Graph Neural Networks for Geometry Design in Plastic Regime

Mikel M. Iparraguirre, David González, Icíar Alfaro, Elías Cueto

Applied Mechanics and Bioengineering

Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Tel. +34-976762707, e-mail: *mikel.martinez@unizar.es*

Abstract

In this work, we extend MeshGraphNets to simulate collisions involving plastic material, previously limited to hyperelastic, addressing higher non-linearities and energy dissipation. While previous work estimated only the displacement field, our approach also resolves the complete tensor stress field, under different boundary conditions and geometries.

Introduction

Plasticity governs many phenomena surrounding us [2], and its comprehension is crucial for numerous industrial processes, such as car design and production. Current state-of-the-art approaches for modeling realistic collisions are based on the Finite Element Method (FEM), an iterative method that provides high-fidelity solutions by rigorously satisfying the laws of physics. However, these methods are still slow at solving such simulations. The reason for this high computing time is the non-linearities arising from large deformations and material behavior, which require the solver to subdivide the problem into multiple iterative steps to guarantee convergence.

On the other hand, data-driven approaches incur an overhead time cost during training but are efficient during inference, making them attractive to computational mechanics. Rather than guaranteeing physics by design, these models regress the physics from the data. The work of [1] introduced MeshGraphNets, presenting the first model based on Graph Neural Networks (GNNs) capable of solving the master-slave collision problem for hyperelastic materials, reporting only results on positions.

Our contributions are: 1) Modeling collisions in the plastic regime for both the position and the stress fields using MeshGraphNets.

2) Employ transfer learning to speed-up training and enhance solution quality.

3) Conduct an analysis of the generalization properties and robustness to unseen geometries.

Modeling Plasticity

The approach followed uses a temporal integrator framework. The model predicts the displacement and stress field of each node for the next time step $t+1$ based on positional information at time t. One key aspect is that the model is trained on one-step predictions only but rollouts are made over long trajectories (>400 steps) at inference.

Our master-slave collision setup involves a rigid cylinder actuator as the master, imposing displacements, and an irregular hexahedral deformable plate as the slave (Fig.1). The master-slave meshes are treated as a multi graph $G(v, \varepsilon_{mesh})$, where the vertices are the nodes of the meshes and the edges are the distances between connected nodes at initial time t_0 (undeformed) and at time t (deformed). For the collision an additional edge set is created to connect both graphs $G(v, \varepsilon_{mesh}, \varepsilon_{contact})$ based on a radius distance R_w. This second set of edges facilitates the communication of imposed displacements from the actuator to the plate.

The model follows the Encoder-Processor-Decoder architecture proposed in the MeshGraphNets work (Fig.1). The **Encoder** block comprises three different encoders, one for each set of the multigraph, each projecting the set graph into the latent space. The **Processor** block models the collision between the actuator and plate and the plate's behavior under these conditions, updating the graph via message passing blocks. The **Decoder** outputs the nodal displacements and corresponding stress field $t+1$. Each block consists of MLPs with ReLU activation layers.

Experiments and Results

Datasets: The dataset consists of 100 high-fidelity 3D-trajectories, each containing 435 steps, solved with Abaqus in quasi-static conditions. Variability in

the dataset arises from changes in boundary conditions, including fixed faces of the plate, actuator location and its imposed displacement, and variations in plate geometry such as thickness, hole radius [0.05-0.1cm] and location. All trajectories are conducted in two steps: loading and unloading, to highlight the plasticity phenomena. The meshes are tetrahedral elements, with approximately 600 to 800 each. Due to quasi-static, time is a pseudo-time variable representing incremental steps, not physical time.

The dataset is split into 80% train, 10% validation, and 10% test, each with varying dataset features. Additionally, we generate the test extra set, consisting of 20 trajectories with greater hole radius [0.1-0.15cm] to analyze the generalizability of our model to out-of-distribution geometries (extrapolation)

Experiments: The first experiment models the displacement and the single von Mises stress fields. Achieving less than 1% relative errors in positions (q) and 10% in von Mises stress ($\sigma_{v.mises}$) for the whole rollout. The model shows robustness to extrapolated geometries as shown in Fig. 2.a.

The second experiment models the displacement and the complete 6-tensor stress fields. Achieving less than 1% relative errors in positions (q) and 10% for the whole 6-tensor stress field ($\sigma_{complete}$). Also presenting robustness to extrapolated geometries as shown in Fig. 2.b.

In the first experiment the model was trained from scratch, reaching convergence after around 10 M training steps. In contrast, in the second experiment we used the pre-trained encoder-processor from the previous model and added a new decoder to output the displacements and completed the 6-tensor stress field. This allowed us to reduce down to 1M the required steps to reach convergence.

The optimal architecture after hyperparameter optimization was a 3-layer MLP of 128 neurons for the encoders, processor, and decoder, with a processor of 15 independent message passing blocks. The experiments were carried out on a single GPU GTX 4090 24GB. Code and dataset are available under request.

Conclusions

The experiments prove that collisions between rigid-plastic bodies can be effectively modeled using MeshGraphNets as a data-driven approach. This architecture leverages the mesh-based topology of our Finite Element Method (FEM) simulations. Despite operating on a nodal architecture through message passing, we can generate solutions for the entire plate at a global scale. The approach overcomes the challenges of modeling plasticity, where the relationship between strain and stress is inherently a local phenomenon; however, the entire deformation field must be solved globally, considering factors such as plastifying partial regions, boundary conditions, and the actuator's displacements.

REFERENCIAS

[1]. PFAFF T, et al. *Learning mesh-based simulation with Graph Networks.* arXiv:2010.03409. International Conference on Learning Representations(ICLR), 2021.

[2]. HAN, W., & REDDY, B. D. (2012). *Plasticity: mathematical theory and numerical analysis* (Vol. 9). Springer Science & Business Media.

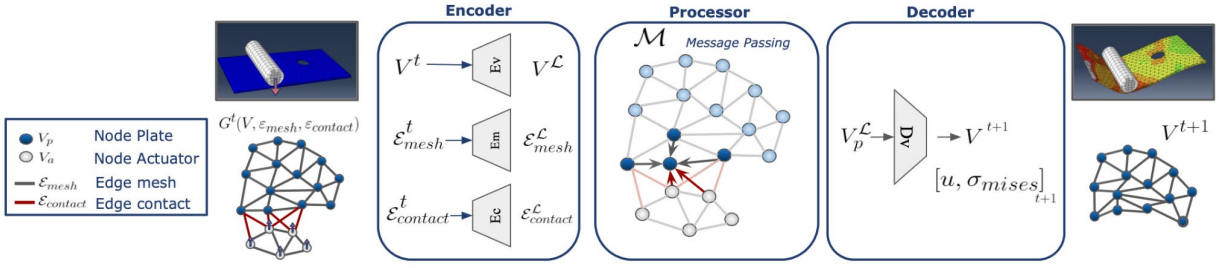
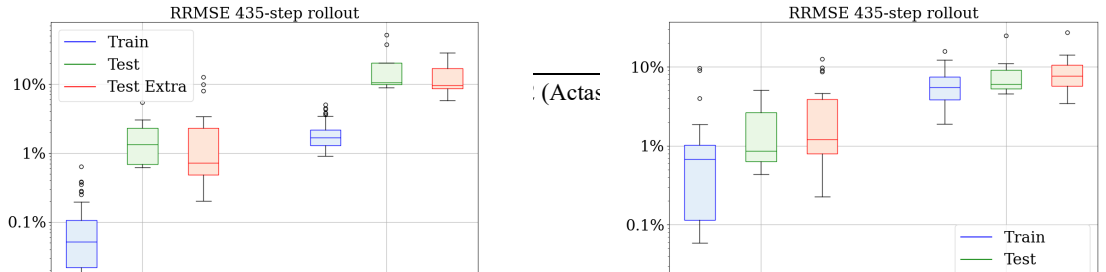


Fig 1. MeshGraphNet Encoder-Processor-Decoder architecture to model collision. **Left**: multi-graph at time t with nodes, edges mesh and edges contact. **Center**: MeshGraphNet: Encoder-Processor-Decoder. **Right**: output at $t+1$ of plate nodes, displacements and stress.



a) b)

Fig 2. Boxplot Relative Root Mean Squared Error (RRMSE) for the train, test (interpolation), and test extra (extrapolation) data splits. The mean is computed across all nodes and steps in the rollout (with 435 steps) for positional (q) and stress (σ) variables. The infinite norm is computed on each step and used as a normalizer for the squared error per variable. **a)** Results of the first experiment, which models displacements and a single von Mises stress field. **b)** Results of the second experiment, which models displacements and a 6-tensor stress field.