INO: Invariant Neural Operators for Learning Complex Physical Systems with Momentum Conservation

Ning Liu
Global Engineering and Materials, Inc.

Yue Yu
Lehigh University

Huaiqian You
Lehigh University

Neeraj Tatikola
Lehigh University

Abstract

Neural operators, which emerge as implicit solution operators of hidden governing equations, have recently become popular tools for learning responses of complex real-world physical systems. Nevertheless, the majority of neural operator applications has thus far been data-driven, which neglects the intrinsic preservation of fundamental physical laws in data. In this paper, we introduce a novel integral neural operator architecture, to learn physical models with fundamental conservation laws automatically guaranteed. In particular, by replacing the frame-dependent position information with its invariant counterpart in the kernel space, the proposed neural operator is by design translation- and rotation-invariant, and consequently abides by the conservation laws of linear and angular momentums. As applications, we demonstrate the expressivity and efficacy of our model in learning complex material behaviors from both synthetic and experimental datasets, and show that, by automatically satisfying these essential physical laws, our learned neural operator is not only generalizable in handling translated and rotated datasets, but also achieves state-of-the-art accuracy and efficiency as compared to baseline neural operator models.

1 INTRODUCTION

Neural operators (Anandkumar et al., 2020; Li et al., 2020c; Lu et al., 2019) have gained popularity in recent years as a form of implicit solution operators to unveil hidden physics of complex real-world physical systems from data. Benefiting from their integral form of the architecture, neural operators learn a surrogate mapping between function spaces, which are resolution independent and can be generalized to different input instances (Kovachki et al., 2021). Resolution independence empowers the learned operator to retain consistent accuracy in prediction regardless of the variation of input resolutions, while being generalizable to different input instances offers the possibility to solve unseen input instances with only a forward pass of the trained network without the hassle of repeating the time-consuming training process. These facts make neural operators excellent candidates for providing surrogate models for complex physical systems (Li et al., 2021; Goswami et al., 2022a).

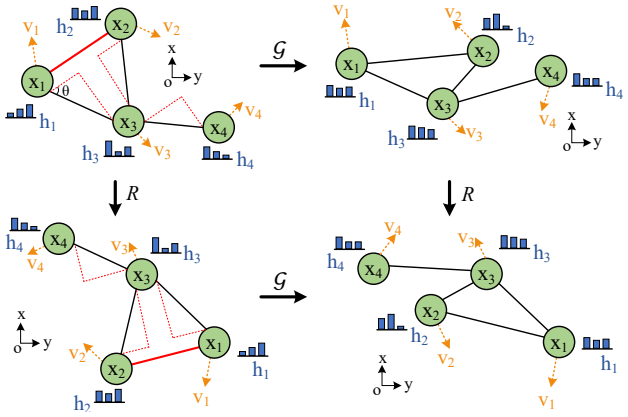

Figure 1: An illustration of the translational and rotational invariance on a mesh grid with INO. Here, $\mathcal{G}$ represents the learnt mapping, R represents the rotation and translation of coordinate frames. To design an invariant neural operator, our key idea is to characterize the interaction between nodes via an invariant kernel. The solid red lines represent the selected local reference edges and the dashed red lines indicate the two components of relative Euclidean distance, $\overline{x_j - x_i} := [|x_j - x_i|\cos\theta, |x_j - x_i|\sin\theta]$, with which we parameterize the proposed kernel form.

Despite of the notable advances in the development of neural operators over traditional neural networks (NNs), their performance highly rely on the amount of available data,

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

especially when the governing PDE is unknown Goswami et al. (2022a). An effective way to alleviate this pathology is to incorporate into the designed architecture the intrinsic preservation of fundamental physical laws in data, as is the case of the conservation of linear and angular momentums in most physical systems. In You et al. (2022b), it was found that incorporating partial physics, such as the no-permanent-set assumption, can enhance the prediction of neural operators in out-of-distribution prediction tasks. However, to the authors' best knowledge, neural operators that preserve the conservation laws of linear and angular momentums have not yet been explored.

To enhance the accuracy of neural operators in physical system predictions, in this work we propose the Invariant Neural Operator (INO), a novel integral neural operator architecture that remains invariant under frame translations and rotations. Specifically, we substitute the frame-dependent coordinate information with its invariant counterpart (cf. Figure 1) in the kernel space, so that the learned kernel is independent of geometric transformations. Compared to existing neural operator methods, the proposed architecture mainly carries four significant advantages. First, different from existing physics-informed neural operators, our approach only requires observed data pairs and does not rely on known governing equations (Goswami et al., 2022b,a; Li et al., 2021; Wang et al., 2021). Therefore, it is readily applicable to learn physical systems directly from experimental measurements (Ranade et al., 2021) or simulation data (Kim et al., 2019), for which the underlying governing equations may not be available. Second, the invariant properties in our approach are realized through built-in kernel functions, which is anticipated to be more robust and efficient than data augmentation techniques (Quiroga et al., 2018). More importantly, through embedding the invariant kernel and updating frame-dependent coordinate information with a separate network, our architecture naturally stems from the interpretation of its layer update as a particle-based method (Karplus & McCammon, 2002; Liu & Liu, 2010), which significantly simplifies model interpretation. Last but not least, analogous to the E(n) equivariance concept in Satorras et al. (2021), our architecture is not limited to two- or three-dimensional invariance. As a matter of fact, it can be easily scaled up to higher dimensions. In summary, the contributions of our work are:

- We propose INO, a novel integral neural operator architecture that is translation- and rotation-invariant, to learn complex physical systems with guaranteed conservation of linear and angular momentums.
- Equipped with the shallow-to-deep initialization technique and a coordinate embedding network, our INO finds a physical interpretation from a particle-based method, and obtains stable predictions as the network proliferates in depth.
- Our approach only requires data pairs and does not rely on *a priori* domain knowledge, while the guaranteed momentum conservation laws improve the learning efficacy, especially in small data regime.
- We demonstrate the expressivity and generalizability of INO across a variety of synthetic and real-world experimental datasets, and show that our learned neural operator is not only generalizable in handling translated and rotated datasets, but also provides improved prediction from the baseline neural operators.

2 BACKGROUND AND RELATED WORK

In this section, we briefly introduce the concept of translational and rotational invariance in classical mechanics, and present its connection to the laws of momentum conservation. Moreover, we review relevant concepts of invariance/equivariance and hidden physical system learning with NNs, which will later become complementary to the proposed INO definition.

Throughout this paper, we use lower case letters to denote vectors, upper case letters for matrices, bold letters for functions, calligraphic letters for operators, and blackboard-bold letters for Banach spaces. For any vector v, we use $|v|$ to denote its l^2 norm. For any function $\boldsymbol{f}$ taking values at nodes $\chi := \{x_1, x_2, \ldots, x_M\}$, $||\boldsymbol{f}||$ denotes its l^2 norm, i.e., $||\boldsymbol{f}|| := \sqrt{\sum_{i=1}^{M}(\boldsymbol{f}(x_i))/M}$. $\mathbb{R}^d$ represents the dimension-d Euclidean space.

2.1 Invariance, Equivariance, and Momentum Conservation Laws

We consider the learning of complex physical responses of a mechanical system, based on a number of observations of the loading field $\boldsymbol{f}_i(x) \in \mathbb{F}(\Omega; \mathbb{R}^{d_f})$ and the corresponding physical system response $\boldsymbol{u}_i(x) \in \mathbb{U}(\Omega; \mathbb{R}^{d_u})$. Here, i denotes the sample index, $\Omega \in \mathbb{R}^d$ is the bounded domain of interests, and $\mathbb{F}$ and $\mathbb{U}$ describe the Banach spaces of functions taking values in $\mathbb{R}^{d_f}$ and $\mathbb{R}^{d_u}$, respectively. To model the physical responses of such a system, we aim to learn a surrogate operator $\mathcal{G} : \mathbb{F} \to \mathbb{U}$, that maps the input function $\boldsymbol{f}(x)$ to the output function $\boldsymbol{u}(x)$.

Let $\mathcal{T}_g : \mathbb{F} \to \mathbb{F}$ be a set of transformation operators for an abstract group g, we say that the operator $\mathcal{G}$ is invariant to g if

$$\mathcal{G} \circ \mathcal{T}_g[\boldsymbol{f}] = \mathcal{G}[\boldsymbol{f}] \,, \tag{1}$$

and $\mathcal{G}$ is equivariant to g if there exists an equivariant transformation $\mathcal{S}_g : \mathbb{U} \to \mathbb{U}$, such that

$$\mathcal{G} \circ \mathcal{T}_g[\boldsymbol{f}] = \mathcal{S}_g \circ \mathcal{G}[\boldsymbol{f}] \,. \tag{2}$$

Considering a mechanical response problem as a practical example in physical systems, we have the input func-

tion $\boldsymbol{f}(x)$ as the initial location and $\boldsymbol{u}(x)$ as the resulting mechanical response in the form of a displacement field. First, let $\mathcal{T}_g$ be a translation on the reference frame, i.e., $\mathcal{T}_g[\boldsymbol{f}] = \tilde{\boldsymbol{f}}$, where $\tilde{\boldsymbol{f}}(x+g) := \boldsymbol{f}(x)$ and $g \in \mathbb{R}^d$ is a constant vector. Translational invariance means that translating the input function $\boldsymbol{f}$ first and then applying the response operator $\mathcal{G}$ will deliver the same result. As such, the resultant physical model does not vary with locations in space, and the Noether's theorem (Noether, 1971) guarantees the conservation of linear momentum. On the other hand, let $\mathcal{T}_g$ be a rotation on the reference frame, which rotates the coordinate x as well as the input function, i.e., $\mathcal{T}_g[\boldsymbol{f}] = \tilde{\boldsymbol{f}}$, with $\tilde{\boldsymbol{f}}(Rx) := R\boldsymbol{f}(x)$ and R being an orthogonal matrix. Rotational equivariance means that rotating the input function $\boldsymbol{f}$ first and then applying the response operator $\mathcal{G}$ will lead to the same result as first applying $\mathcal{G}$ and then rotating the output function. As such, the described physical model does not vary under rotations against the origin, and the Noether's theorem (Noether, 1971) guarantees the conservation of angular momentum.

In this work, the proposed INO is designed to handle the following four types of invariance/equivariance:

1. *Translational Invariance.* Translating the reference frame by $g \in \mathbb{R}^d$ results in an invariant output, i.e., $\mathcal{G}[\tilde{\boldsymbol{f}}](x+g) = \mathcal{G}[\boldsymbol{f}](x)$, where $\tilde{\boldsymbol{f}}(x+g) := \boldsymbol{f}(x)$.

2. *Translational Equivariance.* Translating the reference frame by $g \in \mathbb{R}^d$ results in an equivariant translation of the output, i.e., $\mathcal{G}[\tilde{\boldsymbol{f}}](x+g) = \mathcal{G}[\boldsymbol{f}](x) + g$, where $\tilde{\boldsymbol{f}}(x+g) := \boldsymbol{f}(x)$.

3. *Rotational Invariance.* Rotating the reference frame results in an invariant output, i.e., for any orthogonal matrix $R \in \mathbb{R}^{d\times d}$, one has $\mathcal{G}[\tilde{\boldsymbol{f}}](Rx) = \mathcal{G}[\boldsymbol{f}](x)$, where $\tilde{\boldsymbol{f}}(Rx) := R\boldsymbol{f}(x)$.

4. *Rotational Equivariance.* Rotating the reference frame results in an equivariant rotation of the output, i.e., for any orthogonal matrix $R \in \mathbb{R}^{d\times d}$, one has $\mathcal{G}[\tilde{\boldsymbol{f}}](Rx) = R\mathcal{G}[\boldsymbol{f}](x)$, where $\tilde{\boldsymbol{f}}(Rx) := R\boldsymbol{f}(x)$.

2.2 Learning Hidden Physics

Learning how complex physical systems respond is essential in science and engineering. For decades, physics-based PDEs have been commonly employed to model such systems, and traditional numerical methods (LeVeque, 1992) are developed to solve for unobserved system responses. However, the choice of certain governing PDEs is often determined *a priori*, and these PDEs need to be solved numerically for each specified boundary/initial conditions and loading/source terms, which makes classical PDE-based methods insufficient in expressivity and computationally expensive.

Several recent developments in deep NNs have been devoted to providing an efficient surrogate directly from data (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). Among others, neural operators manifest superiority in predicting physical responses as function mappings. Contrary to classical NNs that operate between finite-dimensional Euclidean spaces, neural operators are designed to learn mappings between infinite-dimensional function spaces (Li et al., 2020a,b,c; You et al., 2022a; Ong et al., 2022; Gupta et al., 2021b; Lu et al., 2019, 2021a; Goswami et al., 2022b; Gupta et al., 2021a). A remarkable advantage of neural operators lies in their resolution independence, which implies that the prediction accuracy is invariant to the resolution of input functions. Moreover, neural operators are generalizable to different input instances, and hence they can serve as efficient surrogates in downstream applications. Furthermore, in contrast to classical PDE-based approaches, neural operators can be trained directly from data, and hence requires no domain knowledge nor pre-assumed PDEs. All these advantages make neural operators a promising tool for learning complex physical systems (Yin et al., 2022a; Goswami et al., 2022b; Yin et al., 2022b; You et al., 2022b; Li et al., 2020a,b,c; Lu et al., 2021b).

Despite the aforementioned advances of neural operators, purely data-driven neural operators still suffer from data challenge. In particular, in order to generalize the solution, they require a large corpus of paired datasets, which is prohibitively expensive in many engineering applications. To resolve this challenge, the physics-informed neural operator (PINO) Li et al. (2021) and physics-informed DeepONets Goswami et al. (2022b); Wang et al. (2021) are introduced, where a PDE-based loss is added to the training loss as a penalization term. However, these approaches still require *a priori* knowledge of the underlying PDEs, which restricts their applications to (known) PDE-solving tasks.

2.3 Integral Neural Operators

Integral neural operators, first proposed in Li et al. (2020a) and further developed in Li et al. (2020b,c); You et al. (2022a,c), have the foundation in the representation of a PDE solution by the Green's function. An integral neural operator is comprised of three building blocks. First, the input function, $\boldsymbol{f}(x) \in \mathbb{F}$, is lifted to a higher-dimension representation via $\boldsymbol{h}(x,0) = \mathcal{P}[\boldsymbol{f}](x) := P[x, \boldsymbol{f}(x)]^T + p$. Here, $P \in \mathbb{R}^{(d+d_f)\times d_h}$ and $p \in \mathbb{R}^{d_h}$ define an affine pointwise mapping. Next, the feature vector function $\boldsymbol{h}(x,0)$ goes through an iterative layer block where the layer update is defined via the sum of a local linear operator, a nonlocal integral kernel operator, and a bias function: $\boldsymbol{h}(\cdot, j+1) = \mathcal{J}_{j+1}[\boldsymbol{h}(\cdot,j)]$, for $j = 0, \cdots, L-1$. Here, $\boldsymbol{h}(\cdot,j)$ is a sequence of functions representing the values of the network at each hidden layer, taking values in $\mathbb{R}^{d_h}$. $\mathcal{J}_1, \cdots, \mathcal{J}_L$

are the nonlinear operator layers, which will be further discussed in the later contents. Finally, the output $\boldsymbol{u}(\cdot) \in \mathbb{U}$ is obtained through a projection layer. A common practice is to project the last hidden layer representation $\boldsymbol{h}(\cdot, L)$ onto $\mathbb{U}$ as: $\boldsymbol{u}(x) = \mathcal{Q}[\boldsymbol{h}(\cdot, L)](x) := Q_2\sigma(Q_1\boldsymbol{h}(x, L)+q_1)+q_2$. Here, $Q_1 \in \mathbb{R}^{d_Q \times d_h}$, $Q_2 \in \mathbb{R}^{d_u \times d_Q}$, $q_1 \in \mathbb{R}^{d_Q}$ and $q_2 \in \mathbb{R}^{d_u}$ are the appropriately sized matrices and vectors that are part of the trainable parameter set. σ is an activation function, which is often taken to be the popular rectified linear unit (ReLU) function.

Let $\mathcal{D} := \{(\boldsymbol{f}_i, \boldsymbol{u}_i)\}_{i=1}^N$ be a support set of observations where the input $\{\boldsymbol{f}_i\} \subset \mathbb{F}$ is a sequence of independent and identically distributed (i.i.d.) random fields from a known probability distribution μ on $\mathbb{F}$, and $\boldsymbol{u}_i(x) \in \mathbb{U}$, possibly noisy, is the observed corresponding solution. We aim to learn the system response by building a surrogate operator:

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

where the parameter set θ is obtained by solving the following optimization problem:

$$\begin{aligned} \min_{\theta\in\Theta} \mathcal{L}_{\mathcal{D}}(\theta) &= \min_{\theta\in\Theta} \mathbb{E}_{\boldsymbol{f}\sim\mu} \left|\left| \tilde{\mathcal{G}}[\boldsymbol{f}; \theta] - \mathcal{G}[\boldsymbol{f}] \right|\right| \\ &\approx \frac{1}{N} \min_{\theta\in\Theta} \sum_{i=1}^{N} \left|\left| \tilde{\mathcal{G}}[\boldsymbol{f}_i; \theta] - \boldsymbol{u}_i \right|\right| . \end{aligned} \tag{3}$$

The particular choice of an integral neural operator varies by the architecture of the iterative layer block, $\mathcal{J}_{j+1}$. In Li et al. (2020a), graph neural operators (GNOs) are proposed, where the iterative kernel integration is invariant across layers, i.e., $\mathcal{J}_1 = \mathcal{J}_2 = \cdots = \mathcal{J}_L := \mathcal{J}^{GNO}$, with the update of each layer network given by

$$\begin{aligned} \boldsymbol{h}(x, j+1) &= \mathcal{J}^{GNO}(\boldsymbol{h}(x, j)) \\ &:= \sigma\left(W\boldsymbol{h}(x, j) + \int_\Omega \boldsymbol{m}(x, y)\boldsymbol{h}(x, j)dy + c \right) , \end{aligned} \tag{4}$$

$$\boldsymbol{m}(x, y) := \kappa\left(x, y, \boldsymbol{f}(x), \boldsymbol{f}(y); v\right) . \tag{5}$$

Here, $W \in \mathbb{R}^{d_h \times d_h}$ and $c \in \mathbb{R}^{d_h}$ are learnable tensors, and $\kappa \in \mathbb{R}^{d_h \times d_h}$ is a tensor kernel function that takes the form of a (usually shallow) NN with learnable parameters v. Since the layer update in integral neural operators is formulated as a continuous integral operator, the learned network parameters are resolution-independent: the learned W, c, and v are close to optimal even when used with different resolutions. Besides GNOs, when both the domain and the discretized points are structured, Fourier Neural Operators (FNOs) (Li et al., 2020c) and Multiwavelet-based Operators (MWT) (Gupta et al., 2021b) can be employed. In FNOs, the fast Fourier transform is employed to evaluate the integrals, which presents superior efficiency. Nevertheless, despite the rapid advancement in neural operators, existing methods fail to preserve the invariance/equivariance properties under translation and rotation operations.

2.4 Invariant and Equivariant Neural Networks

Recently, invariant and equivariant NNs have been developed in the context of convolutional neural networks (Lang & Weiler, 2020; Chirikjian, 2000; Knapp, 2001) and graph neural networks (GNNs) (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016), and their effectiveness is demonstrated via a variety of machine learning tasks, such as in image classification (Cohen & Welling, 2016a,b; Weiler & Cesa, 2019; Romero & Cordonnier, 2020) and dynamical system modelling (Rezende et al., 2019; Satorras et al., 2021). To achieve equivariance, the authors in Thomas et al. (2018); Fuchs et al. (2020) proposed to utilize spherical harmonics to compute a set of basis for transformations between higher-order representations. As another line of efforts (Schütt et al., 2017; Klicpera et al., 2020; Anderson et al., 2019; Miller et al., 2020; Satorras et al., 2021), GNNs were considered based on a message passing framework Brandstetter et al. (2022), in which the translational and rotational invariance/equivariance were imposed by specially designed edge and node update operations. However, the above-mentioned invariant/equivariant networks are restricted to a discrete "vector-to-vector" mapping, and the learned parameters cannot be reused in networks of different input/output resolutions, which hinders their application to learn hidden physics laws in the form of function mappings. Therefore, the goal in this work is to design neural operators and impose minimal physical law assumptions as the translational and rotational invariance/equivariance, so as to provide a data-driven model form that learns complex physical systems with guaranteed momentum conservation.

3 NEURAL OPERATORS WITH MOMENTUM CONSERVATION

In this section, we develop the invariant/equivariant architecture based on integral neural operators. Our developments have two-folds. First, a node embedding update scheme is proposed, that is physically invariant and preserves invariance/equivariance to translations and rotations on a continuous domain. The essential idea is to devise a message passing neural network where its arguments and relevant representation embeddings are invariant to transformations. As such, we can convert transformation-sensitive representations to their transformation-invariant counterparts. Second, to handle general domains and accelerate training, we also modify the GNO architecture in Eqs. (4)-(5) in such a way that each layer resembles a discretized time-dependent nonlocal equation You et al. (2022a). As a result, our proposed architecture can be seen as a resemblance with translational and rotational invariant/equivariant nonlocal differential equations, allowing for generalization of its optimal parameters from shallow to deep networks.

To establish a transformation-invariant kernel in Eq. (5), we introduce two types of transformation-invariant quantities as arguments to the kernel function: the vector Euclidean norm of the edge between x and y, i.e., $|y-x|$, and the orientation of the vector $y-x$ with respect to a local reference vector in the undeformed coordinates. For example, the local reference edge in a rectangular domain can be either the horizontal or vertical edge of the rectangle. In the perspective of numerical implementation, one can take the vector formed by any two fixed nodes as the reference edge, as illustrated in Figure 1. In physical problems with 2D domains, $\Omega \subset \mathbb{R}^2$, we pass in as kernel arguments the decomposed Euclidean norm in the following form:

$$\overline{y-x} := [|y-x|\cos\theta, |y-x|\sin\theta]\,, \tag{6}$$

where x and y are the source and target nodes connected by the edge, and θ denotes the computed local orientation. Similarly, for $\Omega \subset \mathbb{R}^3$, three kernel arguments are passed in, based on two angles from the computed local orientation. Here, we point out that the idea of parameterizing the edge feature with its Euclidean norm, $|y-x|$, was also employed in the equivariant graph neural network proposed in Satorras et al. (2021). However, our approach has for the first time considered the local edge orientation together with its Euclidean norm, which makes the resulting network more expressive. As will be demonstrated in the ablation study in Section 4, an Euclidean norm-based kernel architecture may not be sufficiently expressive.

INO for scalar-valued functions. We first consider the scenario where the output function takes scalar values, i.e., $d_u = 1$, and the physical system is both translation- and rotation-invariant. Examples in this category involve the prediction of energies in environmentally-powered systems (Cammarano et al., 2012), pressure monitoring in subsurface flows (Fumagalli & Scotti, 2011), and the prediction of damage field in brittle fracture (Fan et al., 2022). In this context, we propose the the following INO-scalar architecture: for the lifting block, we only pass in the Euclidean norm of the input function:

$$\boldsymbol{h}(x,0) = \mathcal{P}[\boldsymbol{f}](x) := P|\boldsymbol{f}(x)| + p\,, \tag{7}$$

where $P, p \in \mathbb{R}^{d_h}$. Then, for the iterative layer, we introduce a fictitious time step, τ, and regard different layer features as the solution of a time-dependent nonlocal equation at different time instances:

$$\begin{aligned}\boldsymbol{h}(x,(j+1)\tau) &:= \boldsymbol{h}(x,j\tau) + \\ &\tau\sigma\left(W\boldsymbol{h}(x,j\tau) + \int_\Omega \boldsymbol{m}(x,y)\boldsymbol{h}(y,j\tau)dy + c\right)\,, \quad (8)\\ \boldsymbol{m}(x,y) &:= \kappa\left(\overline{y-x}, |\boldsymbol{f}(x)|, |\boldsymbol{f}(y)|; v\right)\,. \quad (9)\end{aligned}$$

Finally, the projection block is taken as a 2-layer multilayer perceptron (MLP), same as the other integral neural operators. Herein, we notice that the architecture in Eq. (8) resembles the time-dependent nonlocal equation: if we divide both sides of Eq. (8) by the fictitious time step, τ, the term $(\boldsymbol{h}(x,(j+1)\tau) - \boldsymbol{h}(x,j\tau))/\tau$ corresponds to the discretization of a first order derivative so that this architecture can indeed be interpreted as a nonlinear differential equation in the limit of deep layers, as $\tau \to 0$. Hence, in practice we can employ the shallow-to-deep learning technique (Haber et al., 2018; You et al., 2022a), that corresponds to training the network for increasing values of network layers and using optimal parameters obtained with L layers as initial guesses for the $\tilde{L}$-layer INO. Here $\tilde{L} > L$. Moreover, we point out that all network arguments are translational and rotational invariant, and hence we have the following theorem, with its detailed proof provided in Appendix A:

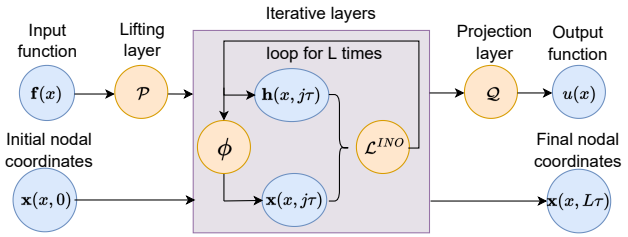

Figure 2: An illustration of the proposed INO architecture. We start from input $\boldsymbol{f}(x)$ and the initial coordinate $\boldsymbol{x}(x,0) := x$. Then, the iterative layers are built as integral operators based on invariant kernel functions, to obtain a frame-invariant layer feature update $\boldsymbol{h}(x,j)$ and a frame-dependent coordinate update $\boldsymbol{x}(x,j)$ that embeds the coordinate rotation information. Lastly, we project the last hidden layer representation (for scalar-valued functions) and the coordinate update (for vector-valued functions) to the target function space.

Theorem 1 (Invariance for INO-scalar)**.** *The INO-scalar architecture proposed in Eqs.* (7)-(9) *is translational and rotational invariant. That means, when translating the reference frame by $g \in \mathbb{R}^d$ and rotating it by an orthogonal matrix $R \in \mathbb{R}^{d\times d}$, the following property holds true:*

$$\tilde{\mathcal{G}}[\tilde{\boldsymbol{f}};\theta](Rx+g) = \tilde{\mathcal{G}}[\boldsymbol{f};\theta](x)\,,$$

where $\tilde{\boldsymbol{f}}(Rx+g) := R\boldsymbol{f}(x)$.

INO for vector-valued functions. We now further consider the case where the output function takes vector values, and hence the output should rotate equivariantly with the rotation of the reference frame. Under this scenario, rotation-equivariant property is required to achieve the conservation of angular momentum. As such, besides the layer feature function $\boldsymbol{h}(x,j\tau)$, $j = 0,\cdots,L$, we further introduce an additional coordinate function, $\boldsymbol{x}(x,j\tau)$, which is defined on domain Ω and takes values in $\mathbb{R}^d$. Then, the key is to preserve invariance to rotations on $\boldsymbol{h}$, as well as equivariance on $\boldsymbol{x}$. In light of this, when translational invariance and rotational equivariance are desired, we propose the fol-

lowing INO-vector architecture (cf. Figure 2): for the lifting block, we provide the Euclidean norm of the original feature to $\boldsymbol{h}$ and carry the coordinate information into $\boldsymbol{x}$:

$$\boldsymbol{h}(x,0) = \mathcal{P}[\boldsymbol{f}](x) := P|\boldsymbol{f}(x)| + p\,, \tag{10}$$

$$\boldsymbol{x}(x,0) := x\,, \tag{11}$$

with $P, p \in \mathbb{R}^{d_h}$. Then, the $(l+1)$-th iterative layer network update of a L-layer INO is defined as,

$$\boldsymbol{h}(x,(j+1)\tau) := \boldsymbol{h}(x,j\tau) + \tau\sigma\left(W\boldsymbol{h}(x,j\tau) + \int_\Omega \boldsymbol{m}(x,y)\boldsymbol{h}(y,j\tau)dy + c\right), \tag{12}$$

$$\boldsymbol{x}(x,(j+1)\tau) := \boldsymbol{x}(x,j\tau) + \tau\int_\Omega (x-y)\phi(\boldsymbol{m}(x,y)\boldsymbol{h}(y,j\tau);w)dy\,, \tag{13}$$

$$\boldsymbol{m}(x,y) = \kappa\left(\overline{y-x}, |\boldsymbol{f}(x)|, |\boldsymbol{f}(y)|; v\right)\,. \tag{14}$$

Finally, we define the projection block and the output function as,

$$\boldsymbol{u}(x) = \mathcal{Q}[\boldsymbol{x}(x,L\tau)](x) := \boldsymbol{x}(x,L\tau) - x\,. \tag{15}$$

Here, κ and ϕ are two separate (usually shallow) MLPs for computing edge and coordinate messages, with v and w being the corresponding trainable parameters, respectively. In Eq. (13), ϕ takes as input the edge embeddings and outputs a scalar representing the weight associated with its neighboring node. The nodal positions are then updated as the weighted sum of coordinate differences. When considering the output function $\boldsymbol{u}$ as the displacement field and $\boldsymbol{x}$ as the updated position of material points, the proposed INO-vector architecture can be seen as an analogue to the particle-based methods, since both approaches describe the motion of a particle by the summation of forces due to its neighboring particles. Additionally, the INO architecture preserves the continuous integral treatment of the interactions between nodes that characterizes neural operators. As a result, INO also permits resolution independence with respect to the inputs, and hence serves as a surrogate operator between function spaces. Formally, we have the following theorem, with the detailed proof provided in Appendix A:

Theorem 2 (Invariance/equivariance for INO-vector)**.** *The INO-vector architecture proposed in Eqs.* (10)-(15) *is translation-invariant and rotation-equivariant. That means, when translating the reference frame by $g \in \mathbb{R}^d$ and rotating it by an orthogonal matrix $R \in \mathbb{R}^{d\times d}$, the following property holds true:*

$$\tilde{\mathcal{G}}[\tilde{\boldsymbol{f}};\theta](Rx+g) = R\tilde{\mathcal{G}}[\boldsymbol{f};\theta](x)\,,$$

where $\tilde{\boldsymbol{f}}(Rx+g) := R\boldsymbol{f}(x)$.

4 EXPERIMENTS

In this section, we demonstrate the empirical effectiveness of INOs. Specifically, we conduct experiments on both

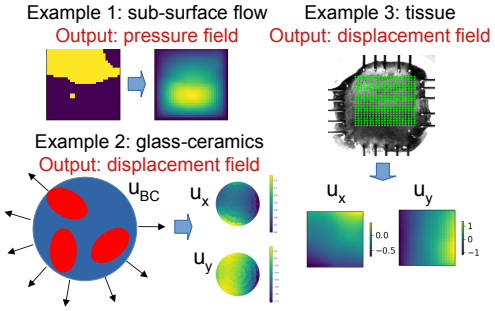

Figure 3: Illustration of input/output settings in our three exemplar problems.

2D synthetic and real-world datasets, and compare the proposed INO against three data-based neural operator baselines – GNOs Li et al. (2020a), FNOs Li et al. (2020c), and MWTs Gupta et al. (2021a), a physics-based neural operator baseline (PINO Li et al. (2021)), and an equivariant GNN (EGNN Satorras et al. (2021)). Herein, we employ $L = 4$ iterative layers for all neural operators. All experiments are tested using PyTorch with Adam optimizer. For fair comparison, we tune the hyperparameters for each method, including the learning rates, decay rates, and regularization coefficient, to minimize the error on the validation dataset. In all tests, we report the averaged relative error, $||\boldsymbol{u}_{i,pred} - \boldsymbol{u}_i||/||\boldsymbol{u}_i||$, as the comparison metric (lower means better). An illustration of our three test examples are provided in Figure 3, with further details of each dataset and experimental settings provided in Appendix B. The source code and the corresponding datasets are available at `https://github.com/ningliu-iga/invariant_neural_operators`.

4.1 Synthetic dataset: sub-surface flow

As a benchmark for scalar-valued output functions, we consider the modeling of 2D sub-surface flow through a porous medium with heterogeneous permeability field. The high-fidelity synthetic simulation data in this example is described by the Darcy flow, which has been considered in a series of neural operator studies (Li et al., 2020a,b,c; Lu et al., 2021b; You et al., 2022a,c). Specifically, the governing differential equation is defined as:

$$\begin{aligned} -\nabla\cdot(\boldsymbol{f}(x)\nabla\boldsymbol{u}(x)) &= 1,\ \forall\ x\in\Omega := [0,1]^2, \\ \text{subjected to}\quad \boldsymbol{u}_{BC}(x) &= 0,\ \forall\ x\in\partial\Omega, \end{aligned} \tag{16}$$

where $\boldsymbol{f}$ is the conductivity field, and $\boldsymbol{u}$ is the hydraulic head (both are scalar-valued functions). In this context, we aim to learn a solution operator of the Darcy's equation that maps each realization of the conductivity field $\boldsymbol{f}(x)$ to the corresponding solution field $\boldsymbol{u}(x)$. For training, we employ the dataset from Li et al. (2020a), where the conductivity field $\boldsymbol{f}(x)$ is modeled as a two-valued piecewise constant function with random geometries such that the two

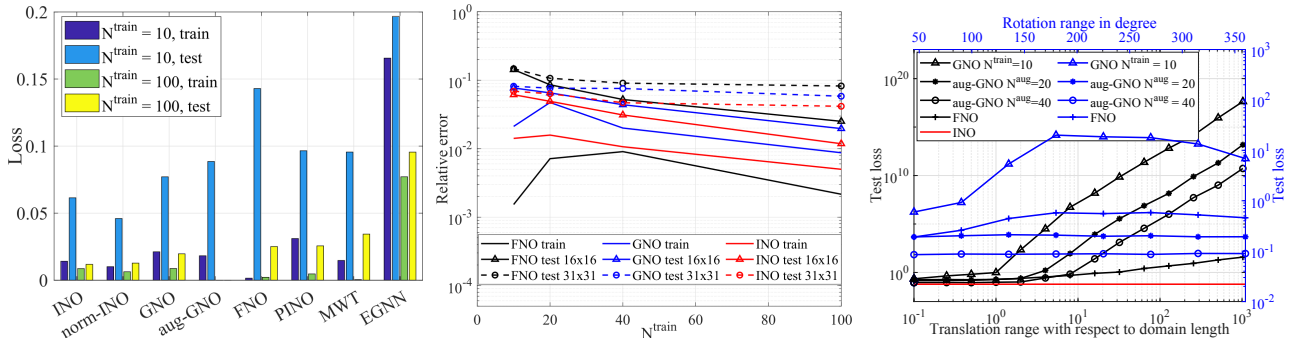

Figure 4: Results for cases with scalar-valued output function. Left: comparison between INO, GNO, norm-INO, aug-GNO, and other baseline models in small and medium data regimes. Middle: comparison between FNO, GNO, and INO with varying numbers of training samples and test on grids with different resolutions. Right: Generalization results to test samples with translated and rotated reference frames.

values have a ratio of 4. Then, the ground-truth solutions of $\boldsymbol{u}(x)$ were generated using a second-order finite difference scheme on a fine resolution of 241×241 grids and downsampled to 16×16 and 31×31. Herein, we split the provided test samples into 40 samples as validation for hyperparameter tuning and 40 samples as test. To investigate the performance of each model under small data regime, we train with $N^{\text{train}} = \{5, 10, 20, 40, 100\}$ numbers of labelled data pairs. The dimension of representation is set to $d_h = 64$, and the kernel κ is modeled by a 3-layer MLP with width $(n, 512, 1024, d_h^2)$, where n is the number of kernel arguments for each architecture. For 2D problems, $n = 6$ for GNOs and $n = 4$ for INOs.

Ablation study. We first conduct an ablation study on the proposed algorithm, with four settings: 1) The original INO-scalar architecture; 2) The original GNO architecture; With this test, we aim to investigate the expressivity of our invariant kernel compared to the original kernel of (5). 3) The INO-scalar architecture, with its kernel κ depending on the Euclidean norm of the edge only (denoted as "norm-INO"); With this test, we study if the local edge orientation, θ, plays an important role in the model. 4) The GNO architecture with training data augmentation (denoted as "aug-GNO"), where we train the GNO model with additional translated/rotated samples. Specifically, we translate the reference frame by setting $\tilde{\Omega} \in R([C_x, 1 + C_x] \times [C_y, 1 + C_y])$. Herein, the randomly generated constants $C_x, C_y \sim \mathcal{U}[-1, 1]$, where $\mathcal{U}[-1, 1]$ denotes the uniform distribution on $[-1, 1]$. Similarly, for rotation we randomly generate $C_\theta \sim \mathcal{U}[0, 2\pi]$ and rotate the reference coordinates counter-clockwisely. For each training sample, we repeat the above process to augment the training set for 3 times. With this test, we investigate if our invariant architecture outperforms an invariance-agnostic approach with data augmentation. On each of these four settings, we report the training and test errors with $N^{\text{train}} = \{10, 100\}$[1] to study the performance of each model in small and medium data regimes. Unless otherwise stated, all trainings and tests are performed on 16×16 structured grids. We plot the results in the left of Figure 4, with further error comparisons provided in Table 1 of Appendix B.

As shown in the left of Figure 4, INOs outperform GNOs in both small and medium data regimes. When $N^{\text{train}} = 100$, INOs present 1.2% test error while GNOs have 2.0% test error. When $N^{\text{train}} = 10$, INOs have 6.1% test error, which also outperforms GNOs (with 7.7% test error) by 20%. This is possibly due to the fact that INOs have only 4 arguments in its kernel function κ, while GNOs have 6. Hence, INOs are less likely to overfit with small and medium data. Surprisingly, the data-augmentation strategy did not help much, and the aug-GNOs show a similar accuracy to the original GNOs. On the other hand, when comparing the performance between INOs and norm-INOs, we notice that norm-INOs achieves the best performance in the small data regime, possibly due to the fact that it further reduces the number of kernel arguments to 3. However, when we increase N^{train} to 100, the performance of norm-INOs deteriorates due to the lack of flexibility. Hence, in this paper we focus on INOs with a 4-argument kernel, since it outperforms GNOs in both small and medium data regimes, and has better expressivity than the norm-INOs in the medium data regime.

Comparison with more baselines. We now present the comparison with other baselines methods by comparing the test errors of INO-scalar with GNO, FNO, MWT, PINO, and EGNN. To obtain a fair comparison, the numbers of trainable parameters for all models are within the range of $[4.2M, 5.0M]$ (see Table 2 in Appendix B for further details). As shown in the left of Figure 4 and Table 1, our proposed INOs have achieved the best accuracy in both small ($N^{\text{train}} = 10$) and medium ($N^{\text{train}} = 100$)

[1] Since the GNO model could not finish its training in the "aug-GNO, $N^{\text{train}} = 100$" case within 72 hours, we only report the results from $N^{\text{train}} = 10$ for this setting.

data regimes. Here, we note that in the PINO architecture, we follow Li et al. (2021) and add the error associated with the PDE governing equation as a penalization term in the loss. The penalty parameter was tuned together with other hyperparameters, so as to optimize the validation error. With this test, we aim to compare the efficacy of INOs with another popular physics-informed approach, where full physics knowledge is infused via a soft constraint (when the governing equation is known). As can be seen from the results, INO still outperforms PINO even in the small data regime, where the best PINO presents 9.7% error. Generally, when the PDE loss has a large weight, PINO gets closer to PINN and suffers from slow convergence. On the other hand, our INOs embed the physics knowledge in the architecture instead of a soft constraint, and do not suffer from the challenges in optimization.

Performance with the change of N^{train} and resolutions. Next, we further compare our INOs with two popular baseline neural operators, FNOs and GNOs, for $N^{\text{train}} = \{5, 10, 20, 40, 100\}$. For the purpose of testing generalization properties with respect to resolution, we train all models on 16×16 structured grids, and consider two testing datasets: the "original-resolution" dataset with 16×16 grids, and a "fine-resolution" dataset with 31×31 grids. In the middle plot of Figure 4, we report the training and test errors on both original- and fine-resolution datasets for each model, with further visual comparison on an exemplar test instance provided in Figure 7 of Appendix B. One can observe that INOs consistently out-performe GNOs and FNOs. Among all three neural operators, in the small data regime FNOs have the smallest training error and the largest test error, indicating that they suffer more from overfitting. This observation is also consistent with the findings in You et al. (2022a,c). When comparing the test results across two resolutions, we observe that test errors at different resolutions remain on a similar scale for all three methods. This fact again verifies the capability of neural operators in handling different resolutions. On the contrary, the EGNN model learns a vector-vector mapping rather than a function-function mapping, which makes it not resolution-invariant: testing EGNN on 31×31 grid samples yields 492% error.

Translation and rotation tests. Lastly, we verify the capability of INOs in handling translated and rotated samples. In the translated test dataset, we translate the reference frame of each test sample by shifting it onto a new domain, $\tilde{\Omega} \in [C_x, 1+C_x] \times [C_y, 1+C_y]$. Here the movement of domain is randomly generated as $C_x, C_y \sim \mathcal{U}[-C, C]$ and the constant C defines the translation range. Similarly, in the rotated dataset we randomly generate $C_\theta \sim [0, C]$ for each test sample, and rotate the reference frame by C_θ counterclockwisely. The test errors of GNOs and INOs are reported in the right plot of Figure 4, along with results from GNOs with data-augmentation, where N^{aug} represents the total number of augmented training samples in addition to N^{train}. Perhaps unsurprisingly, while INOs exhibit invariant performance on translated and rotated datasets, the performance of GNOs deteriorates as the range of translation/rotation, C, increases. When comparing the results from original GNOs and aug-GNOs, we notice that the data-augmentation trick is generally helpful in handling transformations, although it increases the training dataset size and requires longer training time. Similar observation holds for FNOs: in Figure 6 of Appendix B we plot the test results of INO, GNO, and FNO on an exemplar instance with translated and rotated frame coordinates, respectively, where one can see that the INO solutions are invariant while the solutions from GNO and FNO are not.

4.2 Synthetic dataset: glass-ceramics deformation

In this example, we study material deformation in a glass-ceramic specimen as a prototypical exemplar on the heterogeneous material response prediction. A glass-ceramic material is the product of controlled crystallization of a specialized glass composition, which results in a microstructure of one or more crystalline phases within the residual amorphous glass (Prakash et al., 2022; Serbena & Zanotto, 2012). We consider an idealized microstructural realization on a circular domain with radius$= 0.4$, which is subject to displacement-type loading on its boundary. This microstructure realization is composed of randomly distributed crystals embedded in a glassy matrix, such that the crystals occupy 40% of the volume. To generate the training and test samples, we adopted the mechanical parameters in Fan et al. (2022) and employed the quasi-static linear peridynamic solid (LPS) solver to generate the high-fidelity simulation data. We train $N^{\text{train}} = \{10, 40, 100\}$ numbers of labelled data pairs, while validate/test the model with 40/40 samples, respectively. In this example, the specimen deformation is driven by the loading on its boundary, and hence our goal is to learn the mapping from $\boldsymbol{f}(x) := [\boldsymbol{u}_{BC}(x), \boldsymbol{a}(x)]$ to $\boldsymbol{u}(x)$, where $\boldsymbol{u}_{BC}(x)$ stands for (padded) Dirichlet-type boundary condition, and $\boldsymbol{a}(x)$ provides the microstructure information such that $\boldsymbol{a}(x) = 0$ if the material point x is glass and $\boldsymbol{a}(x) = 0$ if x is crystal. The INO-vector architecture is employed. Here, we emphasize that the domain is no longer structured and hence FNOs and MWT are not applicable. Therefore, in this example we compare the performances of GNOs and INOs.

In the left plot of Figure 5, we report our experimental results on $N^{\text{train}} = \{10, 40, 100\}$ samples. The proposed INOs obtain the lowest relative errors on the test dataset compared to GNOs. Furthermore, in the right plot of Figure 5, we study the performance of both neural operators on translated and rotated test samples. One can see that the error from INOs is invariant, whereas GNOs errors increase with the increase of the translation and rotation ranges.

Dataset	Glass-Ceramics			Biological Tissue	
N^{train}	10	40	100	10	100
GNO, train	9.48%	4.09%	1.58%	9.63%	6.76%
GNO, test	31.25%	10.16%	8.50%	38.36%	14.15%
INO, train	4.43%	7.20%	7.28%	4.50%	3.65%
INO, test	**12.65%**	**8.19%**	**7.94%**	**17.37%**	**6.38%**
FNO, train	-	-	-	8.49%	3.95%
FNO, test	-	-	-	36.73%	8.49%
MWT, train	-	-	-	12.39%	2.24%
MWT, test	-	-	-	38.84%	7.57%

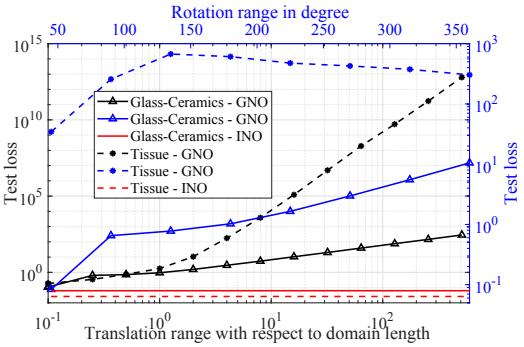

Figure 5: Results for cases with vector-valued output function. Left: Training and test errors in small and medium data regimes, where bold numbers highlight the best method for each case. Right: Generalization to test samples with translated and rotated reference frames.

4.3 Real-world dataset: biological tissue deformation

We now take one step further to demonstrate the performance of our method on a real-world physical response dataset not generated by solving PDEs. We consider learning the mechanical response of multiple biological tissue specimens from DIC displacement tracking measurements (You et al., 2022b). In this example the constitutive equations and material microstructure are both unknown, and the dataset has unavoidable measurement noise. In this task, we aim to model the tissue response by learning a neural operator mapping the boundary displacement loading to the interior displacement field. We train with $N^{\text{train}} = \{10, 100\}$ numbers of samples, validate with 40 samples, and test the learned model with 4500 samples. Since there is no known governing PDE, the PINO architecture is not applicable. Hence, we compare INOs with other three neural operator baselines (GNOs, FNOs, and MWTs). We note that experimental measurements are generally not provided on Cartesian grids. To test FNOs and MWTs, we apply a cubic spline interpolation to the displacement field, to obtain measurements on a structured grid. The results are provided in Figure 5. As one can see, INOs again perform the best. Interestingly, compared with GNOs, in this example INOs halve the test errors not only in the small data regime ($N^{\text{train}} = 10$), but also in the medium data regime ($N^{\text{train}} = 100$). This is possibly due to the fact that the experimental measurement noise makes the learning with a small number of samples more challenging. Therefore, this example validates the robustness of our INOs not only in a small data regime, but also on noisy real-world datasets.

5 CONCLUSION

We proposed INO to learn complex physical systems with guaranteed momentums conservation. The key is to design the network architecture that preserves the translational and rotational invariance/equivariance properties. Our approach finds a physical interpretation from a particle-based method, and only requires observed data pairs with minimal physical assumptions. We demonstrate with both synthetic and real-world datasets the expressivity and generalizability of the proposed INOs, and show that the guaranteed momentum conservation improves the learning efficacy, especially in small and noisy data regimes. INO is not only generalizable in handling translated and rotated datasets, but also provides improved prediction from the baseline neural operator models. Herein, we point out that our INOs represent the first conservation law guaranteed neural operator architecture, and we believe that it is a novel and promising framework applicable to many examples in physical system learning.

Acknowledgments

The authors would like to thank the reviewers for their careful reading and valuable suggestions that help improve the quality of the paper.

Y. Yu, H. You, and N. Tatikola 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.

References

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.

Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

Besnard, G., Hild, F., and Roux, S. "finite-element" displacement fields analysis from digital images: applica-

tion to portevin–le châtelier bands. *Experimental mechanics*, 46(6):789–803, 2006.

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

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

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.

Cammarano, A., Petrioli, C., and Spenza, D. Pro-energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pp. 75–83. IEEE, 2012.

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.

Chirikjian, G. S. *Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups*. CRC press, 2000.

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

Cohen, T. S. and Welling, M. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016b.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Fan, Y., You, H., Tian, X., Yang, X., Li, X., Prakash, N., and Yu, Y. A meshfree peridynamic model for brittle fracture in randomly heterogeneous materials. *arXiv preprint arXiv:2202.06578*, 2022.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

Fumagalli, A. and Scotti, A. Numerical modelling of multiphase subsurface flow in the presence of fractures. *Communications in Applied and Industrial Mathematics*, 3 (1):2038–0909, 2011.

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. *arXiv preprint arXiv:2207.05748*, 2022a.

Goswami, S., Yin, M., Yu, Y., and Karniadakis, G. E. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022b.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34:24048–24062, 2021a.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021b. URL https://openreview.net/forum?id=LZDiWaC9CGL.

Haber, E., Ruthotto, L., Holtham, E., and Jun, S.-H. Learning across scales—multiscale methods for convolution neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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.

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.

Karplus, M. and McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nature structural biology*, 9(9):646–652, 2002.

Kim, M., Winovich, N., Lin, G., and Jeong, W. Peri-net: Analysis of crack patterns using deep neural networks. *Journal of Peridynamics and Nonlocal Modeling*, 1(2): 131–142, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Knapp, A. W. Representation theory of semisimple groups: an overview based on examples. 2001.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Lang, L. and Weiler, M. A wigner-eckart theorem for group equivariant convolution kernels. *arXiv preprint arXiv:2010.10952*, 2020.

LeVeque, R. J. *Numerical methods for conservation laws*, volume 214. Springer, 1992.

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, 2020b.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A., et al. 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. *arXiv preprint arXiv:2111.03794*, 2021.

Liu, M. and Liu, G. Smoothed particle hydrodynamics (sph): an overview and recent developments. *Archives of Computational Methods in Engineering*, 1(17):25–76, 2010.

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, 2021a.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *arXiv preprint arXiv:2111.05512*, 2021b.

Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*, 2020.

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.

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.

Prakash, N., Deng, B., Stewart, R. J., Smith, C. M., and Harris, J. T. Investigation of microscale fracture mechanisms in glass-ceramics using peridynamics simulations. *Journal of American Ceramic Society*, 2022.

Quiroga, F., Ronchetti, F., Lanzarini, L., and Bariviera, A. F. Revisiting data augmentation for rotational invariance in convolutional neural networks. In *International conference on modelling and simulation in management sciences*, pp. 127–141. Springer, 2018.

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

Rezende, D. J., Racanière, S., Higgins, I., and Toth, P. Equivariant hamiltonian flows. *arXiv preprint arXiv:1909.13739*, 2019.

Romero, D. W. and Cordonnier, J.-B. Group equivariant stand-alone self-attention for vision. In *International Conference on Learning Representations*, 2020.

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

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8 (1):1–8, 2017.

Serbena, F. C. and Zanotto, E. D. Internal residual stresses in glass-ceramics: A review. *Journal of Non-Crystalline Solids*, 358(6-7):975–984, 2012.

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

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.

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

Yin, M., Ban, E., Rego, B. V., Zhang, E., Cavinato, C., Humphrey, J. D., and Em Karniadakis, G. Simulating progressive intramural damage leading to aortic dissection using deeponet: an operator–regression neural network. *Journal of the Royal Society Interface*, 19(187): 20210670, 2022a.

Yin, M., Zhang, E., Yu, Y., and Karniadakis, G. E. Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, 2022b.

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

You, H., Zhang, Q., Ross, C. J., Lee, C.-H., Hsu, M.-C., and Yu, Y. A physics-guided neural operator learning approach to model biological tissues from digital image correlation measurements. *arXiv preprint arXiv:2204.00205*, 2022b.

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. *arXiv preprint arXiv:2203.08205*, 2022c.

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.

Supplementary Material: INO: Invariant Neural Operators for Learning Complex Physical Systems with Momentum Conservation

A DETAILED DERIVATIONS

A.1 Proof for Theorem 1

Herein, we provide the proof for the translation- and rotation-invariant properties of INO-scalar.

Proof. Notice that $|\tilde{\boldsymbol{f}}(Rx+g)| = |R\boldsymbol{f}(x)| = |\boldsymbol{f}(x)|$, hence for the lifting layer (7) we have

$$\tilde{\boldsymbol{h}}(Rx+g,0) = \mathcal{P}[\tilde{\boldsymbol{f}}](Rx+g) = P|\tilde{\boldsymbol{f}}(Rx+g)| + p = P|\boldsymbol{f}(x)| + p = \boldsymbol{h}(x,0).$$

Substituting it into the iterative layer, we can prove by induction that the j-th layer feature function is invariant, i.e., $\tilde{\boldsymbol{h}}(Rx+g,j\tau) = \boldsymbol{h}(x,j\tau)$, $j = 0,\cdots,L$. Since the derivation for the lifting layer has already provided the base case, i.e., for $j=0$, it suffices to prove the induction step. Since $\overline{y-x}$ is frame-invariant, we have

$$\tilde{\boldsymbol{m}}(Rx+g,Ry+g) = \kappa(\overline{y-x}, \left|\tilde{\boldsymbol{f}}(Rx+g)\right|, \left|\tilde{\boldsymbol{f}}(Ry+g)\right|) = \kappa(\overline{y-x}, |\boldsymbol{f}(x)|, |\boldsymbol{f}(y)|) = \boldsymbol{m}(x,y)$$

and hence given that $\tilde{\boldsymbol{h}}(Rx+g,j\tau) = \boldsymbol{h}(x,j\tau)$, for the $(j+1)$-th layer we have

$$\begin{aligned}\tilde{\boldsymbol{h}}(Rx+g,(j+1)\tau) :=& \tilde{\boldsymbol{h}}(Rx+g,j\tau) + \tau\sigma\left(W\tilde{\boldsymbol{h}}(Rx+g,j\tau) + \int_\Omega \tilde{\boldsymbol{m}}(Rx+g,Ry+g)\tilde{\boldsymbol{h}}(Ry+g,j\tau)dy + c\right)\\ =& \boldsymbol{h}(x,j\tau) + \tau\sigma\left(W\boldsymbol{h}(y,j\tau) + R\int_\Omega \boldsymbol{m}(x,y)\boldsymbol{h}(x,j\tau)dy + c\right) = \boldsymbol{h}(x,j\tau),\end{aligned}$$

where $\tilde{\Omega} := \{Rx+g : x\in\Omega\}$, $j = 0,\cdots,L-1$. Similarly, for the projection layer we obtain

$$\tilde{\mathcal{G}}[\tilde{\boldsymbol{f}}](Rx+g) = Q_2\sigma(Q_1\tilde{\boldsymbol{h}}(Rx+g,L\tau)+q_1)+q_2 = Q_2\sigma(Q_1\boldsymbol{h}(x,L\tau)+q_1)+q_2 = \mathcal{G}[\boldsymbol{f}](x).$$

□

A.2 Proof for Theorem 2

We now provide the detailed proof for the translation-invariant and rotation-equivariant properties for INO-vector.

Proof. Following the proof of Theorem 1, we have

$$\tilde{\boldsymbol{h}}(Rx+g,j\tau) = \boldsymbol{h}(x,j\tau),\ j=0,\cdots,L,$$
$$\tilde{\boldsymbol{m}}(Rx+g,Ry+g) = \boldsymbol{m}(x,y).$$

Then, for the additional coordinate function, $\boldsymbol{x}(x,j\tau)$, we have

$$\tilde{\boldsymbol{x}}(Rx+g,0) = Rx+g = R\boldsymbol{x}(x,0)+g.$$

Moreover, assume that $\tilde{\boldsymbol{x}}(Rx+g,j\tau) = R\boldsymbol{x}(x,j\tau)+g$, for the $(j+1)$-th layer we have

$$\begin{aligned}\tilde{\boldsymbol{x}}(Rx+g,(j+1)\tau) :=& \tilde{\boldsymbol{x}}(Rx+g,j\tau) + \tau\int_\Omega (Rx+g-Ry-g)\phi(\tilde{\boldsymbol{m}}(Rx+g,Ry+g)\tilde{\boldsymbol{h}}(Ry+g,j\tau);w)dy\\ =& R\boldsymbol{x}(x,j\tau)+g+\tau\int_\Omega R(x-y)\phi(\boldsymbol{m}(x,y)\boldsymbol{h}(y,j\tau);w)dy\\ =& R\left(\boldsymbol{x}(x,j\tau)+\tau\int_\Omega (x-y)\phi(\boldsymbol{m}(x,y)\boldsymbol{h}(y,j\tau);w)dy\right)+g = R\boldsymbol{x}(x,(j+1)\tau)+g.\end{aligned}$$

Therefore, we obtain $\tilde{\boldsymbol{x}}(Rx+g, j\tau) = R\boldsymbol{x}(x, j\tau) + g$ for $j = 0, \cdots, L$. Finally, we have

$$\tilde{\mathcal{G}}[\tilde{\boldsymbol{f}}](Rx+g) = \tilde{\boldsymbol{x}}(Rx+g, L\tau) - (Rx+g) = R\left(\boldsymbol{x}(x, L\tau) - x\right) = R\mathcal{G}[\boldsymbol{f}](x).$$

□

A.3 A translation- and rotation-equivariant architecture

Besides the translation-invariant and rotation-equivariant problem we have discussed in the main text, for some physical problems a translation- and rotation-equivariant INO-vector architecture would be desired. For instance, in particle-tracking problems, the current location of each molecular is of interest, which would be translation- and rotation-equivariant with the change of reference frames. In this case, one can employ the same lifting and iterative block architectures as in (10)-(14), and modify the projection block as

$$\boldsymbol{u}(x) = \mathcal{Q}[\boldsymbol{x}(x, L\tau)](x) := \boldsymbol{x}(x, L\tau). \tag{17}$$

For this architecture we have the following theorem:

Theorem 3 (Equivariance for INO-vector)**.** *The INO-vector architecture proposed in Eqs.* (10)-(14) *and* (17) *is translation- and rotation-equivariant. That means, when translating the reference frame by $g \in \mathbb{R}^d$ and rotating it by an orthogonal matrix $R \in \mathbb{R}^{d\times d}$, the following property holds true:*

$$\tilde{\mathcal{G}}[\tilde{\boldsymbol{f}}; \theta](Rx+g) = R\tilde{\mathcal{G}}[\boldsymbol{f}; \theta](x) + g\,,$$

where $\tilde{\boldsymbol{f}}(Rx+g) := R\boldsymbol{f}(x)$.

Proof. The proof can be trivially obtained following the same argument as in Theorem 2. □

B PROBLEM DEFINITIONS AND NETWORK SETTINGS

We note that, in order to demonstrate the full expressivity of INO and carry out fair comparisons with GNO, MWT, and FNO, all the involved integral operators are evaluated over the entire domain. However, in practical applications, one can strike a balance between computational time and accuracy by evaluating the integral over a ball of a smaller radius. Moreover, to prevent overfitting, an early stopping scheme is employed: the training process is stopped if the validation loss does not drop in 60 epochs, and validation is performed only when the train loss is improved over the last saved model. For fair comparison, the hyperparameters are tuned for each method, including the learning rates (tuned in the range $[1e-2, 1e-4]$), decay rates (tuned out of $\{0.5, 0.7, 0.9\}$), and regularization coefficients (tuned in the range $[1e-2, 1e-5]$). In PINOs, we further tune the penalty coefficient for the PDE loss in the range $[0.1, 10]$. All the models are trained up to a total of 2000 epochs, and the learning rate is decayed at the interval of every 50 epochs. In all the examples, the dimension of representation is set to $d_h = 64$, and the kernel κ is modeled by a 3-layer MLP with width $(n, 512, 1024, d_h^2)$, where n is the number of kernel arguments for each architecture. Finally, the output $\boldsymbol{u}(x) \in \mathbb{R}^{d_u}$ at each point x is obtained via a projection layer in the form of a 2-layer MLP with width $(d_h, 2d_h, d_u)$.

B.1 Synthetic dataset: sub-surface flow

The detailed numerical results for ablation study and empirical experiments (displayed in Figure 4) are provided in Table 1. The number of trainable parameters are listed in Table 2. Furthermore, a demonstration of the predictions of INO, GNO, and FNO on randomly translated and rotated datasets is displayed in Figure 6. In order to demonstrate the resolution independence nature of INO, an example is illustrated in 7 where INO is trained on 16×16 grids and predicted directly on 31×31 and 61×61 grids.

Model, Dataset, Resolution	Number of training samples			
	10	20	40	100
FNO, train, 16×16	0.153%	0.715%	0.906%	0.216%
FNO, test, 16×16	14.29%	8.582%	5.235%	2.510%
FNO, test, 31×31	14.75%	10.69%	9.072%	8.244%
GNO, train, 16×16	2.117%	4.730%	2.002%	0.872%
GNO, test, 16×16	7.708%	6.536%	4.389%	1.966%
GNO, test, 31×31	8.167%	7.659%	7.579%	5.842%
INO, train, 16×16	1.414%	1.580%	1.068%	0.500%
INO, test, 16×16	6.145%	**4.922%**	**3.110%**	**1.182%**
INO, test, 31×31	**6.900%**	**6.345%**	**4.693%**	**4.167%**
norm-INO, train, 16×16	1.008%	-	-	0.627%
norm-INO, test, 16×16	**4.598%**	-	-	1.273%
PINO, train, 16×16	3.105%	-	-	0.468%
PINO, test, 16×16	9.652%	-	-	2.560%
aug-GNO N^{aug}=20, train, 16×16	10.036%	-	-	-
aug-GNO N^{aug}=20, test, 16×16	18.050%	-	-	-
aug-GNO N^{aug}=40, train, 16×16	1.822%	-	-	-
aug-GNO N^{aug}=40, test, 16×16	8.850%	-	-	-
MWT, train, 16×16	1.472%	1.212%	0.094%	0.044%
MWT, test, 16×16	9.554%	7.370%	5.558%	3.436%
EGNN, train, 16×16	16.56%	-	-	7.710%
EGNN, test, 16×16	19.67%	-	-	9.550%

Table 1: Results for the sub-surface flow problem, where bold numbers highlight the best method for the test datasets on original-resolution (16×16 grids) and refined-resolution (31×31 grids), respectively.

model	INO	GNO	FNO	PINO	MWT	EGNN
nparams	4,739,201	4,740,353	4,219,841	4,219,841	4,565,185	4,677,156

Table 2: Total number of parameters of each model for the sub-surface flow problem.

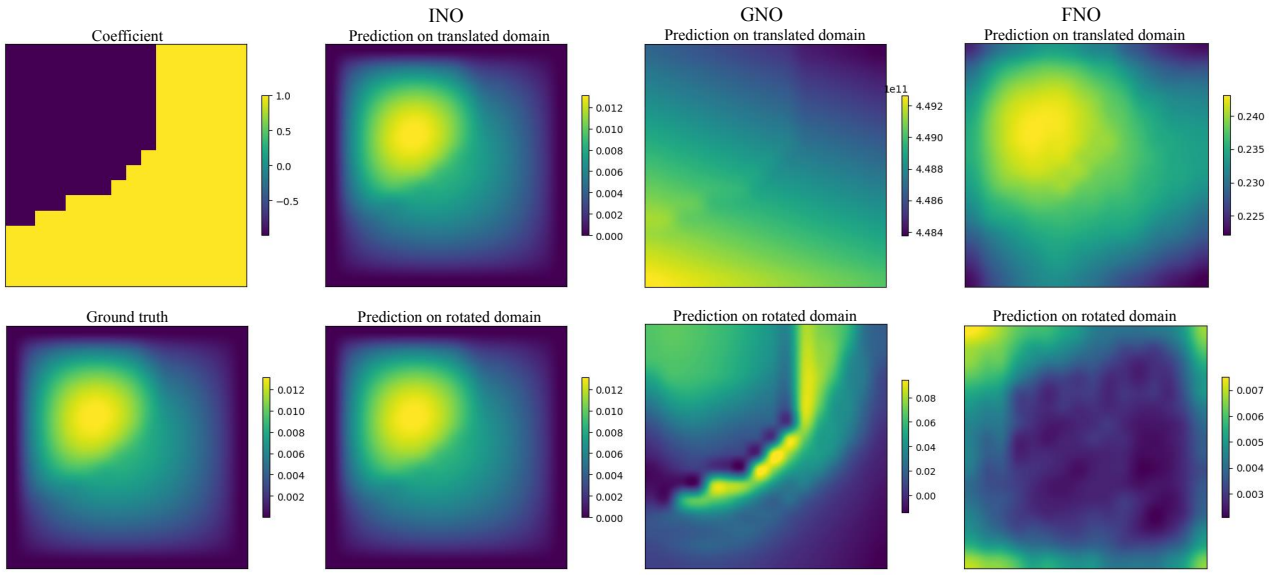

Figure 6: Demonstration of the predictions of INO, GNO, and FNO on randomly translated and rotated domains.

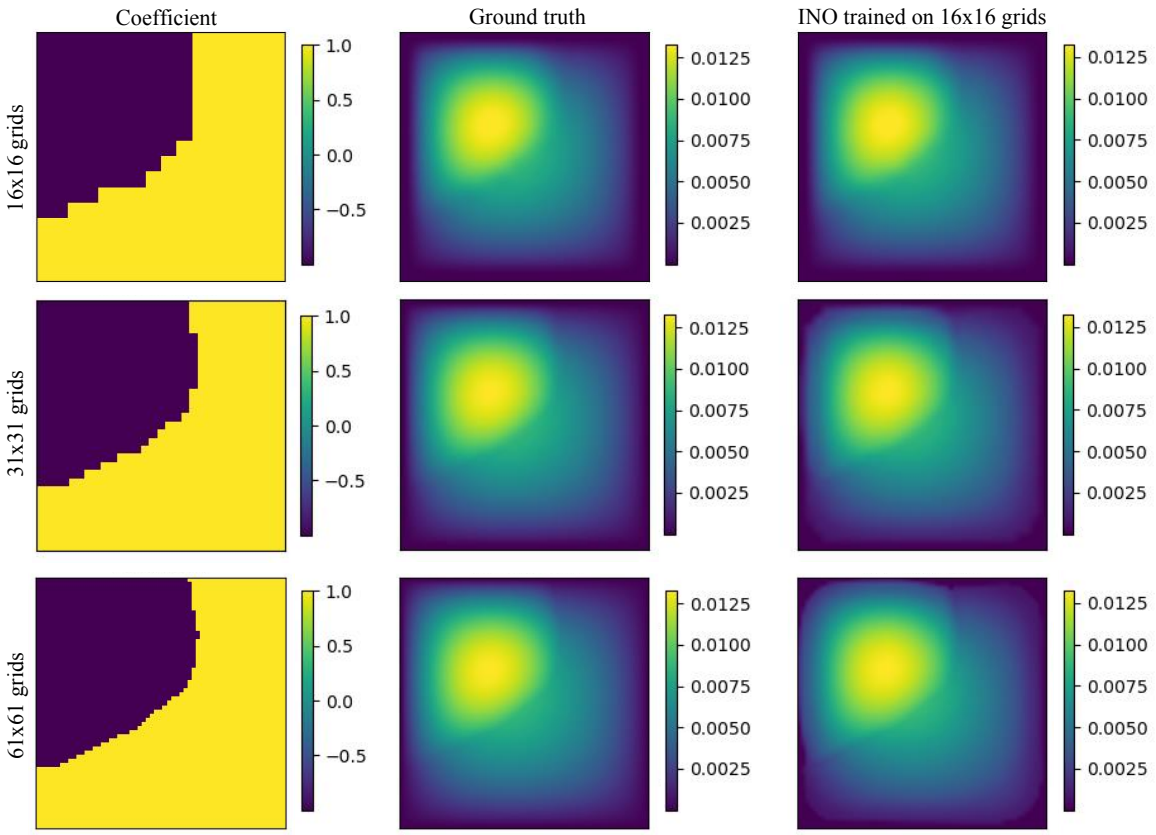

Figure 7: Demonstration of the cross-resolution predictability of INO trained on 16×16 grids and predicted directly on 31×31 and 61×61 grids.

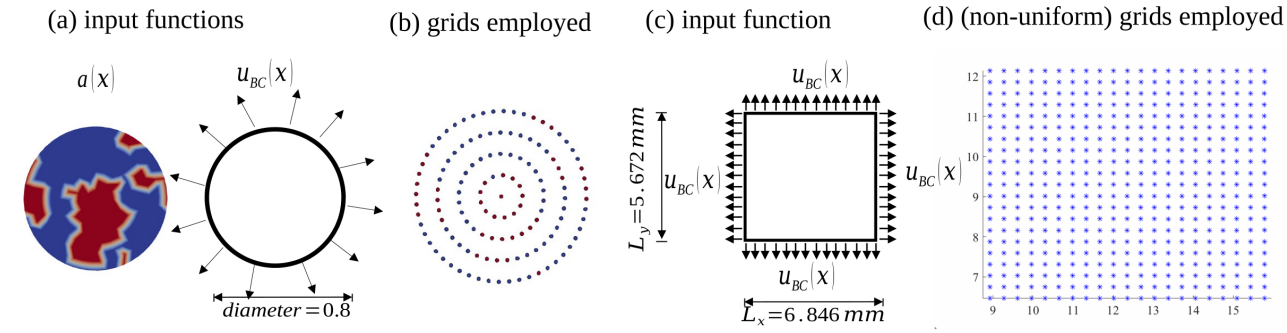

Figure 8: Input functions and (non-uniform) grids employed in examples 2 and 3.

B.2 Synthetic dataset: glass-ceramics deformation

Following Fan et al. (2022), we model the glass-ceramic material deformation with the linear peridynamic solid (LPS) model:

$$\begin{aligned}\mathcal{L}_\delta \boldsymbol{u} :=& -\frac{1}{m(\delta)} \int_{B_\delta(x)} (\lambda(x,y) - \mu(x,y))\, K(|y-x|)\, (y-x)\, (d(x)+d(y))\, dy \\ & -\frac{8}{m(\delta)} \int_{B_\delta(x)} \mu(x,y) K(|y-x|) \frac{(y-x)\otimes(y-x)}{|y-x|^2} (\boldsymbol{u}(y) - \boldsymbol{u}(x))\, dy = 0\,, \quad \text{for } x \in \Omega\,, \\ \boldsymbol{u}(x) :=& \boldsymbol{u}_{BC}(x)\,, \quad \text{for } x \in \mathcal{BB}\Omega\,,\end{aligned} \tag{18}$$

where the nonlocal dilatation $d(x)$ is defined as

$$d(x) := \frac{1}{m(\delta)} \int_{B_\delta(x)} K(|y-x|)(y-x) \cdot (\boldsymbol{u}(y) - \boldsymbol{u}(x))\, dy, \text{ for } x \in \Omega \cup \mathcal{B}\Omega\,. \tag{19}$$

Here, $\mathcal{B}\Omega$ and $\mathcal{B}\mathcal{B}\Omega$ are the one-layer and two-layer nonlocal boundaries, respectively, which will be defined later. The kernel function K and nonlocal volume $m(\delta)$ are defined as

$$K(|y-x|) = \frac{1}{|y-x|}\,, \quad m(\delta) := \int_{B_\delta(\mathbf{0})} K(|y|)|y|^2 dy = \frac{2\pi\delta^3}{3}\,. \tag{20}$$

$\mu(x,y)$, $\lambda(x,y)$ are the averaged shear modulus and Lamé first parameters of material points x and y.

To model the glass-ceramic specimen, we consider a circular domain with the same material parameters as provided in Fan et al. (2022). The microstructure realization is randomly generated such that the volume ratio between crystal and glass is around 2:3. To generate the displacement under different boundary conditions, we consider the horizon size $\delta = 0.3$, the nonlocal domain as $\Omega = \{x : |x| \leq 0.4\}$, and the nonlocal boundary regions as $\mathcal{B}\Omega = \{x : 0.4 < |x| \leq 0.7\}, \mathcal{B}\mathcal{B}\Omega = \{x : 0.4 < |x| \leq 1.0\}$. For each sample, the nonlocal boundary condition $\boldsymbol{u}_{BC}(x)$ is generated from a Gaussian random field with $\mathbf{G}(x) = Re\left(\begin{bmatrix} \sum_{k_1,k_2} \xi_k \exp(i2\pi k \cdot x/D)\mathcal{U}[0,1] \\ \sum_{k_1,k_2} \xi_k \exp(i2\pi k \cdot x/D)\mathcal{U}[0,1] \end{bmatrix} \right)$, where $k = (k_1, k_2)$, $D = 2.8$, $\xi_k = (k_1^2 + k_2^2)^{-5/4}$, $\mathcal{U}[0,1]$ represents the uniform distribution on $(0,1)$, and $k_1, k_2 \in \{-15, -14, \ldots, 13\}$. The Gaussian random field $\mathbf{G}(x)$ is created on a square domain $[-1.4, 1.4] \times [-1.4, 1.4]$ with structured grids, and the nonlocal boundary condition $\boldsymbol{u}_{BC}$ is obtained by cubic interpolation on unstructured grids within the region $\mathcal{B}\mathcal{B}\Omega$. The corresponding high-fidelity displacement field $\boldsymbol{u}(x)$ is generated employing the meshfree solver as described in Fan et al. (2022). Here, we point out that in this example the domain shape and the grids are not structured, hence the FNO and MWT models are not applicable and we focus on the comparison between GNOs and INOs.

B.3 Real-world dataset: biological tissue deformation

We briefly introduce the experimented specimen and the corresponding biological tissue dataset. In this example, we employ the opensource dataset provided in `https://github.com/fishmoon1234/IFNO-tissue`, where a tricuspid valve anterior leaflet tissue with an effective testing area of 9×9 mm was mounted on a biaxial testing device. Then, various loadings were applied on the boundary of this tissue specimen, and the corresponding displacement was recorded for each material point by employing the digital image correlation (DIC) toolkit of the BioTester's software. The unstructured coordinate locations of the DIC-tracked grids were directly used to train the proposed INO and reference GNO models. To create a structured grid for FNOs and MWTs, we further applied a cubic spline interpolation to the displacement field on a structured 21×21 node grid. The dataset consists of a total of $26,523$ time instants (samples), which is further divided into $22,023$ for training and $4,500$ for testing. In our example, we focus on the small training data regime, by randomly select 10 or 100 numbers of samples for the purpose of training, then validate the model on all $4,500$ test samples and provide the averaged relative error for displacement fields.