Journal of Computational Physics 461 (2022) 111205

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

RotEqNet: Rotation-equivariant network for fluid systems with symmetric high-order tensors

Liyao Gao [a], Yifan Du [b,1], Hongshan Li [c,1], Guang Lin [d,*]

[a] Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA
[b] Department of Mechanical Engineering, John Hopkins University, Baltimore, MD, USA
[c] Amazon, Santa Clara, CA, USA
[d] Department of Mathematics and School of Mechanical Engineering, Purdue University, West Lafayette, IN, USA

ARTICLE INFO

Article history:
Received 11 May 2020
Received in revised form 20 February 2021
Accepted 4 April 2022
Available online 11 April 2022

Keywords:
Machine learning
Tensor analysis
Rotation-equivariant
Fluid systems

ABSTRACT

In the recent application of scientific modeling, machine learning models are largely applied to facilitate computational simulations of fluid systems. Rotation symmetry is a general property for most symmetric fluid systems. However, in general, current machine learning methods have no theoretical guarantee of Rotation symmetry. By observing an important property of contraction and rotation operation on high order symmetric tensors, we prove that the rotation operation is preserved via tensor contraction. Based on this theoretical justification, in this paper, we introduce Rotation-Equivariant Network (RotEqNet) to guarantee the property of rotation-equivariance for high order tensors in fluid systems. We implement RotEqNet and evaluate our claims with four case studies on various fluid systems. The property of error reduction and rotation-equivariance is verified in these case studies. Results are showing the high superiority of RotEqNet compared to traditional machine learning methods.

1. Introduction

With recent developments in data science and computational tools, machine learning algorithms have been increasingly applied in different engineering and science areas to model physical phenomena. The data from physical experiments and numerical simulations are a source of knowledge about the physical world, on which data-driven methods could be performed to extract new physical laws [1–6]. For example, in turbulence RANS modeling in fluid mechanics, traditional modeling methods have failed in many flow scenarios. A unified RANS model that can successfully describe complex flows, including boundary layer, a strong rotation, separation still does not exist according to the author's knowledge [7,8]. On the other hand, advanced measurement and direct numerical simulations provide plenty of data that could be utilized to establish and validate new models. With the above argument, data-driven methods are particularly suitable for turbulence modeling and some other areas in physics and engineering. There have been many attempts to discover new turbulence models using machine learning methods. Milano and Koumoutsakos [9] reconstruct near-wall flow applying neural networks and compared their results with linear methods (POD). Zhang and Duraisamy [10] used Gaussian process regression combined with an artificial neural network to predict turbulent channel flow and bypass transition. Beck, Flad, and Munz

* Corresponding author.
E-mail addresses: marsgao@uw.edu (L. Gao), dyifan1@jhu.edu (Y. Du), lihongshan8128@gmail.com (H. Li), guanglin@purdue.edu (G. Lin).
[1] Equal contribution.

https://doi.org/10.1016/j.jcp.2022.111205
0021-9991/

[11] applied residual neural network for Large Eddy Simulation. Chen et al. proposed an ODE network to generally learn differential equations [12].

The physical laws often appear in the form of tensorial equalities which inherently obey certain types of symmetry. For example, the constitutive laws in fluid and solid mechanics should obey translation and rotation invariance [13]. Physical symmetries mostly appear in the form of invariance under continuous group actions on the tensorial equalities. The preservation of symmetries is essential for simulation algorithms to capture physical reality, and often difficult to achieve in both traditional numerical methods and data-driven machine learning methods. For example, neural networks and most of the deep learning methods do not naturally guarantee rotation equivariance when they are used as approximators between the Cartesian components of tensors. Although the universal approximator theorem guarantees convergence to any compact supported functions [14] including rotation equivariant functions, the symmetry is not generally preserved for training results from finite sample points.

There have been different attempts to achieve rotation equivariance in deep learning of turbulence RANS models, which also appear as local tensorial equality between mean velocity gradient and Reynolds stress. Data augmentation is one simple way of improving symmetry preservation in deep learning methods of physical laws. In [15], an augmented dataset is generated by rotating the existing sample points in arbitrarily different directions. The training cost increases significantly while the prediction is still far from rotation equivariance. Data augmentation method achieves rotation equivariance at infinite sample limit by universal approximation theorem, but normally fails on finite sample set.

Another method that has been explored in previous research requires the construction of integrity basis on first and second-order tensors. In [15,16], Reynolds stress is expressed as a general expansion of nonlinear integrity basis multiplied by scalar functions of invariants of strain rate and rotation rate tensors. Machine learning is performed to find these scalar functions of tensor invariants of strain rate and rotation rate tensors. Mathematically this expansion comes from an application of the Caylay-Hamilton theory. The special case used in [15,16] is derived by S.B. Pope in [17]. Although such construction is general and possible for higher-order tensors and tensor tuples containing multiple tensors, the number of this basis and the derivation complexity will grow exponentially and become prohibitive for real applications [18,19].

There have been previous works considering group-equivariance with convolutional neural networks (CNN) in image recognition. A general method has been proposed using group convolution [20–22]. Based on the idea of using convolution, several methods composed a steerable filter for rotation-equivariance in convolutional neural networks [23–27]. However, these works cannot be applied directly to physical systems. One of the most important reasons is that the rotation operation on the image is different from rotation operation on physical systems. The rotation on image is a global coordinate transformation, which is different from tensor rotation [28]. Additionally, these methods have a strong restriction that this model must be built on CNNs. CNNs are not well suitable for the approximation of tensorial equations, which lack the spatial scale similarity that is often explored in a natural image by CNNs.

In this paper, we establish Rotation-Equivariant Network (RotEqNet), a data-driven framework for learning equalities from tensorial data which exactly satisfies rotation-equivariance with finite samples. We first establish a general method to achieve rotation equivariance on the learning of equalities from 1st and 2nd order tensorial data using diagonalization. Then through tensor contractions, the problem of preserving rotation equivariance for higher order tensorial data is reduced to lower order problems, which have been established in previous step. In the realization of rotation equivariance, a rotation matrix can be extracted from the sample data, which is applied to transform sample data to its standard position. Standard position algorithm is proven to truncate infinite rotation group into a learnable finite group in Lemma 3.1. Therefore, the learning rules based on standard position are forming a quotient space of the original rules in random rotated plural position [23,29]. In this way, RotEqNet lowers the training difficulty of a randomly positioned dataset and leads to finite learnability of RotEqNet in Theorem 1. Further, RotEqNet is also proven to be rotation-equivariant in Theorem 2. These advantages of RotEqNet would result in an observable error reduction compared to previously introduced data-driven methods.

We applied RotEqNet into four different case studies ranging from second-order, third-order, and fourth-order. These case studies are designed based on Newtonian fluids, Large-eddy simulations, and Electrostriction. Improved performances could be observed for using RotEqNet. We show the rotation error could be reduced to machine precision, and the prediction errors are reduced for 99.6%, 32.17%, and 54.63% for second, third, forth-order case studies respectively. The error from random rotation reaches machine precision for RotEqNet in all tested cases. Our contribution in this paper is three-fold:

1. We design a position standardization algorithm that could find the standard position for arbitrarily rotated tensors. We prove this algorithm would truncate infinite rotation group into finite groups for machine learning.
2. We propose and analyze RotEqNet which could obtain rotation-equivariance property with finite training samples. RotEqNet has the theoretical guarantee to improve generalization with lower testing errors.
3. We implement RotEqNet with various case studies to show its improvement to baseline methods with data augmentation. The generalization error on random rotations reaches machine precision.

2. Problem description

In this paper, we aim to build a Rotation-equivariant model with machine learning algorithms using finite samples. Firstly, for a machine learning model, it is generally not built up with rotation-equivariance. As a typical solution, data augmentation has been largely applied by increasing training data with random rotations. However, to guarantee the learnability from a continuous rotation group, it will require infinite amount of samples. This would harm the performance of machine learning model leading to a high generalization error. We formalize the above problems into the following parts.

2.1. Rotation-equivariant machine learning: idea and definition

Group equivariance is an important property for most physical systems. Typical cases of group equivariance could be rotation group equivariance, scaling group equivariance, and translation group equivariance. Modeling rotation group equivariance in fluid dynamics prediction is critical to real life applications like atmospheric air currents simulation. Mathematically, group equivariance is a property defined as follows.

Definition 2.1 *(Rotation-equivariant functions).* A mapping $f : X \to Y$ to commute from X to Y under rotation group actions. Specifically, let $R \in SO(n)$ be a rotation action. f is rotation-equivariant if

$$f(R(x)) = R(f(x)), \quad \forall R \in SO(n),\ x \in X. \tag{1}$$

Consider supervised learning models as functions, name a parametric machine learning model $\mathcal{M} : X \to Y$. For a rotation operation R, we define this ML model to be rotation-equivariant as the following:

Definition 2.2 *(Rotation-equivariant ML model).* A model $\mathcal{M}$ is rotation-equivariant if

$$\mathcal{M}(R(x)) = R(\mathcal{M}(x)), \quad \forall R \in SO(n),\ x \in X. \tag{2}$$

Here, notice this property is generally not true as long as $\mathcal{M}$ is non-linear. Modern machine learning algorithms are typically non-linear, which are not naturally Rotation-equivariant.

2.2. Learnability with finite samples

Machine learning could only be trustworthy if it is learnable within finite samples. Suppose we have a set of dataset which is finite learnable. From probably approximately correct (PAC) learning [30] point of view, we can assume a hypothesis class $\mathcal{H}$ which is learnable with finite samples $D = \{(x_i, y_i) : x_i \in X, y_i \in Y\}_{i=1}^n$.

Finite learnable typically represents the data that can be shattered into finite sets. If an arbitrary large subset can be shattered, since its VC dimension is ∞, this would not be finite learnable. Building on a finite learnable $\mathcal{H}$, we further define the extension of this hypothesis class with a group action by the following:

Definition 2.3 *(Group action on hypothesis class).* Suppose an unknown distribution function $\mathcal{D}$ can be learned by a realizable hypothesis class $\mathcal{H}$. G is a group. The G-extension of this hypothesis class of $\mathcal{H}$ is $G(\mathcal{H})$, which is realizable on the group action G on $\mathcal{D}$.

Since $\mathcal{H}$ is realizable on $\mathcal{D}$, according to Realizability Assumption, we know there exists $h^* \in \mathcal{H}$ such that the training loss $L_{\mathcal{D},f}(h^*) = 0$. For a realizable hypothesis class on $G(\mathcal{D}) = \{(g(x), g(y)) : \forall g \in G\}$, it can be only true on a hypothesis space with larger expressive power. We denote this extended hypothesis class by $G(\mathcal{H})$.

We introduce a Lemma here to qualify when the cardinality of G is finite for $G(\mathcal{H})$, we still have finite learnability.

Lemma 2.4 *(Finite learnability). Suppose G is a group, and $\mathcal{H}$ is a finite learnable hypothesis class. If $|G|$ is finite, $G(H)$ is finite learnable. If $|G|$ is infinite, $G(H)$ is not finite learnable.*

We include the proof in Appendix B via VC dimension [31,30].

Remark 1. For a rotation group $G_{SO(3)}$, in general, $G_{SO(3)}(\mathcal{H})$ is not finite learnable.

To guarantee $\mathcal{M}$ could learn $G_{SO(3)}(\mathcal{H})$, applying data augmentation requires infinite amount of training samples to guarantee the learnability (**Remark 1**). In the practice of fluid dynamic modeling, we could only generate finite data for ML models. The requirement of infinite samples would cause the trained ML model to be ill-conditioned. In our paper, we aim to solve the problem of rotation-equivariant machine learning with finite number of samples. This would allow ML models could be easily generalized on the testing set, with low training effort and high accuracy.

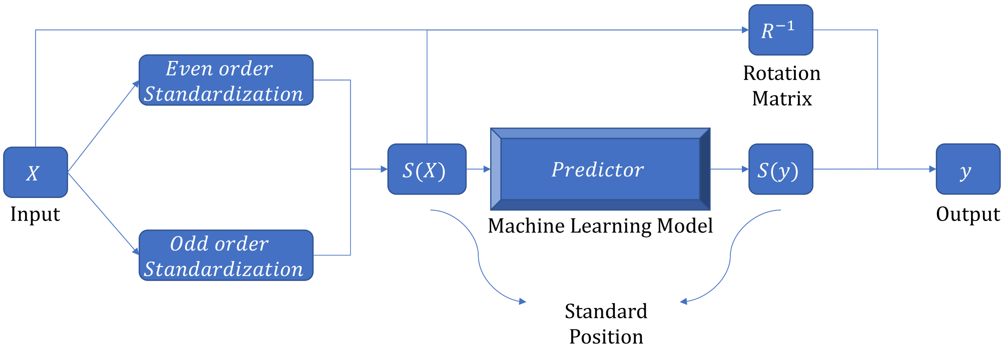


Fig. 1. Rotation-equivariant Network Architecture.

3. Rotation equivariant network

In this section, we propose a Rotation Equivariant Network (RotEqNet) architecture to learn constitutive laws of higher tensors in fluid systems with rotation-equivariance. The general process of RotEqNet is described in Fig. 1 with 3 steps:

1. This step is *position standardization*. For an input data X, $S(X)$ is the standard position of X. The algorithm treats input tensors differently for even and odd orders.
2. Train a machine learning model based on $(S(X), S(y))$. The machine learning model in the middle of Fig. 1 only needs to learn the standard positions of X, y.
3. Denote the kernel machine learning model in the middle of Fig. 1 as $\mathcal{M}_\Phi$. After $\mathcal{M}_\Phi$ is trained, in the inference stage, suppose for an input X, $X = R(S(X))$, the RotEqNet is defined as

$$\mathcal{M}_\mathcal{R} = R^{-1}(\mathcal{M}_\Phi(R(S(X)))) \tag{3}$$

as the prediction of X. Here, R is determined by position standardization algorithm S.

3.1. Position standardization algorithm for high order tensors

Let $\mathcal{D}$ denote as our data set. The first stage of RotEqNet is to find a good representative of randomly rotated tensors. We will call this representative the sample in *standard position* (X_s, Y_s). S is the position standardization algorithm. $\Phi(T)$ denotes a set generated by all standard positions of T.

The position standardization algorithm is defined as follows. For a tensor T, we aim to find a representative in the orbit $O(T)$, (where $O(T) = \{R \cdot T | R \in SO(n)\}$), as the standard position of T [32]. In our algorithm, we first perform tensor contraction to higher-order tensors. Then, we use diagonalization and QR factorization to obtain rotation information R in lower-dimension. Finally, R is used to rotate the tensor to its standard position. This final operation, using lower-dimension R to find standard position in original dimension, is compatible as shown in Lemma A.3.

The purpose of position standardization algorithm is to perform data reduction on the rotation group. Instead of performing machine learning on $G_{SO(3)}(\mathcal{H})$, we wish to distill ancillary information and learn only on $G_\Phi(\mathcal{H})$ where G_Φ is finite.

3.1.1. Position standardization: tensor of even-order

Given a symmetric tensor of even-order $T^n \in \otimes^n V$ (n is even). Let C denote a sequence of contraction along the first two axes until we reach a second-order tensor (Fig. 2). Applying C to T^n we get:

$$T^2 = C(T^n) \tag{4}$$

Then we find the orthonormal eigen-vectors of T^2 and use them to form the orthonormal matrix R that diagonalize T^2. Technically, orthonormal eigen-vectors are not unique with random sign and permutation (see Appendix). We choose R where the first row of R is positive and sort the order of eigen-vectors (ascending from left to right in rotation matrix R) using the corresponding eigenvalues. This ensures this process is well-defined.

$$T^2 = R^{-1} \times D \times R \tag{5}$$

Since R is an orthonormal matrix, we have

$$R^{-1} = R^T \tag{6}$$

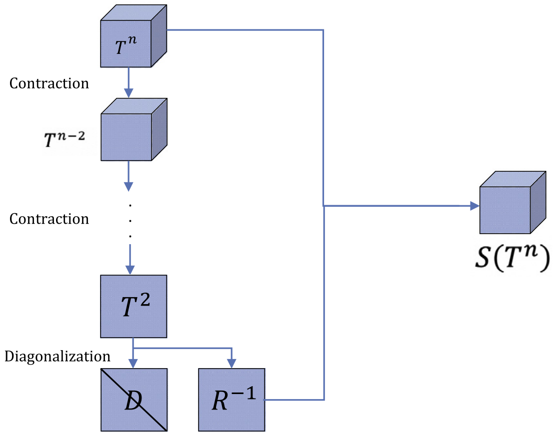


Fig. 2. Rotation-invariant extraction for even-order.

We will call D the standard position of T^2. We write $R(T^2) = D$ to shorten the notation

Since contraction and rotation are compatible by Lemma A.3. We can apply R to T^n before we apply contraction, and we will have

$$C(R(T^n)) = D \tag{7}$$

For the even tensor T^n, we could obtain

$$S(T^n) = R^{-1}(T^n), \tag{8}$$

as its standard position.

3.1.2. Position standardization: tensor of odd-order

Given a symmetric tensor of an odd-order tensor $T^n \in \otimes^n V$ (n is odd). Let C denote a sequence of contraction along the first two axes until we reach a third-order tensor (Fig. 3). Applying C to T^n we get:

$$T^3 = C(T^n) \tag{9}$$

After we obtain T^3, we could obtain three different order-one tensors by contracting it along axes (2,3), (1,3) and (1,2). Name the contracted results, which are first-order tensors i.e. vectors, V_1, V_2, V_3. We could get an second-order tensor by concatenating them.

$$T^2 = (V_1, V_2, V_3) \tag{10}$$

Then, we perform QR factorization to obtain rotation matrix R. We can execute the QR decomposition via Gram-Schmidt to ensure R is well-defined.

$$T^2 = R \times U^2, \tag{11}$$

For odd tensor, we define:

$$S(T^n) = R^{-1}(T^n) \tag{12}$$

The pseudocode of our proposed algorithm is documented in Algorithm 1.

3.2. Theoretical analysis of rotation-equivariant property

In our analysis, we aim to show the rotation-equivariant property is able to be obtained with finite samples for RotEqNet. The first step is to analyze the property of standard positions of any T would form a finite set. Then, we construct rotation-equivariant property based on our model's algebraic structure.

3.2.1. Main theorems and proofs

Since we will perform machine learning on $\Phi(T)$ instead of $G_{SO(3)}(T)$, we wish to show that $\Phi(T)$ is finite. We introduce the following Lemma to show Φ is a finite group.

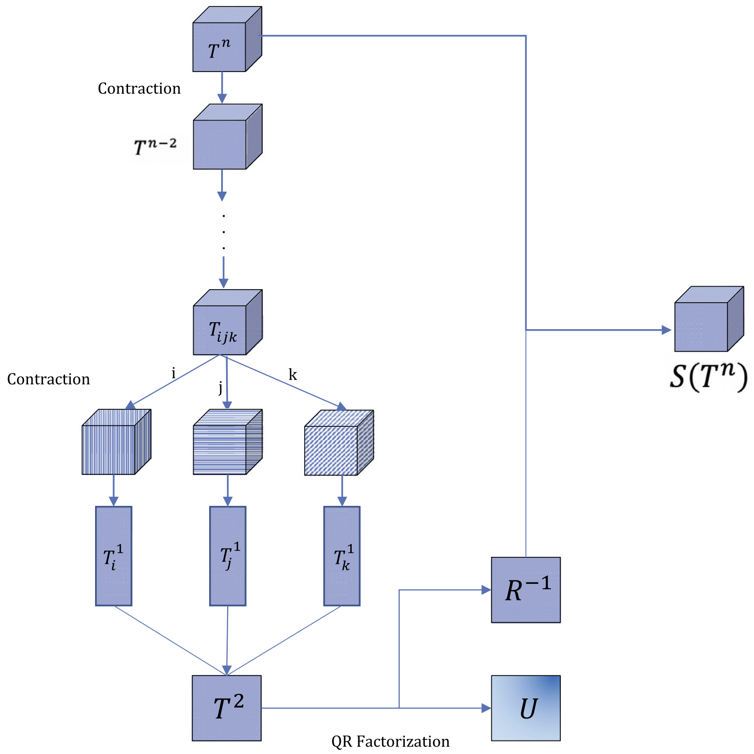


Fig. 3. Rotation-invariant extraction for odd-order.

Algorithm 1 Position standardization algorithm.

function POSITIONSTANDARDIZATION(T)
  $T^n = T$
  if T is in even **then**
    # For even cases
    while $order(T^n) \neq 2$ **do**
      $T^{n-2} = contraction(T^n \to T^{n-2}, 1, 2)$
      $T^n = T^{n-2}$
    end while
    $T^2 = T^n$
    $RotMat = leftEigenvector(T^2)$
    $T_s = tensorRotation(RotMat, T)$
  else
    # For odd cases
    while $order(T^n) \neq 3$ **do**
      $T^{n-2} = contraction(T^n \to T^{n-2}, 1, 2, 3)$
      $T^n = T^{n-2}$
    end while
    $T^3 = T^n$
    $V_1, V_2, V_3 = contraction(T^3 \to T^1, 2, 3),\ contraction(T^3 \to T^1, 1, 3),\ contraction(T^3 \to T^1, 1, 2)$
    $T^2 = concat(V_1, V_2, V_3)$
    $RotMat = QR(T^2)$
    $T_s = tensorRotation(RotMat, T)$
  end if
  return R, T_s
end function

Lemma 3.1. *For all standard positions of T, let $\Phi(T) = \{S(R(T)) : \forall R \in SO(3)\}$ be the set of all standard positions (derived by algorithm S) of randomly rotated T. We have*

1. *Φ is a group.*
2. *$\Phi(T)$ is a finite set and*

$$\begin{cases} |\Phi(T)| = 1, & T \text{ is second-order or odd-order tensor,} \\ |\Phi(T)| = 8, & T \text{ is even-order } (\geq 4) \text{ tensor.} \end{cases} \tag{13}$$

This lemma is purely technical by verifying different orders of tensors. We provide the proof in Appendix D. Using Lemma 3.1, we can construct finite learnability of RotEqNet in the following theorem.

Theorem 1. *Suppose H is realizable hypothesis class of $\mathcal{D}$, $G_\Phi(H)$ is a realizable hypothesis class of $G(\mathcal{D})$. If H is finite learnable, $G_\Phi(H)$ is finite learnable.*

Theorem 1 is a direct application of Lemma 3.1. Using Theorem 1 we can obtain finite learnability of the kernel of RotEqNet. Compare to infinite sample requirement for data augmentation, RotEqNet truncates the infinite rotation group into finite groups, resulting in only finite sample requirement. We provide a proof in Appendix E.

Theorem 2 *(Rotation-equivariant). Suppose a well-trained RotEqNet $\mathcal{M}_\mathcal{R}$ is trained with dataset $D = \{X_i, y_i\}_{i=1}^n$ with loss $L_D(\mathcal{M}_\mathcal{R}) = 0$. This well-trained $\mathcal{M}_\mathcal{R}$ is rotation-equivariant,* i.e. *for all rotation $R \in SO(n)$ and input tensor X_i*

$$\mathcal{M}_\mathcal{R}(R(X_i)) = R(\mathcal{M}_\mathcal{R}(X_i)) \tag{14}$$

Theorem 2 completes the proof of rotation-equivariant property of RotEqNet. We provide details in Appendix F.

3.2.2. Analysis on testing error improvements

Comparing the data augmentation, the design of RotEqNet improves the model by reducing the following three errors: approximation, optimization, and generalization. We could define generalization error from testing loss decomposition. The testing loss $L_{test}(\mathcal{M}^\theta)$ can be decomposed into the following,

$$\begin{aligned} L_{test}(\mathcal{M}^\theta) &= L_{train}(\mathcal{M}^\theta) + L_{test}(\mathcal{M}^\theta) - L_{train}(\mathcal{M}^\theta) \\ &= \min_\theta L_{train}(\mathcal{M}^\theta) && (Approximation\ error) \\ &\quad + L_{train}(\mathcal{M}^\theta) - \min_\theta L_{train}(\mathcal{M}^\theta) && (Optimization\ error) \\ &\quad + L_{test}(\mathcal{M}^\theta) - L_{train}(\mathcal{M}^\theta) && (Generalization\ error). \end{aligned} \tag{15}$$

For approximation errors, the truncation of the $SO(3)$ group will relax the difficulty in approximation. The learning task of RotEqNet only targets on a much smaller functional space with nearly $\mathbb{R}^{n\times n}/SO(3)$. This will lower the approximation error. Further, the optimization error could also be controlled based on finite sample size. Different from data augmentation, RotEqNet can achieve convergence within finite time. The generalization error is also proven to be smaller specifically on the generalization on the entire $SO(3)$ group. Applying RotEqNet, we expect to see loss reduction numerically compared to data augmentation.

4. Case studies

In this section, we conduct cases studies to show the performance of RotEqNet. In the following subsections, we include second-order, third-order, and fourth-order testing equations. The interpretation of experimental results is included in each subsection separately.

4.1. Machine learning model and evaluation metrics

Model setup. The machine learning model we apply here is neural networks and random forests because of the ability of these two models to approximate arbitrary functions. For Neural Networks, in our implementation, the logistic activation function is used as an activation function for every neuron. The number of neurons for two layers is 512 and 4, respectively. Adam optimizer [33] is applied as the algorithm for optimization, and the learning is set up to be 1×10^{-3}. We also set the batch size to be 64. For random forests, 100 estimators are set up with mean squared error as the criterion. The maximum depth of random forests is 3 to lower the chance of overfitting. We used Sklearn for implementation [34]. The computation environment of this experiment is CPU.

Prediction error in MSE. The effect of error reduction is evaluated for the first. A standard position predictor is trained by standard positions derived from training data using Algorithm 2. Then, the prediction algorithm is applied to both training and testing set to obtain the training and testing performances. The validation error $\mathcal{E}_{MS}$ is defined as the Mean Squared Loss using the formulation that:

$$\mathcal{E}_{MS} = \frac{\sum_{i=1}^N (y_i - \mathcal{M}(X_i))^2}{N} \tag{16}$$

In Eqn. (16), N is the number of data in dataset D, $\mathcal{M}$ is the trained machine learning model, and $(X_i, y_i) \in D$ describes the input-output pair of the dataset. This evaluation $\mathcal{E}_{MS}$ represents the MSE of model $\mathcal{M}$ with dataset D.

Rotation-equivariance error. There are two metrics in this case study. The first metric, as defined in the following, describes the error of rotation-equivariance for the model itself.

$$\mathcal{E}_R(\mathcal{M}) = \mathbb{E}\left[\|\mathcal{M}(R(X_0)) - R(\mathcal{M}(X_0))\|_2^2\right], \tag{17}$$

R is a random variable with all possible rotations and $X_0 \in X$. Note here if a parameterized model $\mathcal{M}$ have $\mathcal{E}_M = 0$, we know from definition that $\mathcal{M}$ is rotation-equivariant.

To empirically evaluate $\mathcal{E}_M$, first, we randomly generate data (X_0, y_0). Then we apply the rotation set $\{R_i\}_{i=1}^{10000}$ with 10,000 random rotation matrices to (X_0, y_0). To fully investigate the property of rotation-equivariant, there are two different measurements of the rotation-equivariant property of a specific model.

Another error metric here is the standard L2 error defined as $\mathcal{E}_D$ is the error of model with respect to data, which is defined in the following:

$$\mathcal{E}_D(\mathcal{M}) = \mathbb{E}\left[\|\mathcal{M}(R(X)) - R(y)\|^2\right],$$

R is a random variable with all possible rotations and (X_0, y_0) is an input-output pair from the learning target.

4.2. Case study from Newtonian fluid: a second-order linear case

4.2.1. Problem statement and data generation

Newtonian fluid is a type of fluid such that its viscous stress changes based on its flow. In this experiment, we aim to use simulation data to demonstrate this rule of Newtonian fluid. This would serve as a case study with second-order linear equations. Let $\sigma \in \mathbb{R}^{3\times 3}$ be stress tensor, $p \in \mathbb{R}$ pressure and $S \in \mathbb{R}^{3\times 3}$ strain rate. The rule of Newtonian fluid is a second-order physical equation which satisfies the following condition [35]:

$$\sigma = -pI + \mu S \tag{18}$$

Another definition of the equation for Newtonian fluid would use the velocity vector field, defined as ∇v. ∇v could be expressed as a 3×3 matrix. Using this definition, the equation of Newtonian fluid could also be written as:

$$\sigma = -pI + \mu(\nabla v + \nabla v^T) \tag{19}$$

This could also be considered as the definition of strain rate Based on this definition, we could observe that $S = \nabla v + \nabla v^T$, and it is symmetric since $S = S^T$. Since S is symmetric and I is an identity matrix, σ is also symmetric. Therefore, defining an arbitrary rotation matrix R, this system is equipped with the property of rotation-equivariant that $R(\sigma) = R(-pI + \mu S)$.

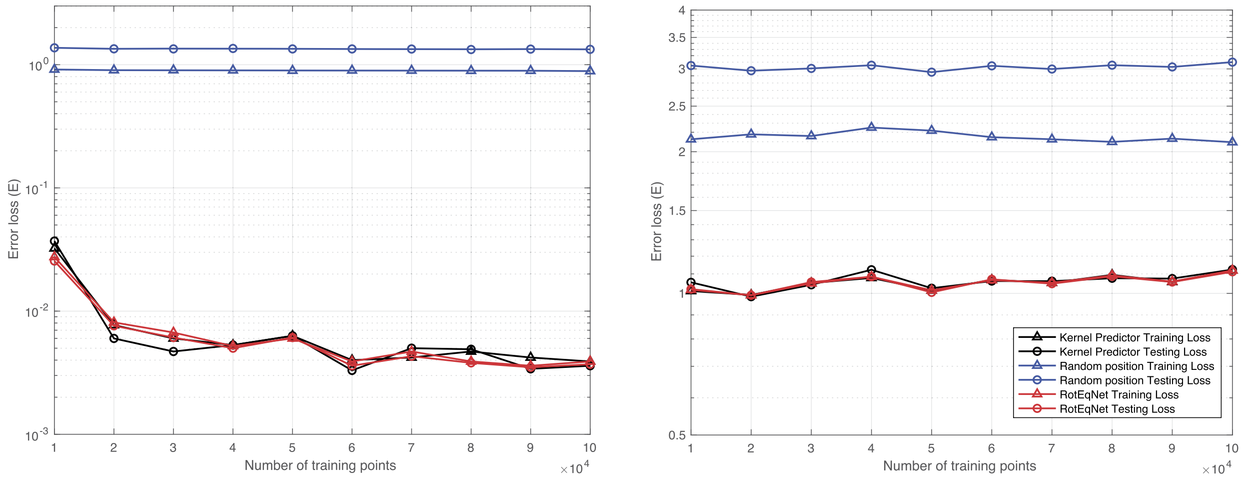
To quantify the stress for Newtonian fluid simulation, it would be useful to be able to predict the Newtonian fluid stress, given the simulation of pressure and velocity vector field. Based on this scenario, in this subsection, we provide a case study for the machine learning model on inputting the shear of Newtonian fluid and prediction of the stress.

Based on Eqn. (19), we first generate random data to obtain ∇v and p. The generation of random numbers in ∇v follows a normal distribution from range $(0, 1)$. Derived from generated ∇v and p, we could obtain σ from Eqn. (19). We denote the dataset as $D = \{x_i, y_i\}_{i=1}^{N}$. To form a proper dataset D with N elements for a machine learning model for Newtonian fluid, the input x is set up to be a vector where $x \in \mathbb{R}^{10}$. Specifically, x is composed by p and flattened S in Eqn. (18). The output $y \in \mathbb{R}^9$ is a vector which is the flattened result of matrix σ derived by p and S. The dataset D is set up as the description above. To compare the difference of our method to the baseline method, we trained two models with the same hyper-parameter using different amounts of training data, ranging from $10,000, 20,000, \ldots, 100,000$. 85% of generated data is used for training and 15% of data is used for testing. A rotation set with 10,000 random rotation matrices is also generated for evaluating the property of rotation-equivariant, denoted by $\{R_i\}_{i=1}^{10000}$.

4.2.2. Results

Fig. 4(a) shows the error reduction property of RotEqNet. This plot consists of three groups of experimental groups. The first experiment group focuses on the accuracy of the baseline model, a single feed-forward Neural Network, on raw data with random rotated positions. As represented by blue curves in Fig. 4(a), the triangle curve represents training error and the circle curve represents testing error. The second experiment group is RotEqNet with Neural Network as the standard position predictor. As represented by red curves in Fig. 4(a), the triangle curve represents training error, and the circle curve represents testing error. For 100,000 training samples, the testing error of RotEqNet is 0.0037, and the testing error of the baseline method is 1.333. We could observe a huge error reduction 99.56% in training and 99.60% in testing. The last experiment group, represented by a black curve in the figure, reports the performance of standard position predictor with standard position only. This experiment would explain why RotEqNet would improve performance since training with standard positions would be an easier task compared to raw data.

Further, Fig. 4(b) shows the error reduction effect of RotEqNet using Random Forest as a standard position predictor. Similarly, as shown in Fig. 4(b) with blue curves, it represents the performance of the baseline method (Random Forests).



(a) Neural Networks

(b) Random Forests

Fig. 4. Error of training with baseline model with random position, RotEqNet, and standard position predictor with the standard position for (a) Neural Networks and (b) Random Forests in the case study of Newtonian Fluid. Different colors represent different experimental groups. The RotEqNet model is trained with random positions and tested with random positions (red curves). Baseline models that trained and tested on raw data are shown as blue curves. The performances of standard position predictors that trained and tested with only standard positions are also shown as black curves. Training errors are shown with lines marked with triangles, testing errors are shown with lines marked with circles. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 1
Evaluation of Rotation-equivariant property between baseline model and RotEqNet.

Model	Baseline (NN)	RotEqNet (NN)	Baseline (RF)	RotEqNet (RF)
$\mathcal{E}_M$	0.6973	8.4594×10^{-32}	0.5177	**0.0**
$\mathcal{E}_D$	0.4872	**0.0011**	0.6243	0.0592

The second experiment group is RotEqNet with Random Forests as the standard position predictor. As shown in Fig. 4(b) in red curves, the triangle curve represents training error and the circle curve represents testing error. We could observe a huge error reduction, 99.56% in training and 99.72% in testing, for RotEqNet compared to the error of using only the Random Forest predictor. The last experiment group (black curves in the figure) trains the standard position predictor with standard position only. As stated before, this could also serve as a reason for the error reduction effect for RotEqNet on random forests.

According to the reported results, RotEqNet has a good generalization result without overfitting. For cases training with raw data for baseline models, the testing error is typically higher compared to training error. For example, the difference is training and testing errors are 0.44 for Neural Networks, 1.01 for Random forests when $N = 100,000$. This represents that for both Neural Networks and Random Forests would be easy to overfit this task on Newtonian Fluid. By contrast, RotEqNet would help to reduce this difference in training and testing error. As we could observe from the training and testing error of RotEqNet, the errors are much lower. When $N = 100,000$, there are only 0.0002 for Neural Networks and 0.0078 for Random Forests. In the case of linear second-order equations, the application of RotEqNet would result in better-generalized results in learning.

As shown in Table 1, for both baseline methods, using neural networks and random forests, there are large errors for $\mathcal{E}_M$ and $\mathcal{E}_D$. The baseline methods have no theoretical guarantee that it has the property of rotation-equivariant. However, there is error reduction for both machine learning models when applying with RotEqNet's architecture. For RotEqNet with Neural Networks as standard position predictor, we have $\mathcal{E}_M = 8.4594 \times 10^{-32}$ and $\mathcal{E}_D = 0.0011$. For RotEqNet with Random Forests as standard position predictor, we have $\mathcal{E}_M = 0.0$ and $\mathcal{E}_D = 0.0592$. This very low error of $\mathcal{E}_M$ could guarantee the rotation-equivariant property of RotEqNet.

Fig. 5 is a visualization of our tested case with contour plot of RotEqNet and baseline models. The x-axis is p and y-axis is $S_{0,0}$. The color map is representing the numerical result of $\sigma_{0,0}$. RotEqNet is reaching lower error with similar contour pattern compared to true data. We could observe that the plotting of RotEqNet (the top middle plotting) has nearly identical trend and numerical range to true data. The plotting of baseline method (the bottom middle plotting) is mainly predicting values in range [0.0, 1.0), which causes its color to be blue and white. Therefore, we could make a further conclusion that RotEqNet has a better result in flow simulation for the second-order linear cases.

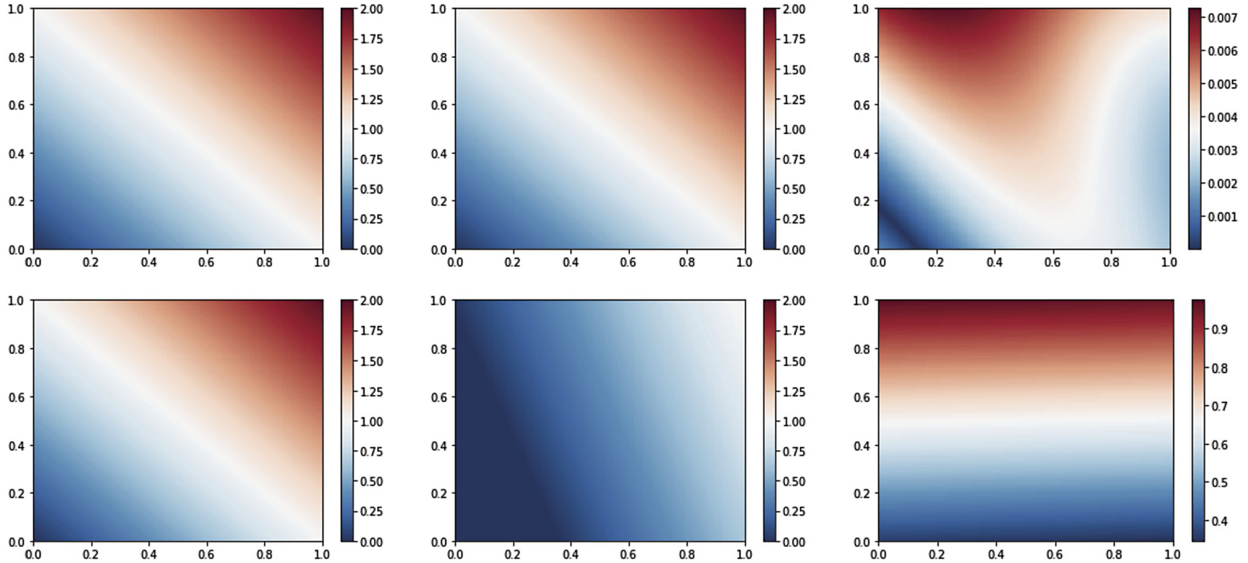


Fig. 5. Comparison of contour plot with RotEqNet and baseline method for the second-order linear case study. First row: RotEqNet, second row: baseline (Neural Network). Left column: true data, middle column: predicted solution, right column: L1 difference between the true data and prediction.

4.3. Case study from large Eddy simulation: a second-order nonlinear case

4.3.1. Problem statement and data generation

In this case, we consider the subgrid model of large eddy simulation (LES) of turbulent flow by Kosovic [36]. In this case study, as formulated previously in [37,38], we hope to obtain a learned model by simulation data from LES. This would serve as a case study with second-order non-linear equations. LES is defined as:

$$\tau_{ij} = -(C_s\Delta)^2 \left\{ 2\sqrt{2S_{mn}S_{mn}}S_{ij} + C_1\left(S_{ik}S_{kj} - \frac{1}{3}S_{mn}S_{mn}\delta_{ij}\right) + C_2\left(S_{ik}\Omega_{kj} - \Omega_{ik}S_{kj}\right)\right\} \tag{20}$$

Here τ_{ij} is subgrid stress, which is a symmetric traceless 2nd order tensor. S_{ij} and Ω_{ij} are symmetric and anti-symmetric parts of velocity gradient tensor G_{ij}, where $\mathrm{Tr}(G) = 0$. Further, C_s, Δ, C_1, C_2 are all constants. The configuration of constants above is reported in the next subsection.

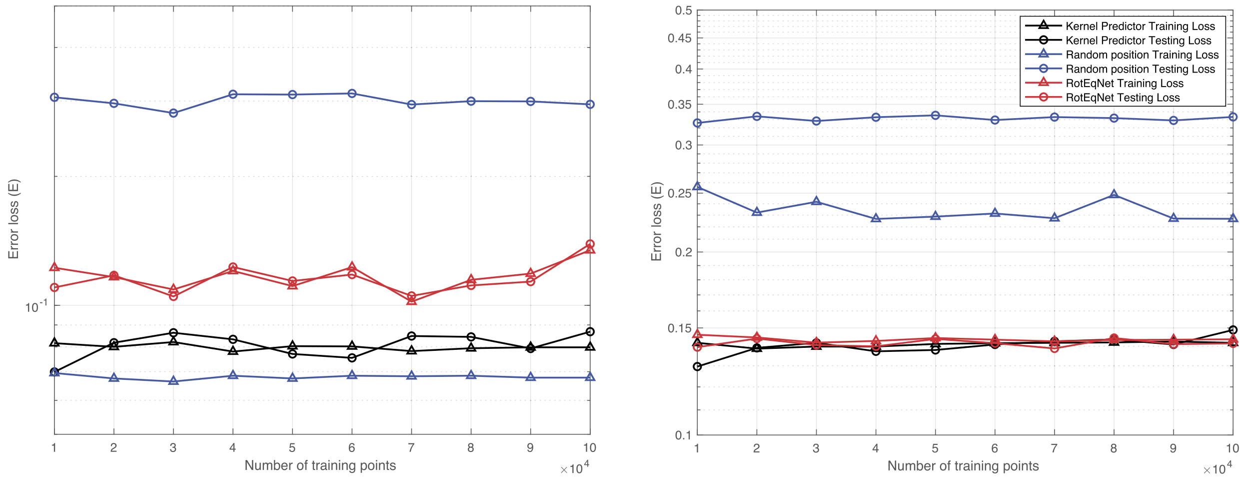
In order to qualify the subgrid stress for LES, this study aims to predict the subgrid stress, given the simulation of velocity gradient tensor. This case study for the machine learning model on inputting the velocity gradient tensor.

Based on Eqn. (20), we first generate random data to obtain a simulated velocity gradient tensor G_{ij}. The generation of random numbers follows a normal distribution from range $(0, 1)$, and G_{ij} is obtained from a random matrix G_{raw} by subtracting $\frac{1}{3}\mathrm{Tr}(G_{raw})$ from diagonal position. This would keep $\mathrm{Tr}(G) = 0$. S_{ij} and Ω_{ij} could be obtained from G_{ij} by getting its symmetric and anti-symmetric parts. For the setup of constants, $C_s = 0.4$, $\Delta = 0.4$, $C_1 = C_2 = 1.0$. τ_{ij} is computed from the above setting with Eqn. (20). Denote the dataset as, $D = \{x_i, y_i\}_{i=1}^{N}$. To form a proper dataset D with N elements for a machine learning model for Newtonian fluid, the input x is set up to be a vector where $x \in \mathbb{R}^9$. Specifically, x is composed by flattened G_{ij}. The output $y \in \mathbb{R}^9$ is a vector, which is the flattened result of matrix τ derived by G and other constants. To compare the difference of our method to the baseline method, we trained two models with the same hyper-parameter using different amounts of training data, ranging from $10,000, 20,000, \ldots, 100,000$. 85% of generated data is used for training, and 15% of data is used for testing. A rotation set with 10,000 random rotation matrices is also generated for evaluating the property of rotation-equivariant, denoted by $\{R_i\}_{i=1}^{10000}$.

4.3.2. Results

The effect of error reduction is evaluated, which the validation error $\mathcal{E}_{MS}$ is defined as the Mean Squared Loss using the formulation in Eqn. (16). This evaluation $\mathcal{E}_{MS}$ represents the expected error of model $\mathcal{E}_M$ with dataset D.

Fig. 6(a) shows the error reduction effect of RotEqNet with Neural Network as a standard position predictor for second-order nonlinear cases with three groups of experimental groups. The first experiment group focuses on the accuracy of the baseline method on raw data with random rotated positions. As shown in Fig. 6 with blue curves, triangle curve represents training error and the circle curve represents testing error. The second experiment group is RotEqNet, with Neural Network as a standard position predictor. As shown in Fig. 6(a) in red curves. For 100,000 training samples, the testing error of RotEqNet is 0.1391, and the testing error of the baseline method is 0.2946, with 52.77% of error reduction. The performances of the last experiment group are marked as black curves in the figure, with only standard position trained for standard position predictor.



(a) Neural Networks (b) Random Forests

Fig. 6. Error of training with baseline model with random position, RotEqNet, and standard position predictor with the standard position for (a) Neural Networks and (b) Random Forests in the case study of large eddy simulation. Different colors represent different experimental groups. The RotEqNet model is trained with random positions and tested with random positions (red curves). Baseline models that trained and tested on raw data are shown as blue curves. The performances of standard position predictors that trained and tested with only standard positions are also shown as black curves. Training errors are shown with lines marked with triangles, testing errors are shown with lines marked with circles.

Table 2
Evaluation of Rotation-equivariant property between baseline model and RotEqNet.

Model	Baseline (NN)	RotEqNet (NN)	Baseline (RF)	RotEqNet (RF)
$\mathcal{E}_M$	0.2804	1.8895×10^{-16}	0.3264	**0.0**
$\mathcal{E}_D$	0.2029	**0.005**	0.4799	0.2090

Based on the experimental results, firstly, RotEqNet could reach a better learning performance compared to simply applying Neural Networks (baseline method). Training with standard positions could lower the training difficulty, and therefore RotEqNet could obtain better performance. Further, Fig. 6(b) shows the error reduction effect of RotEqNet using Random Forest as a standard position predictor. The general performance of using Random Forests as a standard position predictor is relatively worse compared to using Neural Networks as a standard position predictor. In Fig. 6(b), blue curves represent the performance of training with raw data by Random Forests (baseline method); red curves represent the performance of RotEqNet; black curves represent the performance of standard position predictor trained with standard positions. We could observe an error reduction for 36.63% in training and 57.58% in testing for RotEqNet with Random Forests.

Moreover, RotEqNet has a good generalization result without overfitting. Applying raw data in learning directly on baseline models, the testing error is much higher compared to the training error. For example, the difference is training and testing errors are 0.0068 for Neural Networks, 0.1068 for Random forests when $N = 100,000$. It is also observable in Fig. 6(a) that the training error of the baseline model with raw data is the lowest, while the testing error of the baseline model is the highest. In this case study, Neural Networks are worse for the sake of overfitting compared to Random Forests. By contrast, introducing the architecture RotEqNet would help to reduce this difference in training and testing error. As we could observe from the training and testing error of RotEqNet, the errors are much lower. When $N = 100,000$, there are only 0.0046 for Neural Networks and 0.0022 for Random Forests. In this case study of LES, the application of RotEqNet would result in better-generalized results in learning.

As shown in Table 2, for both baseline methods, using neural networks and random forests, there are large error for $\mathcal{E}_M$ and $\mathcal{E}_D$. The baseline methods have no theoretical guarantee that it has the property of rotation-equivariant. However, there is an error reduction for both machine learning models when applying with RotEqNet's architecture. For RotEqNet with Neural Networks as standard position predictor, the errors are $\mathcal{E}_M = 1.8895 \times 10^{-16}$ and $\mathcal{E}_D = 0.005$; for RotEqNet with Random Forests as standard position predictor, the errors are $\mathcal{E}_M = 0.0$ and $\mathcal{E}_D = 0.2090$. This very low error of $\mathcal{E}_M$ for RotEqNet could guarantee the rotation-equivariant property for nonlinear second-order cases.

Fig. 7 is a visualization of our tested case with contour plot of RotEqNet and baseline models. These contour plots are resulted from changing $G_{0,0}$ using range $(0, 1)$. The color map is representing the numerical result of $\tau_{0,0}$. RotEqNet is reaching lower error with similar contour pattern compared to true data. We could observe that the plotting of RotEqNet (the top middle plotting) has identical trend and numerical range to true data. The plotting of baseline method (the bottom middle plotting) is mainly predicting values in range $[10, 15)$, which causes its color to be blue. We could make a further conclusion that RotEqNet is performing better in this case of second-order non-linear flow simulation.

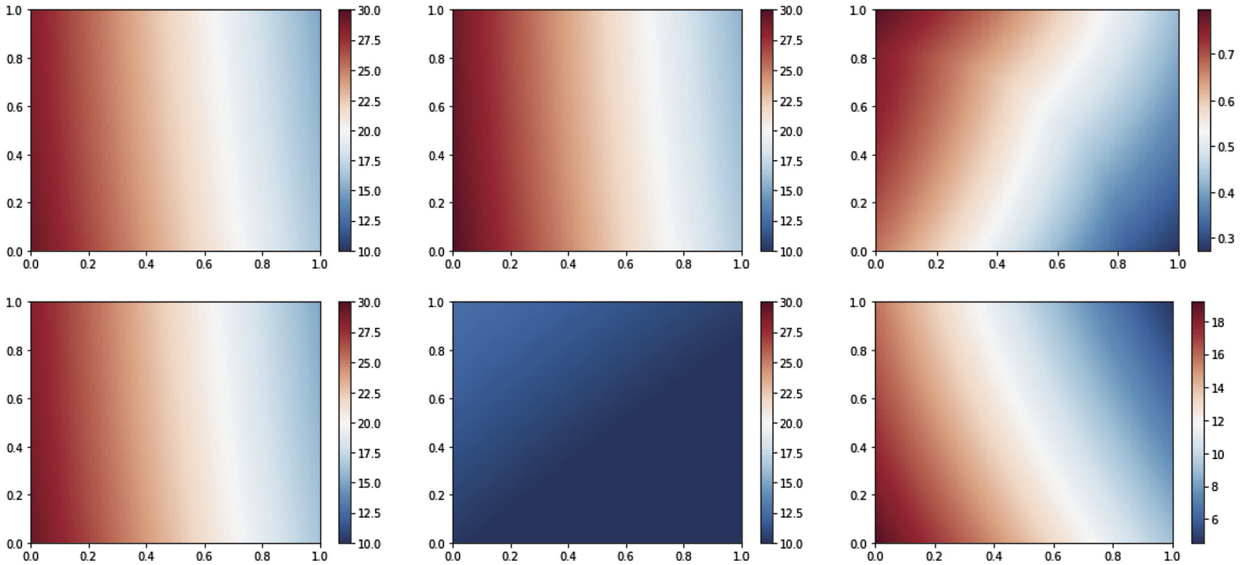


Fig. 7. Comparison of contour plot with RotEqNet and baseline method for the second-order nonlinear case study. First row: RotEqNet, second row: baseline (Neural Network). Left column: true data, middle column: predicted solution, right column: L1 difference between the true data and prediction.

4.4. Case study from testing Newtonian fluid equation: a third-order case

4.4.1. Problem statement and data generation

In this section, we study the performance of RotEqNet for tensor with odd-order. In this case, we specifically set a third-order test equation. We used a test equation here revised from Newtonian fluid equation from Eqn. (19). We name this testing equation as 'testing Newtonian fluid equation' for simplicity. The testing equation revised from Newtonian fluid equation can be described as:

$$\sigma = -pI + \mu(\nabla v + \nabla v^T), \tag{21}$$

where $\sigma \in \mathbb{R}^{3\times3\times3}$ is testing stress, $p \in \mathbb{R}$ is testing pressure, and $v \in \mathbb{R}^{3\times3\times3}$ is testing velocity field. $I \in \mathbb{R}^{3\times3\times3}$ is the identity third-order tensor.

Based on this testing equation, we could observe that $(\nabla v + \nabla v^T)^T = \nabla v + \nabla v^T$. Since $\nabla v + \nabla v^T$ is symmetric, and I is symmetric, we have σ is also symmetric. Therefore, defining an arbitrary rotation matrix R, this system is equipped with the property of rotation-equivariant that $R(\sigma) = R(-pI + \mu(\nabla v + \nabla v^T))$.

In order to qualify stress for testing the Newtonian fluid equation, this study aims to predict the stress, given the simulation of pressure and velocity gradient tensor. This case study for the machine learning model on inputting the pressure and velocity gradient tensor.

Based on Eqn. (21), we first generate random data to obtain ∇v and p. The generation of random numbers in ∇v follows a normal distribution from range $(0, 1)$. σ could be obtained using the Eqn. (21), derived from generated ∇v and p. Denote the dataset as, $D = \{x_i, y_i\}_{i=1}^N$. To form a proper dataset D with N elements for a machine learning model for Newtonian fluid, the input x is set up to be a vector where $x \in \mathbb{R}^{28}$. Specifically, x is composed by p and flattened $(\nabla v + \nabla v^T)$ in Eqn. (21). The output $y \in \mathbb{R}^{27}$ is a vector which is the flattened result of matrix σ. The dataset D would set up in the description above. To compare the difference of our method to the baseline method, we trained two models with the same hyper-parameter using different amounts of training data, ranging from $10,000, 20,000, \ldots, 100,000$. 85% of generated data is used for training and 15% of data is used for testing. A rotation set with 10,000 random rotation matrices is also generated for evaluating the property of rotation-equivariant, denoted by $\{R_i\}_{i=1}^{10000}$.

4.4.2. Results

Fig. 8(a) shows the error reduction effect of RotEqNet with Neural Network as a standard position predictor for third-order cases with three groups of experimental groups. The first experiment group focuses on the accuracy of the baseline model (Neural Network) on raw data with random rotated positions as shown in Fig. 8(a) with blue curves. The second experiment group is RotEqNet, with Neural Network as standard position predictor as shown in Fig. 8(a) in red curves. For 100,000 training samples, the testing error of RotEqNet is 1.6770 and the testing error of baseline method is 2.4723 with 32.17% of error reduction. The performances of the last experiment group are marked as black curves in the figure, with only standard position trained for standard position predictor.

Based on the experimental results, for the third-order testing equation, RotEqNet could reach a better learning performance compared to the baseline method. Training with RotEqNet could lower the training difficulty, and therefore RotEqNet

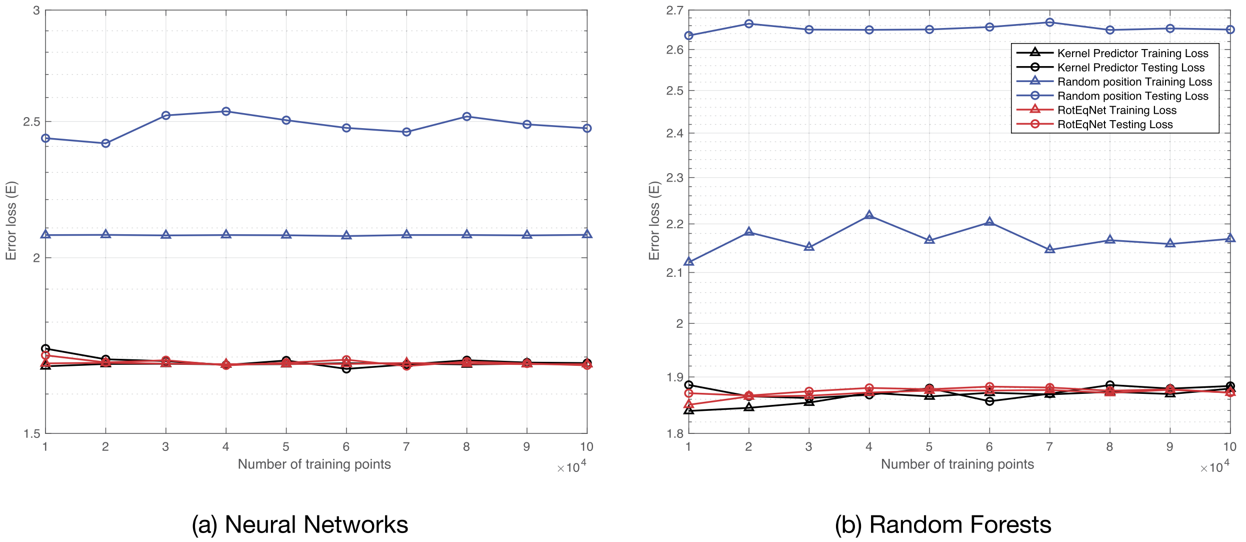


Fig. 8. Error of training with baseline model with random position, RotEqNet, and standard position predictor with the standard position for (a) Neural Networks and (b) Random Forests in the case study of testing Newtonian Fluid equation. Different colors represent different experimental groups. The RotEqNet model is trained with random positions and tested with random positions (red curves). Baseline models that were trained and tested on raw data are shown as blue curves. The performances of standard position predictors that trained and tested with only standard positions are also shown as black curves. Training errors are shown with lines marked with triangles, testing errors are shown with lines marked with circles.

Table 3
Evaluation of Rotation-equivariant property between baseline model and RotEqNet.

Model	Baseline (NN)	RotEqNet (NN)	Baseline (RF)	RotEqNet (RF)
$\mathcal{E}_M$	1.0526	7.9563×10^{-30}	0.1410	$\mathbf{3.2561 \times 10^{-30}}$
$\mathcal{E}_D$	2.8454	2.6992	3.0788	**0.4085**

could obtain better performance. Moreover, RotEqNet has good generalization capability without overfitting. As shown in the blue curves of Fig. 8, if we apply raw data in learning directly on baseline models, the testing error is much higher compared to the training error. In this case study, introducing the architecture RotEqNet would help to reduce this difference in training and testing error. As we could observe from the difference of training and testing error of RotEqNet, the errors are much lower. When $N = 100,000$, the differences of RotEqNet for training and testing errors are only 0.0033 for Neural Networks and 0.0002 for Random Forests. In this case study of testing the Newtonian fluid equation, the application of RotEqNet would result in better-generalized results in learning.

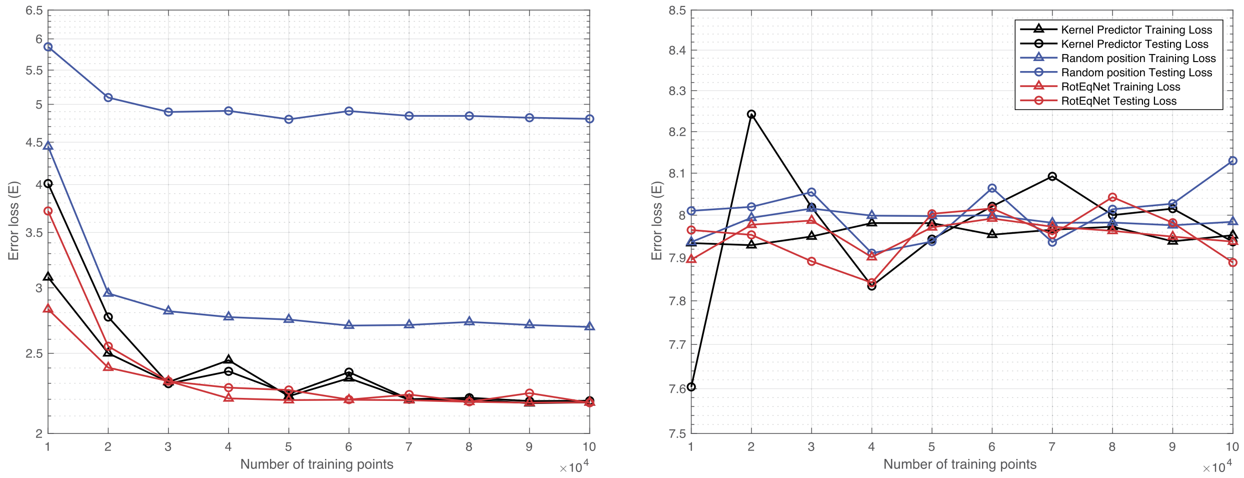
Further, Fig. 8(b) shows the error reduction effect of RotEqNet using Random Forest as a standard position predictor. The general performance of using Random Forests as a standard position predictor is relatively worse compared to using Neural Networks as a standard position predictor. In Fig. 6(b), blue curves represent the performance of training with raw data by Random Forests (baseline method); red curves represent the performance of RotEqNet; black curves represent the performance of Random Forest trained with standard positions. For the first point, we could observe an error reduction for 0.90% in training and 6.84% in testing for RotEqNet with Random Forests. As another point, RotEqNet is also obtaining a better-learned model for the model's capability in generalization. The testing error of the baseline method is observably higher than testing error, while the training and testing performance of RotEqNet is approximately the same. As suggested in Fig. 8(a), in second-order nonlinear cases, RotEqNet could reach a generalized learning result with remarkably lower error compared to baseline methods.

As shown in Table 3, for both baseline methods, using neural networks and random forests, there are large error for $\mathcal{E}_M$ and $\mathcal{E}_D$. The baseline methods have no theoretical guarantee that it has the property of rotation-equivariant. However, there is an error reduction for both machine learning models when applying with RotEqNet's architecture. For RotEqNet with Neural Networks as standard position predictor, the errors are $\mathcal{E}_M = 7.9563 \times 10^{-30}$ and $\mathcal{E}_D = 2.6992$; for RotEqNet with Random Forests as standard position predictor, the errors are $\mathcal{E}_M = 3.2561 \times 10^{-30}$ and $\mathcal{E}_D = 0.4085$. This very low error of $\mathcal{E}_M$ for RotEqNet could guarantee the rotation-equivariant property for third-order linear cases.

4.5. Case study from electrostriction: a fourth-order case

4.5.1. Problem statement and data generation

This case study focuses on a linear relationship of fourth-order tensor. Nye [39] has introduced a fourth-order tensor in modeling elastic compliances and stiffnesses, which has been investigated using machine learning methods [40,41]. Gen-



(a) Neural Networks (b) Random Forests

Fig. 9. Error of training with baseline model with random position, RotEqNet, and standard position predictor with the standard position for (a) Neural Networks and (b) Random Forests in the case study of Electrostriction. Different colors represent different experimental groups. The RotEqNet model is trained with random positions and tested with random positions (red curves). Baseline models that trained and tested on raw data are shown as blue curves. The performances of standard position predictors that trained and tested with only standard positions are also shown as black curves. Training errors are shown with lines marked with triangles, testing errors are shown with lines marked with circles.

erally, in the study of the properties of a crystalline and anisotropic elastic medium, a fourth-order tensor coefficient will typically be applied to model the relationship between two symmetric second-order tensors [42]. In this case, we study Electrostriction, a property causing all electrical non-conductors to change their shape under the application of an electric field. The relationship is described as:

$$T_{ij} = S_{ijkl} V_{kl} \tag{22}$$

Here $T_{ij} \in \mathbb{R}^{3\times3}$ is a symmetric traceless second-order strain tensor. $S_{kl} \in \mathbb{R}^{3\times3}$, $S_{kl} = S_k S_l$ where V_k and V_l are first-order electric polarization density. Note here V_{kl} is symmetric. $V_{ijkl} \in \mathbb{R}^{3\times3\times3\times3}$ is the electrostriction coefficient.

Based on the formulation above, this system is symmetric. Since S_{ij} is symmetric, $T_{ij}^T = (V_{ijkl} S_{ij})^T = S_{kl} V_{klij} = T_{ij}$. This could guarantee that T_{ij} is also symmetric. Due to the face that the system is symmetric, applying a random rotation matrix R, $R(T) = R(VS)$.

In order to qualify strain for study on Electrostriction, we aim to predict the strain, given the simulation of electrostriction coefficient and electric polarization density.

Based on Eqn. (22), we first generate random data to obtain simulated electrostriction coefficient tensor V_{ijkl} and electric polarization density tensor S_{ij}. The generation of random numbers follows a normal distribution. T_{ij} is computed from above setting using V_{ijkl} and S_{ij}. Denote the dataset as, $D = \{x_i, y_i\}_{i=1}^N$. To form a proper dataset D with N elements for a machine learning model for the study on Electrostriction, the input x is set up to be a vector where $x \in \mathbb{R}^{90}$. Specifically, T_{ij} is composed by flattened V_{ijkl} and S_{ij}. The output $y \in \mathbb{R}^9$ is a vector, which is the flattened result of second-order tensor T. To compare the difference of our method to the baseline method, we trained two models with the same hyper-parameter using different amounts of training data, ranging from $10,000, 20,000, ..., 100,000$. 85% of generated data is used for training, and 15% of data is used for testing. A rotation set with 10,000 random rotation matrices is also generated for evaluating the property of rotation-equivariant, denoted by $\{R_i\}_{i=1}^{10000}$. We use Numpy to generate this simulated dataset by generating a random symmetric fourth-order tensor V, and second-order tensor S. T is computed from generated V and S by Eqn. (22).

4.5.2. Results

The effect of error reduction is evaluated for the first. The validation error $\mathcal{E}_{MS}$ is defined as the Mean Squared Loss using the formulation in Eqn. (16). This evaluation $\mathcal{E}_{MS}$ represents the expected error of model $\mathcal{M}$ with dataset D. Fig. 9 shows the performance of Neural Networks and Random Forests as standard position predictor separately. It is observable that in high-order cases, Neural Networks outperform Random Forests. Details will be demonstrated in the following paragraphs.

We are starting with Neural Networks, Fig. 9(a) shows the error reduction effect of RotEqNet with Neural Network as a standard position predictor. As shown in blue curves, the first experiment group focuses on the accuracy of the baseline model on raw data with random rotated positions. The second experiment group is RotEqNet marked with red curves. As shown in black curves, it shows the performance of the standard position predictor trained by standard position. For 10,000 training samples, the testing error of RotEqNet is 4.0106 and the testing error of baseline model is 8.6458 with 53.61% of

Table 4
Evaluation of Rotation-equivariant property between baseline model and RotEqNet.

Model	Baseline (NN)	RotEqNet (NN)	Baseline (RF)	RotEqNet (RF)
$\mathcal{E}_M$	0.0324	$\mathbf{3.5723 \times 10^{-35}}$	0.1410	2.9985×10^{-34}
$\mathcal{E}_D$	0.0951	**0.0202**	0.4241	0.0004

error reduction. The testing performance of the standard position predictor is only evaluated on the testing set with only standard positions. It will be helpful to explain the reason for the improved performance of RotEqNet.

To interpret the experimental results, firstly, RotEqNet could reach a better learning performance compared to simply applying Neural Networks (baseline method). A dataset with only standard positions has lower training difficulty compared to random positions. This claim is supported by black curves in Fig. 9(a), the performance of the standard position predictor is much better than the baseline model. RotEqNet could obtain better performance for utilizing rotation symmetry as a prior, and training standard position predictor with only standard positions. Moreover, RotEqNet has a good generalization result without clear overfitting. The training error and testing error of RotEqNet is considerably close to each other, and sometimes, the testing error of RotEqNet is even slightly better than its training error. By contrast, applying raw data in learning directly on $M_{baseline}$ would result in an overfitted model. The testing error is much higher compared to the training error. To demonstrate the improved learning result in generalization, for example, when $N = 100,000$, the difference between training and testing errors for RotEqNet is only 0.0024 while the difference of the baseline method is 2.1118. As a quick conclusion, for Neural Networks as a standard position predictor, the application of RotEqNet would be better compared to the baseline method.

Further, Fig. 9(b) shows the error reduction effect of RotEqNet using Random Forest as a standard position predictor. At first glance, we could find that the curves for Random Forests are quite messy without certain patterns like Fig. 9(a). The general performance of using Random Forests as a standard position predictor is worse in both aspects of performance and generalization. We could observe a training error reduction for 0.58% and testing error reduction of 2.96%. Even if we could still see the general error of RotEqNet seems to be slightly lower than the baseline method. This result is not comparable to the error reduction performance with setting Neural Networks as a standard position predictor. As another point, selecting Random Forests as a standard position predictor fails to extract learning rules with the standard position. As we could observe the black curves in Fig. 9(b) is not showing an improved performance as good as using Neural Networks. Finally, the learned model of RotEqNet is also not getting a model with better generalization capability. There is no significant reduction of overfitting error compared to the baseline method.

As shown in Table 4, for both baseline methods, using neural networks and random forests, there are large error for $\mathcal{E}_M$ and $\mathcal{E}_D$. The baseline methods have no theoretical guarantee that it has the property of rotation-equivariant. However, there is an error reduction for both machine learning models when applying with RotEqNet's architecture. For RotEqNet with Neural Networks as standard position predictor, the errors are $\mathcal{E}_M = 3.5723 \times 10^{-35}$ and $\mathcal{E}_D = 0.0202$; for RotEqNet with Random Forests as standard position predictor, the errors are $\mathcal{E}_M = 2.9985 \times 10^{-34}$ and $\mathcal{E}_D = 0.0004$. This very low error of $\mathcal{E}_M$ for RotEqNet could guarantee the rotation-equivariant property for fourth-order linear cases.

5. Discussion and conclusion

The large error reduction observed in case studies raises new opportunities in solving the problem of the physical system with rotation symmetry. Most physical systems have the property of rotation symmetry, and currently, there exist few works that could provide a theoretical guarantee to this property for machine learning methods. A key point in this problem is to design a properly defined algorithm to obtain rotation invariant for high-order tensors. This paper has demonstrated RotEqNet with theoretical and experimental results aiming to solve the problem of rotation symmetry for high-order fluid systems.

In our work, we define the position standardization algorithm to extract rotation-invariants, which is compatible for high-order tensors. It allows us to utilize the rotation-invariants of high-order tensors using contraction with diagonalization, and QR factorization. The theoretical guarantee is shown in Theorem 1, and the algorithm is shown in Algorithm 1. RotEqNet is built on Algorithm 1 with a standard position predictor which only deals with rotation-invariants. By setting kernel predictor with Neural Networks and Random Forests, these two methods are compared with baseline methods in four different case studies focusing on second-order linear, second-order nonlinear, third-order linear, and fourth-order linear cases.

There are three conclusions to address from the observation of case studies.

First, the design of the position standardization algorithm is trustworthy. We aim to define a position standardization algorithm to extract standard position, simplifying the learning task to a smaller function space. In our case, the position standardization algorithm can truncate infinite rotation group into a finite group. We could observe in most of the cases, training kernel predictors with only standard positions could reach lower error in the MSE sense.

Second, RotEqNet is guaranteed to have the property of Rotation-equivariant with finite samples. As we could observe from the results of case studies, the rotation error $\mathcal{E}_M$ of RotEqNet could reach machine precision. The perseverance of the property of Rotation-equivariant shows proves Theorem 2. Further, RotEqNet will have a lower test error. Under this situation, adding with the property of Rotation-equivariant, this would cause RotEqNet could generalize this system with any rotation.

The two conclusions above are causing the error reduction for RotEqNet. Since we may have a random sign group U for high-order even tensors, Neural Networks is the best model because of its flexibility to approximate arbitrary functions. We only reported the performance of Neural Networks and Random Forests following previous work of Ling [15]. As described in Sec. 4.5.2, the performance of Random Forests is limited compared to Neural Networks. Also, as a general trend in previous experiments, Neural Networks are usually reaching better performance compared to Random Forests. In conclusion, we believe the application of Neural Networks as a kernel predictor has a series of advantages than other machine learning models.

We would like to mention that we only apply relatively small machine learning models and shallow neural networks. Since applying deeper models could also help to have better performances considering rotation-equivariance. In accordance to our main focus, we only use small models to avoid possible lurking effects by deep models. The application of deeper models will certainly help to allow the flow simulation to reach the state-of-the-art, which is a feasible extension for future applications.

For future work, there are mainly three directions to contribute: a better definition of standard position, application to other groups, and generalization to non-symmetric systems. For the first direction, for the current definition, the rotation invariant of odd-order tensors is not reaching equivalent performance as even-order tensors. It would be a good work for revising the definition of standard position for odd tensors. Second, besides rotation symmetries, there are also physical systems with other group-equivariant properties such as scaling and transaction. This work could provide a method in solving problems with other groups, but the detailed design of an algorithm should differ from case to case. Third, current work could only deal with the symmetric system. However, for a general case, if the system is not symmetric, there are certain methods to use RotEqNet in a symmetric system for solving a non-symmetric system. One of the possible approaches is to deal with PP^T. This is a good intuition to extend our current work into non-symmetric physical systems.

CRediT authorship contribution statement

Liyao Gao, Yifan Du and Hongshan Li conceived the mathematical models, implemented the methods, designed the numerical experiments, interpreted the results, and wrote the paper. Guang Lin supported this study and reviewed the final manuscript. All authors gave final approval for publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

G. Lin gratefully acknowledges the support of the National Science Foundation (DMS-1555072, DMS-1736364, DMS-2053746, and DMS-2134209), and Brookhaven National Laboratory Subcontract 382247, and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0021142.

Appendix A. Preliminaries

A.1. Tensor and its operations

In this section, we first introduce an abstract way in defining tensor. One reason for us to introduce the more abstract way to think about tensors is that it provides a convenient formalism for the operations we will be doing on the tonsorial data discussed in the previous section. The operations are

- Linear transformation
- Contraction

It will also help us proving that these two operations commute, which will be made precise in the subsequent sections. The commutativity of these operations will be used to define and compute a representative among tensorial data that are obtained by rotation.

A.1.1. Abstract definition of tensors

Following [43], fix a vector space V of dimension n over $\mathbb{R}$. A *tensor product* $V \otimes V$ is a vector space with the property that $\mathbb{R}$-bilinear maps $V \times V \to \mathbb{R}$ are in natural one-to-one correspondence with $\mathbb{R}$-linear maps $V \otimes V \to \mathbb{R}$.

The tensor product $V \otimes V$ can be constructed as the quotient vector space $V \times V/C$, where C is generated by vectors of the following types

$$(i)\ (x+y,z)-(x,z)-(y,z) \quad (ii)\ (x,y+z)-(x,y)-(x,z) \quad (iii)\ (ax,y)-a(x,y) \quad (iv)\ (x,ay)-a(x,y) \tag{A.1}$$

where x and y are vectors in V and a is a scalar in $\mathbb{R}$. This means any element in C can be written as a linear combination of vectors of the above form. C is not necessarily a vector space of finite dimension. But the quotient space $V \otimes V$ is. Let $g : V \times V \to V \otimes V$ be the natural projection map, then we use $x \times y$ to denote the image of (x, y) under g.

Let $\langle e_1, \cdots, e_n \rangle$ be a basis of V, then $e_i \otimes e_j$ for $i = 1, ..., n$ and $j = 1, ..., n$ form a basis of $V \otimes V$. This means any vector $p \in V \otimes V$ can be written as

$$\sum_{i,j} a_{ij} e_i \otimes e_j \tag{A.2}$$

for some $a_{ij} \in \mathbb{R}$.

Here are some relations of tensors which come directly as a consequence of the relations generating C:

$$a(e_i \otimes e_j) = ae_i \otimes e_j = e_i \otimes ae_j \tag{A.3}$$

$$(a_i e_i + a_j e_j) \otimes (a_k e_k) = a_i a_k (e_i \otimes e_j) + a_j a_k (e_j \otimes e_k) \tag{A.4}$$

The representation of a tensor in $V \otimes V$ is similar to the representation of a linear map $V \to V$, i.e. a matrix. In fact, there is a natural way to think of a tensor as a linear map:

For each element $e_i \otimes e_j$ in the basis of $V \otimes V$, we can think of it as a linear map $V \to V$ by defining $e_i \otimes e_j(v) = e_i < e_j, v >$, where $<, >$ is the natural inner product on V. Extend the definition linearly to every element in $V \otimes V$, we obtain a way to identify $V \otimes V$ as the space of linear map $V \to V$. In fact, the tensor $\sum_{i,j} a_{i,j} e_i \otimes e_j$ corresponds to the linear map represented by the matrix $[a_{ij}]$.

We have defined the tensor product $V \otimes V$ over V. The definition/construction of order k tensor $\overbrace{V \otimes \cdots \otimes V}^{k}$ follows the same course. We will denote order k tensor by $\otimes^k V$.

The basis of $\otimes^k V$ is given by $e_{i1} \otimes \cdots \otimes e_{ik}$, where $ij = 1, ..., n$ and $j = 1, ..., k$. With respect to this basis, any order k tensor can be written as $\sum_{i1,\cdots,ik} a_{i1,...,ik} e_{i1} \otimes \cdots \otimes e_{ik}$. Analogous to the order 2 case, we can think of an order k tensor as a k-dimensional matrix, the typical way a tensor in physical experiments are represented.

We will use T^k to denote a tensor of order k, i.e. a vector in $\otimes^k V$. k is called the rank of the tensor.

A.1.2. Rotation on tensors: a linear transformation

A linear transformation on higher-order tensor is a generalization of a linear transformation on first-order tensor, i.e. a vector. Rotation is a special kind of linear transformation where $R \in SO(n)$. An important connection in our paper is to under rotation operation on tensors in matrix form.

Let $g : V \to V$ be a linear transformation. Use the basis $\langle e_1, \cdots, e_n \rangle$ of V, we can represent by

$$g(e_i) = \sum_{j=1}^{n} a_{ij} e_j \tag{A.5}$$

Let $M(g)$ denote the matrix representation of g with respect to the basis $\langle e_1, \cdots, e_n \rangle$. Then

$$M(g) = [a_{ij}]^t \tag{A.6}$$

i.e. the transpose of the matrix $[a_{ij}]$

The map g naturally induces a map $\otimes^k g$ on $\otimes^k V$. On the basis element $e_{i1} \otimes \cdots \otimes e_{ik}$, the action of $\otimes^k g$ is defined as

$$e_{i1} \otimes \cdots \otimes e_{ik} \mapsto g(e_{i1}) \otimes \cdots \otimes g(e_{ik}) \tag{A.7}$$

For any tensor $T \in \otimes^k V$, we will use $g(T)$ to denote the extension of g on $\otimes^k V$

There is a convenient way to represent linear transformation of 2-tensor as matrix multiplication.

For a 2-tensor $T = \sum_{i,j} b_{ij} e_i \otimes e_j$, use $M(T)$ be the matrix whose (i, j) term is b_{ij}.

Lemma A.1. *Rotation operation by matrix R on second-order tensor (matrix) is a change of basis operation.*

$$M(R(T)) = M(R) \times M(T) \times M(R)^t, \tag{A.8}$$

where $\times$ here means the usual matrix multiplication.

Proof. We will use column vector convention to represent vectors in V. Let v_1 and v_2 be vectors in V. Then

$$M(v_1 \otimes v_2) = M(v_1) \times M(v_2)^t \tag{A.9}$$

Then,

$$M(R(v_1 \otimes v_2)) = M(R(v_1) \otimes R(v_2)) \tag{A.10}$$
$$= M(R(v_1)) \times M(R(v_2))^t \tag{A.11}$$
$$= M(R) \times M(v_1) \times M(v_2)^t \times M(R)^t \tag{A.12}$$
$$= M(R) \times M(v_1 \otimes v_2) \times M(R)^t \tag{A.13}$$

Therefore,

$$M(R(T)) = M(R) \times M(T) \times M(R)^t \quad \square \tag{A.14}$$

Remark A.2. Rotation operation by matrix R on first-order tensor (vectors) T could be viewed as

$$M(R(T)) = M(R) \times M(T). \tag{A.15}$$

Proof. We will use column vector convention to represent vectors in V. Let v_1 be vector in V. Then

$$M(R(v_1)) = M(R) \times M(v_1) \tag{A.16}$$

Therefore,

$$M(R(T)) = M(R) \times M(T) \quad \square \tag{A.17}$$

As we have shown in this subsection, one could use matrix form of rotation operation with certain rules of matrix multiplication to perform a rotation on tensor. In the following proofs of this paper, we applied this idea to perform rotation operation on tensors via matrix multiplication.

A.1.3. Contraction on tensors: reduction of order

Let $\langle , \rangle$ be the standard inner product on V. Using this inner product, we can define the contraction of a tensor. It "merges" vectors on the specified axes using the inner product and reduces the rank of the tensor by 2. Formally, let $C(a,b)$ denote the contraction along axis-a and axis-b. Here, the axis means the ordinal of V in $\otimes^k V$. For example, axis-1 refers to the first copy of V in $\otimes^k V$.

On the element $\otimes_{j=1}^k v_{ij}$, $C(a,b)$ acts on it by pairing v_{ia} and v_{ib} via the inner product $\langle , \rangle$, i.e.

$$C(a,b)(v_{i1} \otimes \cdots \otimes v_{in}) = \langle v_{ia}, v_{ib} \rangle v_{i1} \otimes \cdots \check{v}_{ia} \cdots \check{v}_{ib} \cdots \otimes v_{in} \tag{A.18}$$

where $\check{v}$ means v is not present.

We can then define $C(a,b)$ on $\otimes^k V$ by extending linearly. When $k = 2$, contraction is nothing other than taking the trace of the corresponding matrix.

Lemma A.3. *Let $R : V \to V$ be a rotation. Let $T \in \otimes^k V$, then*

$$C(a,b)(R(T)) = R(C(a,b)(T)) \tag{A.19}$$

Proof. Since both $C(a,b)$ and g are linear, we may assume that T is of the form $v_{i1} \otimes \cdots \otimes v_{in}$.

$$C(a,b)(g(T)) = C(a,b)(g(v_{i1}) \otimes \cdots \otimes g(v_{in})) \tag{A.20}$$
$$= \langle g(v_{ia}), g(v_{ib}) \rangle g(v_{i1}) \otimes \cdots g(\check{v}_{ia}) \cdots g(\check{v}_{ib}) \cdots \otimes g(v_{in}) \tag{A.21}$$

Since g is a rotation, it preserves the inner product i.e.

$$\langle g(v_{ia}), g(v_{ib}) \rangle = \langle v_{ia}, v_{ib} \rangle \tag{A.22}$$

So

$$C(a,b)(g(T)) = C(a,b)(g(v_{i1}) \otimes \cdots \otimes g(v_{in})) \tag{A.23}$$
$$= \langle v_{ia}, v_{ib} \rangle g(v_{i1}) \otimes \cdots g(\check{v}_{ia}) \cdots g(\check{v}_{ib}) \cdots \otimes g(v_{in}) \tag{A.24}$$
$$= g(C(a,b)(T)) \quad \square \tag{A.25}$$

Lemma A.3 shows an interesting connection between rotation operation and contraction. To understand this lemma, it represents that contraction of tensor is compatible with a linear transformation if this linear transformation is a rotation. This is an important lemma which is the foundation of entire analysis in this paper. We would further utilize this lemma for extracting its rotation operation from higher (arbitrary) orders.

A.2. Supervised learning setup

In our problem, given data set $\mathcal{D} = \{X_i; y_i\}_{i=1,\ldots,N}$. The data set contains N input-output pairs $(X_i; y_i)$. The input here is a tensor tuple:

$$X_i = [X_1, X_2, \ldots, X_{N_x}] \tag{A.26}$$

N_x, is the length of X_i. Normally, we only have one output.

Generally speaking, following the definition of [44,45], parametric supervised learning can be viewed as a type of a model composed from two parts. The first part is a predictor. Given a model $\mathcal{M}$, we have:

$$\hat{y} = \mathcal{M}(X_i) \tag{A.27}$$

where $\hat{y}$ is the prediction output of learning model $\mathcal{M}$. As stated, it predicts value based on input X_i.

The second part is an optimizer, which updates the model based on a loss function. A typical loss function would be defined as:

$$L(\mathcal{M}) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \mathcal{M}(X_i)\|^2, \tag{A.28}$$

where $\|\cdot\|$ represents 2-norm.

We hope to minimize this loss function. It is formulated by:

$$\min_{\mathcal{M}\in\mathcal{H}} L(\mathcal{M}). \tag{A.29}$$

In our work, we applied Neural Networks from parametric family [46] and Random Forests from nonparametric family [47].

A.3. Modeling symmetric fluid systems via supervised learning

The machine learning approach to the fluid dynamics modeling is to train a supervised learning model $\mathcal{M}$ using X_i as features and Y as label.

In our case, the physical model F is complete with respect to rotation. This means for all rotation $R: \mathbb{R}^n \to \mathbb{R}^n$, $R(p) \in F$ for all $p \in F$. In this case, we can study the tensor fields on F that are rotation equivariant. Those are tensor fields $(X_1, \cdots, X_n, Y)$ such that

$$X_1(R(p), \cdots, X_n(R(p)), Y(R(p)) = R(X_1(s)), \cdots, R(X_n(s)), R(Y(s)) \tag{A.30}$$

Then underlying physics law relating X's to Y is rotation-equivariant, because

$$f(R(X)) = R(Y) = R(f(X)), \tag{A.31}$$

X as a short-hand for $X_1, \cdots, X_n$.

Appendix B. Proof of Lemma 2.4

Proof. We can always construct two steps of learning by first learning $\mathcal{H}$ and then learning the group action G.

By construction, we know that $\mathcal{H}$ is learnable with finite samples.

If group G_n has finite cardinality that $|G_n| = n$, we know the VC dimension of G_n is at most n since it could be shattered by $\log(n)$ subsets in maximal [31].

From this construction, using Theorem 2 in [48], we further know this group transformation G is (ϵ, δ)-learnable with sample size N upper bounded by

$$N = O\left(\frac{\log(n) + \ln\left(\frac{1}{\delta}\right)}{\epsilon}\right), \tag{B.1}$$

where ϵ, δ are constants to control the error.

As long as $|G_n| \le \infty$, since G_n and $\mathcal{H}$ are both learnable with finite samples, we know their combination $G_n(\mathcal{H})$ is still learnable with finite samples.

If group G_∞ has infinite cardinality that $|G_\infty| = \infty$, we know that G cannot always be learned with finite sample from the same derivation above. This will result in $G_\infty(\mathcal{H})$ not learnable with finite sample. $\square$

Appendix C. Note of eigenvectors are nonunique

Consider an eigenvector v of matrix A. By definition, we have

$$Av = cv, \quad c \in \mathbb{R}. \tag{C.1}$$

For $-v$,

$$A(-v) = c(-v) \iff Av = cv. \tag{C.2}$$

We know both v and $-v$ is an eigenvectos of A. Therefore, for $A \in \mathbb{R}^{3\times 3}$, consider a eigen-decomposition that $A = RDR^T$, there will be 2^3 ways of eigenvectors R, with random sign applied. In our implementation, we always fix the first row of R to be positive. This process guarantees algorithm S is well-defined.

Appendix D. Proof of Lemma 3.1

Proof. We discuss two situations separately in this proof: **second-order and odd orders** and **even orders** (≥ 4).

odd-order tensors. In position standardization algorithm S, the derivation of odd-order tensors are via QR factorization with sorting eigenvalues. In the following, we will show why the standard position of T through the entire orbit $R(T)$ is unique, equivalently as $|\Phi(T)| = 1$.

Suppose T has odd order. Let C be the sequence of contraction along the first two axes such that $C(T) = T^3$, where T^3 is a third-order tensor as described in the algorithm.

$$T^3 = C(T^n) \tag{D.1}$$

Let V_1, V_2, V_3 be vectors of contraction operation on T^3 via different axes, *i.e.*,

$$V_1 = C(2,3)(T^3) \qquad V_2 = C(1,3)(T^3) \qquad V_3 = C(1,2)(T^3) \tag{D.2}$$

Based on S, we have

$$[V_1 \;\; V_2 \;\; V_3] = R_1 \times U_1 \tag{D.3}$$

In this case,

$$S(T^n) = R_1^{-1}(T^n) \tag{D.4}$$

Consider any rotation operation P acting on T^n. We have,

$$P(T^3) = P(C(T^n)) \tag{D.5}$$

Using QR-factorization,

$$[P \times V_1 \;\; P \times V_2 \;\; P \times V_3] = R_2 \times U_2 \tag{D.6}$$

The standard position of $P(T^n)$ will be defined as:

$$S(P(T^n)) = R_2^{-1}(P(T^n)) \tag{D.7}$$

Using Remark A.2, we could obtain

$$C(2,3)(P(T^3)) = P \times C(2,3)(T^3) = P \times V_1 \tag{D.8}$$

Considering V_2 and V_3, for the same reason, we could know that

$$[P \times V_1 \;\; P \times V_2 \;\; P \times V_3] = P \times [V_1 \;\; V_2 \;\; V_3] \tag{D.9}$$

By reorganizing (D.3), (D.6), and (D.9),

$$[V_1 \;\; V_2 \;\; V_3] = R_1 \times U_1 = P^{-1} \times R_2 \times U_2 \tag{D.10}$$

The QR factorization is not unique with random permutations. We could follow Gram-Schmidt process to remove the permutation group. After this step, QR-factorization is unique. Therefore, we have $U_1 = U_2$. Then,

$$R_2 = P \times R_1 \tag{D.11}$$

Plugging (D.11) into (D.7), comparing the result of (D.4) we have:

$$S(P(T^n)) = R_2^{-1}(P(T^n)) = (R_1^{-1} \times P^{-1} \times P)(T^n) = R_1^{-1}(T^n) = S(T^n) \tag{D.12}$$

At this point, with a random rotation P, we will still have $S(P(T^n)) = S(T^n)$. This means that

$$\Phi(T^n) = \{R_1^{-1}(T^n)\} \to |\Phi(T^n)| = 1. \tag{D.13}$$

This part shows Lemma 3.1 when T has odd-orders.

Even-order tensors. Then we show the even-order cases. Notice that position standardization algorithm relies on eigen-decomposition. Like QR factorization, eigen-decomposition is also not naturally unique. Specifically, consider the decomposition of an arbitrary symmetric matrix $A = RDR^T$. There are various options for R with permutation and random sign. In the following analysis, we wish to show the algorithm S is still able to obtain a finite set with $|\Phi(T)| = 8$. Here, Φ is equivalent as a group action U: a random column sign group which determines the sign of each column of R.

Suppose T has even order. Let C be the sequence of contraction along the first two axes such that $C(T) = T^2$, where T^2 is a second-order tensor as described in the algorithm.

Given arbitrary even high order tensor T, we could perform contraction to a second order tensor T^2 via first two indices:

$$T^2 = C(T^n) \tag{D.14}$$

For T^2, using Lemma A.1, there exists a rotation R such that:

$$T^2 = R(T_s^2). \tag{D.15}$$

Here, R is not unique. We see this from the process of eigen-decomposition. The non-uniqueness is caused by two parts. First, the order of eigenvalues could be random. We can select any order of eigenvalues which will result in different permutations of eigenvectors. Second, the sign of each eigenvector is not specified. A eigenvector only need to satisfy $Av = av$ for matrix A and scalar a. If v is an eigenvector, $-v$ is also an eigenvector. A random sign could be added to each eigenvector.

In position standardization algorithm, we remove the permutation group action by making the eigenvalues ordered. This operation commutes for higher orders.

The second source of non-uniqueness, random sign group, will bring non-uniqueness of standard position for high-order even tensors. Before the proof of the second part, we define this sign group U as

$$U = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \ldots, \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \right\}. \tag{D.16}$$

There are 8 elements in U and let $U_1 = I_{3\times 3}, ..., U_8 = -I_{3\times 3}$. U is a group.

Suppose a tensor $R_0(T)$ rotated from T with rotation R_0. Using eigen-decomposition, we cannot know the sign of R_0. In the position standardization algorithm, we fix R to be positive in the first row to make the algorithm well-defined. However, we can only guarantee the rotation obtained from this algorithm to be one element in set $U(R_0) = \{U_1R_0, U_2R_0, ..., U_8R_0\}$. This is troublesome since we may mistakenly set the standard position to $U_i(T)$ (which should be T instead).

Part 1. Second-order tensors will **not** be affected by random sign group.

Consider a symmetric second-order tensor T^2 which is a matrix. Applying random sign operation $\forall U_i$ on T^2, we have

$$U_i(T^2) = U_i \times T^2 \times U_i = T^2. \tag{D.17}$$

In second-order cases, the standard position will not be affected since applying U_i two times to T^2 will cancel the sign flipping effect.

Based on S the standard position $S(T)$ is defined as

$$S(T^2) = U_i(R^{-1}(T^2)) = R^{-1}(T^2) \tag{D.18}$$

For its standard position $S(P(T^2))$ we have:

$$S(P(T^2)) = (R^{-1} \times P^{-1} \times P)(T^2) = R^{-1}(T^2) = S(T^2) \tag{D.19}$$

To simplify, for a rotation operation P acting on an even high order tensor T,

$$S(P(T)) = S(T). \tag{D.20}$$

Therefore, we know that

$$\Phi(T^2) = \{S(T)\}, \quad |\Phi(T^2)| = 1. \tag{D.21}$$

Part 2. Higher even-order tensors **will be** affected by random sign group.

For higher order tensors, we know

$$U_i(T^n) \neq T^n. \tag{D.22}$$

Suppose the real standard position, $T_s^n = R^{-1}(T^n)$, based on S the standard position from algorithm $S(T)$ is

$$S(T^n) = U_i(R^{-1}(T^n)) = (U_i \times R^{-1})(T^n), \tag{D.23}$$

where U_i is a sign matrix.

Consider a rotation operation P acting on T. For this new tensor, applying contraction we could have:

$$C(P(T^n)) = P(T^2) = U_i((P \times R)(T_s^2)) \tag{D.24}$$

For its standard position $S(P(T))$, since U_i commutes, we know there exist $U_j \in U$ such that

$$S(P(T)) = U_j(R^{-1} \times P^{-1} \times P)(T) = U_j(R^{-1}(T)) = U_j(S(T)). \tag{D.25}$$

To simplify, for a rotation operation P acting on an even high order tensor T,

$$S(P(T)) = U_j(S(T)). \tag{D.26}$$

Therefore, the entire orbit after position standardization will form a set $\Phi(T^n)$ such that

$$\Phi(T^n) = \{U_1(T^n), U_2(T^n), U_3(T^n) ..., U_8(T^n)\}, \quad |\Phi(T^n)| = 8. \tag{D.27}$$

Finally, we show Φ is a group. When T is second or odd-order tensor, we know $\Phi = \{I\}$ which is a trivial group with identity. When T is a higher even-order tensor, the sign group $U = \Phi$. We know Φ is a group in both cases.

By summarizing all situations above, we complete the proof of this Lemma. $\square$

Appendix E. Proof of Theorem 1

Proof. By definition of realizable assumption, we know there exists $h^* \in H$ such that $L_{\mathcal{D},f}(h^*) = 0$. Since G_Φ extend the distribution functional with additional structure with group actions, we also have to increase the space of hypothesis class $G_\Phi(\mathcal{H})$ to guarantee realizable. Since Φ is a finite group, using Lemma 2.4, we know $G_\Phi(H)$ is finite learnable. $\square$

Appendix F. Proof of Theorem 2

Proof. Name RotEqNet as $\mathcal{M}_\mathcal{R}$, kernel classifier as $\mathcal{M}_\Phi$. Consider an input pair (X_i, y_i).

If X_i **has order two or odd-order**, suppose the result of standardize position algorithm S would have $S(X) = P_1(X)$, where P_1 denote a rotation operation.

First, based on the definition of $\mathcal{M}_\mathcal{R}$,

$$\mathcal{M}_\mathcal{R}(X_i) = P_1^{-1}(\mathcal{M}_\Phi(S(X_i))) = P_1^{-1}(\mathcal{M}_\Phi(P_1(X_i))). \tag{F.1}$$

Consider another rotation operation P_2 in the matrix form acting on X, using Lemma 3.1 we know that:

$$S(P_2(X_i)) = S(X_i) = P_1(X_i) = (P_1 I)(X_i) = (P_1 \cdot P_2^{-1})(P_2(X_i)). \tag{F.2}$$

Then, consider $\mathcal{M}_\mathcal{R}(P_2(X))$ the process of RotEqNet is defined as:

$$\mathcal{M}_\mathcal{R}(P_2(X)) = (P_1 P_2^{-1})^{-1}(\mathcal{M}_\Phi(S(P_2(X)))) = (P_2 P_1^{-1})(\mathcal{M}_\Phi(S(P_2(X)))) \tag{F.3}$$

We know that $S(P_2(X)) = S(X)$ from Eqn. (F.2). Therefore, we have $\mathcal{M}_\mathcal{R}(S(X)) = \mathcal{M}_\mathcal{R}(S(P_2(X)))$. Substitute $\mathcal{M}_\mathcal{R}(X)$ back into previous equation,

$$\mathcal{M}_\mathcal{R}(P_2(X)) = P_2 P_1^{-1}(\mathcal{M}_\Phi(S(X))) = P_2(\mathcal{M}_\mathcal{R}(X)) \tag{F.4}$$

We know if X_i has order two or odd-order, RotEqNet is rotation-equivariant.

If X_i **has even-order** ≥ 4, we need to show previous property still holds under an additional sign flipping group U.

Suppose the result of standardize position algorithm S would have $S(X) = V_1 \cdot P_1(X)$, where P_1 is solved via position-standardization algorithm and $V_1 \in U$ is a sign matrix.

First, based on the definition of $\mathcal{M}_\mathcal{R}$,

$$\mathcal{M}_\mathcal{R}(X_i) = (V_1 \cdot P_1)^{-1}(\mathcal{M}_\Phi(S(X_i))) = (V_1 \cdot P_1)^{-1}(\mathcal{M}_\Phi(V_1 P_1(X_i))). \tag{F.5}$$

Consider another rotation operation P_2 in the matrix form acting on X, using Lemma 3.1 we know that:

$$S(P_2(X_i)) = V_2(S(X_i)) = V_2V_1P_1(X_i) = (V_1P_1V_2I)(X_i) = (V_1P_1V_2P_2^{-1})(V_2P_2(X_i)), \tag{F.6}$$

where $V_2 \in U$ is an element of the sign flipping group. By Theorem 2 and realizable assumption, we further know

$$\mathcal{M}_\Phi(V_2(S(X_i))) = V_2S(y_i) = V_2(\mathcal{M}_\Phi(S(X_i))). \tag{F.7}$$

Then, consider $\mathcal{M}_\mathcal{R}(P_2(X))$ the process of RotEqNet:

$$\mathcal{M}_\mathcal{R}(P_2(X)) = ((V_1P_1)(V_2P_2)^{-1})^{-1}(\mathcal{M}_\Phi(S(P_2(X)))) = ((V_2P_2)(V_1P_1)^{-1})(\mathcal{M}_\Phi(S(P_2(X)))) \tag{F.8}$$

We know that $S(P_2(X)) = V_2S(X)$ from Eqn. (F.7). Therefore, we have $V_2\mathcal{M}_\mathcal{R}(S(X)) = \mathcal{M}_\mathcal{R}(V_2S(X)) = \mathcal{M}_\mathcal{R}(S(P_2(X)))$. Substitute $\mathcal{M}_\mathcal{R}(X)$ back into previous equation,

$$\mathcal{M}_\mathcal{R}(P_2(X)) = P_2(V_1P_1)^{-1}(\mathcal{M}_\Phi(S(X))) = P_2(\mathcal{M}_\mathcal{R}(X)) \tag{F.9}$$

We know if X_i has even-order ≥ 4, RotEqNet is rotation-equivariant. □

Appendix G. Training and prediction algorithm of RotEqNet

The training of RotEqNet would first normalize the entire dataset into the standard position, including the input tensor T and output tensor y. This would allow RotEqNet to obtain T_s and y_s. After this step, we would train a neural network using (T_s, y_s). This process is described as Algorithm 2.

Algorithm 2 RotEqNet training algorithm.

```
1: function TRAIN(T, y)
2:     R, T_s = GetRotInvariant(T)
3:     R, y_s = GetRotInvariant(y)
4:     Net = NeuralNetwork()
5:     Net.train(T_s, y_s) return Net
6: end function
```

After the training process, we would obtain a trained neural network focusing on standard position. For a new incoming tensor data with random rotation R, we could use the following process to predict its position.

Algorithm 3 RotEqNet prediction algorithm.

```
1: function PREDICT(T)
2:     R, T_s = GetRotInvariant(T)
3:     y_s = Net.predict(T_s)
4:     y = R^{-1} · y_s return y
5: end function
```

References

[1] Z. Huang, Y. Tian, C. Li, G. Lin, L. Wu, Y. Wang, H. Jiang, Data-driven automated discovery of variational laws hidden in physical systems, J. Mech. Phys. Solids (2020) 103871.

[2] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10561, 2017.

[3] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. 91 (2019) 045002.

[4] J.N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. 814 (2017) 1–4.

[5] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on dns data, Phys. Rev. Fluids 2 (2017) 034603.

[6] Y. Li, T. Zhang, S. Sun, X. Gao, Accelerating flash calculation through deep learning methods, J. Comput. Phys. 394 (2019) 153–165.

[7] P.A. Durbin, Some recent developments in turbulence closure modeling, Annu. Rev. Fluid Mech. 50 (2018) 77–103.

[8] J. Ling, J. Templeton, Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty, Phys. Fluids 27 (2015) 085103.

[9] M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. 182 (2002) 1–26.

[10] Z.J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2460.

[11] A. Beck, D. Flad, C.-D. Munz, Deep neural networks for data-driven les closure models, J. Comput. Phys. 398 (2019) 108910.

[12] T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, 2018, pp. 6571–6583.

[13] G.T. Mase, R.E. Smelser, G.E. Mase, Continuum Mechanics for Engineers, CRC Press, 2009.

[14] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.

[15] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, J. Comput. Phys. 318 (2016) 22–35.

[16] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.
[17] S. Pope, A more general effective-viscosity hypothesis, J. Fluid Mech. 72 (1975) 331–340.
[18] R.W. Johnson, Handbook of Fluid Dynamics, CRC Press, 2016.
[19] G. Smith, On isotropic integrity bases, Arch. Ration. Mech. Anal. 18 (1965) 282–292.
[20] T. Cohen, M. Welling, Group equivariant convolutional networks, in: International Conference on Machine Learning, 2016, pp. 2990–2999.
[21] C. Esteves, Theoretical aspects of group equivariant neural networks, arXiv preprint, arXiv:2004.05154, 2020.
[22] C. Esteves, Y. Xu, C. Allen-Blanchette, K. Daniilidis, Equivariant multi-view networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1568–1577.
[23] M. Weiler, F.A. Hamprecht, M. Storath, Learning steerable filters for rotation equivariant cnns, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 849–858.
[24] X. Cheng, Q. Qiu, R. Calderbank, G. Sapiro, Rotdcf: decomposition of convolutional filters for rotation-equivariant deep networks, arXiv preprint, arXiv:1805.06846, 2018.
[25] M. Finzi, S. Stanton, P. Izmailov, A.G. Wilson, Generalizing convolutional neural networks for equivariance to Lie groups on arbitrary continuous data, arXiv preprint, arXiv:2002.12880, 2020.
[26] L. Gao, H. Li, Z. Lu, G. Lin, Rotation-equivariant convolutional neural network ensembles in image processing, in: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, 2019, pp. 551–557.
[27] L. Gao, G. Lin, W. Zhu, Deformation robust roto-scale-translation equivariant cnns, arXiv preprint, arXiv:2111.10978, 2021.
[28] J.D. Foley, F.D. Van, A. Van Dam, S.K. Feiner, J.F. Hughes, J. Hughes, E. Angel, Computer Graphics: Principles and Practice, vol. 12110, Addison-Wesley Professional, 1996.
[29] Y. Zhou, C. Barnes, J. Lu, J. Yang, H. Li, On the continuity of rotation representations in neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5745–5753.
[30] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, J. ACM 36 (1989) 929–965.
[31] L.G. Valiant, A theory of the learnable, Commun. ACM 27 (1984) 1134–1142.
[32] C.C. Pinter, A Book of Abstract Algebra, Courier Corporation, 2010.
[33] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[35] C.K. Batchelor, G. Batchelor, An Introduction to Fluid Dynamics, Cambridge University Press, 2000.
[36] B. Kosović, Subgrid-scale modelling for the large-Eddy simulation of high-Reynolds-number boundary layers, J. Fluid Mech. 336 (1997) 151–182.
[37] H. Pitsch, Large-Eddy simulation of turbulent combustion, Annu. Rev. Fluid Mech. 38 (2006) 453–482.
[38] R. Matai, Les of Flow over Bumps and Machine Learning Augmented Turbulence Modeling, 2018.
[39] J.F. Nye, et al., Physical Properties of Crystals: Their Representation by Tensors and Matrices, Oxford University Press, 1985.
[40] K. Yang, X. Xu, B. Yang, B. Cook, H. Ramos, N.A. Krishnan, M.M. Smedskjaer, C. Hoover, M. Bauchy, Predicting the Young's modulus of silicate glasses using high-throughput molecular dynamics simulations and machine learning, Sci. Rep. 9 (2019) 1–11.
[41] Z. Liu, C. Wu, M. Koishi, A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials, Comput. Methods Appl. Mech. Eng. 345 (2019) 1138–1168.
[42] L. Walpole, Fourth-rank tensors of the thirty-two crystal classes: multiplication tables, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 391 (1984) 149–179.
[43] M.L. Curtis, Abstract Linear Algebra, Springer Science & Business Media, 2012.
[44] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[45] D. Tao, X. Li, W. Hu, S. Maybank, X. Wu, Supervised tensor learning, in: Fifth IEEE International Conference on Data Mining (ICDM'05), IEEE, 2005.
[46] D.F. Specht, et al., A general regression neural network, IEEE Trans. Neural Netw. 2 (1991) 568–576.
[47] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, R News 2 (2002) 18–22.
[48] S. Hanneke, The optimal sample complexity of pac learning, J. Mach. Learn. Res. 17 (2016) 1319–1333.