for applications in domains with sparse observations, where the number of input points is low. In our experiments we found that in practice the Fourier Interpolation layer does not take up considerable compute time compared to the actual prediction model.

Table 2: MSE after 16 prediction steps. The best perfoming model for each equation has been underlined. Additionally, the best non-grid model has been underwaved. A = Advection, B = Burgers', GD = Gas Dynamics, KS = Kuramoto-Sivashinsky, RD = Reaction-Diffusion, W = Wave. Results of our model have been averaged over 10 runs. Baseline results taken from [17].

model	A	B	GD	KS	RD	W
GrIND (ours)	$1.23 \cdot 10^{-1}$	$2.38 \cdot 10^{-1}$	$4.51 \cdot 10^{-1}$	$2.33 \cdot 10^{0}$	$2.53 \cdot 10^{-1}$	$2.90 \cdot 10^{-1}$
GraphPDE	$1.08 \cdot 10^{0}$	$7.30 \cdot 10^{-1}$	$9.69 \cdot 10^{-1}$	$2.10 \cdot 10^{0}$	$8.00 \cdot 10^{-2}$	$1.03 \cdot 10^{0}$
KernelNN	$8.97 \cdot 10^{-1}$	$7.27 \cdot 10^{-1}$	$8.54 \cdot 10^{-1}$	$2.00 \cdot 10^{0}$	$6.35 \cdot 10^{-2}$	$1.58 \cdot 10^{0}$
Point TF	$6.17 \cdot 10^{-1}$	$5.04 \cdot 10^{-1}$	$6.43 \cdot 10^{-1}$	$2.10 \cdot 10^{0}$	$5.64 \cdot 10^{-2}$	$1.27 \cdot 10^{0}$
PointGNN	$6.61 \cdot 10^{-1}$	$1.04 \cdot 10^{0}$	$7.59 \cdot 10^{-1}$	$2.82 \cdot 10^{0}$	$5.82 \cdot 10^{-2}$	$1.31 \cdot 10^{0}$
NeuralPDE	$2.70 \cdot 10^{-4}$	$6.60 \cdot 10^{-1}$	$4.43 \cdot 10^{-1}$	$1.06 \cdot 10^{0}$	$2.24 \cdot 10^{-2}$	$2.48 \cdot 10^{-1}$
ResNet	$8.65 \cdot 10^{-5}$	$1.86 \cdot 10^{0}$	$4.80 \cdot 10^{-1}$	$1.07 \cdot 10^{0}$	$7.05 \cdot 10^{-3}$	$2.99 \cdot 10^{-1}$
Persistence	$2.39 \cdot 10^{0}$	$6.79 \cdot 10^{-1}$	$1.46 \cdot 10^{0}$	$1.90 \cdot 10^{0}$	$2.76 \cdot 10^{-1}$	$2.61 \cdot 10^{0}$

6 Conclusion and Future Work

In this paper, we introduced GrINd (Grid Interpolation Network for Scattered Observations), a novel approach designed to address the challenge of forecasting physical systems from sparse, scattered observational data. GrINd leverages the high performance of grid-based deep learning approaches for the task of forecasting physical systems from sparse and scattered data. It combines a novel Fourier Interpolation Layer with a NeuralPDE model to predict the evolution of spatiotemporal physical systems.

The empirical evaluations on the DynaBench benchmark dataset reveal that GrINd outperforms existing models, particularly for longer prediction horizons, showcasing the potential of using interpolation-based methods for modeling physical systems, as grid-based models show more numerical stability. Our work also opens up several avenues for future research and development that we summarize in the following.

In future work we would like to investigate the performance of our model on real-world data. One example application includes the weather prediction task, where predominantly the ERA5 reanalysis dataset [12] is used for training deep learning models. While the dataset combines observations with grid-based numerical simulations, our model allows for incorporating live and historical observations directly, by interpolating the scattered measuring stations onto a grid.

While we use the NeuralPDE model in our architecture, as it showed best perfomance in the DynaBench benchmark [17], in principle our approach allows for any grid-based model to be used in its placer. Future investigations could explore the integration of other such models to further improve predictive performance.

Another promising direction for future work is using a permutation-invariant neural network such as Deep Sets [50] or Set Transformers [28] to predict the Fourier Coefficients. This could improve the interpolation layer by enabling pretraining on a wide range of data. Additionally, learning the interpolation layer parameters alongside the NeuralPDE model could lead to better alignment between the interpolation and prediction processes, enhancing overall forecasting accuracy.

Instead of interpolating the data onto a high-resolution grid before learning the dynamics, an alternative approach could be to directly learn the dynamics in Fourier space using neuralODEs [14]. By operating directly in the frequency domain, the model may capture underlying patterns and dependencies more effectively, potentially improving forecasting accuracy.

In summary, GrINd represents a significant contribution in forecasting physical systems from sparse observational data, by leveraging the already high performance of grid models. However, there are still ample opportunities for further research and development to enhance the approach and address additional challenges in predictive modeling for spatiotemporal physical systems.

Disclosure of Interests. The project underlying this publication was funded by the German Federal Ministry of Education and Research under the grant number 16DKWN0099B (MAGNET4Cardiac7T). The responsibility for the content of this publication lies with the authors.

References

1. Alkin, B., Furst, A., Schmid, S., Gruber, L., Holzleitner, M., Brandstetter, J.: Universal physics transformers (2024)
2. Anandkumar, A.: Neural operators for solving pdes and inverse design. In: Proceedings of the 2023 International Symposium on Physical Design. p. 195. ISPD '23, Association for Computing Machinery, New York, NY, USA (Mar 2023). https://doi.org/10.1145/3569052.3578911
3. Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., Stuart, A.: Neural Operator: Graph Kernel Network for Partial Differential Equations. In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (Jun 2019)
4. Ayed, I., de Bézenac, E., Pajot, A., Brajard, J., Gallinari, P.: Learning dynamical systems from partial observations. CoRR **abs/1902.11136** (2019). https://doi.org/10.48550/arXiv.1902.11136
5. Barnett, A.H.: Aliasing error of the exp (β1-z2) kernel in the nonuniform fast fourier transform. Applied and Computational Harmonic Analysis **51**, 1–16 (2021). https://doi.org/https://doi.org/10.1016/j.acha.2020.10.002
6. Barnett, A.H., Magland, J., af Klinteberg, L.: A parallel nonuniform fast fourier transform library based on an "exponential of semicircle" kernel. SIAM Journal on Scientific Computing **41**(5), C479–C504 (Jan 2019). https://doi.org/10.1137/18m120885x
7. Bauer, P., Thorpe, A., Brunet, G.: The quiet revolution of numerical weather prediction. Nature **525**(7567), 47–55 (Sep 2015). https://doi.org/10.1038/nature14956

8. Bi, G., Zeng, Y.: Transforms and Fast Algorithms for Signal Analysis and Representations. Birkhäuser Boston (2004). https://doi.org/10.1007/978-0-8176-8220-0
9. Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., Tian, Q.: Accurate medium-range global weather forecasting with 3D neural networks. Nature **619**(7970), 533–538 (Jul 2023). https://doi.org/10.1038/s41586-023-06185-3
10. Boussard, J., Nagda, C., Kaltenborn, J., Lange, C.E.E., Brouillard, P., Gurwicz, Y., Nowack, P., Rolnick, D.: Towards causal representations of climate model data (2023)
11. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences **113**(15), 3932–3937 (2016). https://doi.org/10.1073/pnas.1517384113
12. C3S: Era5 hourly data on single levels from 1940 to present (2018). https://doi.org/10.24381/CDS.ADBB2D47, https://cds.climate.copernicus.eu/doi/10.24381/cds.adbb2d47
13. Cannon, R.H.: Dynamics of Physical Systems. McGraw-Hill (1967), google-Books-ID: Rix6s2VIlOkC
14. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural Ordinary Differential Equations. In: Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)
15. Cullen, M.J., Davies, T., Mawson, M.H., James, J.A., Coulter, S.C., Malcolm, A.: An Overview of Numerical Methods for the Next Generation U.K. NWP and Climate Model. Atmosphere-Ocean **35**(sup1), 425–444 (Jan 1997). https://doi.org/10.1080/07055900.1997.9687359
16. Dulny, A., Hotho, A., Krause, A.: NeuralPDE: Modelling Dynamical Systems from Data. In: Bergmann, R., Malburg, L., Rodermund, S.C., Timm, I.J. (eds.) KI 2022: Advances in Artificial Intelligence. pp. 75–89. Lecture Notes in Computer Science, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-15791-2_8
17. Dulny, A., Hotho, A., Krause, A.: DynaBench: A Benchmark Dataset for Learning Dynamical Systems from Low-Resolution Data. In: Koutra, D., Plant, C., Gomez Rodriguez, M., Baralis, E., Bonchi, F. (eds.) Machine Learning and Knowledge Discovery in Databases: Research Track. pp. 438–455. Lecture Notes in Computer Science, Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-43412-9_26
18. Falcon, W., The PyTorch Lightning team: PyTorch Lightning (Mar 2019). https://doi.org/10.5281/zenodo.3828935
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (Jun 2016). https://doi.org/10.1109/CVPR.2016.90, ISSN: 1063-6919
20. Iakovlev, V., Heinonen, M., Lähdesmäki, H.: Learning continuous-time {pde}s from sparse data with graph neural networks. In: International Conference on Learning Representations (2021)
21. Jiang, L., Wang, L., Chu, X., Xiao, Y., Zhang, H.: Phygnnet: Solving spatiotemporal pdes with physics-informed graph neural network. In: Proceedings of the 2023 2nd Asia Conference on Algorithms, Computing and Machine Learning. p. 143–147. CACML '23, Association for Computing Machinery, New York, NY, USA (May 2023). https://doi.org/10.1145/3590003.3590029
22. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR). San Diega, CA, USA (2015)

23. Kleinstreuer, C.: Modern Fluid Dynamics: Basic Theory and Selected Applications in Macro- and Micro-Fluidics, Fluid Mechanics and Its Applications, vol. 87. Springer Netherlands, Dordrecht (2010). https://doi.org/10.1007/978-1-4020-8670-0
24. Kopriva, D.A.: Implementing Spectral Methods for Partial Differential Equations. Springer Netherlands (2009). https://doi.org/10.1007/978-90-481-2261-5
25. Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., Anandkumar, A.: Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators (2022)
26. Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., Battaglia, P.: Learning skillful medium-range global weather forecasting. Science (Dec 2023). https://doi.org/10.1126/science.adi2336
27. Larsson, S., Thomée, V.: Partial Differential Equations with Numerical Methods. Springer Berlin Heidelberg (2003). https://doi.org/10.1007/978-3-540-88706-5
28. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 3744–3753. PMLR (09–15 Jun 2019)
29. Lessig, C., Luise, I., Schultz, M.: AtmoRep: Large Scale Representation Learning for Atmospheric Data. In: EGU General Assembly Conference Abstracts. pp. EGU–3117. EGU General Assembly Conference Abstracts (May 2023). https://doi.org/10.5194/egusphere-egu23-3117
30. LeVeque, R.J.: Finite Difference Methods for Ordinary and Partial Differential Equations. Society for Industrial and Applied Mathematics (2007). https://doi.org/10.1137/1.9780898717839
31. Li, L.: A new complexity bound for the least-squares problem. Computers & Mathematics with Applications **31**(12), 15–16 (1996). https://doi.org/https://doi.org/10.1016/0898-1221(96)00072-7, https://www.sciencedirect.com/science/article/pii/0898122196000727
32. Li, Z., Shu, D., Farimani, A.: Scalable transformer for pde surrogate modeling (2023)
33. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., Anandkumar, A.: Multipole Graph Neural Operator for Parametric Partial Differential Equations. In: Advances in Neural Information Processing Systems. vol. 33, pp. 6755–6766. Curran Associates, Inc. (2020)
34. Li, Z., Kovachki, N.B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. In: International Conference on Learning Representations (2021)
35. Long, Z., Lu, Y., Dong, B.: PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. Journal of Computational Physics **399**(C) (Dec 2019). https://doi.org/10.1016/j.jcp.2019.108925
36. Nathaniel, J., Qu, Y., Nguyen, T., Yu, S., Busecke, J., Grover, A., Gentine, P.: Chaosbench: A multi-channel, physics-based benchmark for subseasonal-to-seasonal climate prediction (2024)
37. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)

38. Potts, D., Schmischke, M.: Approximation of high-dimensional periodic functions with Fourier-based methods. SIAM Journal on Numerical Analysis **59**(5), 2393–2429 (jan 2021). https://doi.org/10.1137/20M1354921, arXiv:1907.11412 [cs, math]
39. Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M.V., Nowak, W.: Finite volume neural network: Modeling subsurface contaminant transport. CoRR **abs/2104.06010** (2021). https://doi.org/10.48550/arXiv.2104.06010
40. Pratt, H., Williams, B., Coenen, F., Zheng, Y.: Fcnn: Fourier convolutional neural networks. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 17. pp. 786–798. Springer (2017)
41. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations (Nov 2017). https://doi.org/10.48550/arXiv.1711.10561, type: article
42. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations (Nov 2017). https://doi.org/10.48550/arXiv.1711.10566, type: article
43. Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russel, T., Sanchez-Gonzalez, A., Yang, V.Q., Carver, R., Agrawal, S., Chantry, M., Bouallègue, Z.B., Dueben, P., Bromberg, C., Sisk, J., Barrington, L., Bell, A., Sha, F.: Weatherbench 2: A benchmark for the next generation of data-driven global weather models (2023)
44. Redinbo, G., Manomohan, R.: Fault-tolerant fft data compression. In: Proceedings. 2000 Pacific Rim International Symposium on Dependable Computing. pp. 110–119 (2000). https://doi.org/10.1109/PRDC.2000.897293
45. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.W.: Learning to simulate complex physics with graph networks. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20, JMLR.org (2020)
46. Schiesser, W.E.: The numerical method of lines : integration of partial differential equations. San Diego : Academic Press (1991)
47. Serov, V.: Fourier Series, Fourier Transform and Their Applications to Mathematical Physics. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-65262-7
48. Shi, W., Rajkumar, R.: Point-GNN: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1711–1719 (2020)
49. Shih, Y.h., Wright, G., Andén, J., Blaschke, J., Barnett, A.H.: cufinufft: a load-balanced gpu library for general-purpose nonuniform ffts. In: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 688–697 (2021). https://doi.org/10.1109/IPDPSW52791.2021.00105
50. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep Sets. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
51. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 16259–16268 (October 2021)

A Extension of the FI Layer to non-periodic functions

Originally our FI Layer can only be applied to periodic functions. In practice however, our approach can be extended to any non-periodic function f as follows: We define a surrogate periodic function $g\colon \Omega \to \mathbb{C}$ as

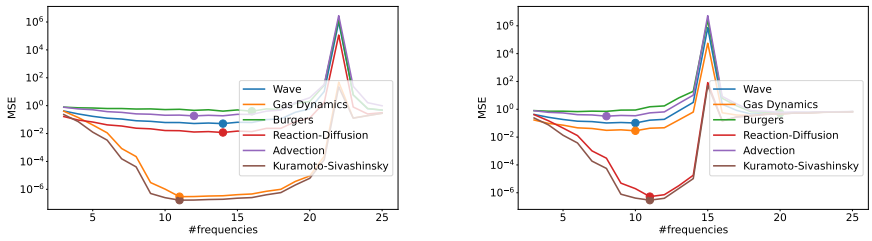
$$g(\mathbf{x}) = g(x_1, x_2, \ldots, x_M) := f(\phi(x_1), \ldots, \phi(x_M)) \tag{9}$$

with $\phi\colon \mathbb{R} \to \mathbb{R}$ defined as $\phi(x) = 2|\frac{1}{2} - x|$.

Intuitively, this corresponds to mirroring the function once in each dimension and concatenating the results along each dimension once. This results in 2^M copies of the original function f each flipped along one or more dimensions, which is periodic in the original domain. The original function f can be recovered and interpolated in the subdomain $\Omega_{sub} = [\frac{1}{2}, 1]^M \subset \Omega$.

B Additional interpolation experiments

We perform additional interpolation experiments on the DynaBench dataset, in which we compare the output of the Fourier Layer with a lower number of observation points (225 and 484) to the full resolution simulation. The results are shown in Figure 5.



(a) *medium* (b) *low*

Fig. 5: Results of the interpolation experiment on the *medium* version of the DynaBench dataset. The plot shows the Mean Squared Error between the interpolation given by our Fourier Interpolation Layer and the full resolution dataset as a function of the number of Fourier frequencies used. The lowest interpolation error for each dataset has been marked with a dot.

Overall the interpolation results for the lower resolution data show that the optimal number of frequencies correlates with the number of observation points. This optimum lies in the interval $[11, 16]$ for the medium resolution version and in the interval $[8, 11]$ for the low resolution version, depending on the type of physical system. When using the Fourier Interpolation Layer for a given number n of observation points this corresponds to the optimal number of frequencies of around $0.6\sqrt{n}$.