Group Equivariant Fourier Neural Operators for Partial Differential Equations

Jacob Helwig [1 *] **Xuan Zhang** [1 *] **Cong Fu** [1] **Jerry Kurtin** [1] **Stephan Wojtowytsch** [2] **Shuiwang Ji** [1]

Abstract

We consider solving partial differential equations (PDEs) with Fourier neural operators (FNOs), which operate in the frequency domain. Since the laws of physics do not depend on the coordinate system used to describe them, it is desirable to encode such symmetries in the neural operator architecture for better performance and easier learning. While encoding symmetries in the physical domain using group theory has been studied extensively, how to capture symmetries in the frequency domain is under-explored. In this work, we extend group convolutions to the frequency domain and design Fourier layers that are equivariant to rotations, translations, and reflections by leveraging the equivariance property of the Fourier transform. The resulting G-FNO architecture generalizes well across input resolutions and performs well in settings with varying levels of symmetry. Our code is publicly available as part of the AIRS library (https://github.com/divelab/AIRS).

1. Introduction

Partial differential equations (PDEs) are widely used to model physical processes that evolve in time and space, including fluid flows (Wang et al., 2020; Bonnet et al., 2022; Eckert et al., 2019), heat transfer (Zobeiry & Humfeld, 2021) and electromagnetic waves (Lim & Psaltis, 2022). Classically, solving a PDE has been viewed as the task of finding a sufficiently smooth function that satisfies a pointwise relationship between derivatives of a different order. A more modern approach is to consider differential operators as (often non-linear) maps between function spaces and utilize techniques of functional analysis to construct and analyze solutions. The first philosophy is present in neural PDE solvers such as physics-informed neural networks (PINNs) (Raissi et al., 2019; Lu et al., 2021b), whereas the second is pursued by neural operators (Lu et al., 2021a; Li et al., 2021a). While PINNs are used to solve equations individually and online, neural operators learn a solution map between function spaces from problem data to the solution offline. The second approach is highly efficient in contexts where the same problem has to be solved often with slightly varied parameters or initial conditions and is the focus of this work.

PDEs capture dynamics of physical processes in which symmetries exist, as visualized in Figure 1. Symmetries of the underlying problem are reflected in the PDE and its solution operator: the laws of physics do not depend on the coordinate system used to describe them. Many differential operators are rotation invariant, including common models for fluid flow, heat propagation and electrodynamics, but asymmetries in the domain of computation can break symmetries in a more global way: for instance, two directions in a cylindrical container behave similarly while the third plays a different role. It is therefore *global* symmetries that solution operators capture. Explicit encoding of these symmetries in network architectures can improve model generalization, interpretability, and sample complexity (Weiler & Cesa, 2019; Worrall & Welling, 2019).

While equivariant architectures have been studied in diverse applications (Thomas et al., 2018; Cohen & Welling, 2017; Weiler et al., 2018; Cohen et al., 2018), most current studies parameterize their convolution kernels in physical space or group space, as opposed to Fourier space. Therefore, the networks are constrained to the resolution of the training data. That is, the trained models may not generalize well to data sampled on a discretization differing from that used in the training data. Additionally, the kernel is most often assumed to have compact, local support, which is effective for sharing information at short distances but requires deep architectures for long-range signal propagation, whereas Fourier convolutions offer an efficient approach for performing global convolutions (Li et al., 2021a).

In this work, we propose the Group Equivariant Fourier Neural Operator (G-FNO). By leveraging symmetries of the Fourier transform, we extend group convolution to the frequency domain and design Fourier layers that are equivariant to rotations, reflections and translations. As a result,

[*]Equal contribution [1]Department of Computer Science & Engineering, Texas A&M University, TX, USA [2]Department of Mathematics, Texas A&M University, TX, USA. Correspondence to: Shuiwang Ji <sji@tamu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

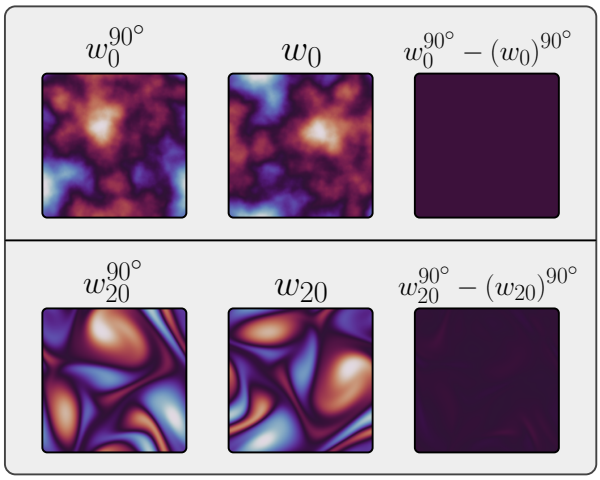


Figure 1: Demonstration of symmetries in the Navier-Stokes equations. We numerically solve the Navier-Stokes equations with a symmetric forcing term as studied in Section 4.3. We find that rotating the initial vorticity field corresponds to a rotated solution vorticity at time $t = 20$ up to numerically introduced artifacts, as seen in the rightmost column. We formally derive this symmetry in Appendix C.3.

the proposed G-FNO leverages symmetries, can generalize across discretizations, and performs a global convolution that efficiently processes information on multiple scales. Experiments show that G-FNO significantly improves the accuracy of PDE solutions even under imperfect symmetries and can generalize to a higher resolution at test time.

2. Related Work

2.1. Neural PDE Solvers

Neural PDE solvers use neural networks to solve PDEs. Physics-informed neural networks (PINNs) (Raissi et al., 2019) parameterize solution functions with neural networks and optimize network parameters to satisfy the constraints imposed by a PDE. In contrast, neural operators (Lu et al., 2021a; Li et al., 2021a) learn mappings between input and output functions directly from training data and can be trained autoregressively (Li et al., 2021a; Brandstetter et al., 2022b). As their combination, physics-informed neural operators (Wang et al., 2021b; Li et al., 2021b) make use of explicit PDE constraints to help neural operators better satisfy underlying physics. Bar-Sinai et al. (2019); Um et al. (2020); Kochkov et al. (2021) integrate neural solvers and classical solvers (Holl et al., 2020) by using trained neural networks to reduce the numerical error on coarse grids. Our work builds upon Fourier Neural Operators (Li et al., 2021a; Kovachki et al., 2023), which have shown promising results in solving PDEs.

2.2. Fourier Neural Operator

Fourier neural operators (FNOs) (Li et al., 2021a; 2020; 2022a; Kovachki et al., 2023) learn to solve PDEs by performing global convolutions via Fourier layers, which are implemented efficiently in the frequency domain using the Fast Fourier Transform (FFT). FNOs process local and global information in parallel through high and low frequency modes (Gupta & Brandstetter, 2022), and reduce computational cost by truncating the highest frequency modes to zero. Additionally, since the convolution operator is learned in the frequency domain, the network is theoretically independent of the resolution of the training data, enabling FNOs to generalize to higher resolution during testing, a task termed *zero-shot super-resolution* (Li et al., 2021a; Boussif et al., 2022).

FNOs have appeared in a variety of applications for dynamics modeling, including optimal control (Hwang et al., 2022), modeling of coastal dynamics (Jiang et al., 2021), modeling of turbulent Kolmogorov flows (Li et al., 2022b), solving stochastic differential equations (Salvi et al., 2022), and forecasting global weather trends (Pathak et al., 2022). Beyond dynamics modeling, Guibas et al. (2022) use Fourier layers to replace spatial self-attention for computer vision tasks.

The FNO architecture has also been extended in recent studies. Poli et al. (2022) propose a new weight initialization scheme and improved efficiency by only applying one Fourier transform per forward propagation. Tran et al. (2023) enable deeper stacks of Fourier layers by applying transforms independently along each of the spatial axes of the input and by proposing a new training strategy. Instead of network design and training, our work focuses on integrating symmetries into FNO architectures by extending group equivariant convolutions to the frequency domain. Specifically, the proposed method parameterizes convolution kernels in the Fourier-transformed group space, and in doing so, allows for a global convolution operator that is equivariant to rotations, reflections, and translations, and can furthermore perform zero-shot super-resolution.

3. Methods

Fourier Neural Operators (FNOs) learn operators mapping an input function to the solution function. For example, for time-dependent PDEs, the input function could be the solution at the current time step, and the output could be the solution at the next time step.

Inspired by the Green's function representation of PDE solutions, FNO alternates between the application of a fixed non-linear map and learned integral operators $\mathcal{K}$, defined as $(\mathcal{K}v)(x) = \int \kappa(x, y)v(y)\mathrm{d}y$, where κ is the Green's function to be learned. By further assuming κ to be invariant

to translations, we can rewrite κ as $\kappa(x, y) = \psi(x - y)$. Consequently, the operator defines a convolution in physical space as

$$(\mathcal{K}v)[x] = \int \psi(x - y)v(y)\mathrm{d}y. \tag{1}$$

By the Convolution Theorem, convolution in physical space can be efficiently implemented as element-wise multiplication in the frequency domain, which gives

$$(\mathcal{K}v)[x] = \mathcal{F}^{-1}\left(\mathcal{F}\psi \cdot \mathcal{F}v\right)[x], \tag{2}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and the inverse Fourier transform. Instead of learning the kernel ψ, FNO directly learns the Fourier transform of ψ, $\mathcal{F}\psi$.

For a translation-invariant Green's function to exist, a PDE must have two properties: linearity and translation-invariance. In practice, most PDEs of interest are non-linear, which FNOs address by including non-linear operations in the learned solution operator. For models in physical sciences, translation-invariance corresponds to the homogeneity of physical space. The assumption is violated in models of heterogeneous materials or boundary phenomena, but generally applies to many situations in fluid mechanics and beyond. If the model is additionally isotropic, the Green's function only depends on the distance $\|x - y\|$, not the direction of $x - y$. In such situations, it is sensible to use computational models which are based on the Green's function representation and respect the invariances it encodes, but allow for non-linear operations when solving non-linear PDEs.

3.1. Encoding Symmetries in the Physical Domain

Equivariant architectures have previously been realized using group convolutions (G-convolutions) via physically-parameterized kernel functions (Cohen & Welling, 2016). It is well-known that convolutional neural networks (CNNs) achieve translation equivariance through convolutional weight-sharing across spatial locations (LeCun et al., 1998). Group equivariant CNNs achieve equivariance to symmetry groups beyond translation by convolving feature maps and kernels defined on these groups. Given a group G, G-convolutions are defined as

$$(f \star \psi)[g] = \sum_{h \in G} \sum_{j=1}^{d_z} f^j(h)\left(L_g\psi^j\right)(h), \tag{3}$$

where both the feature map f and kernel ψ are functions on G mapping to $\mathbb{R}^{d_z}$, and $L_g\lambda := \lambda \circ g^{-1}$ is the group action of G on a function λ. Cohen & Welling (2016) proved the G-equivariance of G-convolutions. Taking $G = \mathbb{Z}^2$, the group of planar translations, G-convolution reduces to the conventional translation-equivariant convolution, i.e., $(f \star \psi)(x) = \sum_{y \in \mathbb{Z}^2} \sum_{j=1}^{d_z} f^j(y)\psi^j(y - x)$, where $(L_x\psi)(y) := \psi(y - x)$ for $x \in \mathbb{Z}^2$.

For the group of translations and 90° rotations (the group $p4$), a single-channel G-convolution kernel is efficiently implemented as a stack of four independent filters defined on $\mathbb{Z}^2$, each representing a rotation by 90°. Applying the group action L_g to the kernel, as in Equation (3), applies a roto-translation to each of the filters and cyclically shifts their order in the stack. The group $p4m$ additionally considers reflection symmetries and increases the number of independent filters to 8. We review how these groups transform functions defined on G and $\mathbb{Z}^2$ in more depth in Appendix C.1.

In the context of solving PDEs, Wang et al. (2021a) apply equivariant models leveraging a range of symmetries for modeling the temporal evolution of the velocity field of ocean currents. This work was later extended to settings where dynamics are only approximately equivariant by relaxing the symmetry constraint on learned kernels (Wang et al., 2022). However, in both cases, the convolution kernel was parameterized in physical space, tying the network to the resolution of the training data and lacking parallel processing of multiscale information as is inherent to Fourier convolutions (Gupta & Brandstetter, 2022). Cohen et al. (2018) explored parameterizing convolution kernels in the frequency domain to achieve a rotation equivariant convolution for spherical functions. However, this approach is not applicable beyond functions defined on the sphere, and furthermore does not encode translation equivariance, as this is not a symmetry of the sphere. While Cohen et al. (2018) are able to leverage an $SO(3)$ Fourier transform to perform $SO(3)$ equivariant convolutions for spherical functions, no such transform exists for performing convolutions that are equivariant to roto-reflections and translations, which presents a challenge for performing G-convolutions in the frequency domain.

3.2. $O(2)$-Equivariance of Fourier Transforms

To gain insight into how we may perform G-convolutions in the frequency domain, we look to characterize the behavior of the Fourier transform of a kernel function $\psi : \mathbb{Z}^2 \to \mathbb{R}$ under the action of G on ψ. In this work, we consider two groups in particular: the group $p4$ generated by 90° rotations and translations, and the group $p4m$ generated by $p4$ and horizontal reflections about the origin. We observe an intuitive symmetry, which we prove in Appendix C.2, that is foundational to performing group convolutions in the frequency domain.

Lemma 3.1. *Given the orthogonal group $O(d)$ acting on functions defined on $\mathbb{R}^d$ by the map $(g, f) \mapsto L_g f$ where $(L_g f)(x) := f(g^{-1}x)$, the group action commutes with the Fourier-transform, i.e. $\mathcal{F} \circ L_g = L_g \circ \mathcal{F}$.*

This result describes the equivariance of the Fourier transform. That is, *applying a transformation from $O(d)$ to*

a function in physical space applies the transformation equally to the Fourier transform of the function. We next use this result for the group of planar roto-reflections $O(2)$ to construct our G-Fourier layer.

3.3. Group Equivariant Fourier Layers

To derive our G-Fourier layers, we use the Convolution Theorem in addition to Lemma 3.1. For the functions $v, \rho : \mathbb{Z}^2 \to \mathbb{R}^{d_z}$ and the translation $x_g \in \mathbb{Z}^2$, this theorem gives that

$$\begin{aligned}(v \star \rho)[x_g] &:= \sum_{x \in \mathbb{Z}^2} \sum_{j=1}^{d_z} v^j(x) \left(L_{x_g} \rho^j\right)(x) \\ &= \sum_{j=1}^{d_z} \mathcal{F}^{-1}\left(\mathcal{F} v^j \cdot \mathcal{F} \rho^j\right)[x_g].\end{aligned} \tag{4}$$

The multiplication in the second line is element-wise and $\mathcal{F}$ is the Discrete Fourier Transform (DFT), which operates on functions defined on $\mathbb{Z}^d$. In contrast, G-convolutions operate on functions defined on a group as $f, \psi : G \to \mathbb{R}^{d_z}$, for which $\mathcal{F}$ is not defined in general. However, the groups G we consider here admit a decomposition, as they are the semidirect product $G = \mathbb{Z}^2 \rtimes S_G$ of the group of translations $\mathbb{Z}^2$ and the stabilizer of G, S_G (Weiler & Cesa, 2019). The stabilizer S_G is defined as transformations that leave the origin invariant, such as rotations for the group $p4$ and rotation-reflections for $p4m$. Therefore, for all group elements $g \in G$, g may be decomposed into a translation $x_g \in \mathbb{Z}^2$ and transformation $s_g \in S_G$ as $g = x_g s_g$, with the action of G similarly decomposed as $L_g = L_{x_g} L_{s_g}$. As shown in the following, this decomposition is key for parameterizing our G-convolution kernels in the $\mathcal{F}$-transformed group space.

For a fixed stabilizer element $s \in S_G$, define $f_s : \mathbb{Z}^2 \to \mathbb{R}^{d_z}$ for all translations $x \in \mathbb{Z}^2$ as $f_s(x) := f(xs)$, and define $\psi_s : \mathbb{Z}^2 \to \mathbb{R}^{d_z}$ analogously using the kernel ψ. Then, for the group element $g = x_g s_g \in G$, our G-Fourier layer is derived from G-convolutions as

$$\begin{aligned}(f \star \psi)[g] &:= \sum_{h \in G} \sum_{j=1}^{d_z} f^j(h) \left(L_g \psi^j\right)(h) \\ &= \sum_{s \in S_G} \sum_{j=1}^{d_z} \sum_{x \in \mathbb{Z}^2} f_s^j(x) \left(L_{x_g} L_{s_g} \psi_{\hat{s}}^j\right)(x) \\ &= \sum_{s \in S_G} \sum_{j=1}^{d_z} \mathcal{F}^{-1}\left(\mathcal{F} f_s^j \cdot \left(L_{s_g} R_{\hat{s}}^j\right)\right)[x_g] \\ &=: \mathcal{F}^{-1}\left(\mathcal{F} f \cdot \left(\hat{L}_{s_g} R\right)\right)[x_g],\end{aligned} \tag{5}$$

where $R_{\hat{s}}^j := \mathcal{F} \psi_{\hat{s}}^j$ is the complex-valued function we aim to learn and the equality in Equation (5) is a result of applying the Convolution Theorem in Equation (4) followed by Lemma 3.1. Additionally, $\hat{s} := s_g^{-1} s$ accounts for the cyclic shift along the stabilizer dimension (absorbed into $\hat{L}_{s_g}$) that we detail in Appendix C.1. We furthermore define the order of operations in the action of S_G on $\psi_{\hat{s}}$ as $(L_{s_g} \psi_{\hat{s}})(x) = \psi_{\hat{s}}(s_g^{-1} x)$, as opposed to $(L_{s_g} \psi_{\hat{s}})(x) = (L_{s_g} \psi)(x \hat{s})$, and similarly for $L_{s_g} R_{\hat{s}}$. This ensures that S_G, which is a subgroup of $O(2)$, is acting on a function defined on $\mathbb{Z}^2$ via L_{s_g}, enabling the use of Lemma 3.1. Notably, Equation (5) shows that we can efficiently perform group equivariant convolutions in the frequency domain by transforming R_s^j by elements s_g from the stabilizer S_G.

Concretely, the ℓ-th G-Fourier layer in the G-FNO architecture $\mathcal{L}^\ell$ maps the feature map $f^\ell \in \mathbb{R}^{d_z \times d_g \times d_x \times d_y}$ to $f^{\ell+1} \in \mathbb{R}^{d_z \times d_g \times d_x \times d_y}$. Here, d_z is the dimension of the latent space, d_g is the number of elements in S_G (i.e., $d_g = 4$ for $p4$, 8 for $p4m$), and $d_x \times d_y$ is the resolution of the input function. Our G-convolution kernel bank in the frequency domain is then $R^\ell \in \mathbb{C}^{d_z \times d_z \times d_g \times d_x \times d_y}$, which we visualize in Figure 2. To manage complexity, we assume *a priori* all modes above a cutoff frequency k are 0, and thus, only a subset of size $k \times k$ of the $d_x \times d_y$ Fourier modes are learnable, with the remaining modes fixed at 0.

Subsetting R^ℓ along the output channel dimension, $R^{\ell,l} \in \mathbb{C}^{d_z \times d_g \times d_x \times d_y}$ represents $\mathcal{F}\psi^{\ell,l}$ for our implicit kernel function $\psi^{\ell,l} : G \to \mathbb{R}^{d_z}$. From Equation (5), $\psi^{\ell,l}$ is then convolved in the frequency domain with f^ℓ to produce the l-th channel of $f^{\ell+1}$ as

$$f^{\ell+1,l}(g) = \mathcal{F}^{-1}\left(\mathcal{F} f^\ell \cdot \left(\hat{L}_{s_g} R^{\ell,l}\right)\right)[x_g]. \tag{6}$$

We denote the operator mapping f^ℓ to $f^{\ell+1}$ over all channels and $g \in G$ by $f^{\ell+1} = \mathcal{F}^{-1}(\mathcal{F} f^\ell \cdot \hat{L}_{S_G} R^\ell)$.

To increase the expressive capacity of our G-Fourier layer, we compose each frequency domain G-convolution with additional equivariant operations. Here, we note that it is key that these operations are applied only along the channel dimension, as aggregating information in physical space would introduce a dependence on the resolution of the training data, whereby reducing the ability of the G-FNO to perform super-resolution. The ℓ-th G-Fourier layer $\mathcal{L}^\ell$ mapping f^ℓ to $f^{\ell+1}$ can then be formally expressed as

$$\mathcal{L}^\ell f^\ell = W_G^\ell f^\ell + G\text{-MLP}^\ell\left(\mathcal{F}^{-1}\left(\mathcal{F} f^\ell \cdot \hat{L}_{S_G} R^\ell\right)\right). \tag{7}$$

Here, W_G^ℓ linearly projects the residual connection using a 1×1 G-Conv layer, and G-MLP$^\ell$ is a shallow 2-layer MLP with `GeLU` activation and with the linear layers replaced with 1×1 G-Conv layers. In Appendix B.6, we consider a steerable parameterization (Cohen & Welling, 2017) of R^ℓ.

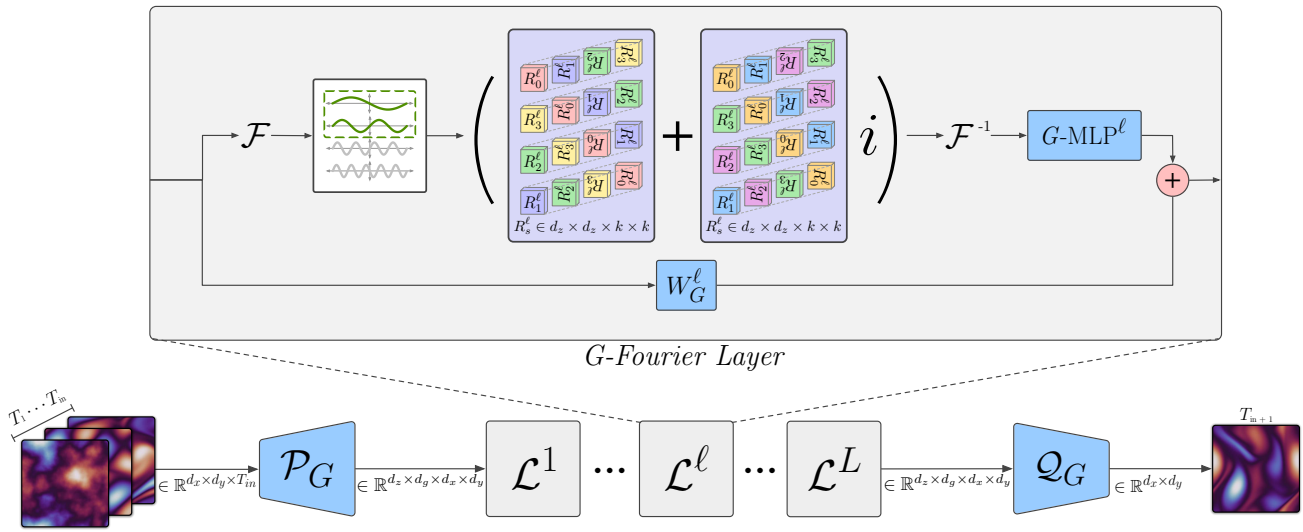


Figure 2: G-FNO-$p4$ architecture, illustrated in the 2D autoregressive form. Bottom: The model raises the input field to a high-dimensional embedding in group space ($\mathcal{P}_G$) and performs group convolutions in the frequency domain ($\mathcal{L}^\ell$) before lowering the G-feature map back to the base space ($\mathcal{Q}_G$). Top: A single G-Fourier layer. We perform G-convolutions in the frequency domain, selecting the k lowest Fourier modes of the input signal. The top row of the real and imaginary parts of the kernel bank illustrate the unrotated kernels, with R_s^ℓ rotated by $s \cdot 90°$ relative to its canonical orientation, $s \in \{0, 1, 2, 3\}$. G-MLP$^\ell$ is a shallow 2-layer MLP with `GeLU` activation and 1×1 G-Conv layers which we apply along the channel dimension of the output. We add a residual connection linearly projected along the channel dimension by W_G^ℓ.

3.4. Group Equivariant Fourier Neural Operator Architecture and Implementation

The G-FNO composes an encoder $\mathcal{P}_G$ with multiple G-Fourier layers followed by a decoder $\mathcal{Q}_G$. Both the encoder and decoder are 1×1 G-convs that lift the 2D input field $f^0 \in \mathbb{R}^{d_{in} \times d_x \times d_y}$ to a higher dimensional embedding in the group space and vice versa. Thus, the overall architecture, visualized in Figure 2, is expressed as

$$\mathcal{Q}_G \circ \mathcal{L}^L \circ \cdots \circ \mathcal{L}^2 \circ \mathcal{L}^1 \circ \mathcal{P}_G. \tag{8}$$

The encoder additionally concatenates a positional encoding describing the location in space and possibly time to each of the input grid points. As this grid does not transform with the input, the concatenation would not be equivariant if the grid were to contain the Cartesian coordinates of the input grid points. Therefore, we construct a positional encoding that is rotation and reflection invariant by letting each of the grid points map to the distance of the point from the center of the grid. This gives a positional encoding that is invariant to transformations of the input, hence preserving the overall equivariance of the architecture. We also note that although this positional encoding renders architectures only approximately translation equivariant, we show in Appendix B.5 that the performance of both the baseline FNO and G-FNO are improved with the inclusion of this grid.

Two additional challenges in efficiently implementing the G-Fourier layer concern the organization of the frequency modes in the DFT and enforcing the Hermitian constraint on R^ℓ. For simplicity in notation, we consider $\psi : G \to \mathbb{R}$ as our implicit kernel function and let our learnable parameter $R : G \to \mathbb{C}$ be the DFT of ψ, with $\psi_s : \mathbb{Z}^2 \to \mathbb{R}$ as ψ for a fixed value of $s \in S_G$ and $R_s : \mathbb{Z}^2 \to \mathbb{C}$ defined identically for R. First, for the transform of a 2D signal, the DFT algorithm places the origin (i.e., the zero frequency) in the upper left corner, followed by the positive modes and then the negative modes along each of the axes. However, transformations from S_G (rotations or roto-reflections) are most naturally applied to R_s with a centered origin. We therefore parameterize R_s as $\mathcal{F}_\pi \psi_s$, where $\mathcal{F}_\pi$ represents the DFT followed by a periodic shift to center the zero-frequency component at the origin. In Equation (5), we must similarly apply $\mathcal{F}_\pi$ to the input signal f_s^j so that the corresponding modes are correctly multiplied with R_s, and replace $\mathcal{F}^{-1}$ with $\mathcal{F}_{\pi^{-1}}^{-1}$, which is the inverse shift followed by the inverse DFT.

Second, because our implicit kernel function ψ_s is real-valued, the Fourier transform R_s will be Hermitian. That is, $R_s(\xi_1, \xi_2) = R_s^*(-\xi_1, -\xi_2)$, where $\cdot^*$ denotes complex conjugation and $(\xi_1, \xi_2) \in \mathbb{Z}^2$. It is important to enforce the Hermitian property in learning R, as it ensures that the result of the multiplication in the frequency domain will also be Hermitian, which in turn ensures that the inverse transform will be real-valued. To enforce this constraint, we only learn the positive modes along the last axis of R_s, since the negative modes, which are necessary in Equation (5) for applying

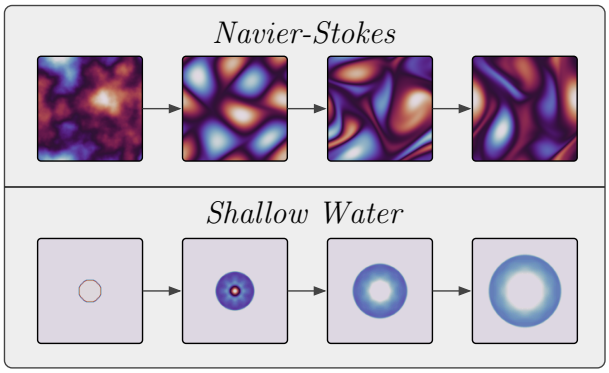


Figure 3: Illustration of the evolution of the Navier-Stokes equations with symmetric forcing (NS-SYM, top) and the shallow water equations (SWE-SYM, bottom).

transformations from S_G, can be directly inferred from the positive modes using the Hermitian property. Maintaining this property has the added benefit that we may use the real FFT algorithm for the forward and inverse transform, which reduces the cost by a factor of roughly 2 compared to the FFT. Further, ensuring the output of our G-Fourier layers is real-valued by enforcing the Hermitian constraint allows us to avoid overhead incurred by complex-valued parameters outside of the frequency domain, i.e., in W_G^ℓ and G-MLP$^\ell$.

We note that the use of Fourier transforms restricts us to studies on the entire space or periodic domains. Numerical discretizations, as we study here, always reduce to the case of a periodic domain, as the DFT implicitly extends the input periodically. Problems on the whole space can be studied, under suitable assumptions, if the length scale of the domain of interest is sufficiently small compared to the assumed domain of periodicity in a discretization.

4. Experiments

We introduce the datasets we consider in Section 4.1, describe our experimental settings in Section 4.2, and present results in Section 4.3.

4.1. Datasets

We evaluate our models on two commonly used PDEs in the field of computational fluid dynamics: the incompressible Navier-Stokes equations and the shallow water equations. The Navier-Stokes equations find broad applications in modeling of turbulent dynamics and hydromechanical systems. The shallow water equations are useful in general flooding events simulation (Takamoto et al., 2022) and weather modeling (Gupta & Brandstetter, 2022).

2D Incompressible Navier-Stokes equations. We consider two versions of the incompressible Navier-Stokes in vorticity form (Li et al., 2021a), given by:

$$\partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) = \nu\Delta w(x,t) + f(x), \quad (9)$$

$$\nabla\cdot u(x,t) = 0, \quad (10)$$

$$w(x,0) = w_0(x). \quad (11)$$

Here, $w(x,t)\in\mathbb{R}$ is the vorticity field we aim to predict, $w_0(x)\in\mathbb{R}$ is the initial vorticity, $u(x,t)\in\mathbb{R}^2$ is the velocity, and $\nu = 10^{-4}$ is the viscosity coefficient. The solution domain we consider is $x\in[0,1]^2, t\in\{1,2,\dots,T\}$, where t enumerates a discretization of the continuous temporal domain and T is defined in **Tasks and Evaluation**.

In Equation (9), f is a forcing term that describes external forces acting on the flow. As we show in Appendix C.3, for the rotated version of a solution to these equations to still be a solution, the forcing term must be invariant to $90°$ rotations. We therefore consider two realizations of these equations with different forcing terms to evaluate the performance of G-FNO in settings with and without global symmetry. The first, studied by Li et al. (2021a) and which we refer to as NS, has the forcing term $f(x_1,x_2) = 0.1(\sin(2\pi(x_1+x_2)) + \cos(2\pi(x_1+x_2)))$, which is not rotation invariant. The second, NS-SYM, has the forcing term $f(x_1,x_2) = 0.1(\cos(4\pi x_1) + \cos(4\pi x_2))$, which is rotation invariant, and hence yields a solution set closed to rotations. We visualize the evolution of the vorticity field w for NS-SYM in Figures 3 and 7 and for NS in Figure 6.

2D Shallow Water equations. As with the Navier-Stokes equations, we consider two versions of the shallow water equations that we refer to as SWE-SYM and SWE, each with different degrees of global symmetry. SWE-SYM is from Takamoto et al. (2022) and models the dynamics of a fluid that is initially confined within a circular dam and subsequently released due to the sudden removal of the dam. The equations are given by:

$$\partial_t h + \nabla\cdot(hu) = 0, \quad (12)$$

$$\partial_t(hu) + \nabla\cdot(huu^\top) + \frac{1}{2}g\nabla h^2 = 0, \quad (13)$$

where $h(x,t)\in\mathbb{R}$ is the depth that we will predict, $u(x,t)\in\mathbb{R}^2$ is the velocity, and $g\in\mathbb{R}$ is acceleration due to gravity. The solution domain is $x\in[-2.5,2.5]^2, t\in\{1,2,\dots,25\}$. The initial depth $h(x,0)$ is given by:

$$h(x,0) = \begin{cases} 2.0 & r < \sqrt{x_1^2+x_2^2} \\ 1.0 & r \geq \sqrt{x_1^2+x_2^2} \end{cases}, \quad (14)$$

where r denotes the radial distance to the center of the circular dam. We visualize the evolution of h in Figures 3 and 5, which demonstrates the high degree of global symmetry present in this data.

SWE is from Gupta & Brandstetter (2022) and models both the vorticity field and pressure field of global winds with a large time step size of 48 hours. Because SWE is defined on a rectangular domain, it lacks the global symmetry present in SWE-SYM, which is evident from Figure 4.

4.2. Experimental Settings

Tasks and Evaluation. We train all models on numerical data that is downsampled from the resolution at which it was generated. In Appendix A.1, we discuss data generation and downsampling in more depth and include training details in Appendix A.2. We evaluate models on a hold-out test set with the same resolution as the training data. Specifically, for NS, we map the ground truth vorticity field up to $t = 10$ to the field at each time step up to $T = 30$. Procedures for NS-SYM are identical, but with $T = 20$ instead. For SWE-SYM, we map the depth of the water at $t = 1$ up to the depth at $T = 25$, while for SWE, we map the pressure and vorticity fields up to $t = 2$ to the fields up to $T = 11$. In Appendix D, we visualize the predicted rollouts for each dataset.

We consider two approaches for advancing time with our neural solvers. 2D models utilize spatial convolutions with autoregressive predictions, while 3D models predict all time steps with a single forward propagation by performing convolutions in space-time. Both 2D and 3D versions of the equivariant models we consider in our experiments encode symmetries in the spatial domain.

After evaluating rollout errors, we study the equivariance by rotating the test set and evaluating the models again. Additionally, we evaluate models on the super-resolution task in which we apply the models trained on the downsampled training data to the test data at the fine resolution at which it was generated. Note that while the 3D models are able to perform super-resolution in space and time, the 2D models can only perform super-resolution in space, as training the models to advance time autoregressively locks the model to this time step. Lastly, to evaluate the efficacy of super-resolution, we used the predictions for each model made on the coarse grid to interpolate the solutions onto the finer grid.

For each of these tasks, we use relative mean square error to evaluate our models (Li et al., 2021a; Tran et al., 2023). Specifically,

$$\text{R-MSE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \frac{\|\hat{y}_i - y_i\|_2}{\|y_i\|_2}, \tag{15}$$

where n_{test} is the number of test PDEs and $\|\cdot\|_2$ is the L_2 norm. $\hat{y}_i$ and y_i denote the predicted solution and ground truth of the i-th test PDE. We also train our models using the one-step version of this loss, where $\hat{y}_i$ and y_i are both only one time step. This training strategy was shown by Tran et al. (2023) to be superior to recurrent training, which we further examine in Appendix B.4.

Baselines. We consider several variants of the FNO, as well as an equivariant U-Net. Coupled with equivariance, this is a particularly interesting comparison, as U-Nets take a sequential approach to processing information on multiple scales, while FNO architectures process multi-scale information in parallel with Fourier convolutions (Gupta & Brandstetter, 2022). Our $p4$ equivariant U-Net, U-NET-$p4$, is a modified version of the architecture studied by Wang et al. (2021a) for dynamics modeling. The version of the FNO we consider is constructed with a non-equivariant version of our G-Fourier layer given in Equation (7) and includes versions of the linear projection W_G^ℓ and G-MLP defined for functions on $\mathbb{R}$ instead of the group space G.

We also include versions of the FNO trained using data augmentation to aid the model in learning symmetries in the data, an idea explored for neural solvers by Brandstetter et al. (2022a). The data augmentation is performed by sampling group transformations from $p4$ (translations and rotations by $90°$) and $p4m$ ($p4$ plus reflections) and applying these transformations to the training data, yielding FNO+$p4$ and FNO+$p4m$, respectively. Note that since FNO performs convolution, it is translation equivariant by design, and therefore, we only need to sample rotations and roto-reflections.

Lastly, we consider RADIALFNO, a frequency domain parameterization of the architecture explored by Shen et al. (2021). Shen et al. (2021) construct equivariant CNNs using a radial kernel function. By Lemma 3.1, the Fourier transform of this kernel will also be radial, and thus, this architecture admits a straightforward extension to a parameterization in the frequency domain. For equivariance to rotations by $90°$, the kernel need not be fully radial and rather just invariant to $90°$ rotations. This increases the capacity of the considered baseline by reducing the degree of weight-sharing required to achieve the desired invariance.

4.3. Results and Analysis

We consider two variants of the G-FNO: G-FNO-$p4$, which employs G- Fourier layers that are equivariant to $90°$ rotations and translations, and G-FNO-$p4m$, which is additionally equivariant to horizontal and vertical reflections about the origin. In Appendix B.1, we analyze inference times and forward memory requirements for all models considered.

4.3.1. INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

Test Rollouts. In Table 1, we present results on both Navier-Stokes datasets for 2D and 3D models. G-FNO gives the lowest test rollout error in all 4 settings, including improving

Table 1: Results on Navier-Stokes with (NS-SYM) and without (NS) global symmetry. 2D models make rollout predictions autoregressively, while 3D models perform convolutions in space-time. We present the relative mean squared error as a percentage averaged over three random seeds for predicted rollouts. Rollouts are of length 10 in the case of NS-SYM and 20 for NS, and conditioned on the first 10 time steps in the trajectory for both datasets.

	2D MODELS				3D MODELS			
	NS		NS-SYM		NS		NS-SYM	
	# PAR. (M)	TEST (%)	# PAR. (M)	TEST (%)	# PAR. (M)	TEST (%)	# PAR. (M)	TEST (%)
FNO	0.93	8.41(0.41)	0.93	4.21(0.12)	4.92	15.84(0.37)	4.92	26.02(0.41)
FNO+$p4$	0.93	10.44(0.47)	0.93	4.80(0.12)	4.92	14.14(0.14)	4.92	<u>15.75</u>(0.33)
FNO+$p4m$	0.93	22.09(1.46)	0.93	13.06(3.29)	4.92	15.32(0.05)	4.92	22.25(0.21)
G-FNO-$p4$	0.85	**4.78**(0.39)	0.85	**2.24**(0.09)	4.80	**11.77**(0.13)	4.80	**10.72**(0.27)
G-FNO-$p4m$	0.84	<u>6.19</u>(0.61)	0.84	<u>2.37</u>(0.19)	3.89	12.71(0.31)	3.89	17.21(1.35)
U-NET-$p4$	3.65	18.40(0.44)	3.65	15.39(0.16)	6.08	24.62(0.29)	6.08	21.82(0.10)
RADIALFNO-$p4$	1.03	9.21(0.26)	1.03	12.81(0.42)	4.98	12.09(0.08)	4.98	17.54(0.60)
RADIALFNO-$p4m$	0.95	10.86(0.18)	0.95	17.39(0.22)	5.63	<u>11.83</u>(0.23)	5.63	17.27(0.17)

Table 2: Super-resolution results on Navier-Stokes with (NS-SYM) and without (NS) global symmetry. We increase the resolution of the test set by a factor of 4 and evaluate models trained on the lower resolution data. For 2D models, the super-resolution is in space, while for 3D models, the time resolution is also increased. In the column SR, we show the super-resolution rollout errors for models predicting directly to the higher resolution, while INT. shows the error for the predictions made at a lower resolution and fine-grained using interpolation.

	2D MODELS				3D MODELS			
	NS		NS-SYM		NS		NS-SYM	
	SR TEST (%)	INT. TEST (%)	SR TEST (%)	INT. TEST (%)	SR TEST (%)	INT. TEST (%)	SR TEST (%)	INT. TEST (%)
FNO	**43.02**(0.18)	**43.14**(0.19)	32.45(1.47)	<u>23.33</u>(0.07)	29.99(0.26)	27.97(0.10)	31.24(0.66)	29.38(0.47)
FNO+$p4$	49.78(8.40)	43.72(0.45)	31.72(1.55)	**23.32**(0.09)	30.36(0.18)	<u>27.27</u>(0.11)	25.25(0.80)	<u>21.20</u>(0.31)
FNO+$p4m$	54.04(4.52)	46.30(1.33)	32.68(0.84)	25.02(1.24)	30.45(0.37)	27.82(0.12)	31.44(1.20)	26.64(0.32)
G-FNO-$p4$	<u>43.41</u>(0.12)	<u>43.51</u>(0.11)	**21.89**(0.05)	23.36(0.04)	**29.62**(0.15)	**27.09**(0.06)	**20.44**(0.73)	**17.71**(0.09)
G-FNO-$p4m$	43.78(0.33)	43.88(0.30)	<u>22.09</u>(0.03)	23.56(0.03)	30.02(0.31)	27.38(0.21)	<u>23.98</u>(1.30)	22.14(0.96)
U-NET-$p4$	92.00(7.22)	43.68(0.82)	70.42(1.66)	24.24(0.12)	114.99(33.94)	30.11(0.41)	79.08(3.60)	25.85(0.03)
RADIALFNO-$p4$	43.73(0.07)	43.86(0.07)	25.47(0.31)	26.52(0.35)	<u>29.92</u>(0.65)	27.52(0.48)	24.87(0.51)	22.83(0.42)
RADIALFNO-$p4m$	43.91(0.69)	44.02(0.71)	27.85(0.22)	28.70(0.05)	29.94(0.60)	27.40(0.45)	24.49(0.14)	22.60(0.14)

the baseline error by over 3.5% on the NS data, which lacks global symmetry. In Appendix B.2, we examine the effect of only maintaining an equivariant representation in the initial layers of the network on this data, and find that performance increases with the number of equivariant layers. The benefit of equivariant architectures, even on data without global symmetries, has been previously noted (Cohen & Welling, 2016), and could be due to the ability of equivariant models to extract local symmetries (Worrall et al., 2017). By contrast, models trained with data augmentation may achieve an approximate equivariance, but are not constrained to maintain equivariant internal representations and struggle to capture local symmetries (Worrall et al., 2017). This could explain why the FNO with data augmentation underperforms the G-FNO in all settings. This observation may also account for data augmentation reducing the performance of the FNO in all settings except 1, including the 2D model on NS-SYM, where the global symmetry may deceptively suggest data augmentation as a reasonable choice.

We also observe that U-NET-$p4$ does not perform well in comparison to architectures performing convolutions in the frequency domain. This could be due to the parallel multi-scale processing mechanisms inherent to Fourier convolutions, which contrasts the sequential multi-scale processing mechanism employed by U-Nets (Gupta & Brandstetter, 2022).

Super-Resolution. In Table 2, we examine the super-resolution capabilities of G-FNO. In 3 out of the 4 settings, G-FNO produces the lowest super-resolution error, giving the second-best error to the baseline FNO only for 2D models on NS. We also perform super-resolution by fine-graining low-resolution predictions using interpolation. For 2D models, we observe for both NS and NS-SYM that the direct G-FNO super-resolution predictions have a lower error than those made with interpolation.

Rotation Test. In Table 3, we rotate the test data by $90°$ counter-clockwise and evaluate the models trained on the unrotated data. Unsurprisingly, we observe that the errors of all equivariant models are invariant to this transformation. We additionally note that the difference in the rotated test error and unrotated test error on NS-SYM is much lower for the FNO than on NS. This empirically demonstrates the

Table 3: Rotation test results on Navier-Stokes with (NS-SYM) and without (NS) global symmetry. We use models trained on the unrotated data to predict rollouts for the rotated PDE. In the case of NS, rotating the PDE does not produce a solution to the original equations as it does for NS-SYM, which we prove in Appendix C.3.

	2D MODELS				3D MODELS			
	NS		NS-SYM		NS		NS-SYM	
	TEST (%)	TEST_{90° (%)	TEST (%)	TEST_{90° (%)	TEST (%)	TEST_{90° (%)	TEST (%)	TEST_{90° (%)
FNO	8.41(0.41)	129.21(3.90)	4.21(0.12)	9.91(0.90)	15.84(0.37)	100.75(2.20)	26.02(0.41)	26.75(0.64)
FNO+$p4$	10.44(0.47)	10.38(0.38)	4.80(0.12)	4.74(0.20)	14.14(0.14)	14.21(0.15)	<u>15.75</u>(0.33)	<u>15.85</u>(0.31)
FNO+$p4m$	22.09(1.46)	22.61(1.54)	13.06(3.29)	12.81(2.80)	15.32(0.05)	15.37(0.03)	22.25(0.21)	22.24(0.20)
G-FNO-$p4$	**4.78**(0.39)	**4.78**(0.39)	**2.24**(0.09)	**2.24**(0.09)	**11.77**(0.13)	**11.77**(0.13)	**10.72**(0.27)	**10.72**(0.27)
G-FNO-$p4m$	<u>6.19</u>(0.61)	<u>6.19</u>(0.61)	<u>2.37</u>(0.19)	<u>2.37</u>(0.19)	12.71(0.31)	12.71(0.31)	17.21(1.35)	17.21(1.35)
U-NET-$p4$	18.40(0.44)	18.40(0.44)	15.39(0.16)	15.39(0.16)	24.62(0.29)	24.62(0.29)	21.82(0.10)	21.82(0.10)
RADIALFNO-$p4$	9.21(0.26)	9.21(0.26)	12.81(0.42)	12.81(0.42)	12.09(0.08)	12.09(0.08)	17.54(0.60)	17.54(0.60)
RADIALFNO-$p4m$	10.86(0.18)	10.86(0.18)	17.39(0.22)	17.39(0.22)	<u>11.83</u>(0.23)	<u>11.83</u>(0.23)	17.27(0.17)	17.27(0.17)

Table 4: Results on Shallow Water Equations with (SWE-SYM) and without (SWE) global symmetry. Rollouts are of length 24 and conditioned on the first time step in the trajectory in the case of SWE-SYM. For SWE, rollouts are of length 9 and conditioned on the first 2 time steps in the trajectory.

	2D MODELS				3D MODELS			
	SWE		SWE-SYM		SWE		SWE-SYM	
	# PAR. (M)	TEST (%)	# PAR. (M)	TEST ($\times 10^{-3}$)	# PAR. (M)	TEST (%)	# PAR. (M)	TEST ($\times 10^{-3}$)
FNO	6.56	18.45(0.89)	0.93	1.22(0.07)	49.57	**41.49**(0.12)	4.92	1.59(0.02)
FNO+$p4$	6.56	33.08(0.20)	0.93	1.33(0.08)	49.57	44.16(0.14)	4.92	1.64(0.02)
FNO+$p4m$	6.56	44.24(1.63)	0.93	1.32(0.07)	49.57	45.06(0.20)	4.92	1.65(0.03)
G-FNO-$p4$	6.36	**14.96**(0.06)	0.85	1.21(0.02)	53.70	43.68(0.55)	4.80	<u>1.44</u>(0.02)
G-FNO-$p4m$	6.23	<u>16.20</u>(0.17)	0.84	1.11(0.17)	56.81	<u>43.46</u>(0.79)	3.89	**1.44**(0.02)
U-NET-$p4$	6.90	28.79(0.08)	3.65	30.86(11.31)	6.08	55.28(1.86)	6.08	8.03(0.35)
RADIALFNO-$p4$	6.79	23.37(0.14)	1.03	<u>0.71</u>(0.03)	53.40	44.12(0.20)	4.98	1.54(0.02)
RADIALFNO-$p4m$	6.84	26.20(0.55)	0.95	**0.70**(0.07)	51.92	44.76(0.15)	5.63	1.49(0.01)

low degree of global symmetry present in NS and further emphasizes the ability of G-FNO to perform well even in settings lacking such symmetry.

4.3.2. SHALLOW WATER EQUATIONS

Test Rollouts. In Table 4, we present results on both shallow water datasets for 2D and 3D models. For the 2D models on SWE-SYM, RADIALFNO gives the best performance, likely due to the compatibility between the radial kernel and radial solution function shown in Figure 3. We note that as evidenced by the Navier-Stokes experiments and the remaining shallow water settings, the radial inductive bias appears to be overly restrictive and does not generalize well beyond this setting. Beyond RADIALFNO, G-FNO gives the lowest error out of the considered baselines. Additionally, in the 3D SWE setting, G-FNO is second to the baseline FNO. We note that all models have an error greater than 40% in this setting, suggesting that the 3D modeling scheme does not work well for SWE. This could potentially be due to the coarse step size representing 48 hours reducing the smoothness of the function being convolved in space-time.

For 2D models on SWE and 3D models on SWE-SYM, G-FNO has the lowest test error. Although SWE lacks global symmetry, which is immediately evident from the non-square domain, G-FNO still improves upon the baseline rollout error by over 2%. We present super-resolution and rotation test results for SWE-SYM in Appendix B.3.

5. Conclusion

In this work, we propose to design FNO architectures that encode symmetries. Specifically, by leveraging symmetries of the Fourier transform, we extend group convolutions to the frequency domain and design G-Fourier layers that are equivariant to rotations, translations, and reflections. We conduct extensive experiments to evaluate our proposed G-FNO. Results show that explicit encoding of symmetries in FNO architectures leads to consistent performance improvements.

Acknowledgements

This work was supported in part by National Science Foundation grant IIS-2006861, and by state allocated funds for the Water Exceptional Item through Texas A&M AgriLife Research facilitated by the Texas Water Resources Institute.

References

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Bonnet, F., Mazari, J. A., Cinella, P., and Gallinari, P. AirfRANS: High fidelity computational fluid dynamics dataset for approximating Reynolds-averaged Navier-Stokes solutions. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022.

Boussif, O., Bengio, Y., Benabbou, L., and Assouline, D. MAgNet: Mesh agnostic neural PDE solver. In *Advances in Neural Information Processing Systems*, 2022.

Brandstetter, J., Welling, M., and Worrall, D. E. Lie point symmetry data augmentation for neural PDE solvers. In *International Conference on Machine Learning*, pp. 2241–2256. PMLR, 2022a.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022b.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Cohen, T. S. and Welling, M. Steerable CNNs. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJQKYt5ll.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. *International Conference on Learning Representations*, 2018.

Eckert, M.-L., Um, K., and Thuerey, N. ScalarFlow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., and Catanzaro, B. Adaptive Fourier neural operators: Efficient token mixers for transformers. In *International Conference on Learning Representations*, 2022.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized PDE modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Holl, P., Koltun, V., Um, K., and Thuerey, N. phiflow: A differentiable PDE solving framework for deep learning via physical simulations. In *NeurIPS workshop*, volume 2, 2020.

Hwang, R., Lee, J. Y., Shin, J. Y., and Hwang, H. J. Solving PDE-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4504–4512, 2022.

Jiang, P., Meinert, N., Jordão, H., Weisser, C., Holgate, S., Lavin, A., Lutjens, B., Newman, D., Wainright, H., Walker, C., et al. Digital twin earth-coasts: Developing a fast and physics-informed surrogate model for coastal floods via neural operators. In *Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021)*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021b.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *arXiv preprint arXiv:2207.05209*, 2022a.

Li, Z., Liu-Schiaffini, M., Kovachki, N. B., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Learning dissipative dynamics in chaotic systems. In *Advances in Neural Information Processing Systems*, 2022b.

Lim, J. and Psaltis, D. MaxwellNet: Physics-driven deep neural network training based on Maxwell's equations. *Apl Photonics*, 7(1):011301, 2022.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Poli, M., Massaroli, S., Berto, F., Park, J., Dao, T., Ré, C., and Ermon, S. Transform once: Efficient operator learning in frequency domain. In *Advances in Neural Information Processing Systems*, 2022.

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

Salvi, C., Lemercier, M., and Gerasimovics, A. Neural stochastic PDEs: Resolution-invariant learning of continuous spatiotemporal dynamics. In *Advances in Neural Information Processing Systems*, 2022.

Shen, P., Herbst, M., and Viswanathan, V. Rotation equivariant operators for machine learning on scalar and vector fields. *arXiv preprint arXiv:2108.09541*, 2021.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. PDEBench: An extensive benchmark for scientific machine learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022. URL https://arxiv.org/abs/2210.07182.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized Fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020.

Wang, R., Walters, R., and Yu, R. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations*, 2021a.

Wang, R., Walters, R., and Yu, R. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning*, 2022.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40): eabi8605, 2021b.

Weiler, M. and Cesa, G. General $E(2)$-equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31, 2018.

Worrall, D. and Welling, M. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 5028–5037, 2017.

Zobeiry, N. and Humfeld, K. D. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101: 104232, 2021.

Table 5: Inference time and forward memory requirements. We analyze the time and space complexity of all models on the NS data over 10,000 forward propagations. As in our experiments, we use a batch size of 20 for the 2D models trained to make predictions autoregressively, and a batch size of 10 for the 3D models that perform convolutions in space-time.

	2D Models			3D Models		
	# Par. (M)	Inference Time (ms)	Forward Memory (GiB)	# Par. (M)	Inference Time (ms)	Forward Memory (GiB)
FNO	0.93	4.27(1.66)	1.83	4.92	8.71(7.54)	3.09
G-FNO-$p4$	0.85	4.50(2.12)	1.97	4.80	10.21(15.62)	4.48
G-FNO-$p4m$	0.84	4.81(3.79)	2.17	3.89	10.73(16.46)	5.41
radialFNO-$p4$	1.03	4.34(2.63)	1.97	4.98	10.81(16.58)	6.24
radialFNO-$p4m$	0.95	4.16(3.29)	2.03	5.63	11.53(17.98)	7.50
U-Net-$p4$	3.65	7.92(4.34)	2.31	6.08	20.29(15.77)	7.45

Table 6: G-Hybrid results on NS. We let the first N layers of the network be equivariant $p4$ and $p4m$ G-Fourier layers, and the remaining $4 - N$ layers be non-equivariant Fourier layers. We report both the test and rotated test errors.

	$p4$			$p4m$		
# G-Fourier Layers	# Par. (M)	Test (%)	Test_{90° (%)	# Par. (M)	Test (%)	Test_{90° (%)
0	0.93	8.41(0.41)	129.21(3.90)	0.93	8.41(0.41)	129.21(3.90)
1	1.14	7.17(0.51)	<u>126.53</u>(3.66)	1.32	<u>7.04</u>(0.41)	128.32(1.83)
2	1.12	6.30(0.82)	127.89(4.99)	1.30	7.32(0.37)	129.15(3.02)
3	1.10	<u>6.12</u>(0.44)	128.73(7.78)	1.27	7.29(0.14)	<u>125.74</u>(9.08)
4	0.85	**4.78**(0.39)	**4.78**(0.39)	0.84	**6.19**(0.61)	**6.19**(0.61)

A. Training and Data Generation Details

A.1. Data Generation

For NS-SYM, we generate 1,000 training trajectories, 100 validation trajectories, and 100 test trajectories using the psuedo-spectral Crank-Nicolson solver from Li et al. (2021a). For NS, we use the data directly from Li et al. (2021a), again with a 1,000/100/100 split. For each trajectory, the boundary conditions are periodic and the initial conditions $w_0(x)$ are sampled from a Gaussian random field. Trajectories were solved numerically on a $256 \times 256 \times 120$ grid and downsampled to $64 \times 64 \times 30$, where the first two dimensions are in space and the third is time.

For SWE-SYM, we used numerical data from Takamoto et al. (2022) generated using the finite volume method. We split 1,000 trajectories into 800 training trajectories, 100 validation trajectories and 100 test trajectories. For each trajectory, the radius of the dam, r in Equation (14), is sampled uniformly from $(0.3, 0.7)$. We downsample the numerical solution from $128 \times 128 \times 100$ to $32 \times 32 \times 25$. Unlike NS and NS-SYM, we performed spatial downsampling using 2×2 mean-pooling, as strided downsampling introduced a significant asymmetry that was not present in the original data.

For SWE, we follow the methods and splits used by Gupta & Brandstetter (2022) to generate 5,600 training trajectories, 1,120 validation trajectories, and 1,120 test trajectories, all with resolution $96 \times 192 \times 88$. We follow Gupta & Brandstetter (2022) in temporally downsampling by a factor of 8 to 11 total time steps spaced 48 hours apart.

A.2. Training

We perform 3 replicates of all experiments and closely follow the training strategy and hyperparameter specification scheme of Li et al. (2021a). We use 4 Fourier layers for all frequency domain models, truncating the transform to the 12 lowest Fourier modes for all 2D models and 8 spatial/6 temporal Fourier modes for all 3D models. In the case of SWE, we increase the number of modes for all 2D models to 32 following Gupta & Brandstetter (2022) and 22 spatial/8 temporal modes for all 3D models. We also replace the Symmetric positional encoding discussed in Appendix B.5 for G-FNO on SWE with the Cartesian encoding, as the non-square 96×192 spatial domain prevents the Symmetric encoding from being invariant to rotations, breaking the equivariance of the model.

For the baseline FNO, the dimension of the latent space is 20. For G-FNO, we offset the extra dimensions added to kernels discussed in Section 3.3 by reducing the number of channels to give a roughly equal number of trainable parameters

Table 7: Super-resolution results on SWE-SYM. We increase the resolution of the test set by a factor of 4 and evaluate models trained on the lower resolution data. For 2D models, the super-resolution is in space, while for 3D models, the time resolution is also increased. In the column SR, we show the super-resolution rollout errors for models predicting directly to the higher resolution, while INT. shows the error for the predictions made at a lower resolution and fine-grained using interpolation.

	2D MODELS		3D MODELS	
	SR TEST ($\times 10^{-3}$)	INT. TEST ($\times 10^{-3}$)	SR TEST ($\times 10^{-3}$)	INT. TEST ($\times 10^{-3}$)
FNO	<u>15.56</u>(2.92)	16.15(0.01)	16.90(0.58)	17.76(0.01)
FNO+$p4$	**14.80**(3.30)	16.16(0.01)	17.19(0.86)	17.76(0.01)
FNO+$p4m$	16.38(3.94)	16.16(0.01)	17.11(0.83)	17.75(0.01)
G-FNO-$p4$	19.39(4.60)	16.15(0.00)	15.87(0.60)	**17.75**(0.00)
G-FNO-$p4m$	31.00(11.27)	16.15(0.01)	16.88(1.19)	<u>17.75</u>(0.00)
U-NET-$p4$	3009.65(2348.93)	35.19(9.48)	167.76(36.77)	19.19(0.13)
RADIALFNO-$p4$	30.40(4.20)	<u>16.12</u>(0.00)	<u>14.48</u>(0.76)	17.76(0.00)
RADIALFNO-$p4m$	22.42(0.95)	**16.12**(0.00)	**13.36**(0.36)	17.76(0.00)

Table 8: Rotation test results for SWE-SYM. We use models trained on the unrotated data to predict rollouts for the rotated PDE.

	2D MODELS		3D MODELS	
	TEST ($\times 10^{-3}$)	TEST$_{90^\circ}$ ($\times 10^{-3}$)	TEST ($\times 10^{-3}$)	TEST$_{90^\circ}$ ($\times 10^{-3}$)
FNO	1.22(0.07)	1.50(0.05)	1.59(0.02)	1.80(0.02)
FNO+$p4$	1.33(0.08)	1.34(0.10)	1.64(0.02)	1.69(0.03)
FNO+$p4m$	1.32(0.07)	1.33(0.07)	1.65(0.03)	1.71(0.03)
G-FNO-$p4$	1.21(0.02)	1.21(0.02)	<u>1.44</u>(0.02)	<u>1.44</u>(0.02)
G-FNO-$p4m$	1.11(0.17)	1.11(0.17)	**1.44**(0.02)	**1.44**(0.02)
U-NET-$p4$	30.86(11.31)	30.86(11.31)	8.03(0.35)	8.03(0.35)
RADIALFNO-$p4$	<u>0.71</u>(0.03)	<u>0.71</u>(0.03)	1.54(0.02)	1.54(0.02)
RADIALFNO-$p4m$	**0.70**(0.07)	**0.70**(0.07)	1.49(0.01)	1.49(0.01)

compared to the baseline FNO. Specifically, in the case of 2D models, the dimension of the latent space for G-FNO-$p4$ and G-FNO-$p4m$ is 10 and 7, respectively. For 3D models, the dimensions are 11 and 7, except in the case of SWE, where we increase the dimension for G-FNO-$p4m$ to 8. We also increased the first-layer latent dimension to 100 for a 48.09M parameter 3D U-NET-$p4$ on SWE. However, the 6.08M parameter version with latent dimension 32 presented in Table 4 improves the test error over the larger model by roughly 10%.

As opposed to the RECURRENT training strategy employed by Li et al. (2021a) for the 2D models, we instead use the TEACHER FORCING strategy, which was shown by Tran et al. (2023) to improve performance. In Appendix B.4, we compare training strategies on NS. We use the Adam optimizer (Kingma & Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.999$, and weight decay 10^{-4}. We use batch size of 20 for 2D models and 10 for 3D models with a cosine learning rate scheduler that starts at 10^{-3} and is decayed to 0. 3D models are trained for 500 epochs or until the validation loss does not improve for 100 successive epochs, while 2D models are trained for 100 epochs. 2D models are trained for less epochs because each PDE in the training set corresponds to 1 training example for the 3D models and $T - T_{in}$ training examples for the 2D models, where T_{in} is the number of time steps being conditioned on. All models are implemented using PyTorch (Paszke et al., 2019) and trained on a single NVIDIA A100 80GB GPU.

B. Additional Results

B.1. Inference Time and Forward Memory Requirements

In Table 5, we present the average inference time across 10,000 forward evaluations and the GPU memory utilized for each model on the NS data. Here, we note that the times reported for 2D models are for predictions only one step into the future, while 3D models perform convolutions in space and time and thus predict all $T = 20$ steps with one forward pass. We use batch size of 20 for 2D models and 10 for 3D models.

Table 9: Training strategy analysis for NS. We present the best validation result and super-resolution test error for three choices of training strategy: MARKOV, where we condition on one time step and predict one time step into the future, RECURRENT, where we condition on several time steps and predict the remaining steps in the rollout autoregressively, and TEACHER FORCING, where we condition on several time steps and predict 1 time step into the future.

	STRATEGY	VALID (%)	SR TEST (%)
FNO	MARKOV	**7.19**(0.28)	61.10(2.58)
	RECURRENT	16.23(0.49)	**42.14**(0.25)
	TEACHER FORCING	8.64(0.19)	43.02(0.18)
FNO+$p4$	MARKOV	10.90(0.24)	60.46(2.92)
	RECURRENT	17.60(0.14)	**41.22**(0.41)
	TEACHER FORCING	**10.70**(0.49)	49.78(8.40)
FNO+$p4m$	MARKOV	**18.26**(0.75)	60.41(2.28)
	RECURRENT	20.81(0.28)	**40.63**(0.20)
	TEACHER FORCING	22.43(1.59)	54.04(4.52)
G-FNO-$p4$	MARKOV	5.06(0.04)	43.35(0.24)
	RECURRENT	13.20(0.06)	**42.36**(0.23)
	TEACHER FORCING	**4.86**(0.32)	43.41(0.12)
G-FNO-$p4m$	MARKOV	**6.59**(0.96)	43.18(0.82)
	RECURRENT	13.93(0.05)	**42.16**(0.34)
	TEACHER FORCING	6.73(0.82)	43.78(0.33)
U-NET-$p4$	MARKOV	33.70(4.00)	110.60(0.92)
	RECURRENT	51.61(7.68)	112.43(3.84)
	TEACHER FORCING	**18.86**(0.53)	**92.00**(7.22)
RADIALFNO-$p4$	MARKOV	**8.43**(0.25)	43.14(0.29)
	RECURRENT	13.51(0.07)	**42.55**(0.05)
	TEACHER FORCING	9.59(0.05)	43.73(0.07)
RADIALFNO-$p4m$	MARKOV	**10.38**(0.19)	43.74(0.09)
	RECURRENT	13.81(0.21)	**42.30**(0.27)
	TEACHER FORCING	11.51(0.68)	43.91(0.69)

B.2. G-HYBRID Experiments on NS

We explore the effect of only maintaining an equivariant representation in the initial layers by composing $N \in \{0, 1, 2, 3, 4\}$ G-Fourier layers and replacing the final $4 - N$ layers with non-equivariant FNO layers. We train 2D models to make predictions autoregressively on the NS task described in Section 4.2 and present rollout errors on the test set and rotated test set in Table 6.

We observe that increasing the number of equivariant layers improves performance, despite the NS data not being globally symmetric, as we prove in Appendix C.3. Equivariant models have been noted to offer benefits on non-symmetric datasets in the past (Cohen & Welling, 2016). The observed benefits could stem from the ability of equivariant layers to learn local symmetries (Worrall et al., 2017).

B.3. SWE-SYM Super-Resolution and Rotation Test Results

In Tables 7 and 8, we present super-resolution and rotation test results for SWE-SYM. For the super-resolution task with 2D models that make rollout predictions autoregressively, the ground truth rollouts during training are $32 \times 32 \times 24$ and $128 \times 128 \times 24$ during testing, where the first two dimensions are spatial and the third is the number of time steps. For 3D models that perform convolutions in space-time, the temporal resolution also increases during testing to $128 \times 128 \times 96$.

B.4. Training Strategy for 2D Models

We consider three different variants of training for 2D models: RECURRENT, TEACHER FORCING, and MARKOV. Li et al. (2021a) used RECURRENT training for their 2D FNO, wherein the model predicts the entire rollout during training autoregressively and the loss is back-propagated through time. Tran et al. (2023) found that TEACHER FORCING and

Table 10: Positional encoding analysis for NS. We present the best validation, test, and rotation test results for three choices of positional encoding: NONE, where we do not encode position, SYMMETRIC, where the encoding for each point is the distance from the middle of the grid, and CARTESIAN, where the encoding is the Cartesian coordinates of the grid point.

	POSITIONAL ENCODING	VALID (%)	TEST (%)	TEST$_{90°}$ (%)
FNO	NONE	9.05(0.19)	8.54(0.36)	130.14(2.06)
	SYMMETRIC	9.01(0.26)	8.95(0.18)	**129.08**(3.95)
	CARTESIAN	**8.64**(0.19)	**8.41**(0.41)	129.21(3.90)
FNO+$p4$	NONE	10.82(0.21)	10.46(0.31)	10.46(0.46)
	SYMMETRIC	**10.63**(0.07)	11.04(0.26)	10.47(0.36)
	CARTESIAN	10.70(0.49)	**10.44**(0.47)	**10.38**(0.38)
FNO+$p4m$	NONE	23.54(0.57)	23.07(1.29)	23.65(1.04)
	SYMMETRIC	23.67(1.31)	23.29(0.44)	23.09(0.29)
	CARTESIAN	**22.43**(1.59)	**22.09**(1.46)	**22.61**(1.54)
G-FNO-$p4$	NONE	**4.45**(0.27)	4.47(0.21)	4.47(0.21)
	SYMMETRIC	4.86(0.32)	4.78(0.39)	4.78(0.39)
	CARTESIAN	4.61(0.25)	**4.39**(0.25)	**4.39**(0.25)
G-FNO-$p4m$	NONE	7.15(0.25)	6.75(0.17)	6.75(0.17)
	SYMMETRIC	**6.73**(0.82)	**6.19**(0.61)	**6.19**(0.61)
	CARTESIAN	6.86(0.17)	6.67(0.32)	6.67(0.32)
U-NET-$p4$	NONE	**18.00**(0.60)	17.95(0.07)	**17.95**(0.06)
	SYMMETRIC	18.86(0.53)	18.40(0.44)	18.40(0.44)
	CARTESIAN	18.84(1.15)	**17.73**(0.13)	18.07(0.27)
RADIALFNO-$p4$	NONE	**9.56**(0.31)	**9.13**(0.09)	**9.13**(0.09)
	SYMMETRIC	9.59(0.05)	9.21(0.26)	9.21(0.26)
	CARTESIAN	9.79(0.31)	9.58(0.05)	24.75(20.20)
RADIALFNO-$p4m$	NONE	11.89(0.34)	11.01(0.04)	11.01(0.04)
	SYMMETRIC	11.51(0.68)	**10.86**(0.18)	**10.86**(0.18)
	CARTESIAN	**11.36**(0.44)	11.20(0.12)	11.84(1.02)

MARKOV training improved performance of FNO architectures relative to RECURRENT training. In both strategies, the model is trained to make predictions only one step into the future conditioned on the ground truth solutions at the previous time steps. However, under the MARKOV strategy, the model is only conditioned on the most recent time step, whereas models trained with TEACHER FORCING are conditioned on several of the most recent time steps.

In Table 9, we compare these training strategies based on their validation errors and super-resolution test errors. In all cases, our findings agree with those of Tran et al. (2023) in that MARKOV and TEACHER FORCING improve results relative to RECURRENT training. While for some models, MARKOV training gives a better error than TEACHER FORCING, we find that MARKOV training significantly reduces the ability of the baseline FNO to perform super-resolution, and thus, we opt to train all models using the TEACHER FORCING strategy.

B.5. Positional Encoding

We consider 3 variants of positional encoding: NONE, where position is not encoded, SYMMETRIC, where the encoding represents the distance of the grid point from the center of the grid, giving a roto-reflection invariant grid, and CARTESIAN, where the encoding is the Cartesian coordinates of each of the grid points. The resulting grid is then concatenated to the input of the network. As the grid is fixed and does not transform with the input, SYMMETRIC breaks translation equivariance while preserving roto-reflection equivariance, while CARTESIAN breaks translation and roto-reflection equivariance. NONE preserves all symmetries.

We present validation, test, and rotation test results for these encodings in Table 10 and find that, although results are mixed, in 5 of the 8 considered models, some form of positional encoding improves the validation error over NONE. In our experiments, we therefore elect to use SYMMETRIC positional encoding for all equivariant models to preserve roto-reflection equivariance and CARTESIAN positional encoding for FNO baselines (FNO, FNO+$p4$, and FNO+$p4m$). The exception is the SWE experiments, where the non-rectangular domain breaks the roto-reflection invariance of the SYMMETRIC encoding.

Table 11: Steerable G-FNO results on NS. We present test and rotation test results for a steerable parameterization of the G-FNO.

	# PAR. (M)	TEST (%)	TEST_{90° (%)
G-FNO-$p4$-STEER	0.83	20.87(1.25)	20.87(1.25)
G-FNO-$p4m$-STEER	0.89	22.58(0.41)	22.58(0.41)
G-FNO-$p4$	0.85	**4.78**(0.39)	**4.78**(0.39)
G-FNO-$p4m$	0.84	<u>6.19</u>(0.61)	<u>6.19</u>(0.61)

We therefore use CARTESIAN encoding for all models on this dataset.

B.6. Steerable parameterization of G-FNO

Steerable CNNs (Cohen & Welling, 2017) construct equivariant convolution layers using a steerable basis $\phi_1, \phi_2, \ldots, \phi_n$. The steerable basis for each layer is constructed offline by solving a linear system of equations dependent on the group representation of the input function and the desired representation of the output function. The steerable kernel ψ is then given by

$$\psi = \sum_{j=1}^{n} \alpha_j \phi_j \tag{16}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the learnable basis coefficients. Steerable convolutions are strictly more general than group convolutions, since choosing the input and output representation as the *regular* representation gives an alternative method for parameterizing group convolutions (Cohen & Welling, 2017). Furthermore, steerable convolutions can achieve equivariance to infinite groups such as continuous rotations (Weiler & Cesa, 2019).

To parameterize the steerable kernel ψ in the frequency domain, we use the linearity of the transform as

$$\mathcal{F}\psi = \sum_{j=1}^{n} \alpha_j \mathcal{F}\phi_j. \tag{17}$$

Thus, to learn a steerable kernel in the frequency domain, it is sufficient to learn the basis coefficients for the transform of the basis functions. This has the added benefit that all of the learnable parameters in the model, i.e., the basis coefficients, are real-valued.

In Table 11, we consider a steerable parameterization of G-FNO for the groups $p4$ and $p4m$. These initial results suggest this parameterization is not ideal, as the models given by the original parameterization are significantly better. Future work should investigate a more effective steerable parameterization so that equivariant frequency domain convolutions can be extended to continuous groups.

C. Proofs and Background

C.1. The Groups $p4$ and $p4m$

In this section, we characterize the groups $p4$ and $p4m$. $p4$ is the group generated by translations and 90° rotations, while $p4m$ is $p4$ with reflections.

For us, an element $g \in p4$ is parameterized by $s_g \in \{0, 1, 2, 3\}$ for a planar rotation and $x_g \in \mathbb{R}^2$ for a translation by x_g. We then let $R_{s_g} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{s_g}$, giving a rotation by $s_g \cdot 90^\circ$. The element g acts on a function $\nu : \mathbb{R}^2 \to \mathbb{R}$ by applying a rotation followed by a translation as

$$(L_g \nu)[x] = \nu\left(R_{s_g}^{-1} x - x_g\right). \tag{18}$$

For discretized functions on a numerical grid $\delta \cdot \mathbb{Z}^2$, the translations x_g are restricted to grid elements. For the sake of simplicity, we focus on this setting in the following, which corresponds to our discretization. The continuous case is perfectly analogous.

Next, for a function $\rho : p4 \to \mathbb{R}$ defined on $p4$, denote $\rho(h) = \rho(s_h, x_h)$. Here, it may be helpful to picture $p4$ as $\{0, 1, 2, 3\} \times \mathbb{Z}^2$, that is, a "stack" of 4 planes. The translation coordinate x_h indexes each of these planes, while the rotation coordinate s_h tracks the relative pose of the corresponding plane as it pertains to rotations, with $s_h = 0$ indicating that the plane is in its canonical orientation, $s_h = 1$ indicating that the plane is rotated by $90°$, etc. Then, g transforms ρ by applying a roto-translation to each of the planes and periodically incrementing the rotation coordinate s_h as

$$(L_g\rho)[h] = \rho\left((s_h + s_g) \bmod 4, R_{s_g}^{-1} x_h - x_g\right). \tag{19}$$

For example, for the "slice" of ρ representing the plane in the second position, i.e., rotated by $180°$, rotating ρ by $180°$ will bring that plane to its canonical orientation, that is $(2 + 2) \bmod 4 = 0$.

Further, for the transformation m in $p4m$, m is parameterized by s_m and x_m, with

$$s_m = \left(s_m^{(1)}, s_m^{(2)}\right) \in \{-1, 1\} \times \{0, 1, 2, 3\}.$$

Let R_{s_m} be a 2D orthogonal matrix corresponding to a rotation by $s_m^{(2)} \cdot 90°$ followed by a horizontal reflection if, and only if, $s_m^{(1)} = -1$. That is, $R_{s_m} = \begin{pmatrix} s_m^{(1)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^{s_m^{(2)}}$. Then, m transforms the function ν similar to before by applying a roto-reflection followed by a translation as

$$(L_m\nu)[x] = \nu\left(R_{s_m}^{-1} x - x_m\right). \tag{20}$$

Lastly, for the function $\eta : p4m \to \mathbb{R}$ defined on $p4m$, again denote $\eta(p) = \eta(s_p, x_p)$. Similar to the $p4$ case, x_p indexes into the "stack" of eight planes comprising $p4m$, while $s_p^{(1)}$ tracks the relative pose of the corresponding plane with respect to horizontal reflections and $s_p^{(2)}$ tracks the rotational pose. Then, m transforms η as

$$(L_m\eta)[p] = \eta\left(s_p^{(1)} \cdot s_m^{(1)}, (s_p^{(2)} + s_m^{(2)}) \bmod 4, R_{s_m}^{-1} x_p - x_m\right). \tag{21}$$

For example, for the "slice" of η representing the plane in the pose $s_p = (-1, 3)$, i.e., reflected and rotated by $270°$, applying a roto-reflection by $90°$ will bring that plane to its canonical orientation, that is $(-1 \cdot -1, (3 + 1) \bmod 4) = (1, 0)$. Note that vertical reflections are accomplished by composing a rotation and horizontal reflection.

C.2. Proof of Symmetry of Fourier transform to $O(n)$

Lemma C.1. *Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, $f : \mathbb{R}^n \to \mathbb{R}$ Lebesgue-integrable and $b \in \mathbb{R}^n$. Consider the function $f_{A,b} : \mathbb{R}^n \to \mathbb{R}$ given by $f_{A,b}(x) = f(Ax + b)$. Then*

$$\mathcal{F}(f_{(A,b)})(\xi) = \frac{e^{-2\pi i \langle A^{-T}\xi, b\rangle}}{|\det A|} \mathcal{F}(f)(A^{-T}\xi)$$

In particular, if A is an orthogonal matrix, then $|\det A| = 1$ and $A^{-T} = A$, so for all $O \in O(n)$:

$$\mathcal{F}(f_{(O,b)})(\xi) = e^{-2\pi i \langle O\xi, b\rangle} \mathcal{F}(f)(O\xi)$$

Proof. We will use the multi-dimensional change of variables formula with the substitution $z = Ax + b$, as well as the identity $\langle \xi, Az\rangle = \langle A^T\xi, z\rangle$.

$$\begin{aligned}
\mathcal{F}(f_{(A,b)})(\xi) &= \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{-2\pi i \langle \xi, x\rangle} f_{(A,b)}(x)\, dx \\
&= \frac{1}{(2\pi)^{n/2} |\det A|} \int_{\mathbb{R}^n} e^{-2\pi i \langle \xi, A^{-1}(Ax+b)\rangle + 2\pi i \langle \xi, A^{-1}b\rangle} f(Ax + b) |\det A| dx \\
&= e^{2\pi i \langle \xi, A^{-1}b\rangle} \frac{1}{(2\pi)^{n/2} |\det A|} \int_{\mathbb{R}^n} e^{-2\pi i \langle \xi, A^{-1}z\rangle} f(z)\, dz \\
&= \frac{e^{2\pi i \langle \xi, A^{-1}b\rangle}}{|\det A|} \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{-2\pi i \langle A^{-T}\xi, z\rangle} f(z)\, dz \\
&= \frac{e^{2\pi i \langle A^{-T}\xi, b\rangle}}{|\det A|} \mathcal{F}(f)(A^{-T}\xi).
\end{aligned}$$

□

C.3. Proof of Navier-Stokes Closure to Action of $O(2)$

Lemma C.2. *Let $U \subseteq \mathbb{R}^2$ be a domain or $U = \mathbb{T}^2 = \mathbb{R}^2/\mathbb{Z}^2$ the flat torus/periodic square in which we identify the top/bottom and left/right sides.*

Suppose that the functions $u : U \to \mathbb{R}^2$, $w, f : U \to \mathbb{R}$ solve the partial differential equation

$$\begin{aligned} \partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) &= \nu \Delta w(x,t) + f(x) \\ \nabla \cdot u(x,t) &= 0 \end{aligned} \tag{22}$$

Take R to be an orthogonal matrix describing a rotation or reflection. Then the functions $u_R : R^{-1}U \to \mathbb{R}^2$ and $w_R, f_R : R^{-1}U \to \mathbb{R}$ also satisfy Equation (22), where:

$$u_R(x,t) \coloneqq R^\top u(Rx,t) \tag{23}$$
$$w_R(x,t) \coloneqq w(Rx,t) \tag{24}$$
$$f_R(x) \coloneqq f(Rx) \tag{25}$$

Note that if f is invariant to the action of $O(2)$, then $f_R = f$, and thus, this result implies that u_R and w_R solve Equation (22).

Lemma C.2 is most meaningful in our context if the domain U is invariant under the rotation R, i.e. if $R^{-1}U = U$. This is true if $U = \mathbb{R}^2$, $U = B_1(0)$ is a disk in $\mathbb{R}^2$, or if $U = \mathbb{R}^2$ and R is a reflection or a rotation by $0°, 90°, 180°$ or $270°$. The case $U = \mathbb{T}^2$ is associated with the study of problems that are periodic in the coordinate directions.

Proof. For brevity of notation, we omit the time variable t.

To show that u_R, w_R and f_R satisfy Equation (22), it is enough to show that:

$$(u_R \cdot \nabla w_R)(x) = (u \cdot \nabla w)(Rx) \tag{26}$$
$$(\Delta w_R)(x) = (\Delta w)(Rx) \tag{27}$$
$$(\nabla \cdot u_R)(x) = (\nabla \cdot u)(Rx) \tag{28}$$

First, to show Equation (26), let $D(\nu) \in \mathbb{R}^{d \times q}$ denote the Jacobian matrix for $\nu : \mathbb{R}^q \to \mathbb{R}^d$. Then:

$$\begin{aligned} (u_R \cdot \nabla w_R)(x) &= u(Rx)^\top R \left(D\left(w(Rx)\right) R\right)^\top \\ &= (u \cdot \nabla w)(Rx) \end{aligned}$$

Next, to show Equation (27), let $\delta_{j,k}$ be the Kronecker delta, which is 1 if $j = k$ and 0 otherwise:

$$
\begin{aligned}
\Delta w_R(x) &= \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\frac{\partial}{\partial x_i} w(Rx) \right) \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial}{\partial x_i} \left(\frac{\partial w}{\partial x_j}(Rx) \sum_{k=1}^{2} \frac{\partial (R_{j,k} x_k)}{\partial x_i} \right) \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial}{\partial x_i} \left(\frac{\partial w}{\partial x_j}(Rx) R_{j,i} \right) \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} R_{j,i} \frac{\partial w}{\partial x_k \partial x_j}(Rx) \sum_{l=1}^{2} \frac{\partial (R_{k,l} x_l)}{\partial x_i} \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} R_{j,i} \frac{\partial w}{\partial x_k \partial x_j}(Rx) R_{k,i} \\
&= \sum_{j=1}^{2} \sum_{k=1}^{2} \frac{\partial w}{\partial x_k \partial x_j}(Rx) (R R^\top)_{j,k} \\
&= \sum_{j=1}^{2} \sum_{k=1}^{2} \frac{\partial w}{\partial x_k \partial x_j}(Rx) \delta_{j,k} \\
&= (\Delta w)(Rx)
\end{aligned}
$$

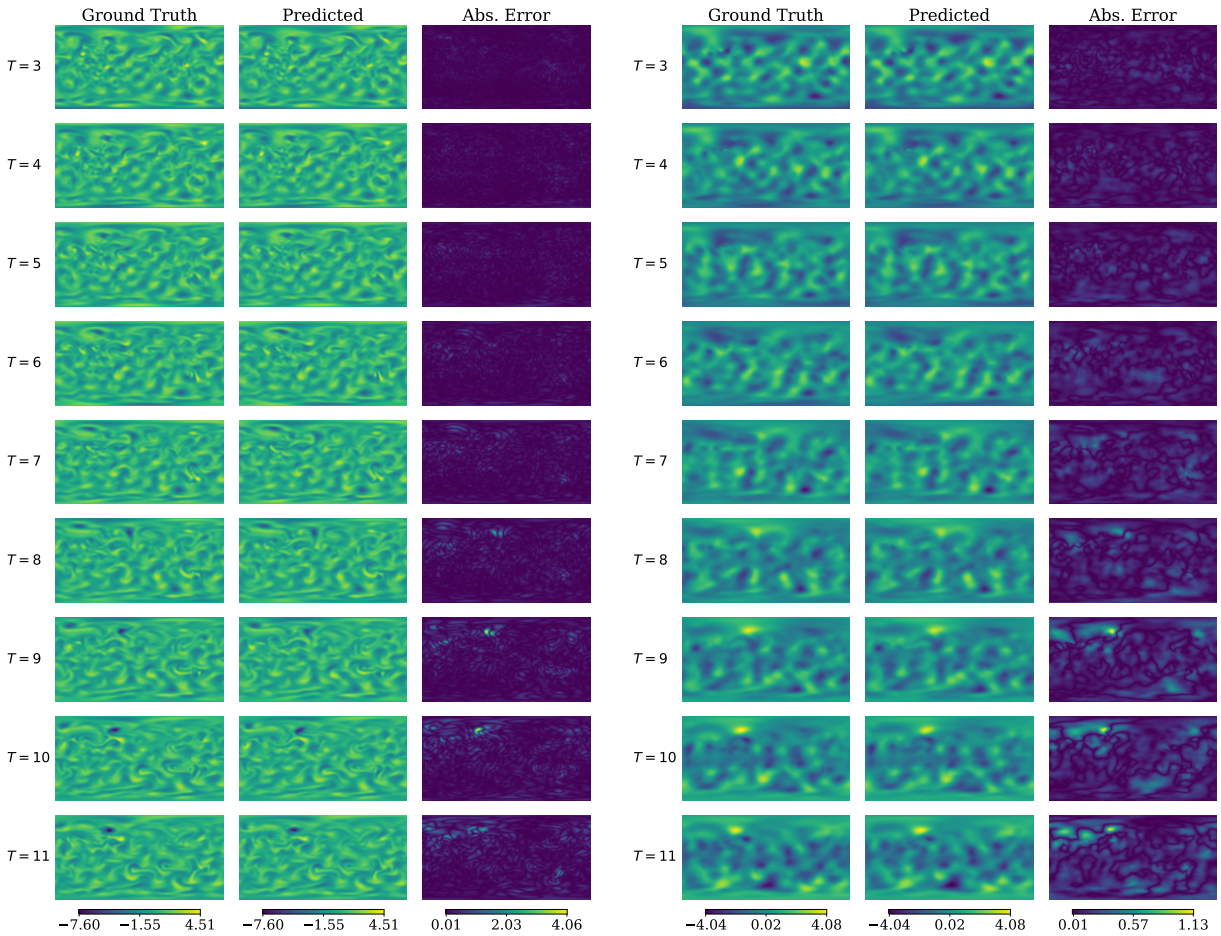
Lastly, to show Equation (28):

$$
\begin{aligned}
\nabla \cdot u_R(x) &= \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(R^\top u(Rx) \right)_i \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial}{\partial x_i} R_{j,i} u_j(Rx) \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} R_{j,i} \frac{\partial u_j}{\partial x_k}(Rx) \sum_{l=1}^{2} \frac{\partial (R_{k,l} x_l)}{\partial x_i} \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} R_{j,i} \frac{\partial u_j}{\partial x_k}(Rx) R_{k,i} \\
&= \sum_{j=1}^{2} \sum_{k=1}^{2} \frac{\partial u_j}{\partial x_k}(Rx) (R R^\top)_{j,k} \\
&= \sum_{j=1}^{2} \sum_{k=1}^{2} \frac{\partial u_j}{\partial x_k}(Rx) \delta_{j,k} \\
&= (\nabla \cdot u)(Rx)
\end{aligned}
$$

□

D. Rollout Visualizations

In this section, we randomly select a trajectory from the test set and visualize the ground truth rollout alongside the rollout predicted by the 2D version of G-FNO-$p4$ trained to make predictions autoregressively. We visualize SWE in Figure 4, SWE-SYM in Figure 5, NS in Figure 6, and NS-SYM in Figure 7.



(a) Vorticity (b) Pressure

Figure 4: Rollout of SWE.

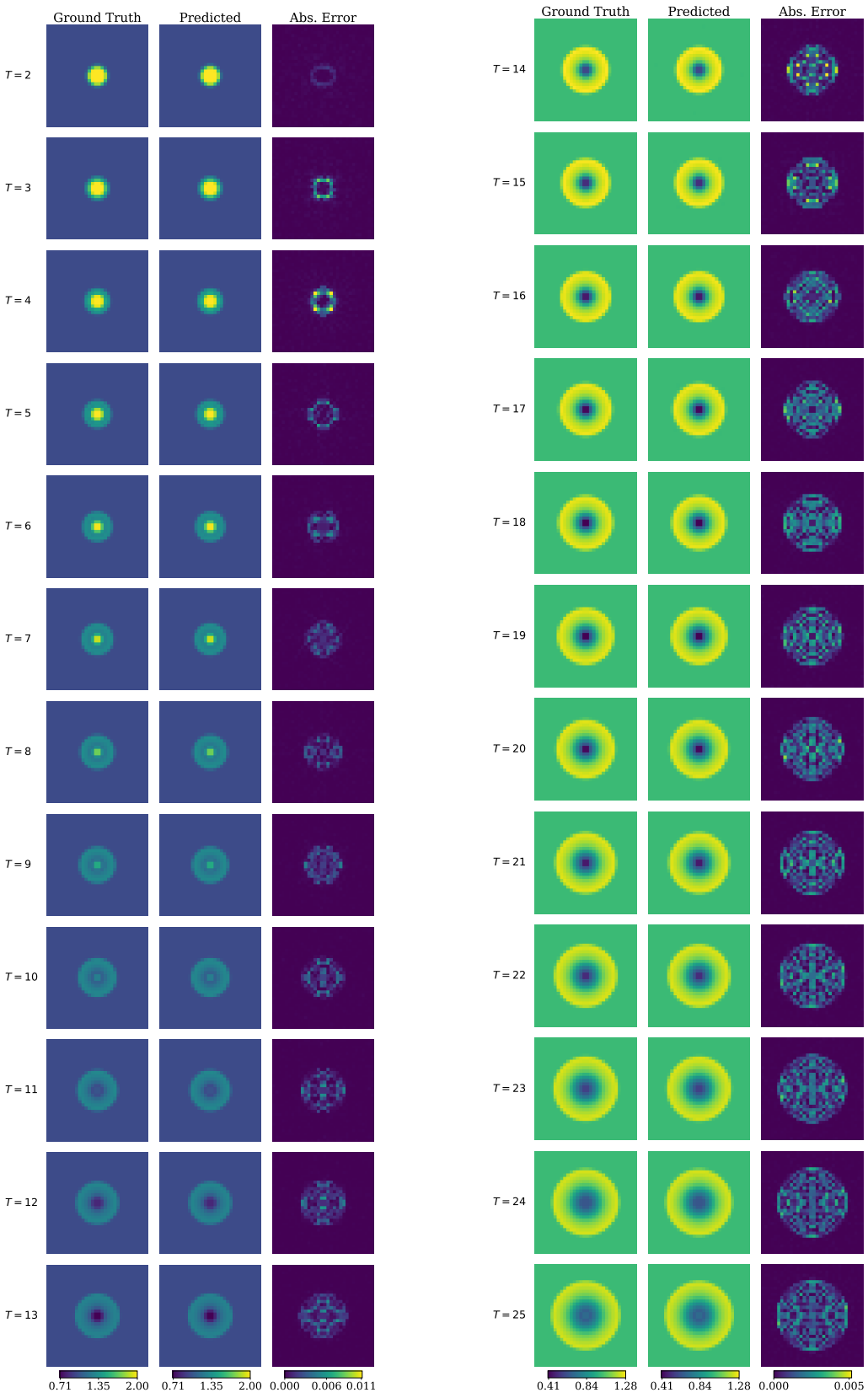


Figure 5: Rollout of SWE-SYM. Note that the scales of the error bars on the left and the right differ.

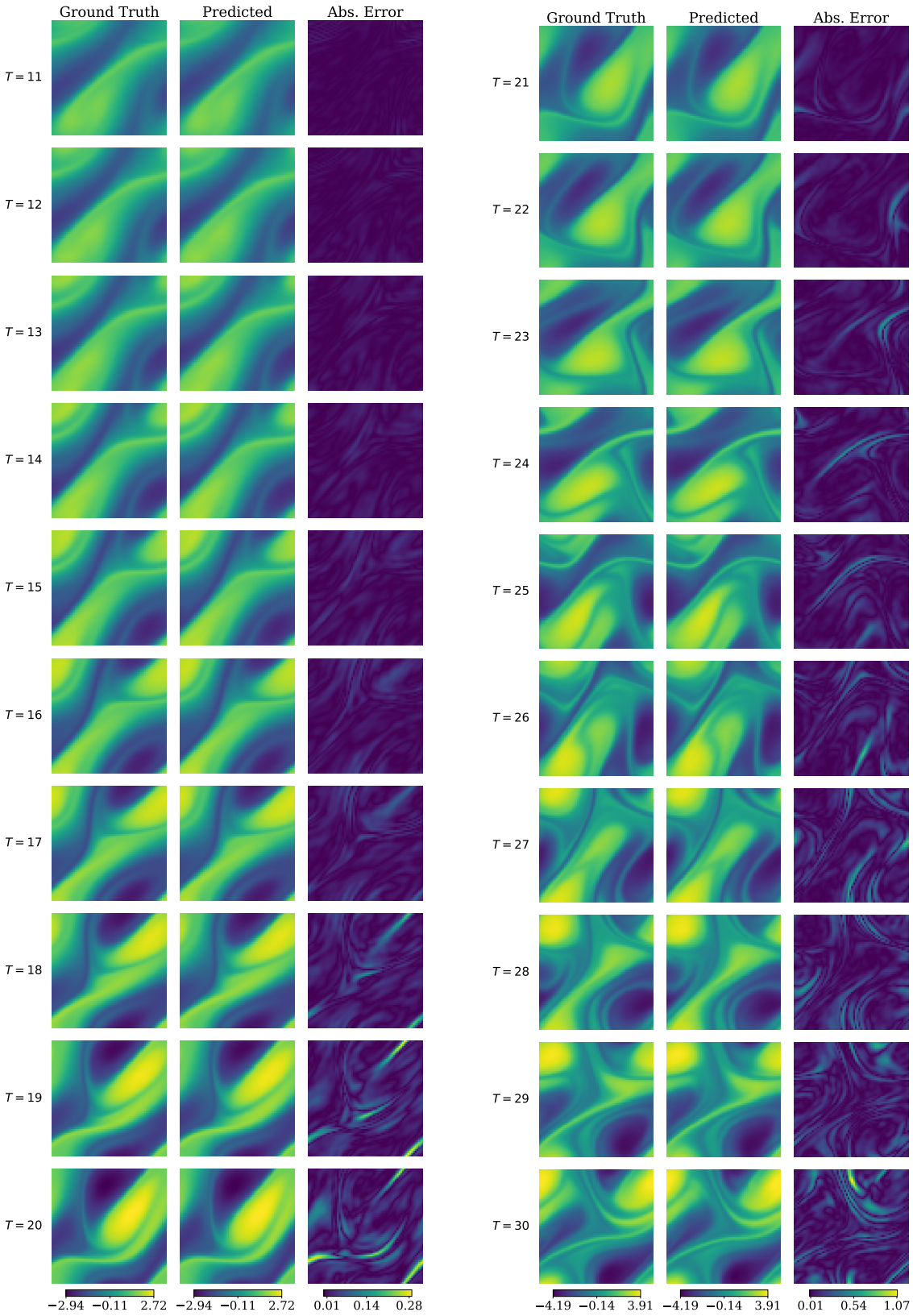


Figure 6: Rollout of Navier-Stokes with non-symmetric forcing (NS). Note that the scales of the error bars on the left and the right differ.

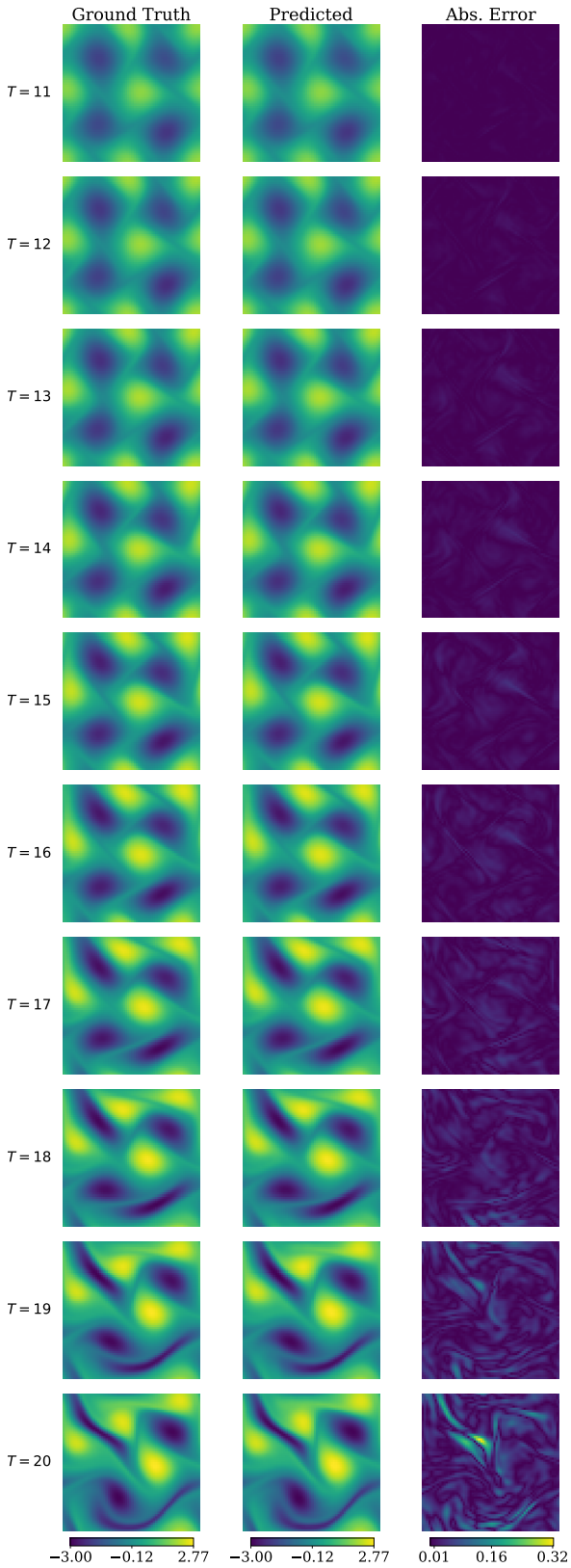


Figure 7: Rollout of Navier-Stokes with symmetric forcing (NS-SYM).