Harnessing the Power of Neural Operators with Automatically Encoded Conservation Laws

Ning Liu [*1] **Yiming Fan** [*2] **Xianyi Zeng** [2] **Milan Klöwer** [3] **Lu Zhang** [2] **Yue Yu** [2]

Abstract

Neural operators (NOs) have emerged as effective tools for modeling complex physical systems in scientific machine learning. In NOs, a central characteristic is to learn the governing physical laws directly from data. In contrast to other machine learning applications, partial knowledge is often known a priori about the physical system at hand whereby quantities such as mass, energy and momentum are exactly conserved. Currently, NOs have to learn these conservation laws from data and can only approximately satisfy them due to finite training data and random noise. In this work, we introduce conservation law-encoded neural operators (clawNOs), a suite of NOs that endow inference with automatic satisfaction of such conservation laws. ClawNOs are built with a divergence-free prediction of the solution field, with which the continuity equation is automatically guaranteed. As a consequence, clawNOs are compliant with the most fundamental and ubiquitous conservation laws essential for correct physical consistency. As demonstrations, we consider a wide variety of scientific applications ranging from constitutive modeling of material deformation, incompressible fluid dynamics, to atmospheric simulation. ClawNOs significantly outperform the state-of-the-art NOs in learning efficacy, especially in small-data regimes. Our code and data accompanying this paper are available at `https://github.com/ningliu-iga/clawNO`.

[*]Equal contribution [1]Global Engineering and Materials, Inc., Princeton, NJ 08540, USA [2]Department of Mathematics, Lehigh University, Bethlehem, PA 18015, USA [3]Earth, Atmospheric and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Correspondence to: Yue Yu <yuy214@lehigh.edu>, Ning Liu <ningliu@umich.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Deep neural networks (DNN) have achieved tremendous progress in areas where the underlying laws are unknown. In computer vision applications convolutional neural networks have been very popular even though it is unclear how the lower-dimensional manifold of "valid" images is parameterized (He et al., 2016; Ren et al., 2015; Krizhevsky et al., 2012). That means, such a manifold has to be discovered in a purely data-driven way by feeding the network vast amounts of data. Another recent popular application of DNNs is the modeling and calibration process of physics-based problems from experimental measurements (Ranade et al., 2021; Schmidt & Lipson, 2009; Çolak, 2021; Jin et al., 2023; Vinuesa et al., 2023). A wide range of physical applications entail the learning of solution operators, i.e., the learning of infinite dimensional function mappings between any parametric dependence to the solution field. A prototypical instance is the case of modeling fluid dynamics, where the initial input needs to be mapped to a temporal sequence of flow states. Like the time integration in numerical modelling, an operator is required that takes the current flow state and maps it to the predicted state one time step later. To this end, the neural operator (NO) (Anandkumar et al., 2020; Li et al., 2020c; Lu et al., 2019; Gupta et al., 2021; Cao, 2021) is introduced, which learns a surrogate mapping between function spaces with resolution independence as well as generalizability to different input instances. These facts make NOs excellent candidates in discovering models for complex physical systems directly from data.

Herein, we consider data-driven model discovery of physics-based problems using NOs (Li et al., 2021; Goswami et al., 2022b). In contrast to computer vision applications, physics-based applications are often at least partially constrained by well-known fundamental laws. As a famous example, the motion of a particle in an external potential should conserve energy and momentum, while the exact form of the potential or the expression for the momentum equations is still unknown and needs to be inferred from observations. However, most of the current NOs have been focused on a pure data-driven paradigm, which neglects these intrinsic conservation of fundamental physical laws in data. As a result, their performances highly rely on the quantity and

coverage of available data.

To improve learning efficacy and robustness in small-data regimes, we propose to encode a series of fundamental conservation laws into the architecture of NOs. Their inference is then constrained to a physically-consistent manifold. Here, we focus on the conservation of mass or volume which leads to a continuity equation for divergence-free flow. The development is based on **two key innovations**. Firstly, when the output function of a NO is divergence-free, the continuity equation is automatically satisfied, guaranteeing conservation of volume and mass. Based on the concept of differential forms, the conservation law is embedded through building divergence-free output functions. Secondly, to evaluate the differential forms, we propose an additional linear layer in NOs, whose weights are prespecified based on high-order numerical differentiations on the given grids. Given an input function and its values on given grids, this layer evaluates the spatial derivatives as weighted linear combinations of neighboring or global points, mapping the function to its approximated derivatives.

Compared to existing NO methods, the proposed architecture mainly carries three significant advantages. First, different from existing physics-informed neural operators (Goswami et al., 2022a;b; Li et al., 2021; Wang et al., 2021), our approach is readily applicable to learn physical systems directly from experimental measurements, since it only requires observed data pairs and does not rely on fully known governing equations. Second, the conservation laws in our approach are realized through built-in architectures, which constrain the output function to a physically consistent manifold independent of noisy measurement and/or scarce data coverage. Third, our architecture is designed to encode general conservation laws at large. Besides the conservation of mass, linear/angular momentum as demonstrated in our examples, it is readily applicable to the conservations of energy and electric charge, and can be easily scaled up to higher dimensions. In summary, the main contributions of our work are:

- We propose clawNO, a novel neural operator architecture to learn complex physical systems with conservation laws baked into the design.
- Our architecture is realized by adding an additional layer that employs numerical differentiation in evaluating the spatial and temporal derivatives as a weighted linear combination of neighboring points. As a result, our design is readily applicable as an add-on to any neural operator architecture, with comparable network size and computational cost.
- ClawNO only requires data pairs and does not rely on *a priori* domain knowledge, while the guaranteed conservation laws improve the learning efficacy, especially in the small-data regime.

2. Background and Related Work

Learning hidden physics. Learning and predicting complex physics directly from data is ubiquitous in many scientific and engineering applications (Ghaboussi et al., 1998; 1991; Carleo et al., 2019; Karniadakis et al., 2021; Zhang et al., 2018; Cai et al., 2022; Pfau et al., 2020; He et al., 2021; Besnard et al., 2006). Amongst these real-world physical problems, the underlying governing laws largely remain unknown, and machine learning models act to discover hidden physics from data through training. Successful examples include graph neural networks (GNNs) in discovering molecular properties (Wieder et al., 2020; Wu et al., 2023a), neural ODEs in constructing strain energy formulation for materials (Tac et al., 2022), and CNNs in steady flow approximation (Guo et al., 2016), etc. In this work, we focus on learning hidden physics from measurements with fundamental physical laws enforced. Examples include learning material deformation model from experimental measurements, where the mass should be conserved, but the constitutive law remains hidden.

Neural operator learning. Among others, NOs possess superiority in discovering physical laws as function mappings. Contrary to classical neural networks (NNs) that operate between finite-dimensional Euclidean spaces, NOs are designed to learn mappings between infinite-dimensional function spaces (Li et al., 2020a;b;c; You et al., 2022a; Ong et al., 2022; Cao, 2021; Lu et al., 2019; 2021; Goswami et al., 2022a; Gupta et al., 2021). As a result, NOs are often employed to manifest the mapping between spatial and/or spatio-temporal data pairs. Compared with classical NNs, the most notable advantages of NOs are their resolution independence and generalizability to different input instances. Moreover, NOs only require data with no knowledge on the underlying governing laws. All these advantages make NOs promising tools to learn hidden physics directly from data.

Despite the aforementioned advances, purely data-driven NOs still suffer from the data challenge: they require a large set of paired data, which is prohibitively expensive in many engineering applications. To resolve this challenge, physics-informed neural operator (PINO) (Li et al., 2021) and physics-informed DeepONets (Goswami et al., 2022a; Wang et al., 2021) are introduced, where a PDE-based loss is added to the training loss as a penalization term. However, these approaches often require *a priori* knowledge of all the underlying physics laws (in the form of governing PDEs). Moreover, the penalization term only imposes these knowledge as a weak constraint.

Imposing partial physical laws in NNs. Several NN models have pursued the direction of imposing partial physics

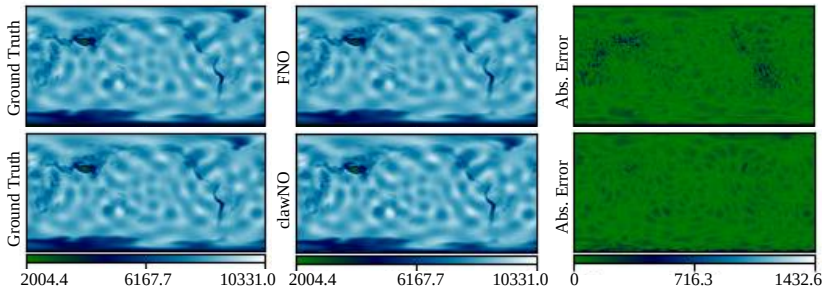


Figure 1. Predictability demo in atmospheric modeling. While FNO plausibly learns the wave propagation patterns, its relative L2 error is twice as big as that of clawNO (cf. Section 4.3), especially in the vicinity of mountains (Fig. 6). ClawNOs automatically satisfy the conservation law and improve physical consistency.

knowledge into the architecture of the model, with the goal of improving efficiency in learning hidden physics. A popular approach is to design equivariant NNs (Satorras et al., 2021; Müller, 2023; Liu et al., 2023; Helwig et al., 2023), which enforces symmetries in a Lagrangian represented by NNs and guarantees that the discovered physical laws do not depend on the coordinate system used to describe them. As pointed out by Sarlet & Cantrijn (1981); Noether (1971), such symmetries induce the conservation of linear and angular momentum in the learned physical system. Towards the conservation law of other quantities, the authors in Keller & Evans (2019); Sturm & Wexler (2022) consider the temporal concentration changes of multiple species. They find that conserving mass via a balancing operation improves the accuracy and physical consistency of NNs. In Richter-Powell et al. (2022), a divergence-free perspective has been considered to impose more general conservation laws in the framework of physics-informed neural networks (PINNs) (Raissi et al., 2019; Cai et al., 2021) with the aid of automatic differentiation to evaluate the derivatives directly. The same problem is also tackled by Négiar et al. (2022); Hansen et al. (2023), where the conservation law is enforced by projecting the solution to the constrained space (Négiar et al., 2022) or encoded by taking the predictive update following an integral form of the full governing equation (Hansen et al., 2023). However, in all these conservation law-informed neural networks the full governing PDEs are employed to formulate the loss function or the predictive updates/projections, and hence they are not applicable to hidden physics discovery.

3. ClawNO: Conservation Law Encoded Neural Operator

We consider the learning of complex physical systems based on a number of observations of the input function values $\mathbf{f}_i(\mathbf{x}) \in \mathbb{F}(\Omega; \mathbb{R}^{p_f})$ and the corresponding output function values $\mathbf{u}_i(\mathbf{x}) \in \mathbb{U}(\Omega; \mathbb{R}^{p_u})$. Here, i denotes the sample index, $\Omega \in \mathbb{R}^p$ is the bounded domain of interest, and $\mathbb{F}$ and $\mathbb{U}$ describe the Banach spaces of functions taking values in $\mathbb{R}^{p_f}$ and $\mathbb{R}^{p_u}$, respectively. To model the physical laws of such a system, we aim to learn the intrinsic operator $\mathcal{G} : \mathbb{F} \to \mathbb{U}$, that maps the input function $\mathbf{f}(\mathbf{x})$ to the output function $\mathbf{u}(\mathbf{x})$. In other words, given the measurements of $\mathbf{f}_i$ and $\mathbf{u}_i$ on a collection of points $\chi = \{\mathbf{x}_j\}_{j=1}^M \subset \Omega$, we seek to learn the physical response by constructing a parameterized surrogate operator of $\mathcal{G}$: $\tilde{\mathcal{G}}[\mathbf{f}; \theta](\mathbf{x}) \approx \mathbf{u}(\mathbf{x})$, where the trainable parameter set θ is obtained by solving the optimization problem:

$$\min_{\theta \in \Theta} \mathcal{L}(\theta) := \min_{\theta \in \Theta} \sum_{i=1}^{N} \frac{1}{N} [C(\tilde{\mathcal{G}}[\mathbf{f}_i; \theta], \mathbf{u}_i)] \tag{1}$$
$$\approx \min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{j=1}^{M} \left|\left| \tilde{\mathcal{G}}[\mathbf{f}_i; \theta](\mathbf{x}_j) - \mathbf{u}_i(\mathbf{x}_j) \right|\right|^2}{\sum_{j=1}^{M} ||\mathbf{u}_i(\mathbf{x}_j)||^2} .$$

Here, C denotes the cost functional that is often taken as the relative mean squared error.

Although our approach is agnostic to any NO architecture, for the purpose of demonstration we consider the generic integral neural operators (Li et al., 2020b;c; You et al., 2022a;b) as the base models. In this context, an L-layer NO has the following form:

$$\tilde{\mathcal{G}}[\mathbf{f}; \theta](\mathbf{x}) := \mathcal{Q} \circ \mathcal{J}_L \circ \cdots \circ \mathcal{J}_1 \circ \mathcal{P}[\mathbf{f}](\mathbf{x}) , \tag{2}$$

where $\mathcal{P}$, $\mathcal{Q}$ are shallow-layer NNs that map a low-dimensional vector into a high-dimensional vector and vice versa. Each intermediate layer, $\mathcal{J}_l$, consists of a local linear transformation operator $\mathcal{W}_l$, an integral (nonlocal) kernel operator $\mathcal{K}_l$, and an activation function σ. The architectures of NOs mainly differ in the design of their intermediate layer update rules. As two popular examples, when considering the problem with structured domain Ω and uniform grid set χ, a Fourier neural operator (FNO) is widely used, where the integral kernel operators $\mathcal{K}_l$ are linear transformations in frequency space. As such, the (l+1)th-layer feature function $\mathbf{h}(\mathbf{x}, l+1)$ is calculated based on the lth-layer feature function $\mathbf{h}(\mathbf{x}, l)$ via:

$$\mathbf{h}(\mathbf{x}, l+1) = \mathcal{J}_l^{FNO}[\mathbf{h}(\mathbf{x}, l)] \tag{3}$$
$$:= \sigma(W_l \mathbf{h}(\mathbf{x}, l) + \mathcal{F}^{-1}[A_l \cdot \mathcal{F}[\mathbf{h}(\cdot, l)]](\mathbf{x}) + \mathbf{c}_l) ,$$

where W_l, $\mathbf{c}_l$ and A_l are matrices to be optimized, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively. On the other hand, the graph neural operator (GNO) is often employed on general irregular domains and grids, where the intermediate layer is invariant with respect to l, i.e., $\mathcal{J}_1 = \cdots, \mathcal{J}_L := \mathcal{J}^{GNO}$, with the update of each layer given by

$$\begin{aligned} &\mathbf{h}(\mathbf{x}, l+1) = \mathcal{J}^{GNO}[\mathbf{h}(\mathbf{x}, l)] \\ &:= \sigma(W\mathbf{h}(\mathbf{x}, l) + \int_\Omega \kappa(\mathbf{x}, \mathbf{y}, \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}); \omega)\mathbf{h}(\mathbf{y}, l)d\mathbf{y} + \mathbf{c}) \,. \end{aligned} \tag{4}$$

Here, $\kappa(\mathbf{x}, \mathbf{y}, \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}); \omega)$ is a tensor kernel function that takes the form of a (usually shallow) NN with parameter ω. In (4), the integral operator is often evaluated via a Riemann sum approximation $\int_\Omega \kappa(\mathbf{x}, \mathbf{y}, \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}); \omega)\mathbf{h}(\mathbf{y}, l)d\mathbf{y} \approx \frac{1}{M}\sum_{j=1}^M \kappa(\mathbf{x}, \mathbf{x}_j, \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}_j); \omega)\mathbf{h}(\mathbf{x}_j, l)$, realized through a message passing GNN architecture on a fully connected graph.

3.1. Built-In Divergence-Free Prediction

Intuitively, a conservation law states that a quantity of a physical system does not change as the system evolves over time. The most well-known examples of conserved quantities include mass, energy, linear momentum, angular momentum, etc. Mathematically, a conservation law can be expressed as a continuity equation defining the relation between the amount of the quantity and the "transport" of that quantity:

$$\frac{\partial \rho}{\partial t}(\mathbf{x}, t) + \nabla_{\mathbf{x}} \cdot \mu(\mathbf{x}, t) = 0 \,, \quad \text{for } (\mathbf{x}, t) \in \Omega \times [0, T] \,, \tag{5}$$

where $\rho : \mathbb{R}^{p+1} \to \mathbb{R}^+$ is the volume density of the quantity to be conserved, $\mu : \mathbb{R}^{p+1} \to \mathbb{R}^p$ is the flux describing how this quantity flows, and $\nabla_{\mathbf{x}}$ denotes the gradient operator with respect to the spatial dimensions. Equation 5 states that the amount of the conserved quantity within a volume can only change by the amount of the quantity that flows in and out of the volume. Taking the mass conservation law as an instance, ρ is the fluid's density, and $\mathbf{u}$ is the velocity at each point, and then the mass flux becomes $\mu = \rho\mathbf{u}$. When considering the (quasi-)static problem, i.e., ρ is independent of time, we have $\nabla_{\mathbf{x}} \cdot \mathbf{u} = 0$, where the divergence is taken only over spatial variables. Such a divergence-free equation is also referred to as the condition of incompressibility in fluid dynamics. Moreover, we note that (5) can always be expressed as a divergence-free equation when considering both spatial and temporal dimensions. As a matter of fact, by taking $\mathbf{X} = (\mathbf{x}, t)$ and $\mathbf{U} = (\mu, \rho)$, (5) can be re-written as $\nabla_{\mathbf{X}} \cdot \mathbf{U} = 0$, there the gradient operator $\nabla_{\mathbf{X}}$ is taken with respect to both $\mathbf{x}$ and t. Then, the conservation law becomes equivalent to the static case, if we consider the augmented variable $\mathbf{X}$ on the augmented domain $\tilde{\Omega} := \Omega \times [0, T]$. Hence, in the following derivation we focus on the static case, with the goal of designing an NO architecture with the continuity equation $\text{div}(\mathbf{u}) := \nabla_{\mathbf{x}}(\mathbf{u}) = 0$ automatically guaranteed for its output function $\mathbf{u}$.

To construct divergence-free NOs, we start from differential forms in $\mathbb{R}^p$, following the derivations in e.g., Barbarosie (2011); Kelliher (2021). More details are provided in Appendix A. Generally, an object that may be integrated over a $k-$dimensional manifold is called a $k-$form, which can be expressed as $\mu = \sum_{1 \le i_1, i_2, \cdots, i_k \le p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k} := \sum_I \mu_I dx_I$. For instance, any scalar function $g(\mathbf{x}) \in C^\infty$ is a $0-$form, and $v = \sum_{i=1}^p v_i(\mathbf{x})dx_i$, $v_i(\mathbf{x}) \in C^\infty$, is a $1-$form. Denote d as the exterior derivative, which is an operation acting on a $k-$form and produces a $(k+1)-$form (e.g., $dg(\mathbf{x}) = \sum_{i=1}^p \frac{\partial g}{\partial x_i} dx_i$), and the Hodge operator as $\star$, which matches to each $k-$form an $(p-k)-$form via $\star dx_I := (-1)^{\sigma(I, I^c)} dx_{I^c}$. Here, $I^c := \{1, \cdots, n\} \backslash I$ is the complement set of I, and $\sigma(I, I^c)$ is the sign of the permutation (I, I^c). A simple calculation yields that $\star\star\mu = (-1)^{k(p-k)}\mu$ and $dd\mu = 0$ for any $k-$form μ.

To see the connection of these definitions to our goal, we note that our output function, $\mathbf{u} : \mathbb{R}^p \to \mathbb{R}^p$, can be equivalently expressed as a $1-$form $\mathbf{u} = \sum_{i=1}^p u_i(\mathbf{x})dx_i$. Then, the divergence of $\mathbf{u}$ can be expressed as $d \star \mathbf{u}$, and the above fundamental properties of the exterior derivative yields $\text{div}(\star d\mu) = d \star \star d\mu = dd\mu = 0$ for any $(p-2)-$form μ. That means, any vector field $\mathbf{u}$ satisfying $\mathbf{u} = \star d\mu$ is divergence-free. Our goal is therefore to parameterize the output function of NOs in the form of $\star d\mu$, or equivalently, as $\star d(\sum_{1 \le i_1, \cdots, i_{p-2} \le p} \mu_{(i_1, \cdots, i_{p-2})} dx_{i_1} \wedge \cdots \wedge dx_{i_{p-2}})) = \sum_{i=1}^p \sum_{j=1}^p \frac{\partial \mu_{ij}}{\partial x_j} dx_i$. Here, $\mu_{ij}(\mathbf{x})$ is a function defined on Ω with $\mu_{ij}(\mathbf{x}) = \mu_{(i,j)^c}$ for $i < j$ and $\mu_{ij}(\mathbf{x}) = -\mu_{(j,i)^c}$ otherwise, as real-valued alternating functions. That means, taking any $\mu := [\mu_{ij}(\mathbf{x})]$ satisfies $\mu_{ij} = -\mu_{ji}$ (a p by p skew-symmetric matrix-valued function), we have $\text{div}(\star d\mu) = 0$, where $\star d\mu = [\text{div}(\mu_1), \cdots, \text{div}(\mu_p)]^T$ and μ_i stands for the $i-$th row of μ.

Remark: The idea of representing the conservation-law informed solution into the divergence of a skew-symmetric matrix-valued function is theoretically studied in Barbarosie (2011); Kelliher (2021), and recently employed in Richter-Powell et al. (2022) with the PINNs architecture. However, we point out that our scope of work is substantially different from Richter-Powell et al. (2022) where the governing equation is given. In PINN architecture the NN is constructed to approximate the solutions of a particular governing equation, i.e., mapping from $\mathbf{x}$ to the solution $\mathbf{u}(\mathbf{x})$. As such, the differential forms are approximated with automatic differentiation in NNs. In our work, we focus on the hidden physics learning problem where the governing equation remains *unknown* and employ the neural operator model to learn the governing law as a function-to-function mapping

directly from data. Besides the fact that we do not rely on an equation-based loss function, in neural operators the differential forms can no longer be approximated via automatic differentiation, and therefore we propose a pre-calculated numerical differentiation layer as will be elaborated below.

3.2. ClawNO Architecture and Implementation

With the above analysis, we now construct an NO architecture with divergence-free output. Concretely, we propose to modify the architecture in (2) as:

$$\tilde{\mathcal{G}}[\mathbf{f};\theta](\mathbf{x}) := \mathcal{D}\circ\mathcal{Q}\circ\mathcal{J}_L\circ\cdots\circ\mathcal{J}_1\circ\mathcal{P}[\mathbf{f}](\mathbf{x})\,, \quad (6)$$

where the projection layer, $\mathcal{Q}$, maps the last layer feature vector $\mathbf{h}(\mathbf{x}, L)$ to a dimension $p(p-1)/2$ vector, such that each dimension in this vector stands for one degree of freedom in the skew symmetric matrix-valued function $\mu(\mathbf{x})$. Then, the last layer $\mathcal{D}$ takes μ as the input and aims to calculate its row-wise divergence $[\text{div}(\mu_1),\cdots,\text{div}(\mu_p)]^T$ as the output function $\mathbf{u}(\mathbf{x})$.

To formulate the last layer $\mathcal{D}$ there remains one more technical challenge: in NOs the output functions are provided on a set of discrete grid points, χ, and therefore the derivatives are not readily evaluated. With the key observation that in classical numerical differentiation methods the derivatives are evaluated as weighted linear combinations of neighboring or global points, we propose to encode these weights in $\mathcal{D}$. As such, $\mathcal{D}$ will have pre-calculated weights and act as a numerical approximation of $[\text{div}(\mu_1),\cdots,\text{div}(\mu_p)]^T$. To demonstrate the idea, in the following we start with uniform grids χ and Ω being a periodic piece of $\mathbb{R}^p$, i.e., the torus $\mathbb{T}^p$, then discuss the scenarios with non-periodic domain and non-uniform grids.

Considering $\mathbb{T}^p := \prod_{i=1}^p[0,L^{(i)}]/\sim$ and $\chi := \left\{(\frac{k^{(1)}L^{(1)}}{N^{(1)}},\cdots,\frac{k^{(p)}L^{(p)}}{N^{(p)}})|k^{(i)}\in[0,N^{(i)}]\cap\mathbb{Z}\right\}$, each component of μ can then be approximated as a truncated Fourier series:

$$\mu_{jk}^{(N^{(1)},\cdots,N^{(p)})}(\mathbf{x}) := \sum_{\xi_1=-\frac{N^{(1)}}{2}}^{\frac{N^{(1)}}{2}-1}\cdots\sum_{\xi_p=-\frac{N^{(p)}}{2}}^{\frac{N^{(p)}}{2}-1}\hat{\mu}_{jk}(\xi_1,\cdots,\xi_p)e^{\frac{i2\pi\xi_1}{L^{(1)}}}\cdots e^{\frac{i2\pi\xi_p}{L^{(p)}}}\,,$$

where $\hat{\mu}_{jk}(\xi_1,\cdots,\xi_p)$ is the Fourier coefficient of μ_{jk}, calculated via $\mathcal{F}[\mu_{jk}]$. Taking derivatives of $\mu_{jk}^{(N^{(1)},\cdots,N^{(p)})}$ and using them to approximate the derivative of μ_{jk}, we have the following result:

Theorem 3.1. $\mathcal{D}$ is a $p(p-1)/2\times M\times M\times p$ tensor with its parameters given by: $\mathcal{D}[\mu] = \star d\mu^{(N^{(1)},\cdots,N^{(p)})} = [(\mathcal{D}[\mu])_1,\cdots,(\mathcal{D}[\mu])_p]^T$, where

$$(\mathcal{D}[\mu])_j = \sum_k\frac{\partial\mu_{jk}^{(N^{(1)},\cdots,N^{(p)})}}{\partial x_k} = \sum_k\mathcal{F}^{-1}\left[i\frac{2\pi\xi_k}{L^{(k)}}\mathcal{F}[\mu_{jk}](\boldsymbol{\xi})\right], \quad (7)$$

then the following error estimate holds true:

$$||\star d\mu - \mathcal{D}[\mu]||_{L^\infty} = O(\Delta x^{m-1})\,,$$

if μ has $m-1$ continuous derivatives for some $m\geq 2$ and a $m-$th derivative of bounded variation. In the special case when μ is smooth, a spectral convergence is obtained. Here, $M := \prod_{i=1}^p N^{(i)}$ denotes the total number of grids, and $\Delta x := \max_{i=1}^p L^{(i)}/N^{(i)}$ is the spatial grid size.

Proof. Derivation of (7) is provided in Appendix E.1. The theorem is an immediate proposition of the Fourier spectral differentiation error estimate (see, e.g., Trefethen (2000, Page 34)). □

We now consider non-periodic domain and/or non-uniform grids. When the domain is non-periodic but the grids are uniform, we employ the Fourier continuation (FC) technique (Amlani & Bruno, 2016; Maust et al., 2022) which extends the non-periodic model output into a periodic function. In particular, the layer parameter for $\mathcal{D}$ in (7) is modified as: $\tilde{\mathcal{D}} := \mathcal{R}\circ\mathcal{D}\circ\mathbf{FC}$, where **FC** and $\mathcal{R}$ are the FC extension and restriction operators, respectively.

When the grids are non-uniform, the spectral method no longer applies and we seek to generate derivatives following the spirit of high-order meshfree methods (Bessa et al., 2014; Trask et al., 2019; Fan et al., 2023). In particular, considering the collection of measurement points $\chi = \{\mathbf{x}_i\}_{i=1}^M$, we seek to generate consistent quadrature rules of the form $\frac{\partial\phi}{\partial x_k}(\mathbf{x}_i) \approx \sum_{\mathbf{x}_j\in\chi\cap B_\delta(\mathbf{x}_i)}(\phi(\mathbf{x}_j)-\phi(\mathbf{x}_i))\omega_{i,j}^{(k)}$ for each $\mathbf{x}_i\in\chi$ and $k=1,\cdots,p$. Here, $B_\delta(\mathbf{x}_i)$ denotes the neighborhood of radius δ of point $\mathbf{x}_i$, and $\omega_{i,j}^{(k)}$ is a collection of to-be-determined quadrature weights pre-calculated from the following optimization problem:

$$\min_{\omega_{i,j}^{(k)}}\sum_{\mathbf{x}_j\in\chi\cap B_\delta(\mathbf{x}_i)}|\omega_{i,j}^{(k)}|,\ \text{s.t.}\ \frac{\partial\phi}{\partial x_k}(\mathbf{x}_i) = \sum_{\mathbf{x}_j\in\chi\cap B_\delta(\mathbf{x}_i)}(\phi(\mathbf{x}_j)-\phi(\mathbf{x}_i))\omega_{i,j}^{(k)},\quad\forall\phi\in\mathbb{P}_m(\mathbb{R}^p)\,, \quad (8)$$

where $\mathbb{P}_m(\mathbb{R}^p)$ is the space of $m-$th order polynomials ($m\geq 1$). Assume that χ is quasi-uniform, the size of $\chi\cap B_\delta(\mathbf{x})$ is bounded, the domain Ω satisfies a cone condition, and δ is sufficiently large, the above optimization problem has a solution (Wendland, 2004), and we further have:

Theorem 3.2. Consider a fixed ratio of $\delta/\Delta x$ where $\Delta x := \sup_{\mathbf{x}_i\in\chi}\min_{\mathbf{x}_j\in\chi\setminus\mathbf{x}_i}||\mathbf{x}_i-\mathbf{x}_j||_{L^2}$, the layer parameters of $\mathcal{D}$ can be formulated as: $\mathcal{D}[\mu] = [(\mathcal{D}[\mu])_1,\cdots,(\mathcal{D}[\mu])_p]^T$, where

$$(\mathcal{D}[\mu])_j(\mathbf{x}_i) = \sum_k\sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)}(\mu_{jk}(\mathbf{x}_l)-\mu_{jk}(\mathbf{x}_i))\omega_{i,l}^{(k)}. \quad (9)$$

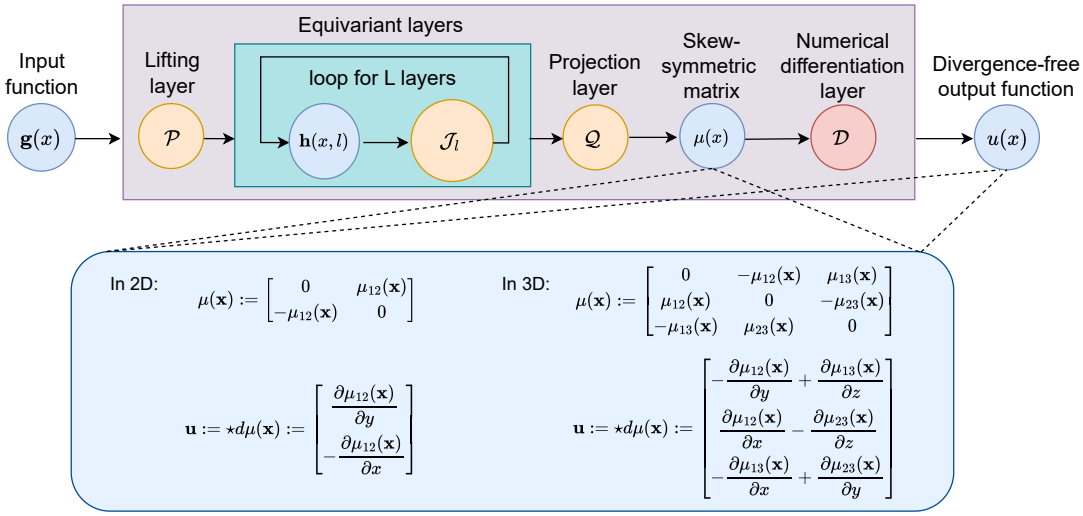


Figure 2. Proposed clawNO architecture. We start from the input function $\mathbf{g}(x)$. After lifting, the high-dimensional latent representation goes through a series of iterative equivariant layers, then gets projected to a function space in the form of an antisymmetric matrix. Lastly, we employ numerical differentiation (layer $\mathcal{D}$ with pre-calculated weights) to obtain the target divergence-free output.

The following error estimate holds true:

$$||\star d\mu - \mathcal{D}[\mu]||_{L^\infty} = O(\Delta x^{m+1}) \,,$$

if μ has $m+1$ continuous derivatives.

Proof. The proof follows a similar argument as in Levin (1998, Theorem 5), see Appendix E.2. □

The proposed architecture can be readily combined with the FNOs in (3) and GNOs in (4). Generally, when measurements are provided on regular domain and uniform grids, we consider FNOs together with the differentiation layer $\mathcal{D}$ approximated in (7) (or its FC-based extension). When the measurements are on non-uniform grids, GNOs can be employed with the differentiation layer $\mathcal{D}$ approximated via meshfree methods (see (9)). Besides vanilla FNOs and GNOs, to further encode laws of physics we also consider their group-equivariant versions as the base models, namely the invariant neural operator (INO) (Liu et al., 2023) for non-uniform grids and the group-equivariant FNO (G-FNO) (Helwig et al., 2023) for uniform grids. As such, the learnt laws of physics do not depend on the coordinate system used to describe them, together with the encoded conservation laws guaranteed.

Remark: Although clawNOs are designed to make divergence-free predictions, they are also applicable to problems involving a mixture of divergence-free and non-divergence-free fields. Assume the number of divergence-free and non-divergence-free variable dimensions are n_1 and n_2, respectively. In this situation, one can set the output dimension of the projection layer to be $n = n_1 + n_2$, and only pass the first n_1 dimensions of the output to the numerical differentiation layer for prediction of divergence-free output, and the remaining n_2 dimensions can be directly trained to predict the non-divergence-free variables in the spirit of a vanilla NO.

4. Experiments

We showcase the prediction accuracy and expressivity of clawNOs across a wide range of scientific problems, including elasticity, shallow water equations and incompressible Navier-Stokes equations. We compare the performance of clawNOs against a number of relevant machine learning techniques. In particular, we select INO (Liu et al., 2023), GNO (Li et al., 2020a), MP-PDE (Brandstetter et al., 2022), and MeshGraphNets (Pfaff et al., 2020) as baselines in graph-based settings, whereas we choose FNO (Li et al., 2020c), G-FNO (Helwig et al., 2023), UNet (Ronneberger et al., 2015), LSM (Wu et al., 2023b), KNO (Xiong et al., 2023a;b), UNO (Rahman et al., 2022), and BOON (Saad et al., 2022) as baselines in datasets with uniform discretizations. We also compare with an additional non-NO baseline, Lagrangian neural networks (LNN) (Cranmer et al., 2020; Müller, 2023), in the first experiment. The relative L2 error is reported as comparison metrics. We perform three replicates of all the experiments with randomly selected seeds, and the mean and standard deviation of the errors are reported. To guarantee a fair comparison, we make sure that the total numbers of trainable parameters in all models are on the same level, and report these numbers for each case. Additional details on the data generation and training strategies are provided in Appendix C.

4.1. Incompressible Navier-Stokes Equation

We start from the incompressible Navier–Stokes equations in vorticity form, which is widely applied to the simulation of sub-sonic flows, especially in hydromechanical and turbulent dynamics modeling. The (known) mass conservation law and (hidden) governing equation are given by:

$$\nabla \cdot \mathbf{u} = 0 \,, \quad \partial_t w + \mathbf{u} \cdot \nabla w = \nu \Delta w + \mathbf{f} \,, \tag{10}$$

where w and ν denote the vorticity and fluid viscosity, respectively, $\mathbf{f} = 0.1(sin(2\pi(x+y)) + cos(2\pi(x+y)))$ represents the external forcing term, and $\mathbf{u}$ is the velocity field that we aim to learn. The system is modeled on a square domain of $[0,1]^2$ and initialized with a random vorticity field w_0 sampled from a random Gaussian distribution. We consider two case studies. The first study predicts a rollout of $T = 20$ subsequent timesteps conditioned on the first $T_{in} = 10$ timesteps, which is consistent with the setting in G-FNOs (Helwig et al., 2023). In the second study, we predict a rollout of $T = 25$ timesteps conditioned on the first $T_{in} = 5$ timesteps. Compared with the first setting, this study aims to provide a longer-term extrapolation with less information from inputs. It is designed to further challenge the proposed model and baselines. Both studies are carried out in three datasets representing small, medium, and large data regimes.

Ablation study We first perform an ablation study in case 1 (i.e., $T = 20$ and $T_{in} = 10$) by adding the proposed divergence-free architecture to FNO and comparing its performance with the vanilla FNO and clawGFNO. The corresponding results are listed in Table 1, where we observe a consistent performance improvement with the proposed divergence-free architecture in all data regimes. In particular, by comparing clawGFNO to GFNO, we observe a boost in accuracy by 59.1%, 37.2% and 61.2% in small, medium and large data regimes, respectively. Our findings are consistent when we compare clawFNO to the vanilla FNO, where an enhancement by 43.8%, 9.0% and 0.7% is obtained, respectively. Therefore, in small training data regimes, imposing the conservation law has enhanced the data efficiency. This argument is further verified when comparing the distribution of test errors for clawGFNO and GFNO as demonstrated in Figure 3, from which one can see that clawGFNO has smaller errors on out-of-distribution test samples. Note that it is expected that the performance improvement of clawNOs becomes less pronounced as we march from the small data regime to the large data regime, as the baseline NOs are anticipated to learn the conservation laws from data as more data becomes available. The performance of clawNOs and the baseline NOs will eventually converge in the limit of infinite data. However, this is typically prohibitive as data generation is expensive in general, especially in the scientific computing community where one simulation can take hours to days to complete. On the other hand, clawGFNO further outperforms clawFNO in all data regimes, benefiting from the equivariant architecture of the model.

Comparison against additional baselines We choose to work with clawGFNO in the 2nd case study owing to its outstanding performance compared to clawFNO. Herein, we compare the performance of clawGFNO against additional baselines in Table 1. Among all baselines in case 1, KNO has achieved the best performance in small data regime, while clawFNO and clawGFNO still outperform it by 37.4% and 40.3%, respectively. By investigating in the direction from small data regime to large data regime, clawGFNO achieves the best performance in all cases and beats the best baselines by 40.3%, 16.2% and 17.6% in case 1 and 39.9%, 26.8% and 9.1% in case 2. ClawGFNO also manifests its robustness as the learning task becomes more challenging across the two case studies, as compared to other baselines whose performance deteriorates notably. Additionally, in order to maintain a similar total number of trainable parameters, we reduce the latent width of GFNO from 20 to 11, leading to possible expressivity degradation in GFNO. In fact, when increasing the latent dimension of GFNO to 20 to match the dimension of FNO, on the large data regime of case 1 the test error is decreased to $1.19\% \pm 0.16\%$, achieving a comparable performance to FNO. Note also that LNN in case 1 seemingly cannot learn the correct solution as the test error remains above 20% across all three data regimes. This is due to the fact that LNN predicts the acceleration from current position and velocity, and then updates the velocity. Since this dataset is relatively sparse in time, it leads to large errors from temporal integration in LNNs.

4.2. Radial Dam Break

In this example, we explore the shallow water equations that are derived from the compressible Navier–Stokes equations, which find broad applications in tsunami and general flooding simulation. Specifically, we simulate a circular dam removal process where the water is initially confined in the circular dam and suddenly released due to the removal of the dam. The (known) mass conservation law and (hidden) governing PDE describing the system are as follows:

$$\begin{aligned} \partial_t h + \nabla \cdot (h\mathbf{u}) &= 0 \,, \\ \partial_t (h\mathbf{u}) + \nabla \cdot \left(h\mathbf{u}\mathbf{u}^T\right) + \frac{1}{2} g \nabla h^2 + g h \nabla b &= 0 \,, \end{aligned} \tag{11}$$

where g is the gravitational acceleration, h describes the water height, $\mathbf{u}$ denotes the velocity field, $(h, h\mathbf{u})$ is the 3D divergence-free field we aim to learn, and b denotes a spatially varying bathymetry. The simulation is performed on a square domain of $[-2.5, 2.5]^2$, and the initial height h_0 is specified as $h_0 = 2.0$ when $r < \sqrt{x^2 + y^2}$ and $h_0 = 1.0$ otherwise, with r being the radial distance to the dam center, which is uniformly sampled from (0.3, 0.7).

Table 1. Test errors and the number of trainable parameters for the incompressible Navier–Stokes problem, where bold numbers highlight the best method. Rollouts are of length T conditioned on the first T_{in} time steps.

Case	2D models	#param	# of training samples		
			10	100	1000
T_{in} = 10, T = 20	clawGFNO	853,231	**12.10%±1.08%**	**4.76%±0.24%**	**1.21%±0.09%**
	clawFNO	928,861	12.68%±0.85%	5.17%±0.27%	1.46%±0.11%
	GFNO-p4	853,272	29.56%±3.32%	7.59%±0.34%	3.11%±0.93%
	FNO	928,942	22.57%±1.87%	5.68%±0.41%	1.47%±0.18%
	UNet	920,845	56.81%±9.91%	16.62%±2.47%	6.07%±0.36%
	LSM	1,211,234	31.79%±2.11%	13.30%±0.15%	5.52%±0.08%
	UNO	1,074,522	21.48%±1.49%	11.41%±0.38%	5.29%±0.09%
	KNO	890,055	20.26%±1.50%	11.40%±3.12%	9.09%±1.70%
	LNN	1,056,768	25.73%±0.96%	24.27%±3.16%	21.36%±2.46%
	BOON	928,942	23.02%±2.00%	6.21%±0.48%	2.25%±0.13%
T_{in} = 5, T = 25	clawGFNO	853,131	**16.02%±0.57%**	**4.84%±0.49%**	**1.57%±0.18%**
	GFNO-p4	853,172	36.35%±4.47%	10.67%±3.00%	2.68%±0.60%
	FNO	928,742	26.65%±5.31%	6.61%±0.83%	1.73%±0.23%
	UNet	919,855	76.12%±14.04%	34.45%±10.59%	7.39%±0.83%

Table 2. Test errors and the number of trainable parameters for the radial dam break problem, where bold numbers highlight the best method. Rollouts are of length 24 conditioned on the first time step, following the setting of Helwig et al. (2023).

3D models	#param	# of training samples		
		2	10	100
clawFNO	4,922,303	**21.24%±1.17%**	**2.69%±0.04%**	1.84%±0.01%
clawGFNO	4,799,732	23.29%±3.10%	2.87%±0.03%	**1.83%±0.01%**
FNO	4,922,303	29.88%±0.17%	4.04%±0.10%	1.88%±0.05%
GFNO-p4	4,799,732	27.31%±2.80%	3.80%±0.28%	1.90%±0.07%
UNet	5,080,113	338.08%±553.62%	5.62%±0.19%	2.98%±0.20%

We report in Table 2 our experimental observations using clawGFNO and clawFNO, along with the comparison against FNO, GFNO, and UNet. ClawNOs continue to achieve the lowest test errors across all three selected datasets, where clawFNO shows the best performance in the small and medium data regimes and clawGFNO becomes slightly more superior in the large data regime. As the radial dam break problem is fairly simple and the solution exhibits strong symmetries, the performances of non-claw baselines quickly converge to clawNOs with only 100 training samples. Nevertheless, clawNOs consistently outstand the best non-claw baseline in performance by 22.2% in small data regime and 29.2% in medium data regime.

4.3. Atmospheric Modeling

In this example, we use the general circulation model SpeedyWeather.jl to simulate gravity waves in the Earth's atmosphere. The shallow water equations for relative vorticity $\zeta = \nabla \times \mathbf{u}$, divergence $\mathcal{D} = \nabla \cdot \mathbf{u}$, and the displacement η from the atmosphere's rest height $H = 8500$ m are described as the given mass conservation law $\frac{\partial \eta}{\partial t} + \nabla \cdot (\mathbf{u}h) = 0$ together with other (hidden) governing laws:

$$\begin{aligned} \frac{\partial \zeta}{\partial t} + \nabla \cdot (\mathbf{u}(\zeta + f)) &= 0\,, \\ \frac{\partial \mathcal{D}}{\partial t} - \nabla \times (\mathbf{u}(\zeta + f)) &= -\nabla^2(\tfrac{1}{2}|\mathbf{u}|^2 + g\eta)\,. \end{aligned} \tag{12}$$

The equations are solved in spherical coordinates with latitude $\theta \in [-\pi, \pi]$, longitude $\lambda \in [0, 2\pi]$ on a sphere of radius $R = 6371$ km. The layer thickness is $h = \eta + H - H_o$ with the Earth's orography $H_o = H_o(\lambda, \phi)$ and gravity $g = 9.81$ ms^{-2}. The Coriolis force uses the parameter $f = 2\Omega \sin(\theta)$ with $\Omega = 7.29 \cdot 10^{-5}$ s^{-1} the angular frequency of Earth's rotation. The simulations start from rest, $\mathbf{u} = 0$, but with random waves in η with wave lengths of about 2000 to 4000 km and maximum amplitudes of 2000 m. These random initial waves propagate at a phase speed of $c_{ph} = \sqrt{gh}$ (about 300 ms^{-1}) around the globe, interacting non-linearly with each other and the underlying Earth's orography. For details on the model setup and the simulation see Appendix C.3.

We list in Table 3 our experimental results along with comparisons against selected baselines. For this problem, since the spherical coordinate system in latitudes and longitudes stay the same and thus the equivariance property plays little role, we directly employ clawFNO as our representative clawNO model, as the latent dimension of clawFNO is much higher than that of clawGFNO for similar total numbers of model parameters. In other words, clawFNO possesses more expressivity power. This is rather important in atmospheric modeling at the entire globe scale, as the wave propagation patterns are more localized compared

Table 3. Test errors for the weather modeling problem and the number of trainable parameters (in millions), where bold numbers highlight the best method. Rollouts are of length 10 conditioned on the first time step.

Model	clawFNO	FNO	GFNO	UNet
#param (M)	49.57	49.57	53.70	51.58
Test error	**6.64%±0.23%**	12.54%±0.26%	14.34%±0.55%	20.89%±0.87%

Table 4. Test errors and the number of trainable parameters for the material deformation problem. Bold numbers highlight the best method.

2D models	#param	# of training samples		
		2	20	100
clawINO	4,747,521	**14.12%±0.23%**	**5.41%±0.67%**	**1.00%±0.02%**
INO	4,747,521	16.75%±1.24%	7.69%±0.51%	2.40%±0.53%
GNO	18,930,498	19.73%±3.60%	12.59%±0.24%	8.51%±5.29%
MP-PDE	18,368,610	23.07%±20.39%	12.38%±0.41%	1.30%±0.55%
MeshGraphNets	18,694,914	38.57%±17.84%	7.35%±0.54%	1.79%±0.81%

to other examples. In this context, clawFNO outperforms the best non-claw baseline by 47% in accuracy. Additionally, it is worth mentioning that the majority of the errors accumulate in the vicinity of mountains, as evidenced in the rollout demonstration in Figures 13-15 (compare to Fig. 6a). This is expected as the interaction with the orography causes wave reflection and dispersion through changes in the wave speed, which generally limits predictability more than in other regions. Furthermore, the training strategy (Appendix C.5) disregards wave numbers higher than 22 which, however, are more pronounced around rough orography. Our simulations resolve orography up to wave number 63 (Appendix C.3).

4.4. Constitutive Modeling of Material Deformation

In this example we test the efficacy of the proposed clawNO in graph-based settings. Specifically, we consider the incompressible material deformation of the Mooney–Rivlin type, where the (hidden) constitutive law reads:

$$-\nabla \cdot \sigma = \mathbf{f} \text{ , where } \sigma = -p\mathbf{I} + 2C_1\mathbf{B} - 2C_2\mathbf{B}^{-1} \text{ , } \quad (13)$$
$$\mathbf{u}|_{\partial\Omega} = \mathbf{u}_D \text{ ,}$$

with Dirichlet boundary conditions, and the (known) mass conservation law $\nabla \cdot \mathbf{u} = 0$ imposed. Here σ is the stress tensor, $\mathbf{u}$ denotes the displacement, $\mathbf{f}$ denotes the body load, λ and μ are the Lame constants. $\mathbf{B}$ is the left Cauchy–Green deformation tensor, $\mathbf{B} := \mathbf{F}\mathbf{F}^T \approx \mathbf{I} + 2\text{sym}(\nabla\mathbf{u})$ for infinitesimal strains. Under this setting, the Dirichlet boundary condition and the body load are treated as inputs. To investigate the efficacy on irregular grids, we consider a circular domain of radius 0.4. We set the material constants $C_1 = C_2 = 0.075$, and the body load $\mathbf{f}$ is randomly generated following:

$$\mathbf{f}(x, y) = \sum_{k_1,k_2=1}^{2} A_{k_1,k_2} \sin(k_1\pi x)\sin(k_2\pi y),$$
$$A_{k_1,k_2} = \exp(-0.1k_1k_2)\xi \text{ , } \quad \xi \sim \mathcal{U}[-2.5, 2.5] \text{ .}$$

We report in Table 4 the numerical results of clawINO, along with the comparison against GNO, INO, MP-PDE and MeshGraphNets in various data regimes. Consistent with the findings on regular grids, clawINO outperforms other graph-based models in all data regimes, beating the best non-claw baseline by 15.7%, 26.4%, and 23.1% in the considered three data regimes, respectively. Note that the total number of parameters in GNO is roughly four times the number of parameters in INO-based models, as the parameters in INO-based models are layer-independent, which also results in an advantage in memory saving.

5. Conclusion

We introduce a series of clawNOs that explicitly bake fundamental conservation laws into the neural operator architecture. In particular, we build divergence-free output functions in light of the concept of differential forms and design an additional layer to recover the divergence-free target functions based on numerical differentiation. We perform extensive experiments covering a wide variety of challenging scientific machine learning problems, and show that it is essential to encode conservation laws for correct physical realizability, especially in small-data regimes.

For future work, we plan to explore more fundamental laws in other application domains. In some applications such as the shock capturing problem (Hansen et al., 2023), the solution regularity deteriorates and error estimates in this work do not apply. It would be interesting to modify the differentiation layer in clawNOs based on discontinuity capturing methods such as the Essentially Non-Oscillatory method (Shu, 1999) and the discontinuous Galerkin method (Huerta et al., 2012). We also point out that the idea of incorporating numerical PDE techniques into NOs can also be extended. For instance, the quadrature weights can provide a more accurate numerical integration in evaluating the cost functional, as compared to the current mean squared error widely employed in NOs.

Acknowledgements

Y. Fan, L. Zhang, and Y. Yu would like to acknowledge support by the National Science Foundation under award DMS-1753031 and the AFOSR grant FA9550-22-1-0197. Portions of this research were conducted on Lehigh University's Research Computing infrastructure partially supported by NSF Award 2019035. The authors would also like to thank the reviewers for the careful reading and valuable suggestions that helped improve the quality of the paper.

Impact Statement

Broader Impact: Conservation laws are fundamentally properties of nature, describing that a particular measurable property of a physical system does not change as the system evolves. In physics-based machine learning applications such as material modeling and climate science, the underlying governing equations are often unknown, while the system should satisfy some fundamental physical laws such as the conservation of mass, momentum and energy. Our work focuses on developing a novel neural network architecture to impose these fundamental physical laws, and therefore contributes mostly to the field of scientific machine learning, especially in the enduring problem of data-driven model discovery from physics-based measurements.

Ethics Statement: The data utilized in our study is solely sourced from publicly available datasets or softwares. No user information was involved. We are committed to upholding the principles of responsible and transparent data usage throughout our work.

References

Amlani, F. and Bruno, O. P. An fc-based spectral solver for elastodynamic problems in general three-dimensional domains. *Journal of Computational Physics*, 307:333–354, 2016.

Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., and Stuart, A. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Barbarosie, C. Representation of divergence-free vector fields. *Quarterly of applied mathematics*, 69(2):309–316, 2011.

Besnard, G., Hild, F., and Roux, S. "finite-element" displacement fields analysis from digital images: application to portevin–le châtelier bands. *Experimental mechanics*, 46(6):789–803, 2006.

Bessa, M., Foster, J., Belytschko, T., and Liu, W. K. A meshfree unification: reproducing kernel peridynamics. *Computational Mechanics*, 53(6):1251–1264, 2014.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, pp. 1–12, 2022.

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940, 2021.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., and Zdeborová, L. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

Çolak, A. B. An experimental study on the comparative analysis of the effect of the number of data on the error rates of artificial neural networks. *International Journal of Energy Research*, 45(1):478–500, 2021.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Do Carmo, M. P. *Differential forms and applications*. Springer Science & Business Media, 1998.

Fan, Y., You, H., and Yu, Y. Obmeshfree: An optimization-based meshfree solver for nonlocal diffusion and peridynamics models. *Journal of Peridynamics and Nonlocal Modeling*, pp. 1–29, 2023.

Ghaboussi, J., Garrett Jr, J., and Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

Ghaboussi, J., Pecknold, D. A., Zhang, M., and Haj-Ali, R. M. Autoprogressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering*, 42(1):105–126, 1998.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. Physics-informed neural operators. *2022 arXiv preprint arXiv:2207.05748*, 2022a.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. Physics-informed neural operators. *arXiv preprint arXiv:2207.05748*, 2022b.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

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

Hansen, D., Maddix, D. C., Alizadeh, S., Gupta, G., and Mahoney, M. W. Learning physical models that can respect conservation laws. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, Q., Laurence, D. W., Lee, C.-H., and Chen, J.-S. Manifold learning based data-driven modeling for soft biological tissues. *Journal of Biomechanics*, 117:110124, 2021.

Helwig, J., Zhang, X., Fu, C., Kurtin, J., Wojtowytsch, S., and Ji, S. Group equivariant fourier neural operators for partial differential equations. *arXiv preprint arXiv:2306.05697*, 2023.

Hoskins, B. J. and Simmons, A. J. A multi-layer spectral model and the semi-implicit method. *Quarterly Journal of the Royal Meteorological Society*, 101(429):637–655, 1975. ISSN 1477-870X. doi: 10.1002/qj.49710142918.

Hotta, D. and Ujiie, M. A nestable, multigrid-friendly grid on a sphere for global spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal Meteorological Society*, 144(714):1382–1397, 2018. ISSN 1477-870X. doi: 10.1002/qj.3282.

Huerta, A., Casoni, E., and Peraire, J. A simple shock-capturing technique for high-order discontinuous galerkin methods. *International journal for numerical methods in fluids*, 69(10):1614–1632, 2012.

Jin, H., Zhang, E., and Espinosa, H. D. Recent advances and applications of machine learning in experimental solid mechanics: A review. *arXiv preprint arXiv:2303.07647*, 2023.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Keller, C. A. and Evans, M. J. Application of random forest regression to the calculation of gas-phase chemistry within the geos-chem chemistry model v10. *Geoscientific Model Development*, 12(3):1209–1225, 2019.

Kelliher, J. P. Stream functions for divergence-free vector fields. *Quarterly of Applied Mathematics*, 79(1):163–174, 2021.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Levin, D. The approximation power of moving least-squares. *Mathematics of computation*, 67(224):1517–1531, 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*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:NeurIPS 2020, 2020b.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020c.

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. *2021 arXiv preprint arXiv:2111.03794*, 2021.

Liu, N., Yu, Y., You, H., and Tatikola, N. Ino: Invariant neural operators for learning complex physical systems with momentum conservation. In *International Conference on Artificial Intelligence and Statistics*, pp. 6822–6838. PMLR, 2023.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

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, 2021.

Maust, H., Li, Z., Wang, Y., Leibovici, D., Bruno, O., Hou, T., and Anandkumar, A. Fourier continuation for exact derivative computation in physics-informed neural operators. *arXiv preprint arXiv:2211.15960*, 2022.

Müller, E. H. Exact conservation laws for neural network integrators of dynamical systems. *Journal of Computational Physics*, 488:112234, 2023.

Négiar, G., Mahoney, M. W., and Krishnapriyan, A. S. Learning differentiable solvers for systems with hard constraints. *arXiv preprint arXiv:2207.08675*, 2022.

Noether, E. Invariant variation problems. *Transport theory and statistical physics*, 1(3):186–207, 1971.

Ong, Y. Z., Shen, Z., and Yang, H. IAE-NET: Integral autoencoders for discretization-invariant learning. 03 2022. doi: 10.13140/RG.2.2.25120.87047/2.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Pfau, D., Spencer, J. S., Matthews, A. G., and Foulkes, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 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.

Ranade, R., Gitushi, K., and Echekki, T. Generalized joint probability density function formulation inturbulent combustion using deeponet. *arXiv preprint arXiv:2104.01996*, 2021.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Richter-Powell, J., Lipman, Y., and Chen, R. T. Neural conservation laws: A divergence-free perspective. *Advances in Neural Information Processing Systems*, 35: 38075–38088, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Saad, N., Gupta, G., Alizadeh, S., and Maddix, D. C. Guiding continuous operator learning through physics-based boundary constraints. *arXiv preprint arXiv:2212.07477*, 2022.

Sarlet, W. and Cantrijn, F. Generalizations of noether's theorem in classical mechanics. *Siam Review*, 23(4): 467–494, 1981.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, volume 139, pp. 9323–9332. PMLR, 2021.

Schmidt, M. and Lipson, H. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Shu, C.-W. High order eno and weno schemes for computational fluid dynamics. In *High-order methods for computational physics*, pp. 439–582. Springer, 1999.

Sturm, P. O. and Wexler, A. S. Conservation laws in a neural network architecture: enforcing the atom balance of a julia-based photochemical model (v0. 2.0). *Geoscientific Model Development*, 15(8):3417–3431, 2022.

Tac, V., Costabal, F. S., and Tepole, A. B. Data-driven tissue mechanics with polyconvex neural ordinary differential equations. *Computer Methods in Applied Mechanics and Engineering*, 398:115248, 2022.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611, 2022.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2022.

Trask, N., You, H., Yu, Y., and Parks, M. L. An asymptotically compatible meshfree quadrature rule for nonlocal problems with applications to peridynamics. *Computer Methods in Applied Mechanics and Engineering*, 343: 151–165, 2019.

Trefethen, L. N. *Spectral methods in MATLAB*. SIAM, 2000.

Vinuesa, R., Brunton, S. L., and McKeon, B. J. The transformative potential of machine learning for experiments in fluid mechanics. *Nature Reviews Physics*, pp. 1–10, 2023.

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, 2021.

Weintraub, S. H. *Differential forms: theory and practice*. Elsevier, 2014.

Wendland, H. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

Wieder, O., Kohlbacher, S., Kuenemann, M., Garon, A., Ducrot, P., Seidel, T., and Langer, T. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

Williams, P. D. The RAW Filter: An Improvement to the Robert–Asselin Filter in Semi-Implicit Integrations. *Monthly Weather Review*, 139(6):1996–2007, June 2011. ISSN 0027-0644, 1520-0493. doi: 10.1175/2010MWR3601.1.

Wu, F., Radev, D., and Li, S. Z. Molformer: Motif-based transformer on 3d heterogeneous molecular graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5312–5320, 2023a.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023b.

Xiong, W., Huang, X., Zhang, Z., Deng, R., Sun, P., and Tian, Y. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *arXiv preprint arXiv:2301.10022*, 2023a.

Xiong, W., Ma, M., Sun, P., and Tian, Y. Koopmanlab: A pytorch module of koopman neural operator family for solving partial differential equations. *arXiv preprint arXiv:2301.01104*, 2023b.

You, H., Yu, Y., D'Elia, M., Gao, T., and Silling, S. Nonlocal kernel network (NKN): A stable and resolution-independent deep neural network. *Journal of Computational Physics*, pp. arXiv preprint arXiv:2201.02217, 2022a.

You, H., Zhang, Q., Ross, C. J., Lee, C.-H., and Yu, Y. Learning deep implicit fourier neural operators (IFNOs) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022b. doi: https://doi.org/10.1016/j.cma.2022.115296.

Zhang, L., Han, J., Wang, H., Car, R., and Weinan, E. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, 120(14):143001, 2018.

A. Related concepts of differential forms

Differential forms aim to provide a unified approach to define integrands. In this section, we briefly review a few related mathematical concepts in the paper, which can be found in Richter-Powell et al. (2022); Do Carmo (1998); Weintraub (2014).

Definition A.1. Let k be a non-negative integer and $\mu(\mathbf{x}) = f(x_1, \cdots, x_p)$ be a smooth function on $\mathbb{R}^p$, a monomial $k-$form on $\mathbb{R}^p$ is an expression $\mu\, dx_{i_1} \wedge \cdots \wedge dx_{i_k} = \mu dx_I$, where $I := \{i_1, \cdots, i_k$. A $k-$form is a sum of monomial $k-$forms, which will be denoted as

$$\mu = \sum_{1 \leq i_1, i_2, \cdots, i_k \leq p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k} := \sum_I \mu_I dx_I.$$

Definition A.2. The wedge product, $\wedge$, is an associative operation on differential forms. When ω is a $k-$form and η is a $l-$form, $\omega \wedge \eta$ is a $(k+l)-$form. Moreover, $\wedge$ satisfies the following properties:

- (Distribution Law) $(\omega_1 + \omega_2) \wedge \eta = \omega_1 \wedge \eta + \omega_1 \wedge \eta$, $\omega \wedge (\eta_1 + \eta_2) = \omega \wedge \eta_1 + \omega \wedge \eta_2$.
- (Associative Law) $(f\omega) \wedge \eta = f(\omega \wedge \eta)$.
- (Skew Symmetry) $\eta \wedge \omega = -\omega \wedge \eta$.

Here ω, ω_1, ω_2 are $k-$forms, η, η_1, η_2 are $l-$forms, and f is a function.

Definition A.3. The exterior derivative (or differential) operator d maps a $k-$form to a $(k+1)-$form. In particular, given a $k-$form $\mu = \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k}$, one has

$$\begin{aligned} d\mu &= \sum_{1 \leq i_1, i_2, \cdots, i_k \leq p} d\mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k} \\ &= \sum_{1 \leq i_1, i_2, \cdots, i_k \leq p} \sum_{1 \leq l \leq p} \frac{\partial \mu_{(i_1, \cdots, i_k)}}{\partial dx_l} dx_l \wedge dx_{i_1} \wedge \cdots \wedge dx_{i_k}. \end{aligned}$$

Moreover, the following properties hold for the exterior derivative operator:

- For each function f, $ddf = 0$, and $df \wedge df = 0$.
- For each $k-$form ω, $dd\omega = 0$.
- For each function f and $k-$form ω, $d(f\omega) = df \wedge \omega + fd\omega$.

Definition A.4. The Hodge $\star$-operator on $\mathbb{R}^p$ is a function that takes a $k-$form to a $(p-k)-$form defined as follows:

- Let $I = \{i_1, \cdots, i_k\}$ be an ordered subset of $\{1, \cdots, p\}$ and $I^c = \{j_1, \cdots, j_{p-k}\}$ be its complement, ordered so that $dx_{i_1} \wedge \cdots dx_{i_k} \wedge dx_{j_1} \wedge \cdots dx_{j_{p-k}} = dx_1 \wedge \cdots \wedge dx_p$, then

$$\star dx_I = \star(dx_{i_1} \wedge \cdots dx_{i_k}) := dx_{j_1} \wedge \cdots dx_{j_{p-k}} = dx_{I^c}.$$

- Given any $k-$form $\mu = \sum_{1 \leq i_1, i_2, \cdots, i_k \leq p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k}$, then

$$\star\mu := \sum_{1 \leq i_1, i_2, \cdots, i_k \leq p} \mu_{(i_1, \cdots, i_k)} \star (dx_{i_1} \wedge \cdots \wedge dx_{i_k}).$$

Then, the following property holds for any $k-$form μ:

$$\star \star \mu = (-1)^{k(p-k)} \mu.$$

Given a vector-valued function on $\mathbb{R}^p$ $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), \cdots, u_p(\mathbf{x})]$, its divergence $\text{div}(\mathbf{u}) = \sum_{i=1}^p \frac{\partial u_i}{\partial x_i}$ can be defined via exterior derivative:

$$\begin{aligned} d \star \sum_{i=1}^p u_i dx_i =& d \sum_{i=1}^p u_i \star dx_i = d \sum_{i=1}^p (-1)^{i-1} u_i dx_{\{1,\cdots,n\}\setminus i} \\ =& \sum_{i=1}^p (-1)^{i-1} du_i \wedge dx_{\{1,\cdots,n\}\setminus i} = \sum_{i=1}^p \frac{\partial u_i}{\partial x_i} dx_{\{1,\cdots,n\}}. \end{aligned}$$

Therefore, one can denote the divergence of $\mathbf{u}$ as $d \star \mathbf{u}$. On the other hand, combining the fundamental properties of $\star$ and d yields $\text{div}(\star d\mu) = d \star \star d\mu = (-1)^{k(p-k)} dd\mu = 0$ for any $k-$form μ. Therefore, when $\mathbf{u} = \star d\mu$, where μ is a $(p-2)-$form, it is guaranteed to be divergence free. Since any $(p-2)-$form μ can be represented as

$$\mu = \sum_{1 \le i_1, \cdots, i_{p-2} \le p} \mu_{(i_1,\cdots,i_{p-2})} dx_{i_1} \wedge \cdots \wedge dx_{i_{p-2}} = \sum_{1 \le i,j \le p} \mu_{(i,j)} \star (dx_i \wedge dx_j),$$

where $\mu_{(i,j)} := (-1)^{\sigma(\{i,j\},\{i_1,\cdots,i_{p-2}\})} \mu_{(i_1,\cdots,i_{p-2})}$ when $\{i,j\} \cup \{i_1, \cdots, i_{p-2}\} = \{1, \cdots, p\}$. And note that the skew symmetry property of $\wedge$ yields $\mu_{(i,j)} := \mu_{ij} = -\mu_{ji}$. We further expand the representation of $\star d\mu$ as:

$$\begin{aligned} \star d\mu =& \star \sum_{1 \le i,j \le p} d\mu_{(i,j)} \star (dx_i \wedge dx_j) \\ =& \star \left(\sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star(dx_i \wedge dx_j) - \frac{\partial \mu_{ji}}{\partial x_j} dx_j \wedge \star(dx_i \wedge dx_j) \right) \\ =& \star \left(\sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star(dx_i \wedge dx_j) + \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star(dx_i \wedge dx_j) \right) \\ =& 2 \star \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star(dx_i \wedge dx_j) = 2 \star \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} \star dx_j \\ =& 2 \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} \star \star dx_j = (-1)^{p-1} 2 \sum_j \left(\sum_i \frac{\partial \mu_{ij}}{\partial x_i} \right) dx_j. \end{aligned}$$

That means, taking any $\mu := [\mu_{ij}(\mathbf{x})]$ satisfies $\mu_{ij} = -\mu_{ji}$ (a p by p skew-symmetric matrix-valued function), we have $\text{div}(\star d\mu) = 0$, where $\star d\mu = [\text{div}(\mu_1), \cdots, \text{div}(\mu_p)]^T$ and μ_i stands for the $i-$th row of μ.

B. Equivariance of the differentiation layer

The projection layer (μ to $\mathbf{u}$) is realized as a numerical divergence operator. Therefore, under the translation of coordinates, $\tilde{\mathbf{x}} = \mathbf{x} + g$, we have

$$\nabla_{\tilde{\mathbf{x}}} \cdot \mu_j = \sum_k \frac{\partial}{\partial \tilde{x}_k} \mu_{jk} = \sum_k \frac{\partial}{\partial x_k} \mu_{jk} = \nabla_{\mathbf{x}} \cdot \mu_j.$$

That means, the divergence operator is invariant under coordinate translation, and therefore $\mathcal{D}$ is also invariant. On the other hand, under the rotation of coordinates, $\tilde{\mathbf{x}} = R\mathbf{x}$, we have the chain rule: $\dfrac{\partial}{\partial x_k} = \sum_l R_{lk} \dfrac{\partial}{\partial \tilde{x}_l}$. Therefore,

$$\nabla_{\tilde{\mathbf{x}}} \cdot (R\mu_j) = \sum_k \frac{\partial}{\partial \tilde{x}_k} (R\mu_j)_k = \sum_k \frac{\partial}{\partial \tilde{x}_k} \sum_l R_{kl} \mu_{jl} = \sum_l \frac{\partial}{\partial x_l} \mu_{jl} = \nabla_{\mathbf{x}} \cdot \mu_j.$$

That means, the coefficients in $\mathcal{D}$ is also invariant under coordinate rotation.

When the previous layers are translational invariant and rotational equivariant, we have $\tilde{\mu}_j(\tilde{\mathbf{x}}) = \tilde{\mu}_j(R\mathbf{x} + g) = R\mu_j(\mathbf{x})$ for the $j-$th row of μ. Therefore, $\mathcal{D}[\tilde{\mu}](\tilde{\mathbf{x}}) = R\mathcal{D}[\mu](\mathbf{x})$, which results in the translational invariant and rotational equivariant output.

C. Data generation and training strategies

C.1. Example 1 – incompressible Navier-Stokes equation

For the two case study datasets in the incompressible Navier-Stokes example, we generate a total of 1,200 samples using the pseudo-spectral Crank-Nicolson solver available in Li et al. (2020c). Here, the models are trained to predict the two velocity components in the two directions. We then split the generated dataset into 1,000, 100 and 100 for training, validation and testing, respectively. A histogram demonstration of the dataset distributions in small, medium and large data regimes is illustrated in Figure 3, where the test set of the small dataset of ntrain=10 samples exhibits a wider data distribution compared to its training dataset, which is aimed to test the data efficiency and out-of-distribution performances of the learned models. The fluid viscosity employed in the physics solver is $\nu = 10^{-4}$, and the timestep size is $\Delta t = 10^{-4}$ s. The solutions are obtained on a 256×256 spatial grid and the total duration of the simulation is 24 s. The obtained solutions are then downsampled to a $64 \times 64 \times 30$ grid, with the 3rd dimension being the temporal dimension. Note that, in downsampling the spatial dimensions, we employ a 2×2 mean pooling. This strategy is suggested in Helwig et al. (2023) to mitigate the spurious numerical errors not existed in the original data.

To provide more details on the model performance, we plot the per-time-step prediction errors in terms of L2 relative error on the test dataset in Figure 4 using the best models trained with ntrain=1000 samples. Perhaps unsurprisingly, the prediction error increases as the prediction time step grows, due to the accumulation of error. All models have a similar growth rate, while clawNOs, together with FNO, significantly outperform other baselines in accuracy. We also list a number of performance metrics in Table 5, including the total number of model parameters, the per-epoch runtime, the inference time, as well as the peak GPU usage. To quantitatively evaluate the divergence of the predicted solutions, we compute the averaged L2-norm of the divergence on the test dataset for all the models trained with ntrain=1000 samples (cf. the last row of Table 5). Because the additional layer in clawNOs are with pre-calculated weights, it barely adds any extra burden into GPU memory compared with its NO base model. Similar observation also applies to the runtime in large data regime. The inference time has a minor increase, due to the fact that the inferred solution will go through the an additional layer which avoidably increases the computational cost. When comparing the divergence of predicted solutions, we can see that both clawNOs predictions have much smaller divergence compared with all baselines. However, we point out that the solution divergence from clawNOs is not exactly zero, due to the numerical errors as discussed in Theorem 3.1.

Remark: The complexity of our clawNOs is very simular to their NO counterparts. Taking clawFNOs for example, the trainable part of the clawNO model consists of two fixed-size MLPs for the lifting layer and the projection layer, and L numbers of Fourier layers between them. Denote d_u as the input function dimension, H as the latent dimension after lifting, M as the total number of grids, m as the number of Fourier modes on each dimension after truncation (which is often taken as half of the number of grids in each dimension, $M^{1/p}$, in practice), and p as the problem dimension. During the lifting layer a vector valued input function taking values in $\mathbb{R}^{d_u}$ is linearly and locally mapped to the first layer feature function $\mathbf{h}(\cdot, 0)$ taking values in $\mathbb{R}^H$, and hence the number of trainable parameters is $Hd_u + H$. Then, each iterative Fourier layers involves the integral with a trainable kernel weight matrix in the Fourier domain, which is of size $2H^2m^p = 2^{1-p}H^2M$, and a local linear transformation which involves $H^2 + H$ numbers of trainable parameters. Then, the last iterative layer feature function, $\mathbf{h}(\cdot, L)$, is projected to the skew symmetric matrix-valued function μ with a two-layer MLP. Since the skew symmetric matrix-valued function μ is of degree of freedom $p(p-1)/2$ at each point $\mathbf{x}$, the projection layer maps a size H input vector to a size $p(p-1)/2$ vector. Assume that the hidden layer of this MLP is of d_Q neurons, the total number of trainable parameters in projection layer will be $Hd_Q + d_Q + d_Q p(p-1)/2 + p(p-1)/2$. Finally, μ will go through the pre-calculated differentiation layer, with $p^2(p-1)M^2/2 = p^2(p-1)m^{2p}/2$ numbers of non-trainable parameters in the FNO case. From the above calculation, we can see that clawFNO involves $(d_Q + H(d_u + d_Q + L + 1) + LH^2) + \frac{(d_Q+1)}{2}p(p-1) + 2LH^2m^p$ numbers of trainable parameters, while the vanilla FNO involves $(d_Q + H(d_u + d_Q + L + 1) + LH^2) + (d_Q + 1)p + 2LH^2m^p$ numbers of trainable parameters. Therefore, the number of trainable parameters in clawFNO and FNO only differs in the second part of their projection layer, where clawFNO has $\frac{(d_Q+1)}{2}p(p-1)$ numbers of parameters and FNO has $(d_Q+1)p$. When $p > 3$, clawFNO will have a larger number of trainable parameters. However, we want to point out that since the number of parameter in the iterative layer ($2LH^2m^p$) grows exponentially with dimension p, it dominates the cost, and the differences between clawFNO and FNO are negligible. This is consistent with what we observed in Table 5: the number of trainable parameters and the GPU cost of clawNOs and their counterparts are almost the same. During the inference, the non-trainable parameters in clawNO will play a role and we therefore observe an increase in the inference time.

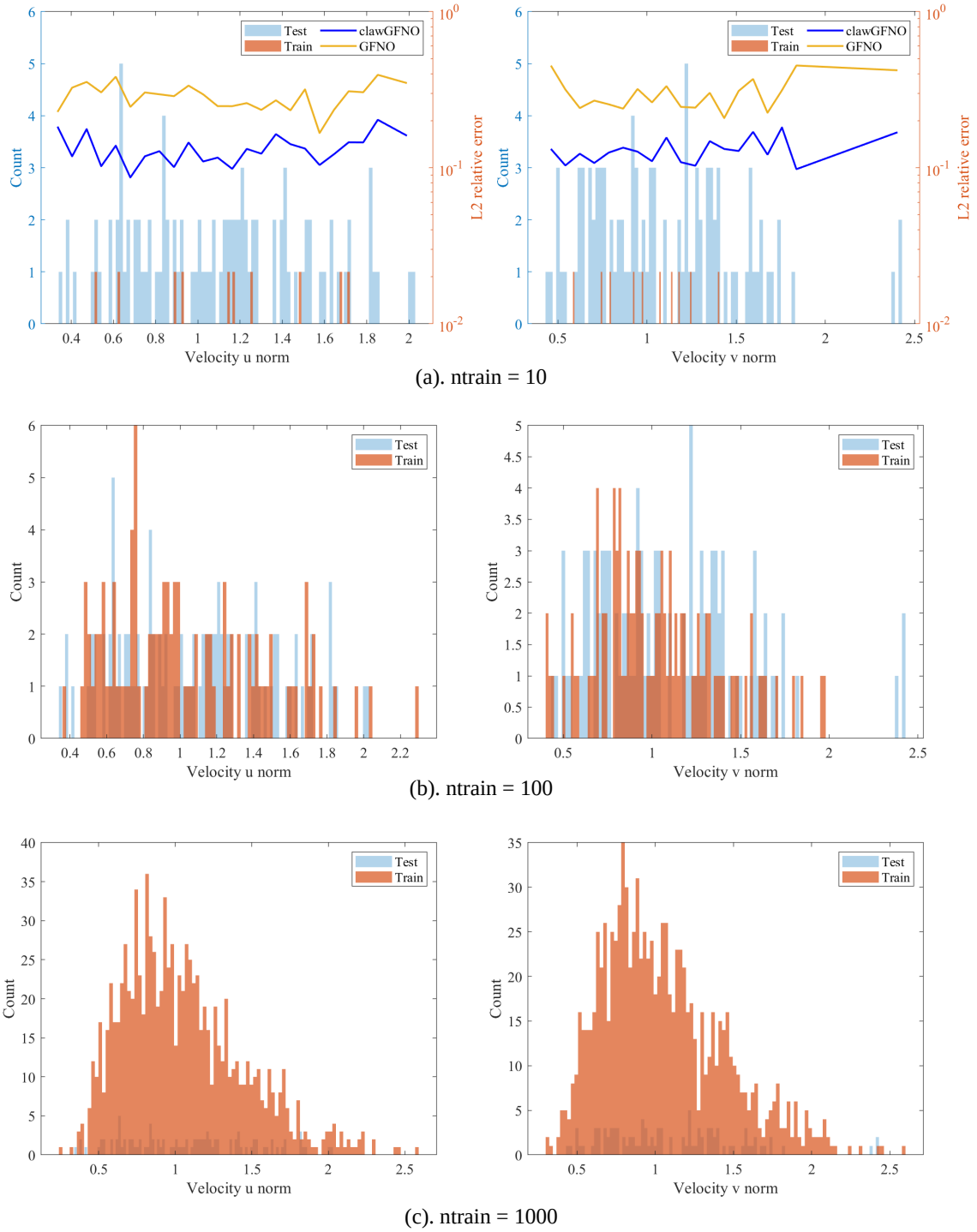
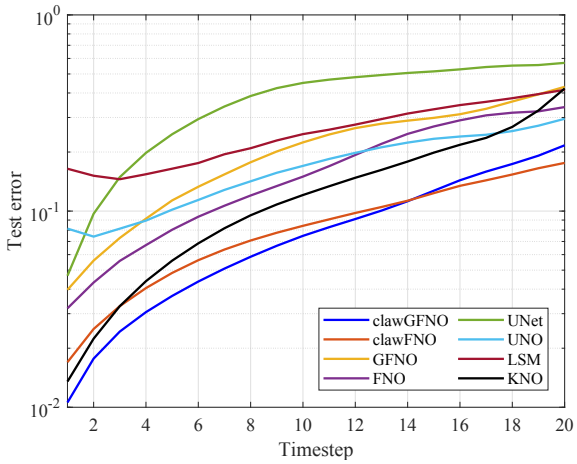


Figure 3. The data distribution of velocity in L2 norm, in the incompressible Navier-Stokes dataset. Left: the x component of velocity. Right: the y component of the velocity. Cases (a)-(c) represent the histogram of sample distributions in the small, medium and large data regimes, respectively, with blue representing the test samples and orange for the training samples. The per-sample relative L2 error on the test set is also plotted in (a), comparing clawGFNO (in navy) with GFNL (in yellow). This result demonstrates the improved the accuracy of clawNO, comparing to its counterpart, in small data regime.



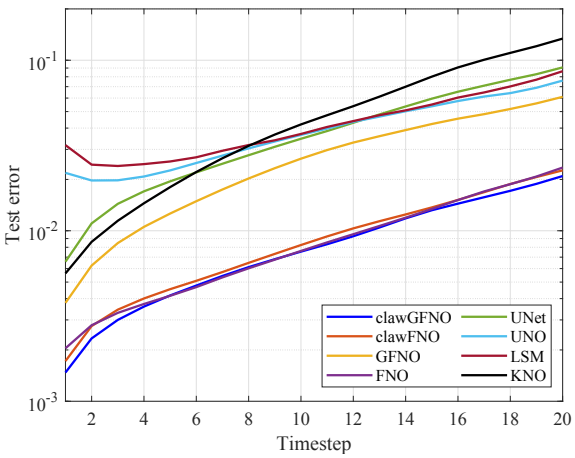


Figure 4. The per-time-step prediction error on the test dataset of the incompressible Navier-Stokes case 1: ntrain=10 (left) and ntrain=1000 (right).

Table 5. Performance comparison of selected models in incompressible Navier-Stokes case 1, in terms of the total number of parameters, the per-epoch runtime, inference time, peak GPU usage, and the L2-norm divergence of prediction. The runtime is evaluated on a single NVIDIA V100 GPU. Note that the case of ntrain=10 requiring more time to run compared to ntrain=100 is due to the reduced batch size of 2, as opposed to the batch size of 20 in ntrain=100.

Case	ntrain	clawGFNO	clawFNO	GFNO	FNO	UNet	LSM	UNO	KNO
nparam (M)		0.85	0.93	0.85	0.93	0.92	1.21	1.07	0.89
runtime (s)	10	7.10	4.76	6.14	3.80	4.85	9.86	4.24	4.31
	100	4.89	2.42	4.39	2.03	2.39	4.98	2.69	2.35
	1000	41.75	19.56	40.06	17.38	20.80	37.86	18.78	18.20
inf. time (s)		0.077	0.072	0.062	0.058	0.075	0.183	0.066	0.045
GPU (GB)	1000	0.68	0.34	0.68	0.34	0.12	13.20	2.14	0.16
L2(div)	1000	1.2e-3	3.8e-4	6.6e-2	4.7e-2	3.5e-1	5.4e-1	5.8e-1	1.8e-1

C.2. Example 2 – radial dam break

In generating the radial dam break dataset, we closely follow the numerical procedure in Takamoto et al. (2022), where we slightly modify the code to output the velocity fields in addition to the water height. The model then aims to learn three channels of (h, hu_x, hu_y). A total of 1,000 samples are generated and are subsequently split into n_{train}, 100, 100 and training, validation, and testing, respectively, with n_{train} the size of the training dataset depending on the adopted data regime. We run the numerical simulation on a $128 \times 128 \times 100$ grid and downsample the obtained solution to $32 \times 32 \times 25$ for training, where the first two dimensions are the spatial dimensions and the last is the temporal dimension. Analogous to the incompressible NS dataset, we perform 2×2 mean pooling in downsampling the spatial dimensions to maintain symmetry in data. In addition, we demonstrate the zero-shot super-resolution predictability of clawNO in Figure 5, where we use the clawFNO trained on 32×32 spatial resolutions to directly make predictions on 128×128 grids.

C.3. Example 3 – atmospheric modeling

As SpeedyWeather.jl uses spherical harmonics to solve the shallow water equations, we set the initial conditions η_0 for the aforementioned random waves through random coefficients of the spherical harmonics. The spherical harmonics are denoted as $Y_{\ell,m}$ with degree $\ell \geq 0$ and order m with $-\ell \leq m \leq \ell$. Using a standard complex normal distribution $\mathcal{CN}(0,1) = \mathcal{N}(0, \frac{1}{2}) + i\mathcal{N}(0, \frac{1}{2})$, the random coefficients $\eta_{\ell,m}$ are drawn for degrees $10 \leq \ell < 20$ from $\mathcal{CN}(0,1)$, but $\eta_{\ell,0} \sim \mathcal{N}(0,1)$ for the zonal modes $m = 0$, and $\eta_{\ell,m} = 0$ otherwise. The wave lengths are $2\pi R\ell^{-1}$, about 2000 to 4000 km.

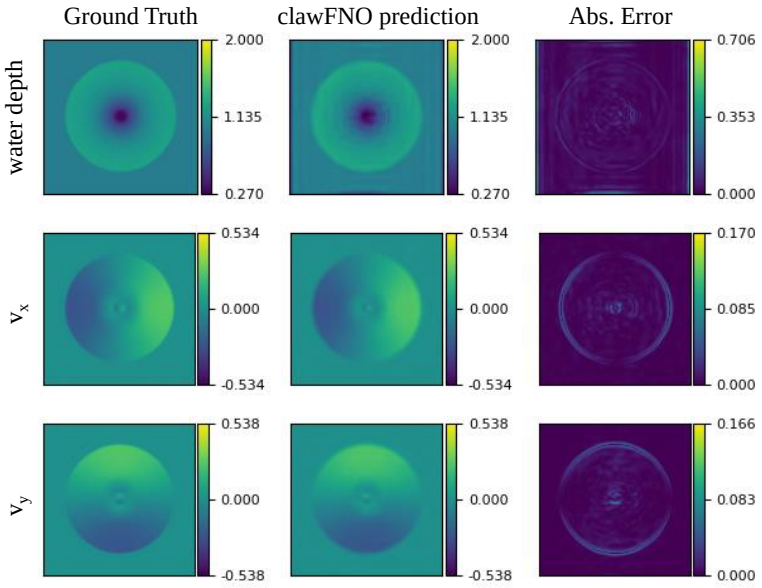


Figure 5. Demonstration of zero-shot super resolution of clawFNO using the radial dam break dataset, where clawFNO is trained on 32×32 spatial resolutions and directly make predictions on 128×128 grids.

The initial u_0, v_0, η_0 on a grid can be obtained through the spherical harmonic transform as

$$\begin{aligned} u_0 &= v_0 = 0, \\ \eta_0 &= A \sum_{\ell=0}^{\ell_{max}} \sum_{m=-\ell}^{\ell} \eta_{\ell,m} Y_{\ell,m}. \end{aligned} \tag{14}$$

The amplitude A is chosen so that $\max(|\eta_0|) = 2000$ m. The resolution of the simulation is determined by the largest resolved degree ℓ_{max}, we use $\ell_{max} = 63$. In numerical weather prediction this spectral truncation is widely denoted as T63. We combine this spectral resolution with a regular longitude-latitude grid of 192x95 grid points ($\Delta\lambda = \Delta\theta = 1.875°$, about 200 km at the Equator, no grid points on the poles), also called a full Clenshaw-Curtis grid because of the underlying quadrature rule in the Legendre transform (Hotta & Ujiie, 2018). Non-linear terms are calculated on the grid, while the linear terms are calculated in spectral space of the spherical harmonics, and the model transforms between both spaces on every time step. This is a widely adopted method in global numerical weather prediction models.

For numerical stability, an implicitly calculated horizontal diffusion term of the form $-\nu\nabla^8\zeta, -\nu\nabla^8\mathcal{D}$, is added to the vorticity and the divergence equation, respectively. The power-4 Laplacian is chosen to be very scale-selective such that energy is only removed at the highest wave numbers, keeping the simulated flow otherwise largely unaffected. In practice, we use a non-dimensional Laplace operator $\tilde{\nabla}^2 = R^2\nabla^2$, such that the diffusion coefficient becomes an inverse time scale of $\tau = 2.4$ hours. The shallow water equations, (12), do not have a forcing or drag term such that the horizontal diffusion is the only term through which the system loses energy over time. The shallow water equations are otherwise energy conserving.

SpeedyWeather.jl employs a RAW-filtered (Williams, 2011) Leapfrog-based time integration with a time step of $\Delta t = 15$ min at T63 resolution. At this time step, the CFL number $C = c_{ph}\Delta t(\Delta x)^{-1}$ with equatorial $\Delta x = 2\pi R\frac{\Delta\lambda}{360°}$ is typically between $C = 1$ and $C = 1.4$, given wave speeds $c_{ph} = \sqrt{gh}$ between 280 and 320 ms^{-1}. Thanks to a centred semi-implicit integration of the linear terms (Hoskins & Simmons, 1975), the simulation remains stable without aggressively dampening the gravity waves with larger time steps or with a backwards implicit scheme. The continuity equation with the centred semi-implicit leapfrog integration reads as (the RAW-filter is neglected)

$$\frac{\eta_{i+1} - \eta_{i-1}}{2\Delta t} = -\tfrac{1}{2}\nabla \cdot (\mathbf{u}_{i+1}h_{i+1}) - \tfrac{1}{2}\nabla \cdot (\mathbf{u}_{i-1}h_{i-1}) \tag{15}$$

with previous time step $i-1$, and next time step $i+1$. The RAW-filter then acts as a weakly dampening Laplacian in time, coupling the tendencies at $i-1, i$ and $i+1$ to prevent a computational mode from growing.

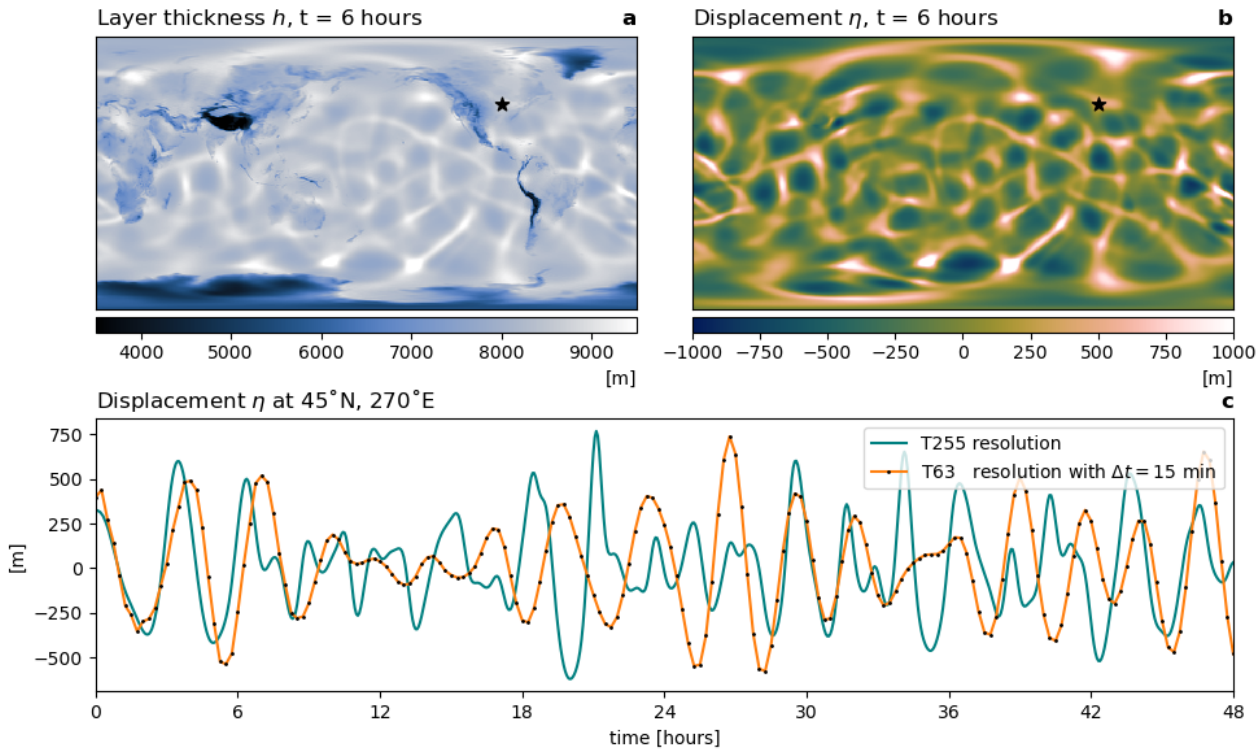


Figure 6. Atmospheric gravity waves as simulated by SpeedyWeather.jl. **a**, Layer thickness h and **b** displacement η after $t = 6$ hours at a resolution of T255 (about 50 km). The layer thickness h includes, in contrast to η, clearly the signal of the underlying orography of Earth. **c** Time series of η over the USA (45°N, 90°W, marked with a black star in **a**,**b**). Black circle markers denote the time step $\Delta t = 15$ min used for model integration and training data. Both simulations, T63 and T255, started from the same initial conditions, illustrating the limited predictability of the shallow water equations.

While the initial conditions contain only waves of wave lengths 2000 to 4000 km shorter waves are created during the simulation due to non-linear wave-wave interactions and interactions with the Earth's orography (Fig. 6). The time scale of these gravity waves is on the order of hours (Fig. 6c), which is why we use a training data sampling time step of $\Delta t = 15$ min. Much longer time steps as used by Gupta & Brandstetter (2022) would therefore fail to capture the wave dynamics present in the shallow water simulations. It is possible to use initial conditions that are closer to geostrophy, such that the presence of gravity waves from geostrophically-unbalanced initial conditions is reduced. In such a setup, the predictability horizon is given by the chaotic vorticity advection and turbulence that evolves over longer time scales, which would justify a longer data sampling time step. However, the Gupta & Brandstetter (2022) setup includes a strong gravity wave in the initial conditions that propagates meridionally, while also including some slowly evolving vortices. Our setup therefore represents a physically clearer defined problem, one that focuses on the non-linear gravity wave propagation in the shallow water system. Following (14), our setup can be easily recreated in other models for further studies. In this context, we generate a total of 1,200 samples and split them into 1,000/100/100 for training/validation/testing, respectively.

In this context, the mass conservation in spherical coordinates can be expressed as: $\frac{\partial \eta}{\partial t} + \nabla \cdot (\mathbf{u}h) = 0$, where η is the displacement from the atmosphere's rest height $H = 8500$ m, $h = \eta + H - H_o$ is the layer thickness, $H_o = H_o(\lambda, \phi)$ is the Earth's orography, and λ, ϕ are the longitude and colatitude, respectively. We note that $\frac{\partial \eta}{\partial t} = \frac{\partial h}{\partial t}$, and $\nabla \cdot (\mathbf{u}h)$ in the spherical coordinate can be rewritten as: $\nabla \cdot (\mathbf{u}h) = \frac{1}{R \sin\theta} \frac{\partial (u_x h)}{\partial \lambda} + \frac{1}{R \sin\theta} \frac{\partial (u_y h \sin\theta)}{\partial \theta}$ (here definitions using latitudes replace the $\sin$ with a $\cos$). Based on this formulation, we first rescale the equation by R, then set the clawNO model to learn $(u_x h, u_y h \sin\theta, Rh \sin\theta)$ and require the output to be divergence-free:

$$(\partial_\lambda, \partial_\theta, \partial_t) \cdot (u_x h, u_y h \sin\theta, Rh \sin\theta) = \frac{\partial (u_x h)}{\partial \lambda} + \frac{\partial (u_y h \sin\theta)}{\partial \theta} + \frac{\partial (Rh \sin\theta)}{\partial t} = \frac{\partial (u_x h)}{\partial \lambda} + \frac{\partial (u_y h \sin\theta)}{\partial \theta} + R \sin\theta \frac{\partial \eta}{\partial t} = 0.$$

As such, the mass conservation equation is guaranteed. Once the output field $\mathbf{u} = (u_x h, u_y h \sin\theta, Rh\sin\theta)$ is obtained, we post-process the prediction to obtain (u_x, u_y, h) (e.g., by dividing the second component $u_y h \sin\theta$ by the third component $Rh\sin\theta$ and multiplying the result by R, to obtain u_y). In this way, we can leave our skew symmetric matrix and numerical differentiation layer unchanged while being able to handle spherical data.

C.4. Example 4 – constitutive modeling of material deformation

We generate the material deformation dataset using an incompressible Mooney–Rivlin material model. We run the numerical simulation on a 80×80 uniform mesh and downsample the solution to a grid of 246 spatial points through interpolation on a circular domain of radius 0.4 that centers at the origin (cf. Figure 7). A total of 300 samples are generated and are subsequently split into n_{train}, 100, 100 for training, validation, and testing, respectively, with n_{train} the size of the training dataset depending on the adopted data regime.

In this example, we impose the kinematic condition of incompressibility with infinitesimal deformation, $div(\mathbf{u}) = 0$. This is equivalent to the mass conservation law only under the assumption that the density remains a constant, i.e., $\rho = const$, and then $div(\mathbf{u}) = 0$ guarantees that the total mass of a volume does not change under any applied deformation, i.e., its total mass is conserved. In solid mechanics, the deformation gradient writes $\mathbf{F} = \mathbf{I} + \nabla\mathbf{u}$, and the total mass of a volume does not change when $det(\mathbf{F}) \equiv 1$. Under an infinitesimal deformation assumption, one has $\frac{\partial u_i}{\partial x_j}\frac{\partial u_k}{\partial x_l} \approx 0$, and therefore $det(\mathbf{F}) \approx 1 + div(\mathbf{u})$. That means, the incompressibility condition is equivalent to $div\mathbf{u} = 0$.

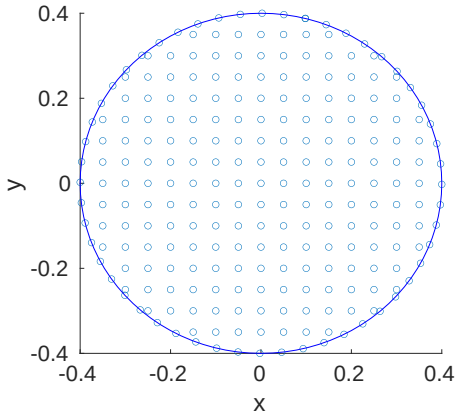


Figure 7. An illustration of the grid points and circular domain uused in example 4.

C.5. Training strategies

We run three replicates for all the experiments and report the mean and standard deviation of the L2 relative error for comparison metrics. For all Fourier-domain models, we closely follow the model setup in Helwig et al. (2023), employing four Fourier layers and keeping only the 12 lowest Fourier modes in all the 2D models and 8 spatial and 6 temporal lowest modes in all the 3D models. An exception is in the atmospheric modeling problem, where we truncate the spatial Fourier modes to 22 for correct physical realizability. For fair comparison, we adopt Cartesian encoding in all the models.

Similar to the model size in Helwig et al. (2023), we set the latent dimension in FNO and clawFNO to 20, whereas we counterbalance the additional dimensions introduced due to equivariance in GFNO and clawGFNO by reducing the latent dimension to 10 in all the 2D models and 11 in all the 3D models. For UNet, in order to arrive at a similar number of model parameters, we increase the first-layer dimension to 11 in the incompressible NS problem, to 15 in the radial dam break problem, and to 96 in weather modeling.

As suggested in Tran et al. (2022) and Helwig et al. (2023), we turn to the teacher forcing strategy to facilitate the learning process. We set the batch size to 20 for all 2D models and 10 for all 3D models, with the exception in small and medium data regimes, where we set batch size to 2 and 1 when the training datasets are of size 10 and 2, respectively. We employ

cosine annealing learning rate scheduler that decays the initial learning rate to 0. All the 2D models are trained for a total of 100 epochs whereas all the 3D models are trained for 500 epochs with an early stop if the validation loss stops improving for consecutive 100 epochs. 2D models are trained with less number of epochs as one training sample in 3D corresponds to $(T - T_{in})$ training samples in 2D. We directly take the baseline models in GFNO (Helwig et al., 2023), and further tune the hyperparameters (i.e., the learning rate and weight decay in the Adam optimizer) in clawNOs. All the experiments are carried out on a single NVIDIA A6000 40GB GPU.

For both of the two graph-based models (i.e., INO and GNO), we closely follow the model setup in Liu et al. (2023) and Li et al. (2020c). In order to be consistent across clawINO, INO and GNO, the latent width is set to 64 and the kernel width is set to 1,024 in all models. We employ a total of 4 integral layers in all models, whereas the shallow-to-deep technique is equipped for initialization in INO-based models. The batch size is set to 2 for all the irregular-mesh models, with the exception in the first case in the constitutive modeling example where a batch size equal to 1 is employed. We adopt the cosine annealing learning rate scheduler that decays the initial learning rate to 0. All the models are trained for 500 epochs with an early stop if the validation loss stops improving for consecutive 200 epochs.

D. Rollout visualizations

We illustrate the rollout of randomly selected trajectories in the test dataset using clawNO predictions, along with the comparisons against ground truth data and the corresponding absolute errors. The rollouts of incompressible NS, radial dam break, and atmospheric modeling are showcased in Figure 8, Figures 10 and 11, Figures 13 and 2, respectively. We also showcase the material deformation prediction in 6 different test samples in Figure 17. To provide a visual comparison across models, we plot the final-step prediction of all the models in the incompressible NS example in Figure 9, the radial dam break example in Figure 12, and the atmospheric modeling example in Figure 16.

E. Detailed error estimates for the differentiation layer

E.1. Proof of theorem 3.1

Since the Fourier spectral differentiation error estimate is a direct result of Trefethen (2000, Page 34), we provide the detailed derivation of (7) in this section. According to the basic properties of Fourier transform, if f is a differential and periodic function on $[0, L]$ with its Fourier representation:

$$f(\mathbf{x}) = \sum_{\xi=-N/2}^{N/2-1} \hat{f}(\xi) e^{i2\pi\xi/L},$$

its derivative can be given as

$$f'(\mathbf{x}) = \sum_{\xi=-N/2}^{N/2-1} \frac{i2\pi\xi}{L} \hat{f}(\xi) e^{i2\pi\xi/L} \approx \mathcal{F}^{-1}\left[\frac{i2\pi\xi}{L}\mathcal{F}[f](\xi)\right].$$

As such, (7) can be obtained by applying the above property to approximate every derivative term $\dfrac{\partial \mu_{jk}^{(N^{(1)},\cdots,N^{(p)})}}{\partial x_k}$.

E.2. Proof of theorem 3.2

It suffices to show that $\left|\dfrac{\partial \psi}{\partial x_k}(\mathbf{x}_i) - \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} (\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i))\omega_{i,l}^{(k)}\right| \leq C\Delta x^{m+1}$ for any $\psi \in C^{m+1}(\Omega)$. Denote $I[\psi](\mathbf{x}_i) := \dfrac{\partial \psi}{\partial x_k}(\mathbf{x}_i)$, $I_{\Delta x}[\psi](\mathbf{x}_i) := \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} (\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i))\omega_{i,l}^{(k)}$ and let ϕ_m denote the $m-$th order truncated Taylor series of ψ about $\mathbf{x}_i$ with associated remainder r_m, such that

$$\psi(\mathbf{y}) = \phi_m(\mathbf{y}) + r_m(\mathbf{y}) = \sum_{|\alpha|\leq m} \frac{D^\alpha \psi(\mathbf{x}_i)}{\alpha!}(\mathbf{y} - \mathbf{x}_i)^\alpha + \sum_{|\beta|=m+1} R_\beta(\mathbf{y})(\mathbf{y} - \mathbf{x}_i)^\beta,$$

where $R_\beta(\mathbf{y}) := \frac{|\beta|}{\beta!}\int_0^1 (1-\tau)^{|\beta|-1} D^\beta u(\mathbf{y}+\tau(\mathbf{y}-\mathbf{x}_i))d\tau$ and therefore $|R_\beta(\mathbf{y})| \leq \frac{1}{\beta!}\max_{|\alpha|=m+1}\max_{\mathbf{y}\in B_\delta(\mathbf{x}_i)}|D^\alpha\psi(\mathbf{y})| \leq C||\psi||_{C^{m+1}}$. We then have

$$
\begin{aligned}
|\psi-\phi_m|(\mathbf{y}) =& |r_m|(\mathbf{y}) = \left|\sum_{|\beta|=m+1} R_\beta(\mathbf{y})(\mathbf{y}-\mathbf{x}_i)^\beta\right| \\
\leq& |\mathbf{y}-\mathbf{x}_i|^{m+1}\sum_{|\beta|=m+1}|R_\beta(\mathbf{y})| \leq C||\psi||_{C^{m+1}}|\mathbf{y}-\mathbf{x}_i|^{m+1}.
\end{aligned}
$$

To bound the approximation error, we apply the triangle inequality and the reproducing condition of polynomial ϕ_m:

$$
\begin{aligned}
|I[\psi](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq& |I[\psi](\mathbf{x}_i) - I[\phi_m](\mathbf{x}_i)| + |I[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \\
=& |I[\psi](\mathbf{x}_i) - I[\phi_m](\mathbf{x}_i)| + |I_{\Delta x}[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)|.
\end{aligned}
$$

Here, the first term vanishes since ϕ_m is the truncated Taylor series of ψ and $m \geq 1$, and for the second term we have

$$
\begin{aligned}
|I_{\Delta x}[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq& \sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)} |\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i) - \phi_m(\mathbf{x}_l) + \phi_m(\mathbf{x}_i)||\omega_{i,l}^{(k)}| \\
=& \sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)} |\psi(\mathbf{x}_l) - \phi_m(\mathbf{x}_l)||\omega_{i,l}^{(k)}| \\
\leq& C||\psi||_{C^{m+1}} \sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)} |\mathbf{x}_l - \mathbf{x}_i|^{m+1}|\omega_{i,l}^{(k)}| \\
\leq& C||\psi||_{C^{m+1}}\Delta x^{m+1} \sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)} |\omega_{i,l}^{(k)}| \leq C\Delta x^{m+1}.
\end{aligned}
$$

Here, the last inequality can be proved following the argument in Levin (1998, Theorem 5): for each fixed k and i, the coefficient $\omega_{i,l}^{(k)}$ is a continuous function of $\mathbf{x}_l$. Moreover, the size of $\chi \cap B_\delta(\mathbf{x})$ is bounded. Thus, for a fixed Δx it follows $\sum_{\mathbf{x}_l\in\chi\cap B_\delta(\mathbf{x}_i)}|\omega_{i,l}^{(k)}| \leq C$ where C is independent of Δx, k, and $\mathbf{x}_i$. And therefore we obtain $|I[\psi](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq C\Delta x^{m+1}$ and finish the proof.

E.3. Performance comparison of different numerical differentiation schemes

To evaluate the empirical performance of the differentiation layer, we compare the performance of the FFT and meshfree methods discussed in Theorems 3.1 and 3.2. In particular, we take an analytical divergence-free function, $\mathbf{f} = [\sin x \sin y, \cos x \cos y]$, and use the numerical differentiation layer to evaluate its divergence and report the L^2 error. This result can be found in Figure 18. Here, the L^2 divergence errors using FFT remain near machine precision irregardless of the variation of grid sizes, because the spectrum generation is exact on the provided analytical function. On the other hand, for the meshfree method we have generated quadrature rules for $5-$th order polynomials, i.e., $m = 5$. A $6-$th order convergence is observed, which is consistent with the analysis in Theorem 3.2.

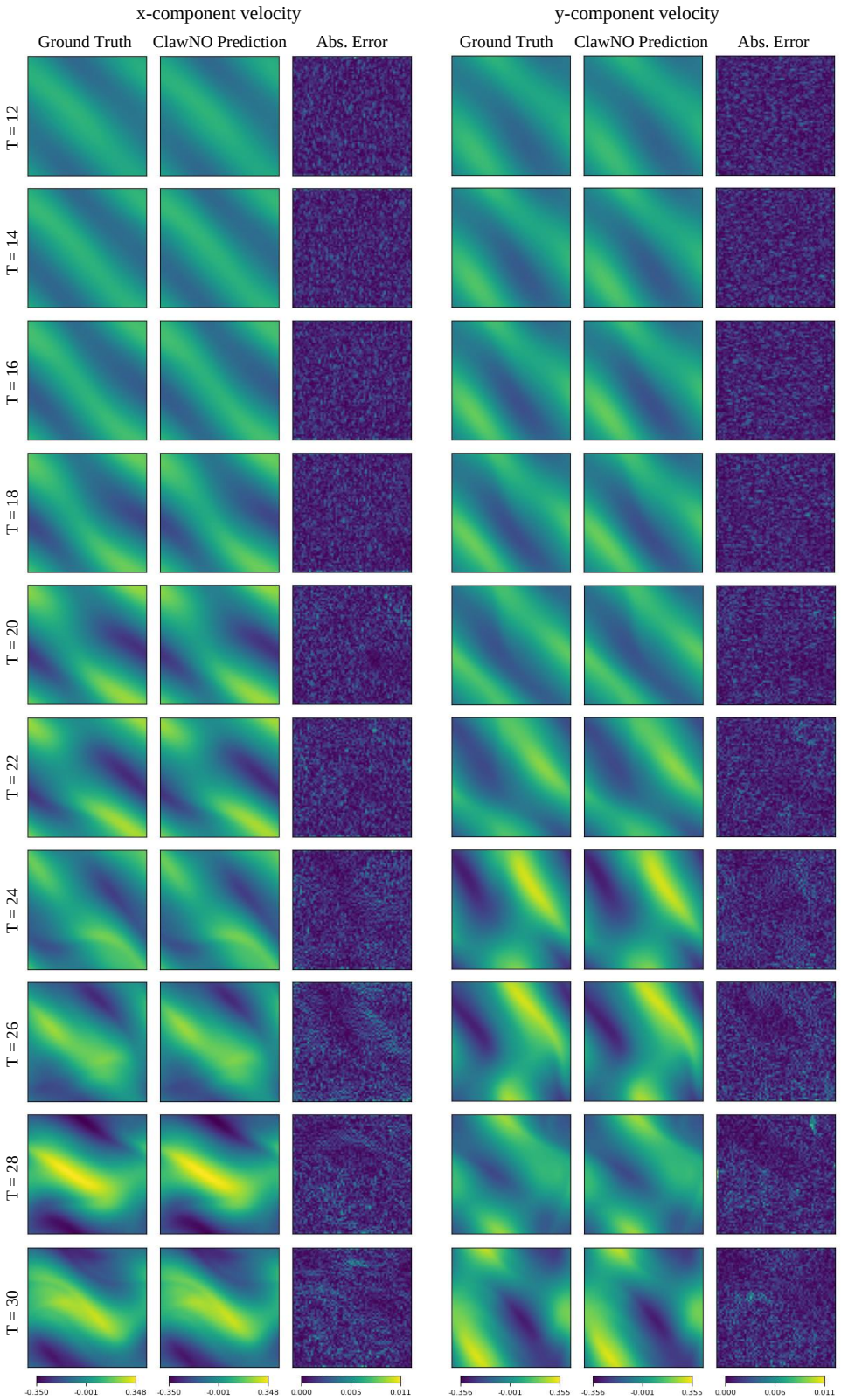


Figure 8. Rollout of incompressible Navier-Stokes case 1.

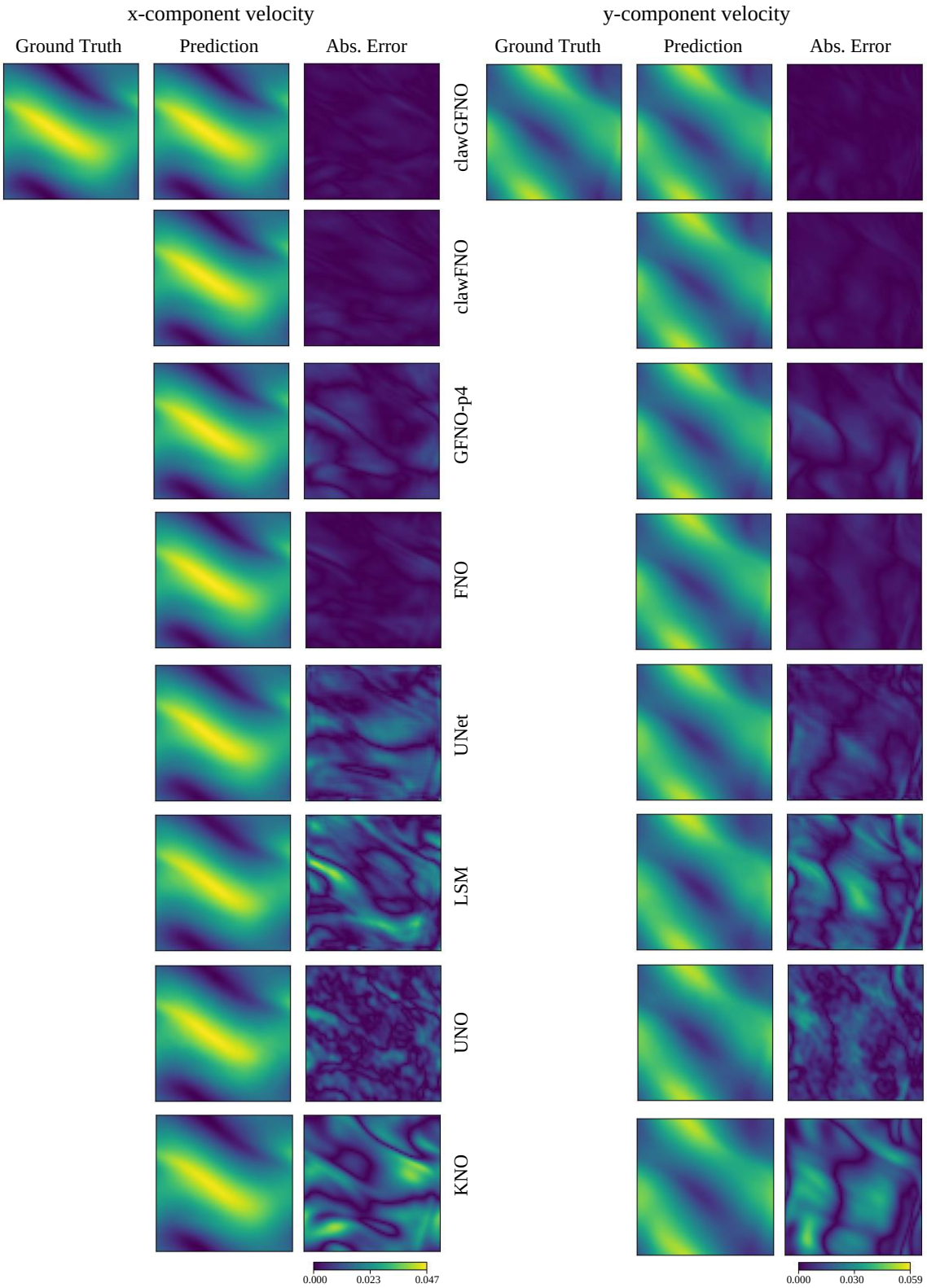


Figure 9. Last-step prediction comparison across models in incompressible Navier-Stokes case 1.

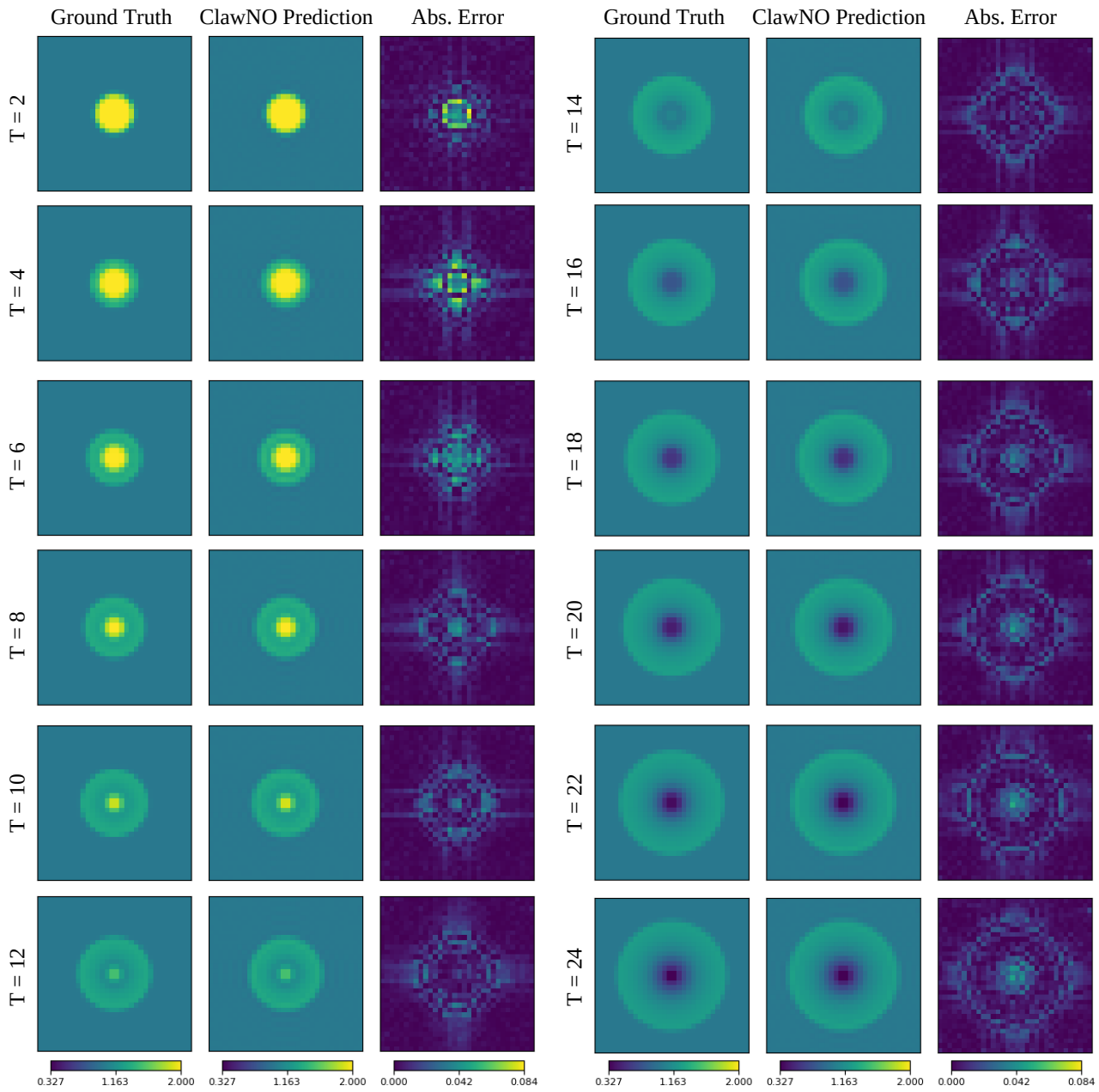


Figure 10. Rollout of water depth in radial dam break modeling.

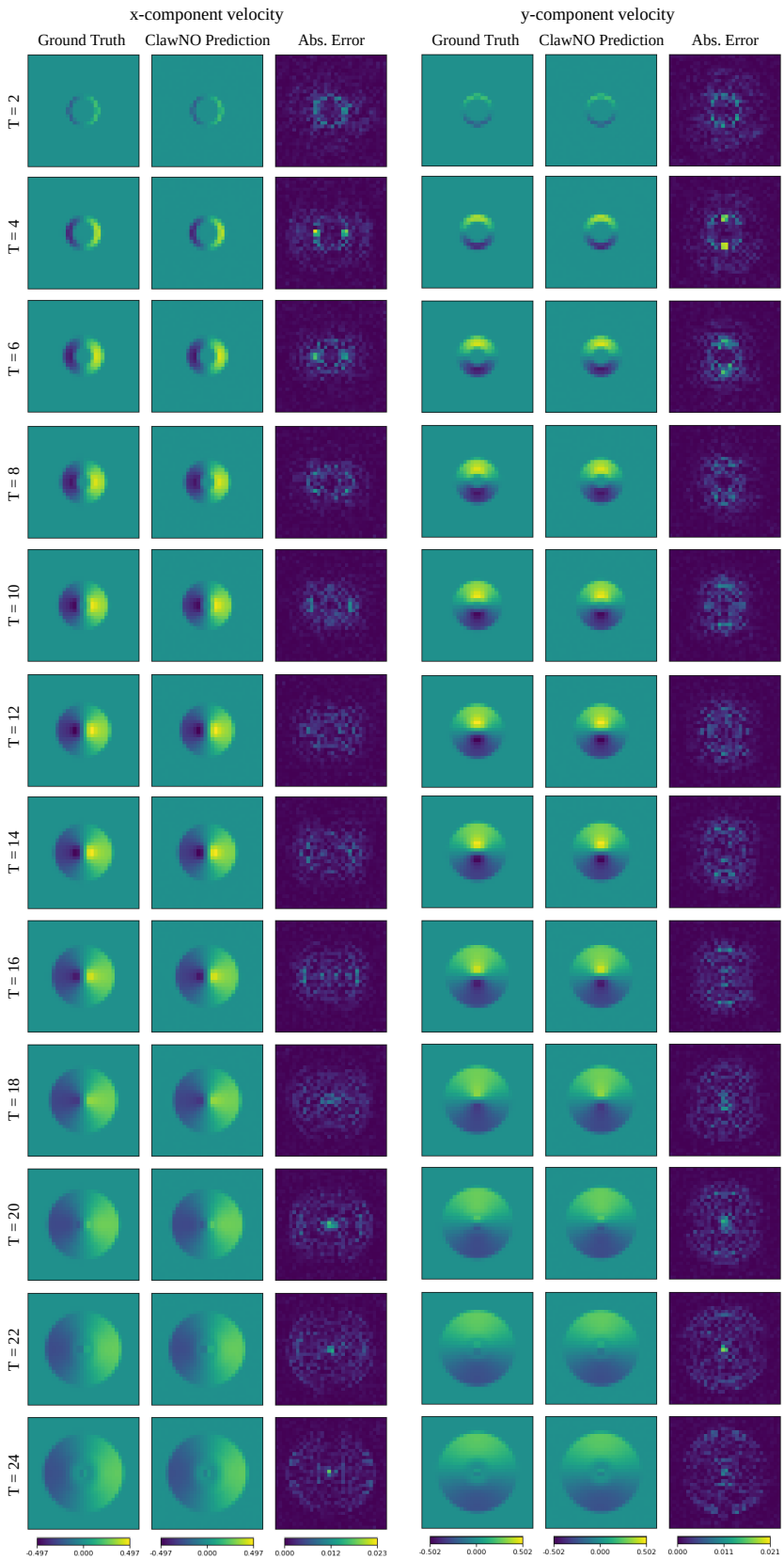


Figure 11. Rollout of velocity in radial dam break modeling.

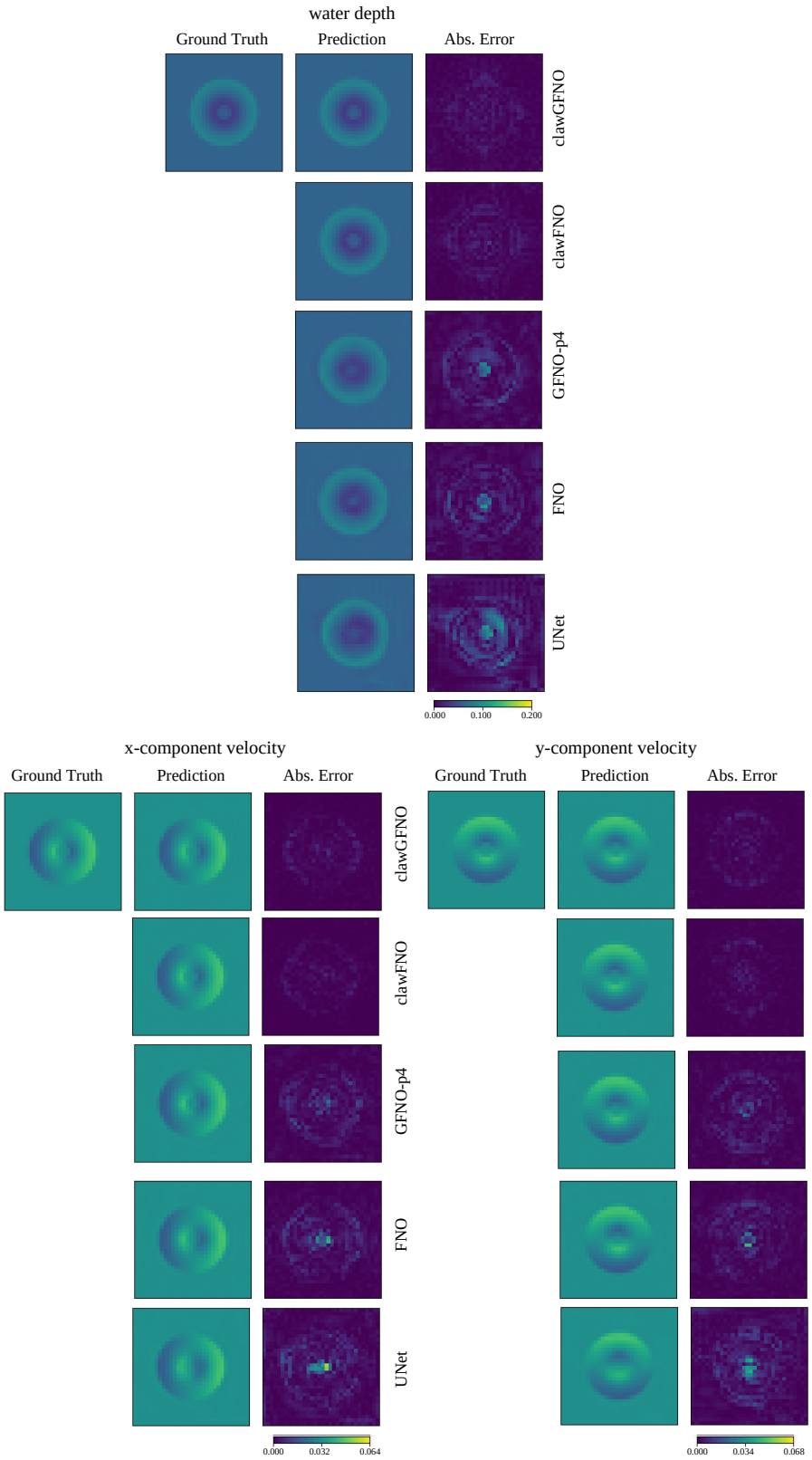


Figure 12. Last-step prediction comparison across models trained with ntrain=10 samples in radial dam break dataset.

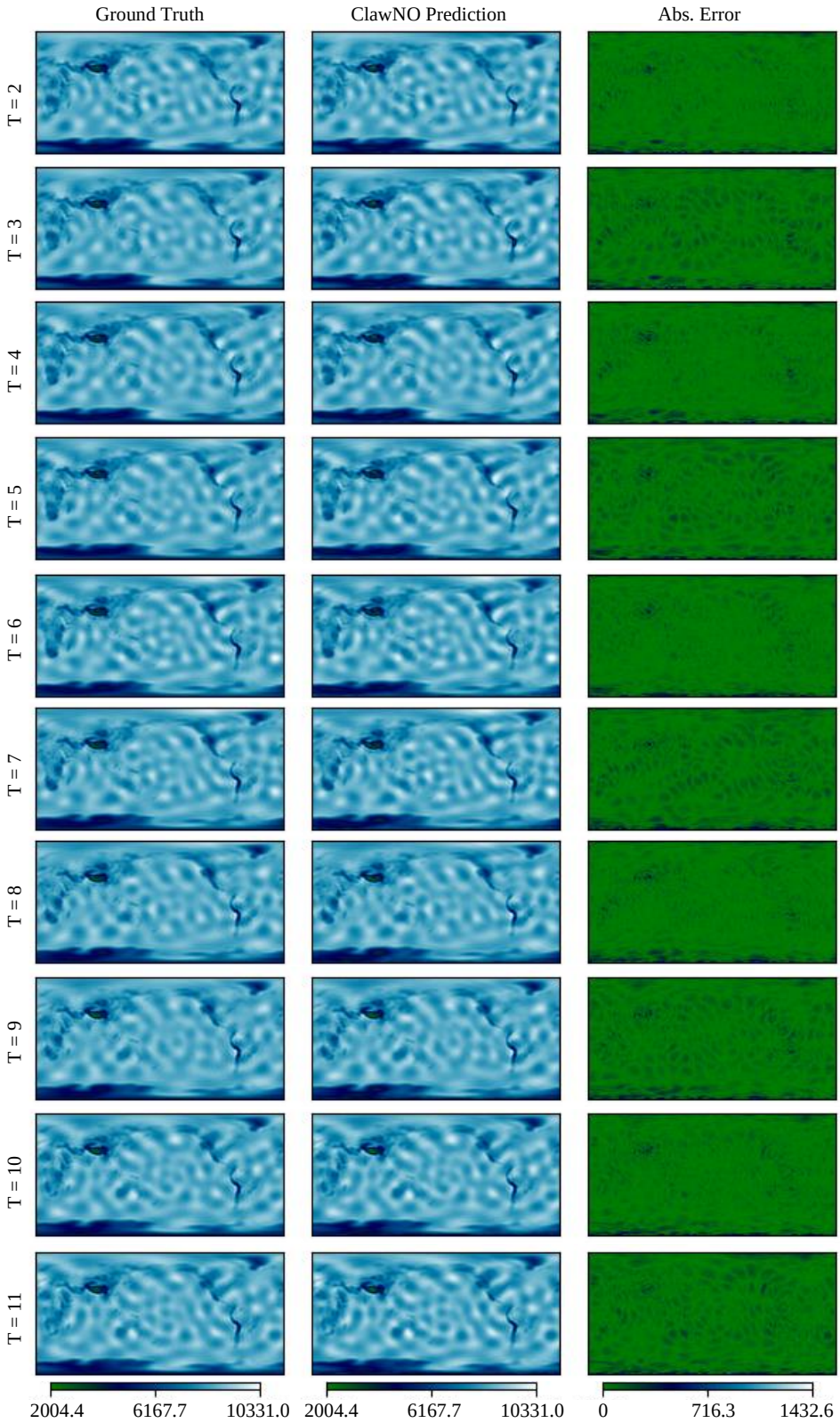


Figure 13. Rollout of layer thickness in atmospheric modeling.

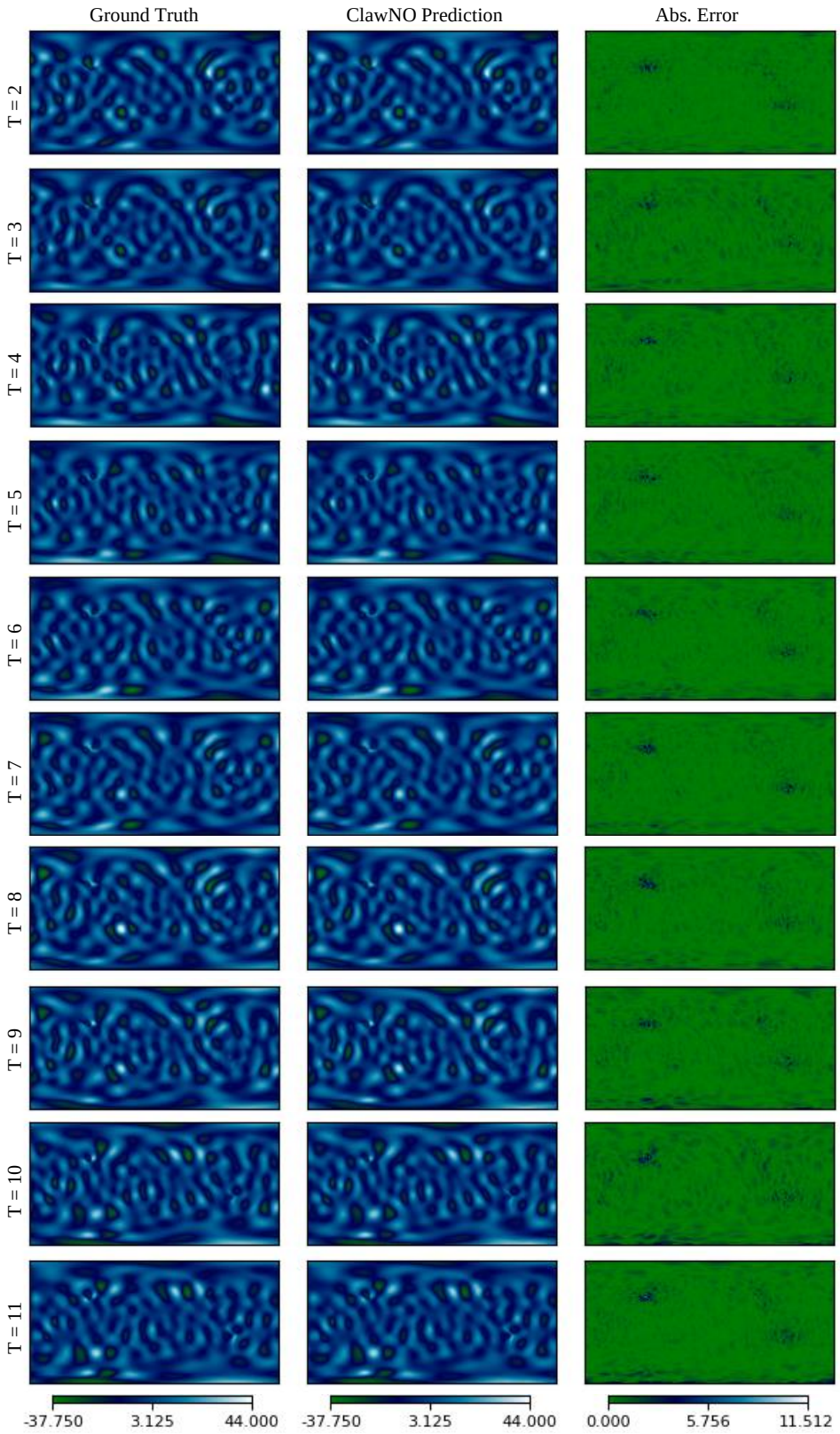


Figure 14. Rollout of zonal wind velocity in atmospheric modeling.

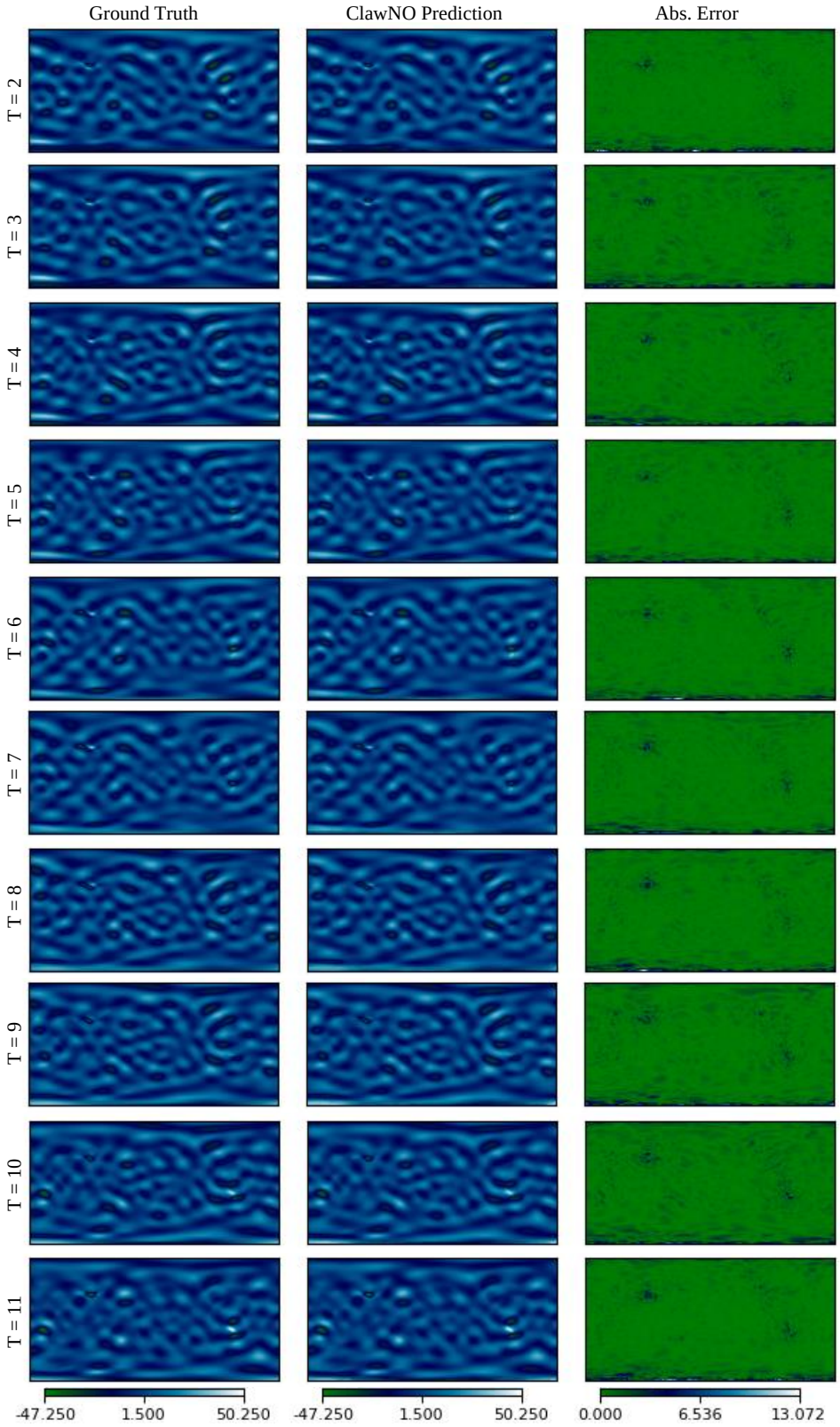


Figure 15. Rollout of meridional wind velocity in atmospheric modeling.

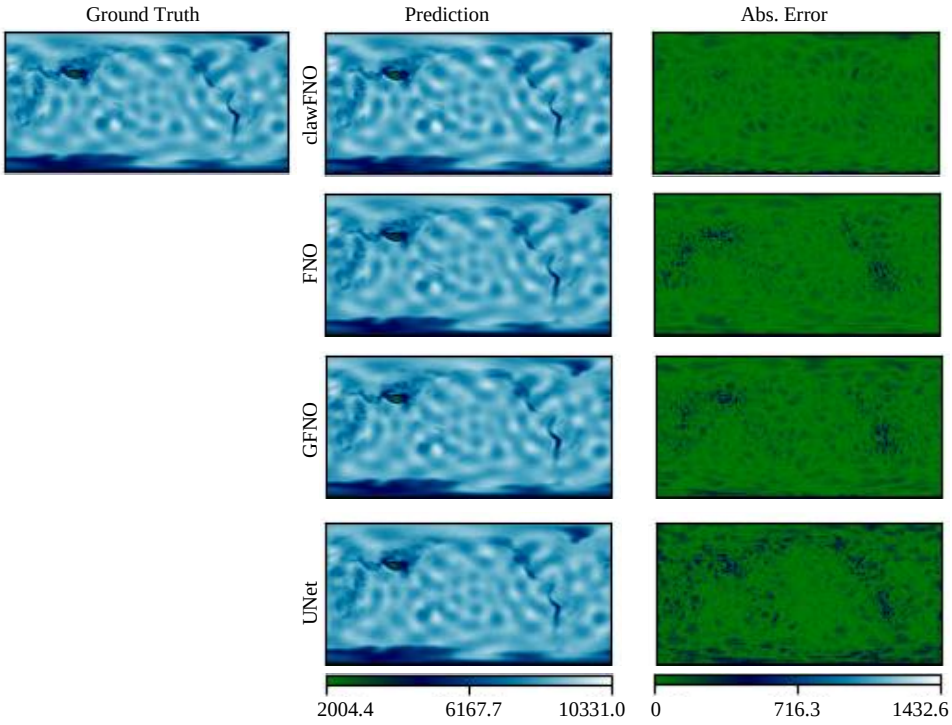


Figure 16. Last-step prediction comparison across models in atmospheric modeling dataset.

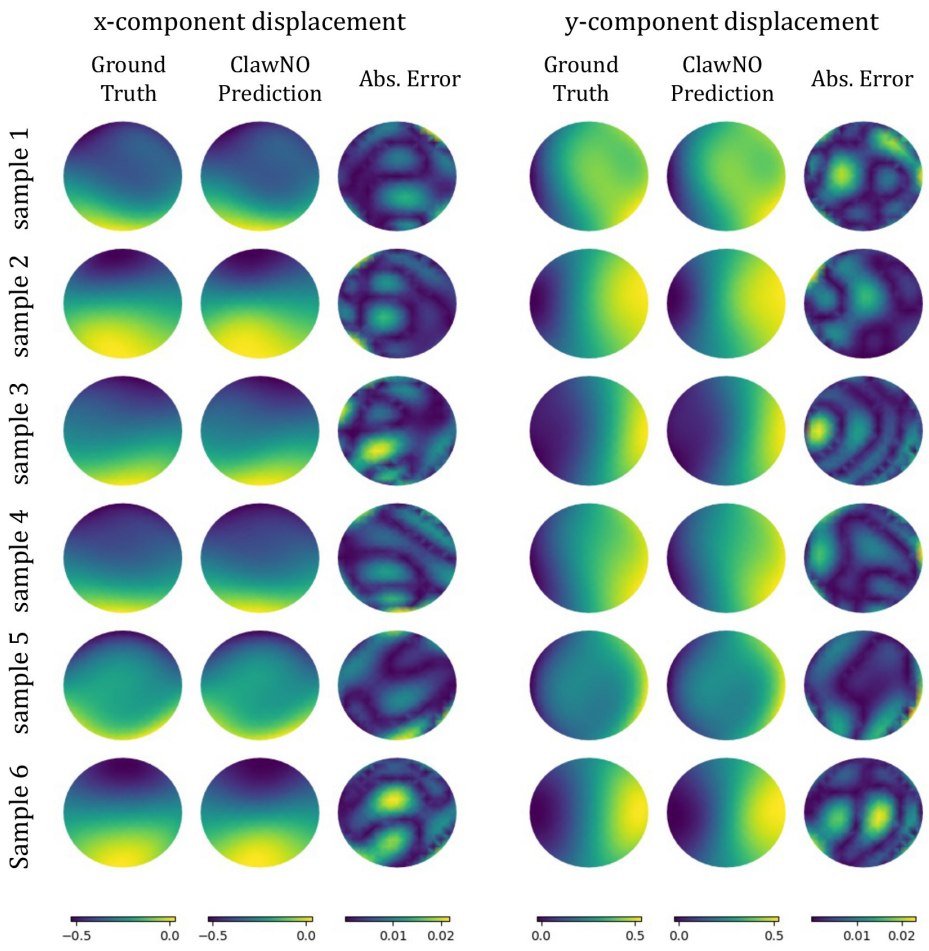


Figure 17. Demonstration of the constitutive modeling of material deformation.

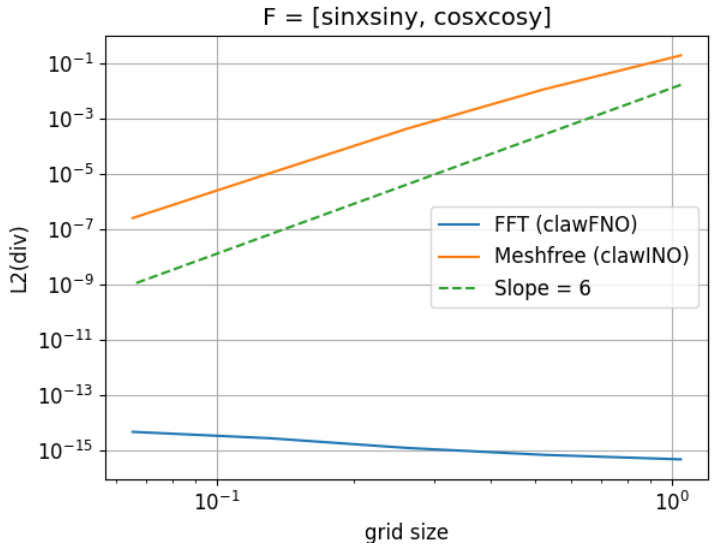


Figure 18. Numerical error of the differentiation layer, with refinement on grids.