INVESTIGATING SUBEQUIVARIANT LEARNABLE MESH SIMULATORS

L. Fredenslund, P. Jensen

University of Copenhagen

P. Adema[*]

University of Amsterdam

ABSTRACT

Learning physical simulators has become a topic of some focus within deep learning, with many current approaches being based on Graph Neural Networks (GNNs). Two papers on this topic are Pfaff et al. [1], proposing MeshGraphNets (MGNs), models capable of accurately replicating mesh-based simulations, and Han et al. [2], proposing a learned rigid-body simulator augmented with the physical prior of rotational subequivariance. We propose two augmentations of MGN using the subequivariant transform from Han et al. [2]. Training and evaluating the modified models at a smaller scale shows that wrapping every non-linear element with a subequivariance transform can enable both accurate predictions and generalization under rotation while increasing data efficiency, at the cost of computational efficiency. Our results show a promising direction for future research on models of larger scales, serving as a compelling proof of concept for subequivariant mesh simulation using GNNs.

Index Terms— GNN, simulation, mesh, equivariance

1. INTRODUCTION

In this paper, we combine the works of Han et al. [2] and Pfaff et al. [1]. Both papers contribute to Graph Neural Network-based physics simulators. Pfaff et al. [1] introduce MeshGraphNets (MGNs) that model a 2D mesh space alongside the 3D world coordinates. This addition lets the network distinguish points close in both world-space and mesh-space from those only close in world-space, leading to a more realistic simulation. While MGNs are translation equivariant, they are not equivariant under rotation, which can hinder performance in systems that exhibit rotational symmetry (such as realistic physics). There are fully equivariant models such as GMN [3] and EGNN [4], but they are not suitable for problems where the symmetry is partially broken, such as with the introduction of a gravitational field. Han et al. [2] proposes a solution to this by introducing a subequivariant model which is rotationally equivariant around only the axis of symmetry.

We extend the model from Pfaff et al. [1] with a version of the subequivariance transform from Han et al. [2], and investigate the empirical performance of the extension under orthogonal transformations.

[*]Work done while at the University of Copenhagen

2. METHODS

2.1. MGN framework overview

MGN simulates a mesh at time t by encoding it into a graph $M^t = (V, E)$, where nodes V are connected by edges E. Each node $i \in V$ includes its current and previous world position, x_i^t and $x_i^{t-1} \in \mathbb{R}^3$, respectively, its current mesh position $u_i^t \in \mathbb{R}^2$ and a one-hot vector n_i, denoting node type.

2.1.1. Encoder

The original inputs are transformed into the following node and edge features: node features consist of the node type, $\boldsymbol{n}_i$, and the velocity vector, $\boldsymbol{q}_i^t = \frac{\boldsymbol{x}_i^t - \boldsymbol{x}_i^{t-1}}{dt}$. Edge features include the relative world and mesh coordinates, along with their Euclidean norms: $\boldsymbol{x}_{ij} = \boldsymbol{x}_i^t - \boldsymbol{x}_j^t$, $\boldsymbol{u}_{ij} = \boldsymbol{u}_i^t - \boldsymbol{u}_j^t$, $\|\boldsymbol{x}_{ij}\|_2$, and $\|\boldsymbol{u}_{ij}\|_2$, respectively. Edge features are made translation invariant by transforming the absolute positions into relative positions. This transformation allows the model's output to be translation equivariant, given that the absolute positions are kept track of separately.

In the encoder, two multilayer perceptrons (MLPs), σ^V for nodes and σ^E for edges, encode these node and edge features into separate higher-dimensional latent spaces.

2.1.2. Processor

The processor consists of K identical message passing blocks; each block first updates the edges and then uses the updated edges to update the nodes as follows:

$$\boldsymbol{e}'_{ij} \leftarrow f^E(\boldsymbol{e}_{ij}, \boldsymbol{v}_i, \boldsymbol{v}_j), \;\; \boldsymbol{v}'_i \leftarrow f^V(\boldsymbol{v}_i, \sum_{j \in \mathcal{N}(i)} \boldsymbol{e}'_{ij}), \qquad (1)$$

Where $\mathcal{N}(i)$ denotes the neighbourhood of node i, and f^E and f^N are MLPs with output dimension d_L.

2.1.3. Decoder

The decoder uses an MLP δ^V, to predict the change in acceleration, denoted $\boldsymbol{p}_i^t$, from the latent node features $\boldsymbol{v}'_i$. The world position of each node at $t+1$ is then updated using a second-order forward-Euler integrator with $dt = 1$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{p}_i^t + 2\boldsymbol{x}_i^t - \boldsymbol{x}_i^{t-1} = \boldsymbol{x}_i^t + \boldsymbol{q}_i^t + \boldsymbol{p}_i^t \qquad (2)$$

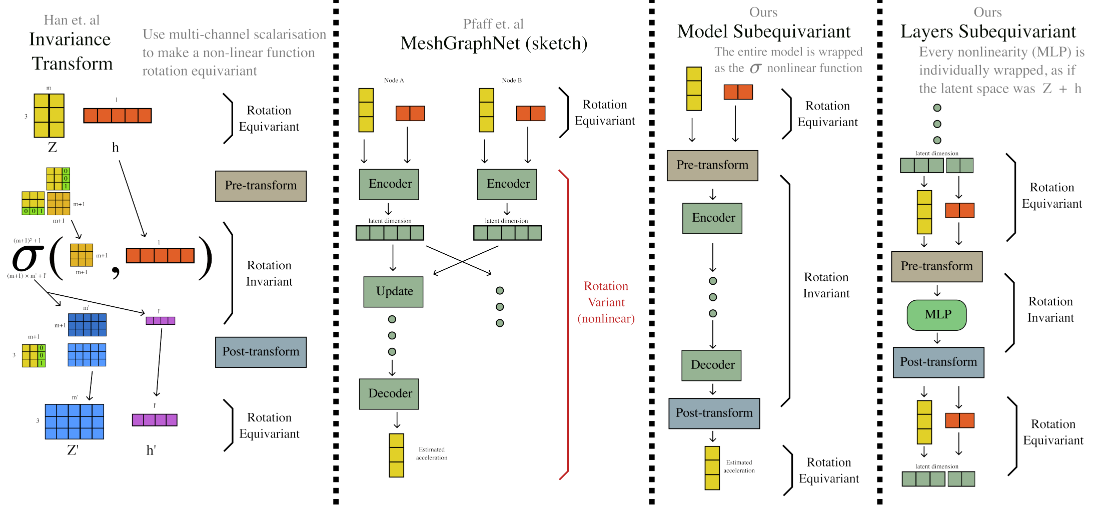

Fig. 1. Illustration of the methods mentioned in this paper: firstly, the invariance transform from Han et al. [2], secondly, a sketch of the MGN [1] architecture, thirdly, the full-model subequivariance augmentation from Sec. 2.3 and finally the layer-wise subequivariance transform from Sec. 2.4.

This update allows MGN to be translation equivariant, provided that the current and previous world positions are translated equally when calculating the current velocity.

2.2. Subequivariance

In this section, we define the notion of subequivariance, specifically $O(3)_{\vec{g}}$-equivariance, explain its relevance in physics simulation models, and show how it can be achieved in models like MGNs. Equivariance under some transformations is desirable to inductively exploit symmetries in the physical problem the model is learning. Equivariance encourages the model to learn the dynamics not covered by the symmetries, improving generalization and increasing performance. Most problems are not fully $E(3)$-equivariant, but a large group of dynamics problems are equivariant to rotation around the axis of an external force. For an external force $\vec{\boldsymbol{g}}$, we refer to such symmetry as $O(3)_{\vec{\boldsymbol{g}}}$-equivariance, or simply subequivariance.

Formally, $E(3)$ refers to the group of all rotation/reflection matrices and translation vectors in 3D space, defined as

$$E(3) = \{(\boldsymbol{R}, \boldsymbol{t}) \mid \boldsymbol{R} \in O(3),\ \boldsymbol{t} \in \mathbb{R}^3\} \tag{3}$$

with $\boldsymbol{t}$ being translation vectors and $\boldsymbol{R}$ being rotation/reflection matrices. $O(3)$ represents the orthogonal group in 3 dimensions, defined by

$$O(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} | \boldsymbol{R}^T \boldsymbol{R} = \boldsymbol{I}\}. \tag{4}$$

Using the definition presented in Han et al. [2], a function $f : \mathbb{R}^{3\times m} \times \mathbb{R}^n \to \mathbb{R}^3 \times m'$ is $E(3)$-equivariant, if for any transformation $g \in E(3)$, we have that $f(g \cdot \vec{\boldsymbol{Z}}, \boldsymbol{h}) = g \cdot f(\vec{\boldsymbol{Z}}, \boldsymbol{h})$, and f is invariant, if $f(g \cdot \vec{\boldsymbol{Z}}, \boldsymbol{h}) = f(\vec{\boldsymbol{Z}}, \boldsymbol{h}), \forall g \in E(3)$. Here, $\vec{\boldsymbol{Z}} \in \mathbb{R}^{3\times m}$ is a matrix of m column-stacked geometric vectors in 3D space. The $\to$ denotes geometric data. $\boldsymbol{h} \in \mathbb{R}^n$ is a vector of non-geometric features. Han et al. [2] defines $O(3)_{\vec{\boldsymbol{g}}}$-equivariance by restricting the group of orthogonal transformations to:

$$O_{\vec{\boldsymbol{g}}}(3) := \{\boldsymbol{R} \in O(3) \mid \boldsymbol{R}\vec{\boldsymbol{g}} = \vec{\boldsymbol{g}}\} \tag{5}$$

Where transformations preserve the fixed direction of $\vec{\boldsymbol{g}}$ in their implementation by appending, Han et al. [2] achieves this by augmenting $\vec{\boldsymbol{Z}}$ with $\vec{\boldsymbol{g}}$ and performing a multichannel scalarization by taking the inner product $\left[\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}\right]^\top \left[\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}\right] \in \mathbb{R}^{m+1\times m+1}$. This inner product is $O(3)_{\vec{\boldsymbol{g}}}$-*invariant*, since the orientation of $\vec{\boldsymbol{Z}}$ relative to $\vec{\boldsymbol{g}}$ is only preserved under $O(3)_{\vec{\boldsymbol{g}}}$ transformations on $\vec{\boldsymbol{Z}}$. Han et al. [2] supplies the scalarized geometric input to an MLP, making it $O_{\vec{\boldsymbol{g}}}(3)$-invariant. The MLP's output is then projected back onto $\left[\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}\right]$, yielding a scalarized form that is $O_{\vec{\boldsymbol{g}}}(3)$-equivariant:

$$f_{\vec{\boldsymbol{g}}}(\vec{\boldsymbol{Z}}, h) = [\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}]V_{\vec{\boldsymbol{g}}}, \quad \text{s.t. } V_{\vec{\boldsymbol{g}}} = \sigma\left([\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}]^\top [\vec{\boldsymbol{Z}}, \vec{\boldsymbol{g}}], \boldsymbol{h}\right), \tag{6}$$

where $V_{\vec{\boldsymbol{g}}}$ is an MLP with output dimension $\mathbb{R}^{m+1\times m'}$. Finally, to make MGNs subequivariant, the function $f_{\vec{\boldsymbol{g}}}$ should be used in place of any MLP that receives input features sensitive to $E(3)$ operations, such as the initial encoders mentioned in Sec. 2.1.1.

2.3. Full-model subequivariance

Unlike the model proposed by Han et al. [2], MGN has a homogeneous latent space for message passing, with no explicit $\vec{\boldsymbol{Z}}, \boldsymbol{h}$ to transform. Despite this, a simple method for making MGN rotation subequivariant using the transformation described in Sec. 2.2 would be to treat the entire model as a single non-linear function and wrap it with the transform, as illustrated in Fig. 1. However, performing the transformation while treating the node-axis as m would result in a model input of variable size per node: the scalarization would output an $(m+1) \times (m+1)$ matrix, meaning that models trained this way would be unable to generalize to different node counts. As such, the only option is to treat the singular set of relative coordinates of the current node or edge as the m-dimension, setting $m = 1$. This has the side-effect of all relative coordinates being scalarized independently, meaning the model cannot determine the relative angles between nodes during message-passing, severely hindering its effectiveness.

2.4. Layer-wise subequivariance

Alternatively, the MGN model can be modified to split its latent space into an explicit $\vec{\boldsymbol{Z}}, \boldsymbol{h}$, allowing every non-linear layer (the MLPs in the encoder, processor and decoder) to be wrapped individually (see Fig. 1). This allows the message-passing layers to have scalarization transforms spanning multiple nodes or edges: an edge update step scalarizes its two neighbour nodes together, informing it of the relative angles between it and its neighbour nodes. In practice, with a latent dimension of 64, we split the latent with $m = 16$, meaning 48 of the 64 latent channels were considered positional (and were transformed), while the remaining 16 were considered non-positional.

3. EXPERIMENTAL SETUP

We train 3 model variants: a baseline model similar to Pfaff et al. [1], a full-model subequivariant extension, and a layer-wise subequivariant extension to the baseline model. In each variant, we reduce the latent size from 128 to 64 and the number of message passing blocks from 15 to 10 due to computational budget limitations. Pfaff et al. [1] found that additional message passing blocks improves performance at the cost of computational budget.

To further limit the computational cost, we train the base and the full-model variation for $5 \cdot 10^5$ steps and the layer-wise subequivariant variation for $2 \cdot 10^5$ steps as opposed to 10^7 steps in Pfaff et al. [1].

We use the *sphere_simple* dataset which does not use dynamic remeshing to further reduce the computational cost. The *sphere_* datasets were the only datasets from Pfaff et al. [1] which feature a limited rotational symmetry making our extensions suitable, but Pfaff et al. [1] did not publish the code for generating world edges for the *sphere_* datasets. Due to time constraints, we were unable to reimplement those. We therefore treat the *sphere_simple* dataset as the *flag_simple* dataset in which there is no world contact between objects.

These limitations have consequences for the absolute results. The training data encodes interactions between the sphere and the flag, but the models get no information about how the interactions should occur. Moreover, we are using smaller models and training them for significantly fewer steps. We will therefore principally look at the relative change due to equivariant rotations around the gravitational axis.

4. RESULTS

We define the equivariant MSE for some MODEL as

$$\mathrm{MSE}_{eqv} = \frac{1}{N}\sum_{i}^{N} \|\mathrm{MODEL}(\vec{x}^{t}) - \mathbf{R}^{-1}\mathrm{MODEL}(\mathbf{R}\vec{x}^{t})\|_2^2, \tag{7}$$

where $\mathbf{R} \in \mathbb{R}^{3\times 3}$ is some rotation matrix around the gravity axis (in our case $45\,\mathrm{deg}$ clockwise), and $\mathbf{R}^{-1}$ is the inverse of that rotation. $\vec{x}_i \in \mathbb{R}^{3\times\mathcal{N}}$ is the world space coordinates for $\mathcal{N}$ nodes at time t. It measures how inconsistent the model is in its predictions when the coordinates are rotated around an equivariant axis. In Table 1, we see that the base model has a relatively high MSE_{eqv}, while the full-model subequivariance extension improves the equivariance considerably and is close to the numerical precision in the early steps indicated by Ground Truth. The layer-wise subequivariance extension achieves a smaller but still significant improvement in MSE_{eqv}.

As the simulation steps are rolled out, the error accumulates faster than the lower bound indicated by Ground Truth, but the subequivariant extensions retain a better MSE_{eqv} compared to the baseline. In Table 2, we report the MSE between the predicted and ground truth positions. In all cases, the error accumulates with time, but there is little difference in the MSE_{abs} between the rotated and non-rotated coordinates. This lack of difference indicates that the error is dominated by the models' ability to learn the dynamics, not rotational equivariance. The only exception is the baseline, which has no mitigation and a higher MSE_{abs} when rotated during the initial steps, indicating that subequivariance can benefit sufficiently capable models. The baseline and layer-wise subequivariance extension generally have lower MSE_{abs} compared to the full-model subequivariance, indicating a tradeoff between MSE_{abs} and MSE_{eqv}. The layer-wise subequivariance is also over 6 times slower than the baseline and full-model subequivariance, running at 5.5 training iterations/s on a Nvidia T4 compared to 35 it/s for the other two variants.

In Fig 2, we see predictions for the different model variants at step 50 and the ground truth for both non-rotated inputs and inputs rotated by $45\,\mathrm{deg}$ around the gravity axis. The baseline model significantly alters its prediction when the inputs are rotated, whereas both subequivariance extensions remain consistent with their predictions under rotation. Furthermore, it should be noted that full-model subequivariance pre-

MSE_{eqv}	Ground Truth	Base	Model Subeqv.	Layer Subeqv.
Step 5	$(2.6 \pm 0.7) \cdot 10^{-16}$	$(3 \pm 1) \cdot 10^{-5}$	$(8 \pm 2) \cdot 10^{-15}$	$(2.3 \pm 0.9) \cdot 10^{-6}$
Step 500	$(2 \pm 1) \cdot 10^{-16}$	$(3 \pm 2) \cdot 10^{-2}$	$(3 \pm 6) \cdot 10^{-5}$	$(10 \pm 3) \cdot 10^{-4}$

Table 1. The subequivariant MSE between predicted positions and predicted positions in rotated coordinate system rotated back. The Ground Truth represents the numerical precision of rotating and rotating back, and works as a lower bound for MSE. We use a rotation of $45 \deg$ clockwise. $\pm$ indicates 1 standard deviation.

MSE_{abs}	Base	Model Subeqv.	Layer Subeqv.
Step 5	$(0.4 \pm 0.2) \cdot 10^{-5}$	$(1 \pm 1) \cdot 10^{-5}$	$(0.2 \pm 0.3) \cdot 10^{-5}$
Step 5 Rot.	$(2.2 \pm 0.5) \cdot 10^{-5}$	$(1 \pm 1) \cdot 10^{-5}$	$(0.4 \pm 0.6) \cdot 10^{-5}$
Step 500	$(6 \pm 4) \cdot 10^{-2}$	$(10 \pm 4) \cdot 10^{-2}$	$(4 \pm 4) \cdot 10^{-2}$
Step 500 Rot.	$(6 \pm 4) \cdot 10^{-2}$	$(10 \pm 4) \cdot 10^{-2}$	$(4 \pm 4) \cdot 10^{-2}$

Table 2. The MSE between predicted positions and ground truth positions. Rot. indicates the inputs have been rotated by $45 \deg$ clockwise. $\pm$ indicates 1 standard deviation.

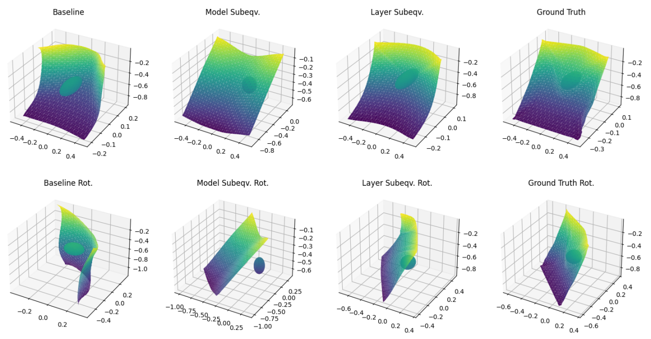

Fig. 2. Comparison of simulation at step 50 for different model variants. Rot. indicates that the input has been rotated by $45 \deg$. The spheres are incorrectly handled in the models due to missing world edges.

diction is generally static: being unable to observe relative angles, it always predicts a velocity around zero. Qualitatively, the layer-wise extension is more consistent with the ground truth after rotation, whereas the full-model extension is more self-consistent. Finally, the baseline is neither self-consistent nor consistent with the ground truth after rotation.

5. DISCUSSION

We show that applying the subequivariant transform from SOMP [2] to the GraphMeshNet [1] model can significantly increase the model's capability to generalize across rotations around the equivariant axis. Table 1 and Fig. 2 show that the base model overfits on the angles present in the training data, while the subequivariant modifications remain generally consistent with their predictions after rotation. This effect is most visible in the early steps of the simulation rollout, where even the full-model equivariance outperforms the base model in MSE_{abs}. Furthermore, although full-model subequivariance decreases prediction accuracy for non-rotated scenes, Table 2 shows that applying the subequivariant transform to every individual layer results in a model with equal, if not better, predictive performance than the base model, even for non-rotated scenes. The ability of the layer-wise subequivariant model to observe only relative angles allows for better accuracy, generalization and data efficiency (using only 40% of the training epochs as the other models) at the price of being six times slower.

Future research directions might include re-implementing more of the MGN model with the subequivariant transforms (as features such as world edges were not included in the code sample from the paper), training full-scale models to investigate whether layer-wise subequivariance also improves non-rotated performance for larger models, and attempting to find a more efficient method for attaining layer-wise subequivariance.

For replicability, all of our training and visualization code is available in a GitHub repository. During this research, we used approximately 110 hours on an NVIDIA T4 GPU, where the final set of models took 18 hours to train on the *sphere_simple* dataset.

6. CONCLUSION

Generalization is necessary for scalable physical simulation, and adding rotation subequivariance to the mesh simulator from MGN is a step towards a more practical GNN-based simulator. At the smaller scale this research has operated at, this addition increases performance under rotations and data efficiency at the price of computation efficiency. Examining these tradeoffs more closely or at a larger scale would be a logical next step for future research.

7. COMPLIANCE WITH ETHICAL STANDARDS

This is a theoretical machine learning study for which no ethical approval was required. We do not expect there to be a possible harmful dual use case for the models trained or described.

8. ACKNOWLEDGMENTS

We would like to thank our Teaching Assistant Kiril Klein for their help in understanding the research assignment, as well as our lecturers for their clear explanations of much of the theory behind these models.

For conducting this study, 50$ of Google Cloud Credits were provided as an educational grant, and were used to run the experiments. The authors have no relevant financial or non-financial interests to disclose.

9. REFERENCES

[1] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia, "Learning mesh-based simulation with graph networks," 2021.

[2] Jiaqi Han, Wenbing Huang, Hengbo Ma, Jiachen Li, Joshua B. Tenenbaum, and Chuang Gan, "Learning physical dynamics with subequivariant graph neural networks," 2022.

[3] Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang, "Equivariant graph mechanics networks with constraints," 2022.

[4] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling, "E(n) equivariant graph neural networks," 2022.