Reference Neural Operators: Learning the Smooth Dependence of Solutions of PDEs on Geometric Deformations

Anonymous Authors[1]

Abstract

For partial differential equations on domains of arbitrary shapes, existing works of neural operators attempt to learn a mapping from geometries to solutions. It often requires a large dataset of geometry-solution pairs in order to obtain a sufficiently accurate neural operator. However, for many industrial applications, e.g., engineering design optimization, it can be prohibitive to satisfy the requirement since even a single simulation may take hours or days of computation. To address this issue, we propose *reference neural operators* (RNO), a novel way of implementing neural operators, i.e., to learn the smooth dependence of solutions on geometric deformations. Specifically, given a reference solution, RNO can predict solutions corresponding to arbitrary deformations of the referred geometry. This approach turns out to be much more data efficient. Through extensive experiments, we show that RNO can learn the dependence across various types and different numbers of geometry objects with relatively small datasets. RNO outperforms baseline models in accuracy by a large lead and achieves up to 80% error reduction.

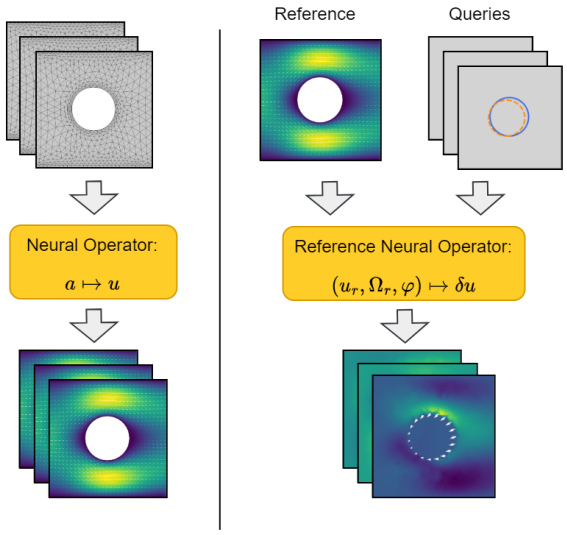

Figure 1. Comparison between two approaches. (Left) Neural operators map directly from geometries/functions a to solutions u, which often requires large amount of data to cover various geometries. (Right) Alternatively, given a reference solution u_r on Ω_r, we hope to query solutions to various deformations of its geometry. Let $\varphi : \Omega_r \mapsto \Omega_q$ be a smooth deformation, a reference neural operator can predict the difference between the queried solution and the pushforward of the reference solution $\delta u = u_q - u_r \circ \varphi^{-1}$.

1. Introduction

Recently, neural operators (Lu et al., 2021; Li et al., 2020a; Bhattacharya et al., 2021; Nelsen & Stuart, 2021; Patel et al., 2021) as surrogate models for solving partial differential equations (PDEs) have rapidly gained attention due to the success in many applications in physics simulation, e.g., weather forecasting, fluid dynamics, etc. (Wang et al., 2023; Bi et al., 2023; Zhang et al., 2023b; Pathak et al., 2022). One of the greatest advantages of neural operators is fast inference speed, which may reduce computational cost by orders of magnitude while maintaining good accuracy. These methods deal with problems on *fixed* domains, e.g., Earth. However, for engineering design, e.g., sensitivity analysis, optimization and robustness study, iterations of solving PDEs on *deformable* domains are required. Existing works (Shukla et al., 2023; Li et al., 2022a; 2023; Hao et al., 2023) of neural operators on deformable domains attempt to learn a mapping from *geometry space to solution space* (**G-S**) of PDEs. Such methods require large amount of training data to achieve satisfying accuracy. Unlike some physics simulation problems in fixed domains, such as weather forecasting where climate data is available from research of meteorology, engineering design usually deals with individual cases and needs to simulate various designs to collect data. For industrial applications, it often takes hours or days to simulate a single design. Meanwhile, the geometric de-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

sign space can be huge, since geometric objects may be a composition of any number of intersection, union and difference. Therefore, learning the mapping from geometries to solutions becomes an extremely challenging and expensive task.

To address this problem, we think out of the box and ask a different question:

> *With a limited budget on training set, can neural operators learn the change of solution corresponding to the change of geometry?*

See Fig. 1 for illustration of the idea. Instead of predicting the solution of an arbitrary design as **G-S**, we ask, given a reference design what impact some geometric deformation will cause to the reference solution. Intuitively, similar changes of different geometry shapes can cause similar impact on the solution. For example, considering a flow through a channel with holes, to enlarge, shrink or shift a hole close to the inlet of the channel will cause similar effect to the flow, regardless of how many holes the channel have. In this paper, we propose a novel approach, *reference neural operators* (RNO) to predict the change of solution due to the change of geometry. **G-S** essentially interpolates sparse data in a vast space, and RNO focuses on estimation in the neighborhood of a reference solution. Based on extensive experiments, compared to **G-S**, RNO turns out to be more data efficient and exhibits superior learning efficacy.

Specifically, RNO takes a reference solution, a corresponding geometry and a deformation to a queried geometry as inputs and outputs its prediction on the difference between reference solution and query solution. In the study of shape optimization, the target difference is defined as material derivative (Sokolowski & Zolésio, 1992). It brings several challenges to us, e.g., how to describe a deformation from reference to query? How to combine multiple input functions on irregular meshes in neural network? Our main contributions include:

- We proposed RNO, a novel type of neural operators on irregular meshes. Given a reference solution (u_r, Ω_r), RNO can estimate the solution of PDEs on deformations of Ω_r.
- We provide a simple method of constructing a domain deformation φ based on the boundary of two domains.
- To handle multiple input functions and strengthen the signal from geometric deformation, we implemented distance-aware cross attention (DACA) in RNO. Since cross attention is known for quadratic complexity and may suffer from scalability issue, additionally, we propose an alternative linear-complexity DACA. To our best knowledge, this is a novel design of linear attention weighted by distance.
- We comprehensively benchmarked RNO in multiple challenging 2D and 3D PDE problems. Results show that even with small size datasets, RNO can learn the smooth dependence of solutions on geometric deformations and outperforms several baseline neural operators.

2. Related Works

Neural operators. Learning solution operators of PDEs using neural networks has gained growing attention. A revolutionary wave starts with DeepONet (Lu et al., 2021) and its followup works (Wang et al., 2021; 2022; Jin et al., 2022). DeepONet is a meshless model, and it can learn various maps from all sorts of functions, e.g., initial data, boundary conditions, coefficient functions, etc. to solutions.

Another principled architecture for neural operators is proposed by (Li et al., 2020b), and Fourier Neural Operator (FNO) (Li et al., 2020a) emerged as a successful special case. One of the key ideas is to formulate neural operators as kernel integrals. Similarly, (Cao, 2021) reinterpreted attention mechanism as kernel integrals and proposed to construct neural operators by transformers. However, all these works are not dedicated to dealing with varying shapes and geometries. In fact, FNO and its variations need to work on regular mesh grids due to the embedded Fast Fourier Transform in FNO layers.

Geo-FNO (Li et al., 2022a) is designed to handle varying shapes and it learns the operator map from geometries to solutions by a composition of deformation and FNO. The deformation map between regular domains to irregular ones should be either provided in advance or learned from data. GINO (Li et al., 2023) combines graph neural network with FNO and deals with varying geometry problems in 3D space. NUNO (Liu et al., 2023) generalizes FNO to irregular meshes by partitioning irregular domains into regular subdomains. GCNN (Chen et al., 2021) learns laminar flow around random 2D objects with graph convolutional neural networks. These works aim to enable generalization of neural operators on shapes and geometries, but such generalization heavily relies on the diversity and the quantity of training samples. For practical applications, where return on investment must be considered, these approaches may be too expensive to apply.

Similarly, (Li et al., 2022b; Hao et al., 2023) improved transformer-structured neural operators both on the versatility of input functions and on the performance. While transformer-based models can naturally handle irregular meshes, they still require enough data to learn the mapping from geometries to solutions.

In contrast to all previous methods, RNO aims to learn the change of solution caused by geometric deformation.

Hybrid methods. There exists a type of hybrid methods that combines ML with traditional numerical solvers. In contrast to pure neural operators that only forward pass once, these methods often require iterations. Neural networks work as a replacement of some expensive parts of iterations in numerical solvers. Some representative works include (Kochkov et al., 2021), an equation-specific method that replaces an expensive component inside computational fluid dynamics (CFD). (Tompson et al., 2017) and (Obiols-Sales et al., 2020) can generalize even to unseen geometries. They also requires being combined with CFD solvers, and the idea is to leverage ML models to accelerate convergence of iterations. (Kahana et al., 2023) implements a more subtle way of interweaving ML models with iterative numerical solvers and can generalize to unseen geometries through transfer learning.

However, all these hybrid methods are either problem specific or require tailored combination with numerical solvers. In contrast, RNO does not involve direct interactions with numerical solvers. For unseen geometries, it can refer to a numerically solved example and predict on various deformations.

Distance-aware transformers. Encoding position information in transformers is a vibrant topic (Dufter et al., 2022). For example, (Wu et al., 2020) proposed distance-aware transformer to re-scale attention weights according to distance between tokens. (Ying et al., 2021) successfully brought transformer to graph-structure problem by encoding structure information including distance between nodes. (Zhang et al., 2023a) theoretically proved that distance is a key ingredient to the expressive power of transformer based GNN models.

3. Method

3.1. Problem Formulation

Let $D \subset \mathbb{R}^n$ be a Lipschitz domain and $T \in \mathcal{T}$ be a signed distance function such that $\Omega_T = \{x \in D | T(x) > 0\}$. Depending on the problem we study, Ω_T can be either a domain with holes or an interior domain of D. Hereafter, Ω_T can stand for a domain, and when it is self-evident Ω_T can also stand for an element in the function space $\mathcal{T}$ to simplify notation. Consider the following PDE problem with Dirichlet boundary condition:

$$\mathcal{N}(u) = 0 \text{ in } \Omega_T \tag{1}$$

$$u = 0 \text{ on } \partial\Omega_T \tag{2}$$

where $\mathcal{N}$ is a differential operator and $\partial\Omega_T$ is the boundary of the domain. Suppose the problem has a unique solution $u_T \in \mathcal{U}_T$, and $\mathcal{U}_T$ is a Banach space of functions on Ω_T.

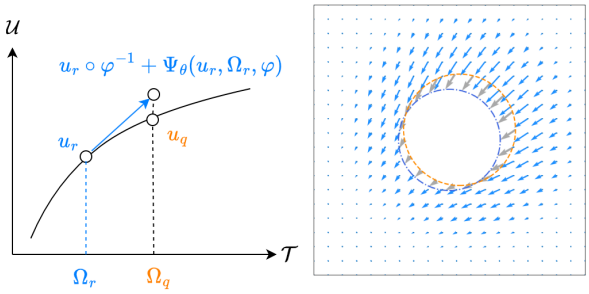

Figure 2. (Left) Given geometry space $\mathcal{T}$ and solution space $\mathcal{U}$, a reference neural operator Ψ_θ learns the material derivative of a solution operator $\mathcal{G} : \mathcal{T} \to \mathcal{U}$ at a given reference geometry Ω_r. It provides an approximation of the solution on a deformed geometry $\Omega_q = \varphi(\Omega_r)$ with $u_r \circ \varphi^{-1} + \Psi_\theta(u_r, \Omega_r, \varphi)$. (Right) The transform (gray vectors) from $\partial\Omega_q$ (orange circle) to $\partial\Omega_r$ (blue circle) is used to construct a vector field that represents a deformation $\varphi^{-1} : \Omega_q \mapsto \Omega_r$.

Also, let $\mathcal{U}$ be a Banach space of functions on D, and assume $\mathcal{U}_T \subseteq \mathcal{U}$ with a family of linear and bounded extension from $\mathcal{U}_T$ to $\mathcal{U}$. Then we can define a solution operator

$$\begin{aligned} G : \mathcal{T} &\to \mathcal{U}, \\ T &\mapsto u_T, \end{aligned} \tag{3}$$

Such operators have some nice properties such as smooth dependence on T within a small neighborhood in $\mathcal{T}$, given some suitable conditions. For elliptic, parabolic and hyperbolic problems, readers can find classic analysis in (Sokolowski & Zolésio, 1992). For Stokes equation and Navier-Stokes equation, shape holomorphy of solutions is analyzed by (Cohen et al., 2018).

Given a reference solution (u_r, Ω_r), we hope to learn about solution u_q on Ω_q, where Ω_q is a small perturbation of Ω_r, i.e., there exists a smooth deformation φ with smooth inverse, i.e., $\varphi, \varphi^{-1} \in C^k(D; D)$, $k \geq 1$, such that

$$\varphi : \Omega_r \mapsto \Omega_q. \tag{4}$$

See Fig. 2 (Left) for conceptual illustration, and the construction of φ will be discussed in Section 3.3. Formally, we define Ψ_θ as a neural operator parameterized by θ,

$$\begin{aligned} \Psi_\theta : \mathcal{U} \times \mathcal{T} \times C^k(\mathcal{T}) &\to \mathcal{U} \\ (u_r, \Omega_r, \varphi) &\mapsto u_q - u_r \circ \varphi^{-1}, \end{aligned} \tag{5}$$

and its training objective is

$$\min_\theta \mathbb{E}[\|\Psi_\theta(u_r, \Omega_r, \varphi) - (u_q - u_r \circ \varphi^{-1})\|], \tag{6}$$

where $\|\cdot\|$ is the norm on $\mathcal{U}$. In practice, the norm usually is chosen to be L^p-norm, $p > 0$, for computational simplicity. In Appendix D, we derive this objective from material derivative.

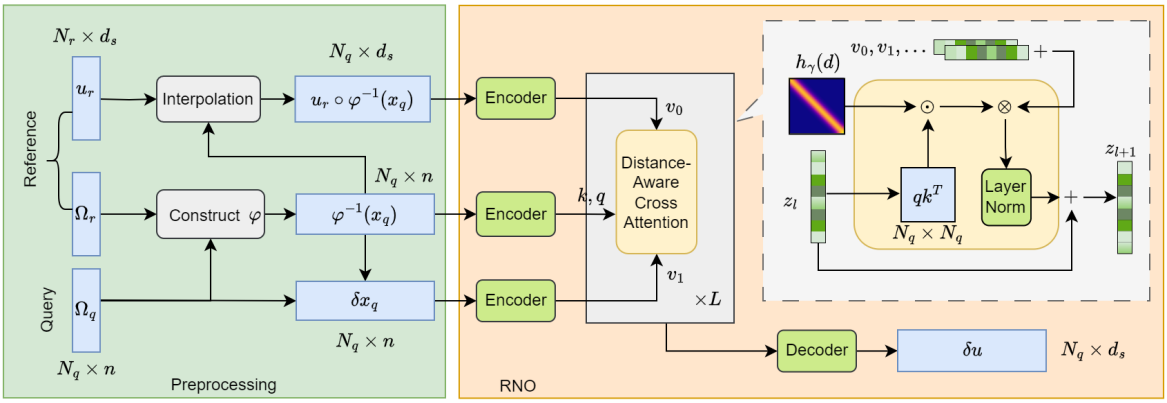

Figure 3. Overview of model architecture which composes of two stages. The first stage is the preprocessing of input data. Input sequences $u_r, \Omega_r = \{x_{r_j}\}_j$ and $\Omega_q = \{x_{q_i}\}_i$ are tensors with shape $N_r \times d_s$, $N_r \times n$, $N_q \times n$, where n, d_s are dimensions of input space and target space. $\delta x_q = \varphi^{-1}(x_q) - x_q$ is the shift of every point $x_q \in \Omega_q$. The second stage is the forward passing of neural network. $\odot$ is element-wise product, $\otimes$ is matrix product, $+$ is element-wise sum. δu is the predicted change of solution, and $\hat{u}_q = u_r \circ \varphi^{-1} + \delta u$.

Suppose every queried domain Ω_q has a set of points $\{x_{q_i}\}_i \subset \Omega_q$ $i = 1, \cdots, N_q$. For N queries, each one is paired with a reference triplet (u_r, Ω_r, φ). Then the objective function (6) is approximated by

$$\min_\theta \frac{1}{N} \sum_q \sum_i \|\Psi_\theta(u_r, \Omega_r, \varphi)(x_{q_i}) - (u_q - u_r \circ \varphi^{-1})(x_{q_i})\|. \tag{7}$$

Let $\delta u = \Psi_\theta(u_r, \Omega_r, \varphi)$ and then predicted solution $\hat{u}_q = u_r \circ \varphi^{-1} + \delta u$. Given ground truth u_q, the above objective is implemented as the metric loss between $\hat{u}_q$ and u_q.

3.2. Reference Neural Operator

Following the principle for neural operators established by (Li et al., 2020b;a), we also formulate the architecture of RNO as a stack of kernel integrals,

$$\Psi = \mathcal{Q} \circ \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1 \circ \mathcal{P}. \tag{8}$$

$\mathcal{P}$ is an encoder that lifts function values to hidden space $\mathbb{R}^s$, where s is feature dimension, and it should be able to encode different numbers of geometries. $\mathcal{L}_l$, $l = 1, \cdots, L$ are integral operator layers. $\mathcal{Q} : \mathbb{R}^s \to \mathbb{R}^{d_s}$ is a decoder that projects hidden variables back to target space $\mathbb{R}^{d_s}$.

3.2.1. Geometry Encoder

If the domain Ω_T we consider have multiple geometry objects, i.e., the boundary of Ω_T has multiple components, $\partial\Omega_T = \cup_i \Gamma_i \cup \partial D$ where ∂D is the boundary of D (or simply $\partial\Omega_T = \cup_i \Gamma_i$ if Ω_T is an interior domain), we need to design an encoder capable of handling these different numbers of components. Inspired by (Molinaro et al., 2023), we encode each change of geometries by a shared encoder $\mathcal{P} : \mathcal{U} \times \mathcal{T} \times C^k(\mathcal{T}) \to \mathbb{R}^s$ and sum their outputs, where s is the dimension of features. Suppose at a point x with value $u_r(x)$, $\mathcal{P}$ is defined as

$$v(x) = \mathcal{P}(x, u_r(x), \Omega_r, \varphi) = \sum_i \mathcal{P}(x, u_r(x), \Gamma_i, \varphi). \tag{9}$$

Unlike (Molinaro et al., 2023) where output of $\mathcal{P}$ is averaged, here we sum all outputs to reflect the composition of changes of all geometry components. For any location $x \in \Omega_q$, we encode it to $\mathcal{P}(x)$ in the latent space $\mathbb{R}^s$. Hereafter, when it is self-evident, we still use x to refer latent encoding of these locations for simplicity.

Here for the encoder, we use the geometric parameters to represent Γ_i's, e.g., centers, radius, length, etc. Thus, we simply takes the difference between geometric parameters p_r, p_q of reference and query to represent φ. Note that we also represent φ as a vector function on mesh grids and combine this information by cross attention. See next sections for cross attention and construction of vector function φ.

3.2.2. Distance-Aware Cross-Attention

In order to handle multiple input functions, e.g., reference solution and deformation, let's consider an integral operator $\mathcal{K} : \mathcal{U} \times \cdots \times \mathcal{U} \to \mathcal{U}$ with M input functions v^j, $j = 1, \cdots, M$,

$$w(x) = \mathcal{K}(v^1, \cdots, v^M)(x) = \int_D \sum_{j=1}^M \kappa_j(x, y) v^j(y) dy \tag{10}$$

Note that we are dealing with irregular meshes deformable geometries. Inspired by (Cao, 2021; Hao et al., 2023), we apply attention mechanism (Vaswani et al., 2017) to approximate the kernels $\kappa_j(\cdot, \cdot)$ on irregular meshes. For

Algorithm 1 Forward function of RNO

Input: Query (x_q, p_q, Γ_q) and reference $((x_r, u_r), p_r, \Gamma_r)$
Preprocessing
$\delta x_q \leftarrow Construct\_Phi(\Gamma_q, \Gamma_r, x_q)$
$x_s \leftarrow x_q + \delta x_q$
$u_{interp} \leftarrow Interpolate((x_r, u_r), x_s)$

Forward passing of neural network
$\hat{u}_q \leftarrow u_{interp} + \Psi_\theta(x_q, u_{interp}, \delta x_q, p_q, p_r)$
Output: $\hat{u}_q$

$\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} \in \mathbb{R}^s$ and a sequence of inputs $X = \{x_i\}_{1 \le i \le N}$,, we have $\boldsymbol{q}(X), \boldsymbol{k}(X), \boldsymbol{v}^j(X) \in \mathbb{R}^{N \times s}$. Then attention works as a kernel and is defined by

$$\textbf{attn}(x, y_i) = \text{softmax}(\boldsymbol{q}(x)\boldsymbol{k}^T(y_i)). \tag{11}$$

Distance weight is helpful to strengthen attention according to spatial relation between elements of $\boldsymbol{q}$ and $\boldsymbol{k}$ (Ying et al., 2021; Zhang et al., 2023a). Considering the change of solution can be strongly related to the location of deformation in some problems, e.g., fluid dynamics, we apply distance weighting and approximate (10) by a distance-aware cross attention (DACA) layer,

$$\boldsymbol{w}(x) \approx \frac{1}{\alpha} \sum_{i=1}^{N} \sum_{j=1}^{M} \textbf{attn}_j(x, y_i) \cdot h_\gamma(d(x, y_i)) \cdot \boldsymbol{v}^j(y_i), \tag{12}$$

where d is a distance function, e.g., Euclidean distance, $h(t) = e^{-\frac{t^2}{\gamma^2}}$ and γ is a hyperparameter. $\alpha = \sum_i \sum_j \textbf{attn}_j(x, y_i) \cdot h_\gamma(d(x, y_i))$ is a normalization factor. Each $\textbf{attn}_j$ can have its own $\boldsymbol{q}^j$ and $\boldsymbol{k}^j$ to learn different kernels $\kappa_j(\cdot, \cdot)$, but the computation cost would be $O(MN^2)$ for each layer. In practice, we find a shared kernel in each layer sufficient for our purpose and keep the cost as $O(N^2)$. In Appendix C, we describe a DACA layer of linear complexity.

Finally, we construct each integral operator layer $\mathcal{L}_l$ as

$$\boldsymbol{v}_{l+1}(x) = \mathcal{L}_l(\boldsymbol{v}_l)(x) = \boldsymbol{v}_l(x) + f\left(\boldsymbol{w}_l(x)\right), \tag{13}$$

where $\boldsymbol{w}_l$ is defined by (12) and f is a composition of a layer normalization and an MLP with nonlinear activation function GELU (Hendrycks & Gimpel, 2016). For input functions v^j of $\boldsymbol{w}_l$, we choose the triplet $\big(\boldsymbol{v}_l, \mathcal{P}_1(u_r \circ \varphi^{-1}(x_q)), \mathcal{P}_2(\varphi^{-1}(x_q) - x_q)\big)$, where $\mathcal{P}_i$'s, $i = 1, 2$, are encoders that lift input function values to hidden variables. See overall architecture in Fig. 3.

3.3. Algorithms

In this section, we deal with the following questions. How to construct a vector function to represent deformation φ? How to obtain a pushforward function $u_r \circ \varphi^{-1}$?

Interpolation for pushforward. Notice that the pushforward of u_r, $u_r \circ \varphi^{-1}(x_q)$, can be obtained by interpolation. One can construct a triangular mesh on Ω_r from reference data $\{(x_{r_i}, u_{r_i})\}_i$, and then interpolate $\{u_r \circ \varphi^{-1}(x_{q_j})\}_j$.

Ref-Query Dataloader. We need a dataloader that loads data in pairs of reference and query. If training dataset is constructed in pairs, where two solutions and geometries in one pair are close to each other, reference and query are paired naturally. If training set is constructed by sampling from geometric design space, for each query, reference can be determined by K-nearest-neighbor. Here the distance between examples can be Euclidean distance between geometric parameters.

In our implementation, a dataset has format $((x, u), p, \Gamma)$, where p stands for the geometry parameters and Γ is the set of boundary points. Then a paired data loaded from the dataloader has format $((x_q, u_q), (p_q, p_r), ((x_r, u_r), \Gamma_r, \Gamma_q))$.

Constructing φ. φ should avoid folding or tearing which means positive Jacobian everywhere. In practice, we take a naïve approach as approximation. **Step 1.** Find shifting vectors from points on boundaries of Ω_r to points on boundaries of Ω_q. This requires 1-to-1 matching between points. If the deformation from reference to query is obtained by some deformation methods, e.g., free-form deformation (Sederberg & Parry, 1986), then the 1-to-1 matching is automatic. Or we can reconstruct boundaries of simple geometries with parameters to create such matching. **Step 2.** For each point x_q, set its shifting vector equal to the shifting vector of the closest boundary point, and then put on the weight by the distance between x_q and the closest boundary point. The weight function is $h(t) = e^{-\frac{t^2}{\gamma_\varphi^2}}$, and γ_φ is another hyperparameter. **Step 3.** Protect the wall ∂D of domain D, i.e., to prevent points being shifted outside of the domain. Let d be the distance between x_r and ∂D. We define a smooth cutoff function on $[0, \infty)$,

$$\eta(d) = \begin{cases} 0, & d = 0 \\ e^{1 - \frac{d_{max}^2}{d^2}}, & 0 < d < d_{max} \\ 1, & d \ge d_{max} \end{cases} \tag{14}$$

where d_{max} is chosen to be the shortest distance between Γ_i's to ∂D. Then apply the cutoff function on all shifting vectors obtained from step 2. Note that except step 1, step 2 and 3 are smooth procedures. Hence, a smooth step 1 guarantees smooth φ. Here, our choice of step 1 may not be smooth, but it provides a simple yet effective approximation of φ. See Fig. 2 (Right) for the illustration of the construction.

Preprocessing and RNO. Given a dataset, all previous procedures are processed before feeding data into forward

Dataset	Component	GNOT	Geo-FNO	R-GNOT	R-FNO-i	R-MIONet	RNO (Ours)
NS2d-360	u	1.6e-1	4.6e-1	1.0e-1	2.7e-1	3.7e-1	**3.9e-2**
	v	4.2e-1	9.0e-1	2.5e-1	7.5e-1	6.6e-1	**1.1e-1**
	p	2.1e-1	4.3e-1	9.5e-2	2.1e-1	3.6e-1	**4.2e-2**
NS2d-sq-360	u	1.1e-1	3.4e-1	8.6e-2	2.3e-1	3.2e-1	**5.3e-2**
	v	3.2e-1	8.5e-1	2.7e-1	7.3e-1	7.0e-1	**1.8e-1**
	p	2.0e-1	4.3e-1	1.6e-1	2.4e-1	4.4e-1	**9.1e-2**
Inductor2d-320	A_z	5.8e-2	1.4e-1	2.5e-2	3.0e-2	3.8e-2	**4.3e-3**
	B_x	6.2e-2	3.4e-1	4.5e-2	5.5e-2	1.1e-1	**1.7e-2**
	B_y	7.0e-2	4.2e-1	5.6e-2	6.0e-2	1.2e-1	**2.2e-2**
Heatsink3d-80	u	1.1e-1	-	5.0e-2	-	2.0e-1	**4.2e-2**
	v	3.2e-1	-	1.9e-1	-	5.6e-1	**1.6e-1**
	w	2.6e-1	-	1.4e-1*	-	4.8e-1	**1.4e-1**
	T	8.4e-3	-	4.3e-3	-	1.9e-2	**3.7e-3**

Table 1. The number following the name of datasets is the size of datasets. The prefix "R-" indicates modified models that take reference solution u_r and deformation φ as inputs. See definition of baseline models in Section 4. **Bold** is the best. *The 2nd digit after the decimal is higher than RNO.

pass of neural networks. The pipeline is summarized in Algorithm 1. Obviously, for scaling purpose, one may move the entire procedure of preparing reference and query pairs outside of the training loop of RNO.

4. Experiments

Datasets. We consider several PDE problems in both 2D and 3D space. Problems are defined on various geometric domains, including different shapes, different positions, different sizes and different numbers of geometric objects. All datasets are generated in pairs. For each pair, the two domains are deformations of each other, and during training and testing the two data samples are used as reference and query for each other. We intentionally limit the size of datasets in order to evaluate all methods for practical use. The dataset sizes of 2D problems are no more than 400, and for more expensive 3D problems, we use a dataset of 80 samples. Training and testing sets are split in ratio $8:2$. For more details of each dataset, please check Appendix A. For all problems, we choose $\gamma_\varphi = 0.1$ to construct deformation φ, see Section 3.3.

- **Steady 2D-Navier-stokes equations** models flows though a 2D channel with holes. There are two types of holes, circle (**NS2d**) and square (**NS2d-sq**). The number of holes ranges from $1 \sim 9$, and the size and the position of holes varies.
- **Maxwell equations (Inductor2d)** is a model of circular shape inductor with different numbers of circular copper wire.
- **Heat transfer and CFD (Heatsink3d)** is a 3D model of heatsink with hexagon shaped pin-fins. The number of pin-fins ranges from $2 \sim 14$. The gap between rows of pin-fins varies.

The above challenging datasets are featured with random number of simple geometries which can be deformed by shifting and resizing. In Appendix B, we benchmarked RNO from a different angle, namely to expand example scenarios to complex geometry shapes such as airfoil. With two more datasets, Airfoil-Euler(Li et al., 2022a) and Airfoil-RANS(Bonnet et al., 2022), we showcase that RNO can flexibly be applied to free-form type of deformations beyond previous examples.

Baselines. Also, we compare RNO to several baseline models. In particular, we modify some of the models to learn the operator: $(u_r, \Omega_r, \varphi) \mapsto u_q$. The modified models are prefixed by "R-".

- **MIONet** (Jin et al., 2022) is similar to DeepONet (Lu et al., 2021) as a mesh-free model, and it requires fixed sensors in the domain. So we interpolate all input data on uniform grids as sensors.
- **Geo-FNO** (Li et al., 2023) is able to learn deformations of the domain and generalize on different geometric shapes. It can work with irregular meshes, so we make no change on this model.
- **FNO** (Li et al., 2020a) should be applied on uniform grids. So we interpolate data on 64 by 64 uniform grids. Additionally, we modify the input of FNO and name it as R-FNO-i.
- **GNOT** (Hao et al., 2023) is versatile with the formats of input functions, including geometric parameters, boundary shapes. etc. So, we use vanilla GNOT as

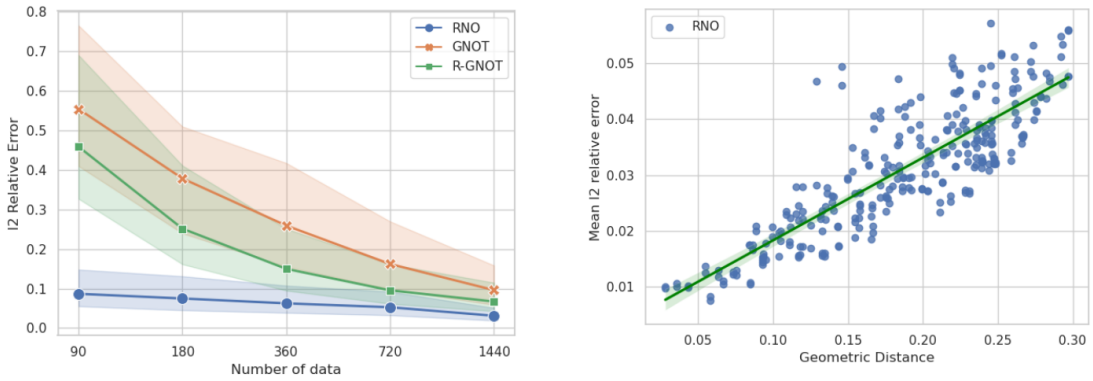

Figure 4. (Left) Mean error of all components versus the size of dataset. (Right) Error versus distance.

one of the baseline models. Additionally, we also use R-GNOT as a baseline.

All baselines except MIONet use up to 4 times more parameters than RNO. MIONet uses roughly 30% fewer parameters than RNO.

Evaluation metric and hyperparameters. The metric of evaluation is l_2 relative error. Suppose u and $\hat{u}$ are ground truth solution and predicted solution,

$$\|u-\hat{u}\|_{rel_2}=\left(\frac{\sum_i|u_i-\hat{u}_i|^2}{\sum_i|u_i|^2}\right)^{\frac{1}{2}}. \tag{15}$$

We train all models with AdamW optimizer (Loshchilov & Hutter, 2017) with cyclical learning rate (Smith, 2017). All experiments are ran 100 epochs with batchsize $1 \sim 4$ depending on the number of nodes in meshes.

4.1. Main Results

In Table 1, we summarize main results of the experiment. There are two types of baseline models, with or without reference solution (u_r, Ω_r) and discretized deformation φ as inputs, distinguished by prefix "R-". Models without "R-" belong to traditional **G-S** neural operators. RNO outperforms all baseline models in all problems.

Note that R-GNOT outperforms GNOT for all problems, indicating that the extra information about reference solution and deformation is helpful for prediction. Meanwhile, R-GNOT is still not as good as RNO, suggesting that the performance of RNO is not only due to extra inputs but a key fact that the target of RNO is approximating the material derivative of solution operators.

The performance of linear RNO (RNO-L) is summarized in Table 6. RNO-L has a little higher error most likely due to the randomness introduced by RFM. We point out that a drawback of linear DACA is $O(d^2D)$ memory cost. When D is taken a high integer value to improve the accuracy of estimation of the distance weight function $h_\gamma(d)$, the memory cost grows by D times compared to vanilla linear attention. However, it can be mitigated by increasing number of heads of attention, with number of heads n_{head}, the memory cost is reduced to $O(\frac{d^2D}{n_{head}})$.

A natural baseline of RNO may be the pushforward of reference solution, $u_r \circ \varphi^{-1}$, i.e., $\Psi_\theta \equiv 0$. We compare the mean l_2 relative error of the pushforward reference solution and RNO. See Table 7 in Appendix E, RNO has smaller errors for all problems, and some error is reduced by $50\% \sim 80\%$. It shows that RNO has essentially learned to predict the difference $u_q - u_r \circ \varphi^{-1}$.

4.2. Scaling Experiment

Size of dataset. In the discussion above, we see that even with small dataset, RNO is able to learn the change of solution due to the change of geometry and hence provides an approximation of the queried solution. We wonder how size of dataset would affect the performance of RNO. We choose the dataset NS2d which has 1440 samples in total. Then we down sample the dataset to smaller datasets with sampling rate $r = 1, 2, 4, 8, 16$. The baseline models are GNOT and R-GNOT, which are two best models besides RNO. In Figure 4 (Left), we plot GNOT and R-GNOT representing vanilla and modified neural operators against RNO on NS2d. The error on three models all decrease as size of dataset increases. Though with more data the gap between RNO and the other methods narrows, RNO consistently outperforms the other two. We emphasize that the ability of learning from small amount of data is highly desirable to shape optimization and other engineering design problem.

Notice that some components are more challenging than others to predict, and l_2 relative errors can be higher than

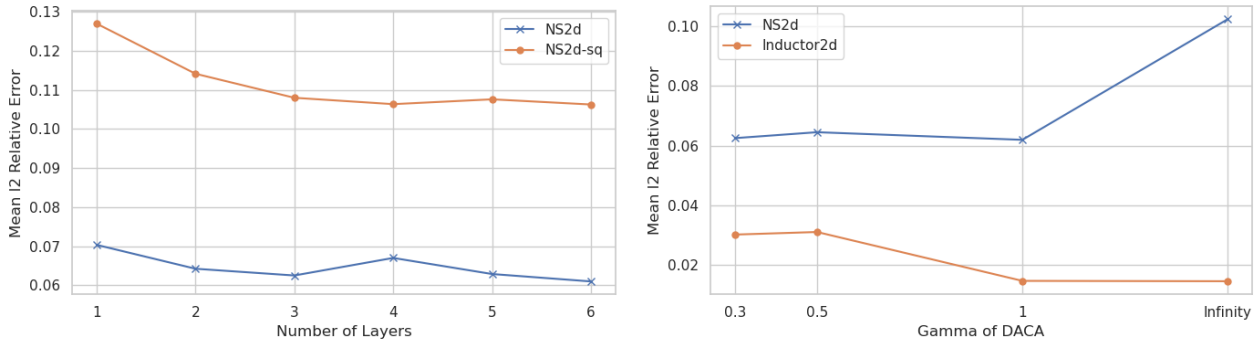

Figure 5. (Left) Error versus number layers. (Right) Error versus γ of DACA.

	u	v	p
GNOT	5.9e-2	1.6e-1	7.1e-2
R-GNOT	4.3e-2	1.1e-1	4.4e-2
RNO	**1.9e-2**	**5.1e-2**	**2.3e-2**

Table 2. Results of models on NS2d-1440.

	NS2d	NS2d-sq	Inductor2d
w/o δu	7.4e-2	1.2e-1	3.6e-2
w/o DACA	9.9e-2	2e-1	3.8e-2
All	**6.3e-2**	**1.1e-1**	**1.5e-2**

Table 3. Ablation study on learning target and DACA.

10%, see Table 1. However, this problem can be mitigated when the size of dataset increases. On the full NS2d dataset with 1440 samples, GNOT and R-GNOT still have an error larger than 10% on v, but RNO achieved reducing error by almost 50%. See Table 2.

Number layers of RNO. The performance of RNO will gain little improvement for number of layers $L > 3$, see Fig. 5 (Left), so we choose $L = 3$ to balance cost and benefit.

4.3. Error Analysis

Since RNO learns to approximate the material derivative of solution operators, a natural question is how geometric distance relates to prediction error In Figure 4 (Right), we plot the mean l_2 relative error of components versus the geometric distance for NS2d, and error grows with distance. Here the geometric distance is defined by Euclidean distance between geometry parameters of reference and query. In general, for non-parametric geometries, we can define the norm of discretized deformation as the geometric distance. See more error analysis for other datasets in Appendix E.

4.4. The Hyperparameter of DACA

The hyperparameter γ in (12) is crucial to the performance of RNO, and it is problem dependent. See Fig. 5 (Right). For NS2d, the mean l_2 relative error increases as γ increases. One way of reasoning this effect is that for fluid dynamics, a local geometry change will most likely impact nearby flow. On the other hand, for Inductor2d, the mean l_2 relative error decreases as γ increases. This interesting opposite trend is due to the well-known fact that magnetic-electronic phenomenon is governed by Maxwell equations, and such elliptic equations have slow-decay kernel functions for solutions (Evans, 2022). Therefore, long distance impact will be significant. In our implementation, for NS2d, NS2d-sq and Heatsink3d, we choose $\gamma = 0.3$. For Inductor2d, we choose $\gamma = \infty$ (disable distance weighting).

4.5. Ablation Study

We conduct ablation study on the components of RNO. Results in Table 3 show that both the target δu and DACA are key components. Values are mean l_2 relative errors. "Without DACA" means to remove entire DACA layers."Without δu" means to learn the operator $(u_r, \Omega_r, \varphi) \mapsto u_q$ instead of $(u_r, \Omega_r, \varphi) \mapsto \delta u$, which is the same as "R-" type of baseline models except for model architecture. To rationalize why predicting δu is better than predicting u_q directly, RNO explicitly determines the pushforward $u_r \circ \varphi^{-1}$ in $\hat{u}$, which reduces the complexity of the target.

5. Conclusion

Learning the operator mapping from geometry shapes to solutions of PDEs is a challenging problem due to the fact that geometry space is extremely huge and complicated. It can be prohibitive to sample sufficient data from geometry space and train neural operators. We propose an alternative approach that neural operators learn the change of a reference solution corresponding to a queried deformation. We conduct extensive experiments and show that RNO can learn this target efficiently. Our method may direct a new and practical path of applying neural operators as surrogate

models for shape optimization and other downstream tasks.

6. Impact Statements

This paper presents work whose goal is to advance the field of Machine Learning as new tools for physics simulation. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

Bonnet, F., Mazari, J., Cinnella, P., and Gallinari, P. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35: 23463–23478, 2022.

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

Chen, J., Hachem, E., and Viquerat, J. Graph neural networks for laminar flow prediction around random two-dimensional shapes. *Physics of Fluids*, 33(12), 2021.

Cohen, A., Schwab, C., and Zech, J. Shape holomorphy of the stationary navier–stokes equations. *SIAM Journal on Mathematical Analysis*, 50(2):1720–1752, 2018.

Dufter, P., Schmitt, M., and Schütze, H. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Jin, P., Meng, S., and Lu, L. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

Kahana, A., Zhang, E., Goswami, S., Karniadakis, G., Ranade, R., and Pathak, J. On the geometry transferability of the hybrid iterative numerical solver for differential equations. *Computational Mechanics*, pp. 1–14, 2023.

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

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

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*, 2020b.

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

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022b.

Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., et al. Geometry-informed neural operator for large-scale 3d pdes. *arXiv preprint arXiv:2309.00583*, 2023.

Liu, S., Hao, Z., Ying, C., Su, H., Cheng, Z., and Zhu, J. Nuno: A general framework for learning parametric pdes with non-uniform data. *arXiv preprint arXiv:2305.18694*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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.

Molinaro, R., Yang, Y., Engquist, B., and Mishra, S. Neural inverse operators for solving pde inverse problems. *arXiv preprint arXiv:2301.11167*, 2023.

Multiphysics, C. Introduction to comsol multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb*, 9 (2018):32, 1998.

Nelsen, N. H. and Stuart, A. M. The random feature model for input-output maps between banach spaces. *SIAM*

Journal on Scientific Computing, 43(5):A3212–A3243, 2021.

Obiols-Sales, O., Vishnu, A., Malaya, N., and Chandramowliswharan, A. Cfdnet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM international conference on supercomputing*, pp. 1–12, 2020.

Patel, R. G., Trask, N. A., Wood, M. A., and Cyr, E. C. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021.

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

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Sederberg, T. W. and Parry, S. R. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pp. 151–160, 1986.

Serrano, L., Le Boudec, L., Kassaï Koupaï, A., Wang, T. X., Yin, Y., Vittaut, J.-N., and Gallinari, P. Operator learning with neural fields: Tackling pdes on general geometries. *Advances in Neural Information Processing Systems*, 36, 2024.

Shukla, K., Oommen, V., Peyvan, A., Penwarden, M., Bravo, L., Ghoshal, A., Kirby, R. M., and Karniadakis, G. E. Deep neural operators can serve as accurate surrogates for shape optimization: a case study for airfoils. *arXiv preprint arXiv:2302.00807*, 2023.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Sokolowski, J. and Zolésio, J.-P. *Introduction to shape optimization*. Springer, 1992.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, pp. 3424–3433. PMLR, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 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.

Wang, S., Wang, H., and Perdikaris, P. Improved architectures and training algorithms for deep operator networks. *Journal of Scientific Computing*, 92(2):35, 2022.

Wu, C., Wu, F., and Huang, Y. Da-transformer: Distance-aware transformer. *arXiv preprint arXiv:2010.06925*, 2020.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T. Do transformers really perform bad for graph representation? arxiv 2021. *arXiv preprint arXiv:2106.05234*, 2021.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of gnns via graph biconnectivity. *arXiv preprint arXiv:2301.09505*, 2023a.

Zhang, Y., Long, M., Chen, K., Xing, L., Jin, R., Jordan, M. I., and Wang, J. Skilful nowcasting of extreme precipitation with nowcastnet. *Nature*, 619(7970):526–532, 2023b.

A. Datasets

All datasets are generated by by COMSOL 6.0. All data has format $((x, u), p, \Gamma)$, where p stands for the geometry parameters and Γ is the set of boundary points. Then we construct a dataloader to load a pair of data as $((x_q, u_q), (p_q, p_r), ((x_r, u_r), \Gamma_r, \Gamma_q))$. All experiments run on NVIDIA Tesla V100.

NS2d and NS2d-sq describes fluid flows through a square channel with holes, namely, $D = [0, 8] \times [0, 8]$, $\Omega = D \backslash \bigcup_{i=1}^{M} R_i$, where $M = 1, \cdots, 9$ and R_i's are disks or squares. Boundary condition is "no-slip". Steady Navier-Stokes equation is

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} = \frac{1}{\text{Re}} \nabla^2 \boldsymbol{u} - \nabla p \tag{16}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{17}$$

where $\text{Re} = 1$ is the Reynolds number and $\boldsymbol{u} = (u, v)$. We sample uniformly the number of holes with random positions and sizes. Additionally, for each sample we perturb its geometry parameters and make them a reference and query pair. In total, there are 1440 samples, 1152 for training and 288 for testing. The pairing of data is kept after splitting. For smaller dataset experiments, we down sample the dataset. Below are some examples of the dataset, see Fig. 6.

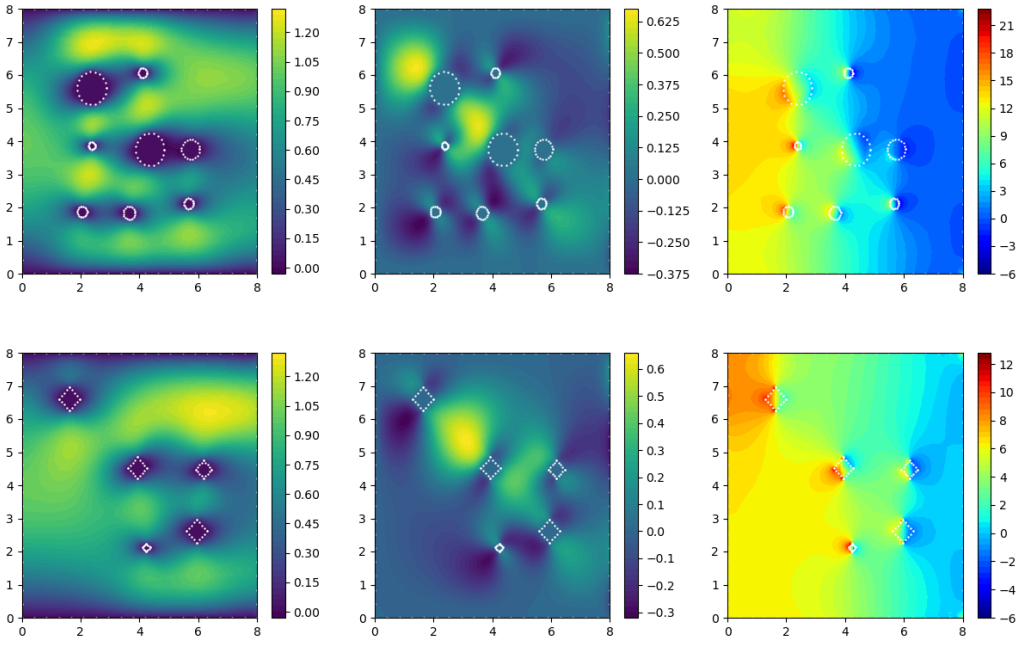

Figure 6. Columns are u, v, p. Rows are circle and square. White dots are boundary points.

Inductor2d models a circular-shape magnetic core wrapped by some number of copper coils. The governing equations are steady Maxwell equations,

$$\nabla \times \boldsymbol{H} = \boldsymbol{J} \tag{18}$$

$$\boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{19}$$

$$\boldsymbol{J} = \sigma \boldsymbol{E} \tag{20}$$

$$\boldsymbol{B} = \mu_0 \mu_r \boldsymbol{H} \tag{21}$$

with interface condition

$$\boldsymbol{n} \times \boldsymbol{A} = 0 \tag{22}$$

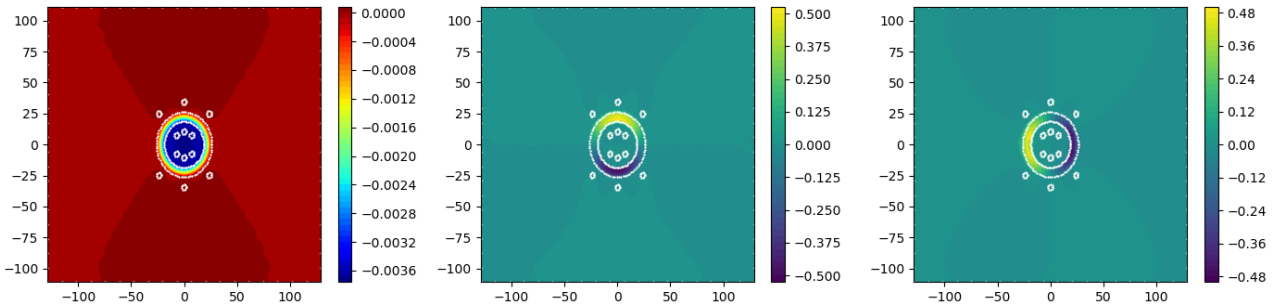

Figure 7. Plots of A_z, B_x, B_y. White dots are boundary points.

where μ_0 is the vacuum permeability and μ_r is the permeability of the magnetic core. For 2d problem, suppose $\boldsymbol{A} = (0, 0, A_z)$ and $\boldsymbol{B} = (\partial_y A_z, -\partial_x A_z, 0)$, the above equations simplify to an elliptic type of equation of A_z. Particularly, if μ_r is a constant, it is a Poisson equation.

Heatsink3d models a heatsink with pin-fins. The detail of the model is omitted. Readers can find details in (Multiphysics, 1998).

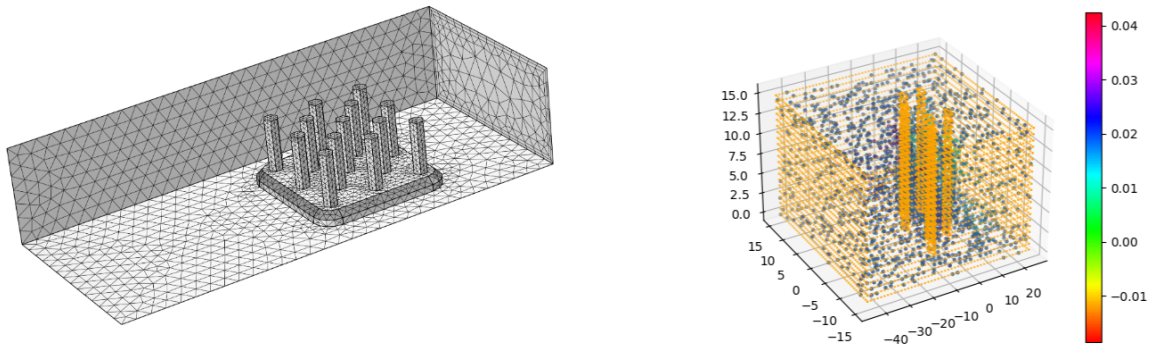

Figure 8. (Left) The mesh of a 3d model of heatsink with 10 hexgon pin-fins. (Right) A data sample of 6 pin-fins. The color of each point in space is the value of a component of u, v, w, T. Orange dots are boundary points.

B. Additional Experiments

In this section, we present benchmark on two additional datasets, Airfoil-Euler (Li et al., 2022a) and Airfoil-RANS (Bonnet et al., 2022) to demonstrate RNO working with complex geometry shapes and free-form type of deformations. The differences between this section and previous experiments are: 1. The number of geometry object is fixed to one, but the type of deformations is extended beyond shifting and resizing to squeezing, stretching, rotation, etc.; 2. When testing, instead of pairing queries and references among testing set, we find the nearest neighbor of a test query from training set and use it as reference. Note that this setting mimics practical usage of RNO in real world. The nearest neighbor is determined by geometric distance, which is implemented as either distance between geometry parameters or norm of deformation.

Airfoil-Euler is a simplified model of airfoil with Euler equation with no viscosity, where the boundary conditions are fixed and also angle of attack is set to zero. The target physical field is fluid density, which is a 1D field. See details in (Li et al., 2022a). We notice that many state of the art neural operators have achieved quite low error rates, order of magnitude $1e-3 \sim 1e-2$ on this dataset, see e.g., (Serrano et al., 2024). One reason is that airfoil area is less than 1% of the whole

domain, and the error is averaged to be low since most area is almost constant and easy to fit. Another reason is that the training set contains 1000 samples. Therefore, we made two changes to the dataset: 1. Cropping the data to a smaller neighborhood of airfoil (see Fig. 10); 2. Limiting training set to 200 samples to fit our setting, which is also called "scarce data regime" in (Bonnet et al., 2022).

Another feature of this dataset is that all mesh grids are deformed from one standard uniform mesh grids, so that the tensors of mesh grids have the same shape. As a result, a natural 1-1 mapping of grids exists among data, which saves the need of constructing deformation from reference to query and the following interpolation, greatly simplifying implementation of RNO. Baseline models are CORAL (Serrano et al., 2024), NU-FNO (Liu et al., 2023), Geo-FNO (Li et al., 2022a) and GNOT (Hao et al., 2023). RNO outperforms all other baselines on this modified dataset, see Table 4. More importantly, it shows RNO can be applied to general deformations. Metric is l_2, $\|u-\hat{u}\|_{l_2}=\frac{1}{N}\sum_i^N|u_i-\hat{u}_i|^2$.

Model	CORAL	NU-FNO	Geo-FNO	GNOT	RNO
l_2 Error	8.1e-2	1.5e-1	7.9e-2	4.0e-2	**3.4e-2**

Table 4. l_2 Error on Airfoil-Euler.

Airfoil-RANS is a much more challenging CFD dataset generated from simulation of Reynolds-Averaged Navier–Stokes (RANS), which contains complex patterns of turbulent viscosity. Besides geometry, the boundary condition varies in the dataset, for example, inlet velocity, angle of attack, Reynolds number, etc. It brings challenge to determine suitable reference data for RNO. Ideally, reference and query should have the same boundary conditions and small difference in geometry. However, we manage to show that RNO still outperforms **G-S** type neural operators with only 180 training data. It shows that the methodology of RNO can be applied to real-life problems where data can be scarce and problems can be complicated.

In our implementation, we use both geometry shape and boundary conditions to determine the top-3 nearest neighbors as reference for training, and use the top-1 neighbor in training set as reference for testing. We found that different target components prefers different strategies of reference. For example, dominant factors for pressure may be inlet velocity and angle of attack. So, a closer reference on these two factor can help predicting pressure. We feel there is great potential to explore, however, due to scope of this paper, we present our primitive study on this dataset.

Baseline models are GNOT and its modified version R-GNOT which adds reference as input. Note that unlike Airfoil-Euler where the mesh grids of every case is deformed from a standard uniform grids, many aforementioned baselines can not be easily implemented here since no such standard grids exists and the number of mesh grids are different from case to case. All metric is l_2. RNO achieves competitive results. See Table 5. We note that, since AirfRANS was not designed to generate data for RNO, there is a potential for RNO to improve. We recommend generating datasets of reference and query pairs with *fixed* boundary conditions (BCs) within a pair (BCs can be different between pairs) and *varying* geometry, such that RNO can learn the change of solution due to the change of variable that we are interested in. Such setting can be particularly useful for e.g. sensitivity and robustness analysis. See qualitative examples in Fig. 11.

Models	u	v	p	ν_t
GNOT	0.51	0.44	0.61	0.31
R-GNOT	**0.30**	**0.19**	*0.11*	**0.19**
RNO	*0.32*	*0.32*	**0.10**	*0.22*

Table 5. **Bold** indicates the best and *Italic* the second. u, v are velocity components, p the pressure, and ν_t the turbulent viscosity.

C. Distance-Aware Linear Attention

To construct a linear attention for (12), we implement random feature mapping (RFM) (Rahimi & Recht, 2007) to approximate the distance weighing function $\exp(-\|\cdot\|^2/\gamma^2)$. The same trick is applied by (Peng et al., 2021) with a different purpose, to modify the softmax function in attention mechanism and achieve linear complexity[1].

[1]We found a typo in the paper of (Peng et al., 2021). In theorem 1. $\boldsymbol{w}_i$ should be sampled from $\mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I}_d)$, and then $\mathbb{E}_{\boldsymbol{w}_i}[\phi(\boldsymbol{x})\cdot\phi(\boldsymbol{y})] = \exp(-\|x-y\|^2\cdot\sigma^2/2)$, instead of $\exp(-\|x-y\|^2/2\sigma^2)$

Let linear attention be $\mathbf{attn}(x, y_j) = \frac{1}{N_{div}} \boldsymbol{q}(x) \cdot \boldsymbol{k}(y_j)$ where $N_{div} = \sum_j \boldsymbol{q} \cdot \boldsymbol{k}_j$ is a normalizing constant (see details of linear attention in (Hao et al., 2023)), and (12) becomes

$$\boldsymbol{w}(x) \approx \sum_j^N \sum_t^M \boldsymbol{q}(x) \cdot \boldsymbol{k}(y_j) \cdot h_\gamma(d(x, y_j)) \cdot \boldsymbol{v}^t(y_j), \tag{23}$$

To approximate $h_\gamma(d(x,y)) = \exp(-\|x-y\|^2/\gamma^2)$, sample $\boldsymbol{\omega}_i$ from $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$ where $\sigma = \frac{\sqrt{2}}{\gamma}$, and then define $\phi : \mathbb{R}^d \to \mathbb{R}^{2D}$ as

$$\phi(x) = \frac{1}{\sqrt{D}} [\sin(\boldsymbol{\omega}_1 \cdot x), \cdots, \sin(\boldsymbol{\omega}_D \cdot x), \cos(\boldsymbol{\omega}_1 \cdot x), \cdots, \cos(\boldsymbol{\omega}_D \cdot x)] \tag{24}$$

Then according to RFM we have $\mathbb{E}_{\boldsymbol{\omega}_i}[\phi(x) \cdot \phi(y)] = \exp(-\|x-y\|^2/\gamma^2)$. Let $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}, \phi(x)$ be indexed by $q_{ab}, k_{ab}, v_{ab}, \phi_{ac}$, where $a = 1, \cdots, N$, $b = 1, \cdots, d$, $c = 1, \cdots, D$. Then let j, m correspond to the first and the second indices a, b, and l corresponds to c, the second index of ϕ. We have

$$\begin{aligned} w_{ab} &= \sum_{m,l,j} q_{am} k_{jm} \phi_{al} \phi_{jl} v_{jb} \\ &= \sum_{m,l} q_{am} \phi_{al} \sum_j k_{jm} \phi_{jl} v_{jb} \\ &= \sum_m q_{am} \sum_l \phi_{al} H_{mlb} \\ &= \sum_m q_{am} H_{amb}, \end{aligned}$$

where $H_{mlb} = \sum_j k_{jm} \phi_{jl} v_{jb}$ and $H_{amb} = \sum_l \phi_{al} H_{mlb}$. The computational cost of H_{mlb}, H_{amb} is $O(Nd^2D)$ and $O(d^2D)$. For all w_{ab}, $1 \le a \le N$ and $1 \le b \le d$, the cost is $O(Nd^2D)$.

The performance of linear RNO is summarized in Table 6,

	NS2d-360			NS2d-1440		
	u	v	p	u	v	p
RNO-L	4.2e-2	1.1e-1	4.1e-2	2.5e-2	6.6e-2	2.6e-2
RNO	3.9e-2	1.1e-1	4.2e-2	1.9e-2	5.1e-2	2.3e-2

Table 6. RNO-L stands for linear RNO and is comparable with RNO in accuracy.

RNO-L has a little higher error most likely due to the randomness introduced by RFM. We point out that a drawback of linear DACA is $O(d^2D)$ memory cost. When D is taken a high integer value to improve the accuracy of estimation of the distance weight function $h_\gamma(d)$, the memory cost grows by D times compared to vanilla linear attention. However, it can be mitigated by increasing number of heads of attention, with number of heads n_{head}, the memory cost is reduced to $O(\frac{d^2D}{n_{head}})$.

D. Learning Objective

Consider a reference solution u_r on Ω_r and a query domain Ω_q. Given a vector field $V = V(x,t)$ and the corresponding flow T_t, recall the material derivative (Sokolowski & Zolésio, 1992) of the solution operator G at Ω_r,

$$\dot{u}(\Omega_r, V) = \lim_{t \to 0} \frac{1}{t} (u_{T_t(\Omega_r)} \circ T_t - u_{\Omega_r}), \tag{25}$$

where $\circ$ is composition of functions and $u_{T_t(\Omega_r)} \circ T_t$ is the pullback of $u_{T_t(\Omega_r)}$. The limit exists in a suitable topology, for example a Sobolev space $W^{m,p}(\Omega)$ or $H_0^1(\Omega)$ (Evans, 2022). Suppose $t = \varepsilon$ and $T_\varepsilon = \varphi$, then we can rewrite the above equation as

$$u_{\Omega_q} \circ \varphi = u_{\Omega_r} + \varepsilon \cdot \dot{u}(\Omega_r, V) + o(\varepsilon). \tag{26}$$

where $u_{\Omega_q} = u_q$. Suppose $\varepsilon \cdot \dot{u}(\Omega_r, V)$ is approximated by a neural operator Ψ_θ. Then for $x \in \Omega_r$,

$$u_q \circ \varphi(x) \approx u_r(x) + \Psi_\theta\left((x, u_r), \Omega_r, \varphi\right). \tag{27}$$

For a set of points $\{x_{q_i}\}_i \subset \Omega_q$, and a collection of Ω_q's of size N, each paired with a reference triplet (u_r, Ω_r, φ), the objective function (6) is approximated by

$$\min_\theta \frac{1}{N} \sum_q \sum_i \|\Psi_\theta(u_r, \Omega_r, \varphi)(x_{r_i}) - (u_q \circ \varphi - u_r)(x_{r_i})\|. \tag{28}$$

Note that this objective is computed on the domain Ω_r. Alternatively, we can transform the problem to Ω_q. Let $\{x_{r_i}\}_i \subset \Omega_r$ and $\varphi(x_{r_i}) = x_{q_i} \in \Omega_q$ for all $i = 0, 1, 2 \ldots$ The above objective can be rewritten as

$$\min_\theta \frac{1}{N} \sum_q \sum_i \|\Psi_\theta(u_r, \Omega_r, \varphi)(\varphi^{-1}(x_{q_i})) - (u_q - u_r \circ \varphi^{-1})(x_{q_i})\|. \tag{29}$$

Let $\delta u = \Psi_\theta(u_r, \Omega_r, \varphi) \circ \varphi^{-1}$, and then predicted solution $\hat{u}_q = u_r \circ \varphi^{-1} + \delta u$. Given ground truth u_q, the above objective is implemented as the metric loss between $\hat{u}_q$ and u_q.

A minor difference between (28) and (29) is due to interpolation. Namely, $u_q \circ \varphi(x_{r_i})$ can be approximated by interpolating u_q on $\varphi(x_{r_i})$ in Ω_q, and $u_r \circ \varphi^{-1}(x_{q_i})$ can be approximated by interpolating u_r on $\varphi^{-1}(x_{q_i})$ in Ω_r. In our implementation, we adopt (7) to leave the target u_q unchanged, but the other way is obviously also valid.

E. Error Analysis

We compare a natural baseline, the pushforward of reference solution, $u_r \circ \varphi^{-1}$, with RNO. See Table 7. It shows that RNO shrinks the gap between $u_r \circ \varphi^{-1}$ and target u_q. In fact, what RNO is actually learning is to fill this gap.

	NS2d	NS2d-sq	Inductor2d	Heatsink3d
$u_r \circ \varphi^{-1}$	1.2e-1	2.5e-1	7.3e-2	1.6e-1
RNO	**6.3e-2**	**1.1e-1**	**1.5e-2**	**8.6e-2**

Table 7. Mean error of the pushforward reference solution and RNO.

In Fig. 9, we plot mean l_2 relative error versus geometric distance for other datasets besides NS2d. Error grows as distance increases except for Inductor2d. Our speculation is that geometric distance defined by the Euclidean distance between parameters of reference and query may not accurately measure deformations in this case. A more accurate alternative for geometric distance can be the norm of deformation φ from reference to query geometry.

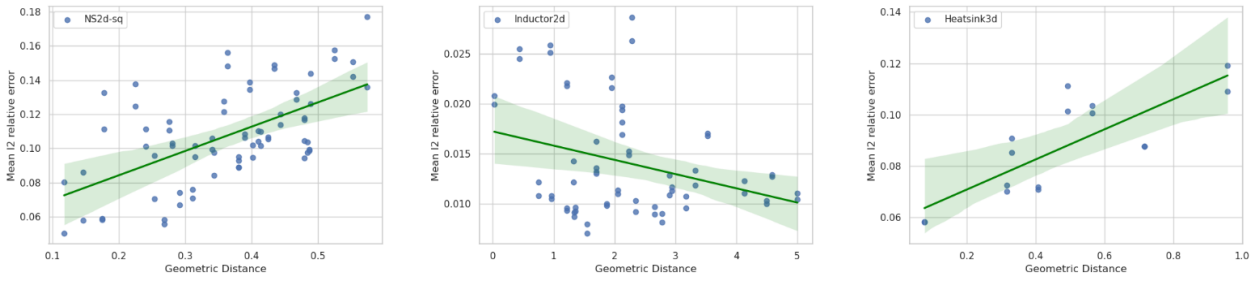

Figure 9. Left to right: Error versus distance of NS2d-sq, Inductor2d, Heatsink3d.

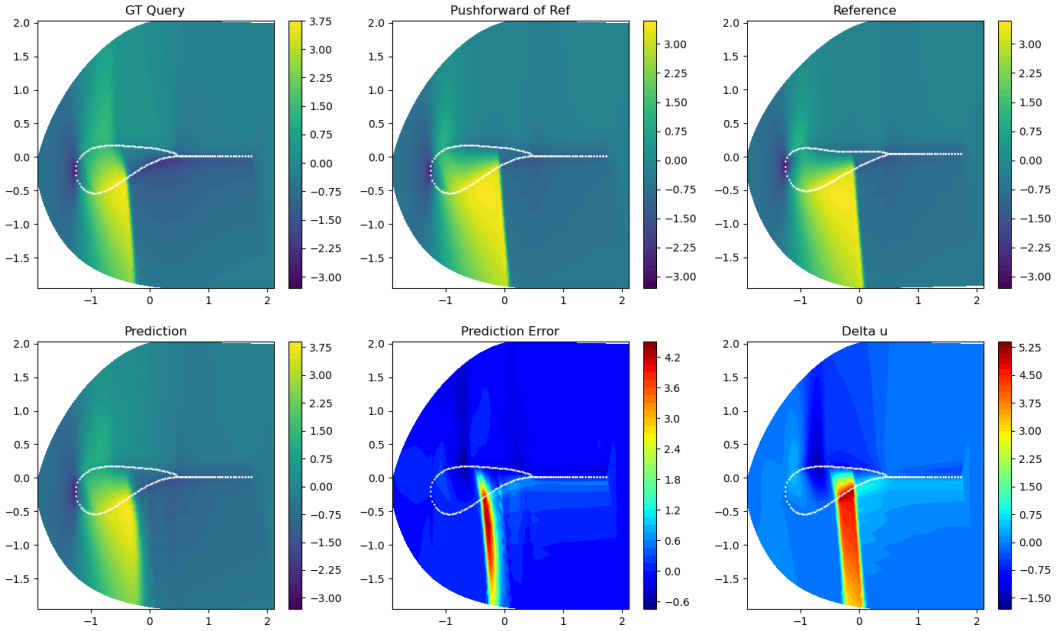

Figure 10. Qualitative example of RNO on Airfoil-Euler.

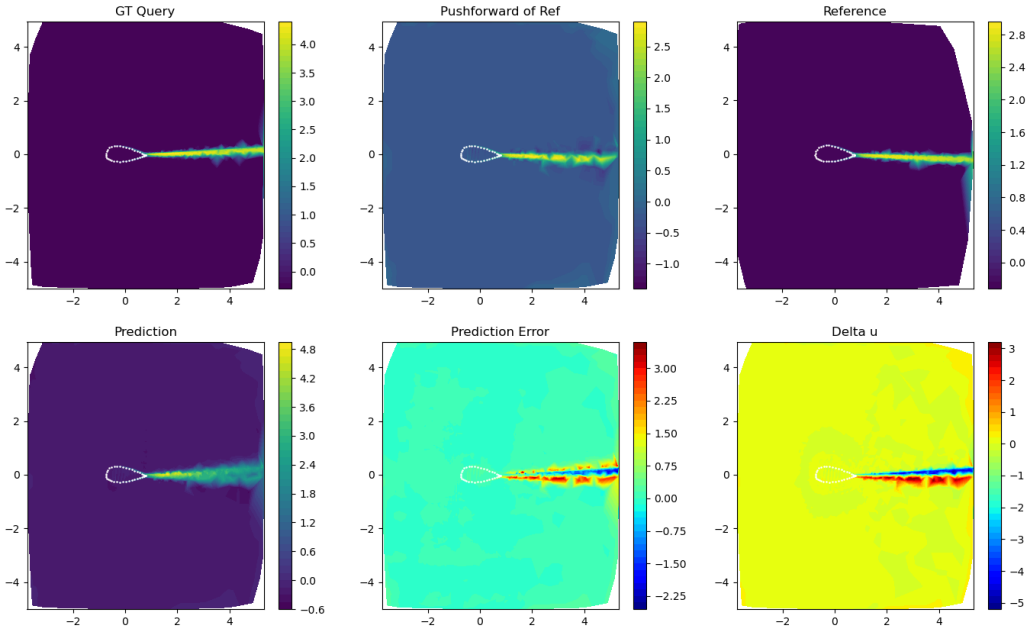

Figure 11. Qualitative example of RNO on the turbulent viscosity of Airfoil-RANS.