MaNGO – Adaptable Graph Network Simulators via Meta-Learning

Philipp Dahlinger* **Tai Hoang** **Denis Blessing** **Niklas Freymuth** **Gerhard Neumann**
Autonomous Learning Robots
Karlsruhe Institute of Technology
Karlsruhe

Abstract

Accurately simulating physics is crucial across scientific domains, with applications spanning from robotics to materials science. While traditional mesh-based simulations are precise, they are often computationally expensive and require knowledge of physical parameters, such as material properties. In contrast, data-driven approaches like Graph Network Simulators (GNSs) offer faster inference but suffer from two key limitations: Firstly, they must be retrained from scratch for even minor variations in physical parameters, and secondly they require labor-intensive data collection for each new parameter setting. This is inefficient, as simulations with varying parameters often share a common underlying latent structure. In this work, we address these challenges by learning this shared structure through meta-learning, enabling fast adaptation to new physical parameters without retraining. To this end, we propose a novel architecture that generates a latent representation by encoding graph trajectories using conditional neural processes (CNPs). To mitigate error accumulation over time, we combine CNPs with a novel neural operator architecture. We validate our approach, Meta Neural Graph Operator (MaNGO), on several dynamics prediction tasks with varying material properties, demonstrating superior performance over existing GNS methods. Notably, MaNGO achieves accuracy on unseen material properties close to that of an oracle model.

1 Introduction

The simulation of complex physical systems is of paramount importance in a wide variety of engineering disciplines, including structural mechanics [1–3], fluid dynamics [4–6], and electromagnetism [7–9]. In particular, simulating object deformations under external forces is essential for applications such as robotics [10–12]. Traditional mesh-based simulations are appealing for such problems due to the accuracy of the underlying finite element method [13, 14]. However, these methods are typically slow and require precise knowledge of the simulation parameters, including material properties of objects.

In contrast, data-driven approaches for simulating complex systems have emerged as a promising alternative to traditional mesh-based simulators [15–17]. Among them, Graph Network Simulators (GNSs) have recently become increasingly popular [18–21, 12, 22]. GNSs encode the simulated system as a graph of interacting entities whose dynamics are predicted using Graph Neural Networks (GNNs) [23]. These models are often orders of magnitude faster than classical simulators [19] while being fully differentiable, making them highly effective for downstream tasks such as inverse design problems [20, 24]. Moreover, these models do not require knowledge of simulation parameters as they directly learn from the training data. However, they must be retrained from scratch for even

*correspondence to philipp.dahlinger@kit.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

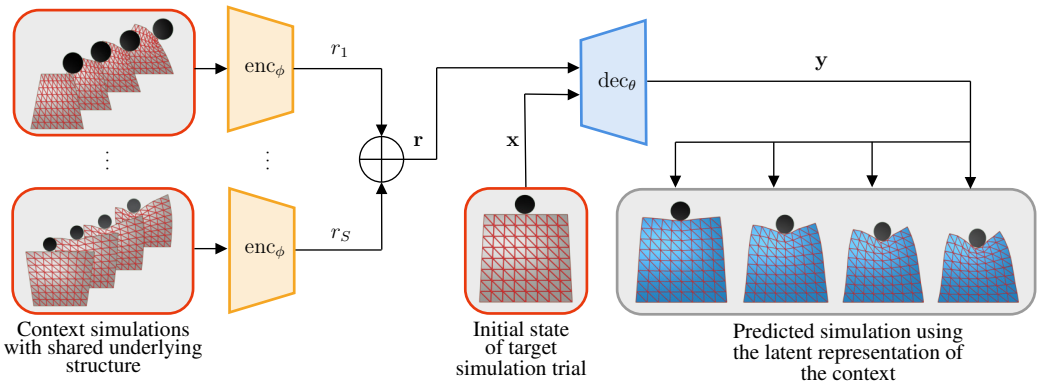

Figure 1: Our proposed Meta Neural Graph Operator (MaNGO) approach: The context set is aggregated to form a latent representation of material properties. Given an unseen initial state, the Graph Network Simulator (GNS) uses this latent representation to generate trials that follow the material laws of the context set, enabling accurate predictions for new conditions. The example prediction aligns perfectly with the ground truth data.

minor variations in physical parameters, and require data collection for each new parameter setting that is often either costly and time-consuming or even impossible.

To overcome this, Sanchez-Gonzalez et al. [25] proposed to use the simulation parameter as conditional information when training the GNS, allowing for generalization to unseen simulation parameters. However, such an approach only works under two assumptions. Firstly, training data must be labeled with the corresponding simulation parameter, and secondly, the simulation parameter must be available at test time for the desired simulation. While the first assumption is mild and often satisfied in simulation settings, the second assumption requires solving the inverse problem of inferring the underlying physical parameters from observed system behavior [26]. Material estimation or system identification can be viewed as a specific instance of this problem [27–32]. However, solving such inverse problems is challenging, as they are typically ill-posed and require explicit knowledge of the governing partial differential equation (PDE) [30, 27, 28].

To overcome this challenge, we investigate data-driven adaptation of GNS – enabling fast and accurate simulations for unknown parameters using only a few simulation trials. Our work builds on the premise that training data from different simulations shares a common 'latent' underlying structure. We aim to learn this structure via meta-learning using Conditional Neural Processes (CNPs). To that end, we propose a novel framework called Meta Neural Graph Operator (MaNGO) that builds on Message Passing Networks (MPNs) and neural operator methods to ensure efficient processing of spatiotemporal data. We validate our approach on several dynamics prediction tasks with varying material properties, demonstrating superior performance over existing GNS methods. Notably, our method achieves accuracy on unseen material properties close to that of an oracle model which has access to the simulation parameters at test time. To summarize, we identify our contribution as follows: (i) we successfully use meta-learning with Conditional Neural Processes (CNPs) for graph network simulators allowing for fast and accurate adaptation to unseen physical parameters. (ii) we identify shortcomings of existing architectures for handling spatiotemporal data and propose a novel GNS architecture. (iii) we provide a set of new benchmark tasks suited for testing the adaptation capability of GNS[2].

2 Preliminaries

Graph and Message Passing Neural Networks. Graph Neural Networks (GNNs) are a class of neural networks designed to process graph-structured data by iteratively updating node representations through localized message passing. Here, a graph is a defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{\mathbf{m}_v^0\}_{v\in\mathcal{V}}, \{\mathbf{m}_e^0\}_{e\in\mathcal{E}})$ with nodes $\mathcal{V}$, edges $\mathcal{E}$, and associated vector-valued node and edge features $\mathbf{m}_v^0$ and $\mathbf{m}_e^0$. A Message Passing Network (MPN) [25, 19], a GNN architecture well-suited for graph-based simulations, consists of K message passing steps, which iteratively update the node and edge features based on

[2]Code: `https://github.com/ALRhub/mango` Dataset: `https://zenodo.org/records/17287535`

the graph topology. Each such step is given as

$$\mathbf{m}_e^{k+1} = f_{\mathcal{E}}^k(\mathbf{m}_e^k, \mathbf{m}_v^k, \mathbf{m}_w^k), \text{ with } e = (v, w), \qquad \mathbf{m}_v^{k+1} = f_{\mathcal{V}}^k(\mathbf{m}_v^k, \bigoplus_{e \in \mathcal{E}_v} \mathbf{m}_e^{k+1}), \tag{1}$$

where $\mathcal{E}_v \subset \mathcal{E}$ are the edges connected to v. Further, $\bigoplus$ denotes a permutation-invariant aggregation operation such as the sum, the max, or the mean. The functions $f_{\mathcal{V}}^m$ and $f_{\mathcal{E}}^m$ are learned Multilayer Perceptrons (MLPs), usually with a residual connection. The network's final output are the node-wise learned representations $\mathbf{m}_v^K$ that encode local information of the initial node and edge features.

Neural Dynamics Prediction. Our goal is to learn the dynamics of a multibody system, i.e., a trajectory of graphs $(\mathcal{G}^{(t)})_{t\in[0,T]}$ from the initial condition $\mathcal{G}^{(0)}$. We follow Brandstetter et al. [33] and group existing approaches into two categories, *neural operators* and *autoregressive methods*. Neural operator methods treat the mapping from initial conditions to solutions at time t as an input–output mapping learnable via supervised learning. Formally, a neural operator F_{NO} predicts the graph at any $t \in [0, T]$ from the initial condition, that is, $\mathcal{G}^{(t)} = F_{\text{NO}}\left(t, \mathcal{G}^{(0)}\right)$. In contrast, autoregressive methods, learn to incrementally update the graph starting from $\mathcal{G}^{(0)}$:

$$\mathcal{G}^{(t+\Delta t)} = F_{\text{AR}}\left(\Delta t, \mathcal{G}^{(t)}\right).$$

Here, F_{AR} is the temporal update and $\Delta t \in \mathbb{R}_{>0}$.

Meta-Learning and Conditional Neural Processes. To formalize the meta-learning problem using conditional neural processes (CNPs) [34], we consider a meta-dataset $\mathcal{D} = \mathcal{D}_{1:L}$ consisting of L, typically small, task datasets $\mathcal{D}_l = \{\mathbf{x}_i^l, \mathbf{y}_i^l\}_{i=1}^{S_l}$ of size S_l. Each task consists of inputs $\mathbf{x}_i^l \in \mathbb{R}^{d_x}$ and corresponding evaluations $\mathbf{y}_i^l \in \mathbb{R}^{d_y}$ of unknown functions f_l, that is, $\mathbf{y}_i^l = f_l(\mathbf{x}_i^l) + \epsilon_i$, where ϵ_s denotes (possibly heteroskedastic) noise. Meta-learning hinges on the idea that tasks share statistical structure, allowing for fast adaptation to a target function f_* based on a small target dataset $\mathcal{D}_* = \{\mathbf{x}_i^*, \mathbf{y}_i^*\}_{i=1}^{S_*}$ of size S_*. To leverage this shared statistical structure, Conditional Neural Processes (CNPs) use the meta-dataset $\mathcal{D}$ to learn how to generate a latent representation $\mathbf{r} \in \mathbb{R}^{d_r}$ from a given set of $(\mathbf{x}, \mathbf{y})$-pairs. At test time, this latent representation is generated from the target dataset $\mathcal{D}_*$ enabling generalization to unlabeled inputs $\mathbf{x}_*$ from the target task without requiring weight adaptation. Formally, to generate the latent representation $\mathbf{r}$ for a set $\mathcal{S} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{S}$ with arbitrary size S, CNPs use a parameterized encoder with a permutation-invariant aggregation method,

$$\mathbf{r} = \bigoplus_{i \in \{1,\dots,S\}} r_i \quad \text{with} \quad r_i = \text{enc}_\phi(\mathbf{x}_i, \mathbf{y}_i), \tag{2}$$

with parameters ϕ and permutation-invariant aggregation method $\bigoplus$. The latent representation $\mathbf{r}$ is then used to predict the mean and variance of a Gaussian distribution over $\mathbf{y}$, given a new input $\mathbf{x}$,

$$p_\theta(\mathbf{y}|\mathbf{x}, \mathcal{S}) = \mathcal{N}\left(\mathbf{y}|\, \text{dec}_\theta^\mu(\mathbf{x}, \mathbf{r}),\ \text{dec}_\theta^\Sigma(\mathbf{x}, \mathbf{r})\right), \tag{3}$$

using a parameterized decoder with parameters θ. For training, the task datasets are further split into context sets $\mathcal{D}_l^c \subseteq \mathcal{D}_l$ to train the CNP on different dataset sizes S_l^c which are used to minimize the negative task log-likelihood

$$\mathcal{L}_l(\phi, \theta) = -\mathbb{E}_{\mathcal{D}_l^c \subseteq \mathcal{D}_l}\left[\sum_{i=1}^{S_l} \log p_\theta(\mathbf{y}_i^l|\mathbf{x}_i^l, \mathcal{D}_l^c)\right] \tag{4}$$

which implicitly depends on ϕ through the encoding of $\mathcal{D}_l^c$. Here, the expectation indicates randomly sampled subsets. Thus, meta-learning aims to maximize the overall task likelihood while conditioning the model on smaller subsets of the data. The complete loss is obtained as $\mathcal{L}(\phi, \theta) = \sum_l \mathcal{L}_l(\phi, \theta)$ and is optimized end-to-end using stochastic gradients with respect to ϕ and θ.

3 Adaptable Graph Network Simulators via Meta-Learning

Having established the foundations, we now explain how meta-learning with Conditional Neural Processes (CNPs) can be used to make graph network simulators adaptable to novel unseen physical parameters. To that end, we start by formalizing our setup and introducing the meta-dataset in

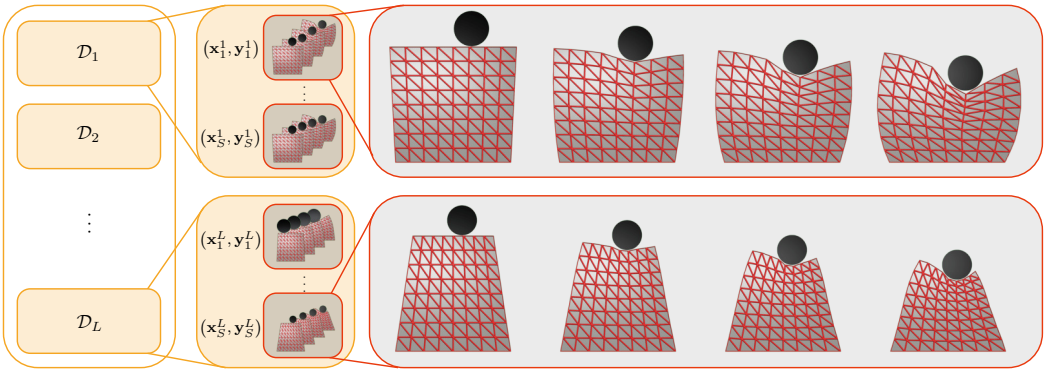

Figure 2: Meta-dataset representation: This figure illustrates the structure of the meta-dataset, consisting of multiple task datasets. Each dataset includes simulations with shared material properties but varying starting conditions. Two example simulations, though starting similarly, produce vastly different results due to their distinct material properties.

Section 3.1. Next, we propose an extension to meta-learning that improves training with additional information that is not available at test-time in Section 3.2. Lastly, we introduce a CNP architecture that is tailored to our setup. Specifically, we propose an encoder that generates a latent representation from spatiotemporal data in Section 3.3. Furthermore, we introduce a novel decoder that addresses shortcomings of existing methods as outlined in Section 3.4.

3.1 Problem Formalization and Setup

The aim is to learn a graph network simulator (GNS) that predicts a sequence of graphs $\mathcal{G}_{1:T} := \{\mathcal{G}_t\}_{t=1}^T$ describing, e.g., mesh deformation over time without knowing some simulation or physical parameters, which we refer to as $\rho \in \mathbb{R}^{d_\rho}$. For example, ρ could be stiffness or compression properties of a deformable object in the simulation. To that end, we use meta-learning to extract latent representations from similar data to generalize to unseen physical parameters. As outlined in Section 2, in order to perform meta-learning we require a meta dataset $\mathcal{D}$ containing L task datasets $\mathcal{D}_l \in \mathcal{D}$ that in turn consist of multiple input-output pairs $(\mathbf{x}_i^l, \mathbf{y}_i^l) \in \mathcal{D}_l$. In our setting, each output is given by

$$\mathbf{y}_i^l = (\mathbf{p}_{1:T}^l, \mathbf{v}_{1:T}^l), \tag{5}$$

where $\mathbf{p}_{1:T}^l, \mathbf{v}_{1:T}^l \in \mathbb{R}^{T \times N \times d}$ denote the positions and velocities of a sequence of N nodes belonging to (possibly deformable) objects in d-dimensional space. An input $\mathbf{x}_i^l$ in our setting is given by

$$\mathbf{x}_i^l = (\mathbf{p}_0^l, \mathbf{v}_0^l, \mathbf{p}_{0:T}^{l,\text{ext}}, \mathbf{v}_{0:T}^{l,\text{ext}}, \mathbf{h}^l), \tag{6}$$

where $\mathbf{p}_0^l, \mathbf{v}_0^l \in \mathbb{R}^{N \times d}$ are the initial node positions and velocities. Here $\mathbf{p}_{0:T}^{l,\text{ext}}, \mathbf{v}_{0:T}^{l,\text{ext}} \in \mathbb{R}^{(T+1) \times N^{\text{ext}} \times d}$ are positions and velocities of 'external' objects, that is, objects that we do not wish to simulate, such as a rigid collider that interacts with the object of interest. Lastly, $\mathbf{h}^l \in \mathbb{R}^{(N+N^{\text{ext}}) \times d_\mathbf{h}}$ represents node features that remain constant over time, such as the node's type (deformable or collider) or whether a force is applied to the node. Initial condition $\mathbf{x}^l$ and simulation result $\mathbf{y}^l$ together result in a sequence of graphs $\mathcal{G}_{1:T}^l$ with $N + N_{\text{ext}}$ nodes. In our work, we assume a fixed graph structure: the connectivity and the number of nodes in $\mathcal{G}_t^l$ is constant over time and task datasets $\mathcal{D}_l$. An illustration of a meta dataset $\mathcal{D}$ is shown in Figure 2.

3.2 Incorporating Simulation Parameters into Meta-Training

For training, we could follow the procedure outlined in Section 2 and optimize the loss defined in Equation (4). However, contrary to the setup discussed in Section 2, we additionally assume access to the simulation parameters for each training task, i.e., $\{\rho^l\}_{l=1}^L$ but not for target tasks $\mathcal{D}^*$ rendering them useless in the standard meta-learning formulation. Here, we slightly extend the meta training such that we obtain additional learning signals from ρ^l. To that end, we use an additional parameterized neural network $f^\psi : \mathbb{R}^{d_r} \to \mathbb{R}^{d_\rho}$ with parameters ψ that aims to predict ρ^l from a

latent representation $\mathbf{r}^l$. Using the following definition for the joint likelihood between $\mathbf{y}$ and ρ, i.e.,

$$p_{\theta,\psi}(\mathbf{y}, \rho|\mathbf{x}, \mathcal{S}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \rho\end{bmatrix} \middle| \begin{bmatrix}\mathrm{dec}_\theta^\mu(\mathbf{x}, \mathbf{r})\\ f^\psi(\mathbf{r})\end{bmatrix}, \begin{bmatrix}\mathrm{dec}_\theta^\Sigma(\mathbf{x}, \mathbf{r}) & 0\\ 0 & 1\end{bmatrix}\right),$$

we obtain a novel per-task loss function as

$$\mathcal{L}_l(\phi, \theta, \psi) = -\mathbb{E}_{\mathcal{D}_l^c \subseteq \mathcal{D}_l}\left[\sum_{s=1}^{S_l} \log p_{\theta,\psi}(\mathbf{y}^l, \rho^l|\mathbf{x}, \mathcal{D}_l^c)\right],$$

where the full loss again sums over the task losses, $\mathcal{L}(\phi, \theta, \psi) = \sum_l \mathcal{L}_l(\phi, \theta, \psi)$. Intuitively, gradients with respect to the encoder parameters ϕ are informed by ρ via f^ψ. As another positive side-effect, we obtain an estimate $\hat{\rho} = f^\psi(\mathbf{r})$ which could be used for downstream tasks.

3.3 Spatiotemporal Encoder

In Section 2, we treated the encoder and decoder of the CNP architecture as black boxes that are responsible for generating a latent representation and a predictive distribution over outputs, respectively. However, since our data consists of non-standard structures, specifically graphs with both spatial and temporal components, we introduce the following novel architectures.

Recall that the encoder of a CNP generates a latent representation $\mathbf{r}^l \in \mathbb{R}^{d_r}$ from a set of input-output pairs, typically a context set $\mathcal{D}_l^c \subseteq \mathcal{D}_l$. Then, for each $(\mathbf{x}_i^l, \mathbf{y}_i^l) \in \mathcal{D}_l^c$ we separately generate a latent representation, i.e., $r_i^l = \mathrm{enc}_\phi(\mathbf{x}_i^l, \mathbf{y}_i^l)$ which are then aggregated in a permutation-invariant fashion to obtain $\mathbf{r}^l$. As discussed in Section 3.1, each input-output pair in our setting has a spatial and temporal component. We first aggregate over the temporal dimension and then over the spatial dimension. To that end, we combine the inputs $\mathbf{x}_i^l$ (see Equation (6)) and $\mathbf{y}_i^l$ (see Equation (5)) as

$$\mathbf{z}_i^l = (\mathbf{p}_{0:T}^l, \mathbf{v}_{0:T}^l, \mathbf{p}_{0:T}^{l,\text{ext}}, \mathbf{v}_{0:T}^{l,\text{ext}}, \mathbf{h}_{0:T}^l),$$

where $\mathbf{h}^l$ is simply copied for each time step. We ensure translation invariance by subtracting the initial mean position. To remove the temporal component, we apply a 1D convolutional neural network, i.e., $\hat{\mathbf{z}}_i^l = \mathrm{CNN}_{\text{1D}}^\phi(\mathbf{z}_i^l)$. Lastly, to obtain a spatial-independent latent representation, we apply a deep set encoder [35] on the node level, that is,

$$r_i^l = \mathrm{enc}_\phi(\mathbf{x}_i^l, \mathbf{y}_i^l) = f_{\text{outer}}^\phi\left(\frac{1}{N + N_{\text{ext}}} \sum_{n=1}^{N+N_{\text{ext}}} f_{\text{inner}}^\phi(\hat{\mathbf{z}}_{i,n}^l)\right),$$

where, f_{outer}^ϕ and f_{inner}^ϕ are neural networks. In this work, we choose deep sets as our spatial aggregation method, as they have demonstrated good performance in graph-level problems similar to ours [36]. Finally, to aggregate a whole set of input-output pairs, we follow Equation (2) and use a permutation invariant aggregation method.

3.4 Meta Neural Graph Operator Decoder

We propose a novel MaNGO decoder architecture that combines elements of MeshGraphNet (MGN) [19] and the Equivariant Graph Neural Operator (EGNO) [22]. EGNO predicts graph sequences using equivariant GNN layers and temporal convolutions in Fourier space. While beneficial in some settings, these equivariance constraints can limit performance when not required. We discuss this limitation in Appendix B, where we mathematically show that EGNO struggles on certain tasks. Alternatively, MGN models graph sequences autoregressively using Message Passing Networks (MPNs), which are highly effective for graph-based simulations [19]. However, autoregressive prediction is not ideal when paired with the current meta-learning training setup, see Appendix E for further discussions. To address these limitations, MaNGO retains the strengths of neural operator methods by replacing the equivariant GNN layers with MPNs and equivariant convolutions with a 1D CNN (see Figure 3). Formally, we use the MaNGO decoder to realize $\mathbf{y} = \mathrm{dec}_\theta(\mathbf{x}, \mathbf{r})$ (omitting the task index l for readability). Thus, we define a single MaNGO block as a mapping

$$(\mathbf{m}_{e,1:T}^k, \mathbf{m}_{v,1:T}^k) \mapsto (\mathbf{m}_{e,1:T}^{k+1}, \mathbf{m}_{v,1:T}^{k+1}),$$

following the notation in Section 2. The initial edge and note features $(\mathbf{m}_{e,1:T}^0, \mathbf{m}_{v,1:T}^0)$ are extracted from $(\mathbf{x}, \mathbf{r})$ by copying $\mathbf{r}$ to every node feature and replicating the initial graph T times. Additionally,

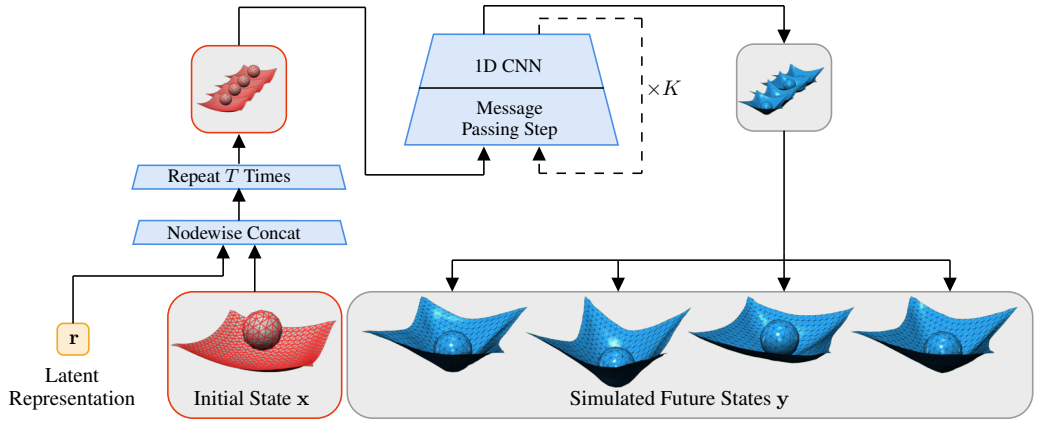

Figure 3: MaNGO Decoder: Our simulator takes a latent representation and an initial state as input. The initial state is combined and iteratively processed to generate a trajectory of graphs. By alternating between a message-passing network for spatial processing and a 1D CNN for temporal processing, the simulator produces accurate dynamic simulations.

we include time embeddings for every time step t and use relative positions as edge features. Further details on the feature creation are provided in Appendix C. As a first step of the MaNGO block, we leverage the MPN for a spatial update, that is, we process the edge and node features according to Equation (1) for each timestep separately to obtain a new tuple $(\mathbf{m}^{k+1}_{e,1:T}, \tilde{\mathbf{m}}^{k+1}_{v,1:T})$. The temporal update is subsequently performed using a 1D residual convolutional layer, that is,

$$\mathbf{m}^{k+1}_{v,1:T} = \text{Conv}^{\theta}_{\text{1D}}(\tilde{\mathbf{m}}^{k+1}_{v,1:T}) + \tilde{\mathbf{m}}^{k+1}_{v,1:T}.$$

After K MaNGO blocks the final node features $\mathbf{m}^{K}_{v,1:T}$ are used for predicting the node positions for every time-step. Specifically, a displacement vector is computed using a parameterized network f^{θ}, that is, $\mathbf{d}_{v,t} = f^{\theta}(\mathbf{m}^{K}_{v,t})$ to obtain node positions $\mathbf{p}_{v,t}$ as $\mathbf{p}_{v,t} = \mathbf{p}_{v,0} + \mathbf{d}_{v,t}$.

The resulting positions $\mathbf{p}_{v,t}$ define the graph sequence $\mathbf{y}$ over time. In this work, we do not predict input-dependent variances, and instead use a fixed $\text{dec}^{\Sigma}_{\theta}(\mathbf{x}, \mathbf{r}) := 1$ to stabilize and simplify the training scheme. If variances are required, for example, for uncertainty estimation, they can be easily predicted from the decoder as well. By alternating between spatial message passing and temporal convolution, the MaNGO simulator efficiently models time-series graph data while avoiding the pitfalls of equivariance over-constraints and autoregressive prediction.

4 Related Work

Learning-based forward simulators. Using deep neural networks to learn physical simulations has become an emerging research direction in scientific machine learning [19, 37, 38]. Deep learning-based approaches have demonstrated success in applications such as fluid dynamics [39, 40], aerodynamics [41, 19], and deformable object simulations [12, 42]. A popular class of learned neural simulators are Graph Network Simulators (GNSs) [43, 25]. GNSs utilize MPNs, a special type of GNN [44, 23] that representationally encompasses the function class of many classical solvers [33]. GNSs handle physical data by modeling arbitrary entities and their relations as a graph. Notably, all previously mentioned GNSs predict system dynamics iteratively from a given state, whereas we directly estimate entire trajectories, improving rollout stability and prediction speed. Related to our approach is the Equivariant Graph Neural Operator (EGNO) [22], which also predicts full trajectories using SE(3) equivariance to model 3D dynamics and capture spatial and temporal correlations. In this work, we adopt the trajectory prediction framework of EGNOs for our decoder (as shown in Figure 3) but remove the equivariance constraint. We justify this choice in Appendix B.

Material estimation. Determining physical parameters from observational data is a challenging and ill-posed problem [26]. Machine learning methods have shown success in inferring material properties from videos [27–29] and point clouds [30], but they rely on knowledge of the underlying

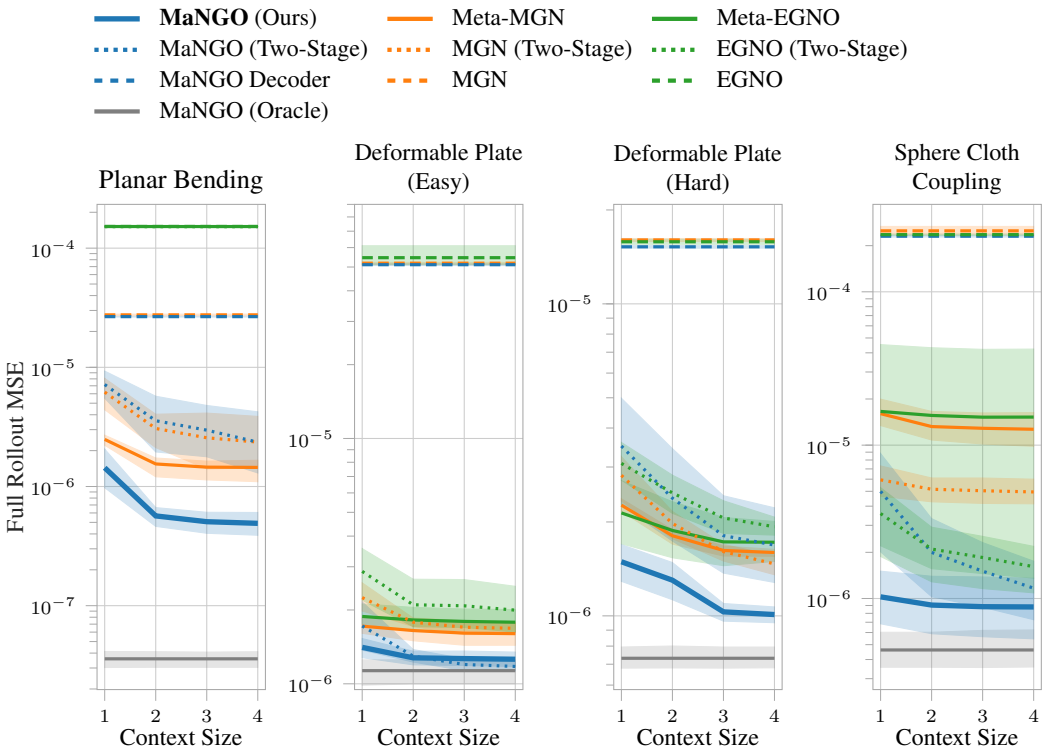

Figure 4: Performance comparison of our proposed methods and baselines across four datasets. We report the mean and 95% confidence interval of the *Full Rollout MSE* over five random seeds. The x-axis indicates the number of context samples used by the meta-learning approaches. We compare (i) meta-learning methods that employ a latent simulation parameter representation, (ii) a two-stage setup that explicitly predicts simulation parameters, and (iii) a baseline model that ignores context and performs direct simulation. The MaNGO (Oracle) variant, which has privileged access to the ground-truth simulation parameters, serves as an upper performance bound. Overall, MaNGO consistently outperforms both non–meta-learning and alternative meta-learning approaches, and achieves results close to the oracle.

PDE. Recently, approaches utilizing pre-trained Graph Network Simulators (GNS) to infer material parameters gained popularity due to their computational efficiency and differentiability [31, 32]. By back-propagating through the learned simulator, these methods estimate latent material codes directly from observations. While the approach in Zhao et al. [31] and ours share this goal, only we aggregate a context set of simulation trials to extract the underlying structure. Furthermore, our method does not require any backward pass and model weight updates, resulting in a faster adaptation.

Meta-learning and Neural Processes. Learning models that can quickly adapt to small context datasets at test time is often referred to as Meta-Learning. This emerging paradigm has found wide applications in fields such as language models [45] and robotics [46, 47], where it is commonly referred to as In-Context Learning. Meta-Learning is typically categorized into two approaches: optimization-based methods with few-shot examples and context-aggregation methods in a zero-shot fashion. A prominent example of the former is Model-Agnostic Meta-Learning (MAML) [48–51], which employs gradient-based updates to adapt the model to new tasks using few examples. In the latter, Neural Processes (NPs) [34, 52–57] aggregate latent features from a variable-sized context set to produce a latent task description that can be used directly during inference.

In this paper, we adopt Conditional Neural Processes (CNPs) [34] as the core mechanism for aggregating context from different simulation trials to produce a latent description representing the simulation parameters. A GNS is then trained to condition on this latent descriptor, enabling it to adapt dynamically to new materials. To the best of our knowledge, this is the first time such a framework has been proposed, combining meta-learning with graph network simulators to quickly adapt to unknown material properties given only few context-examples without requiring retraining or knowledge of the underlying PDE.

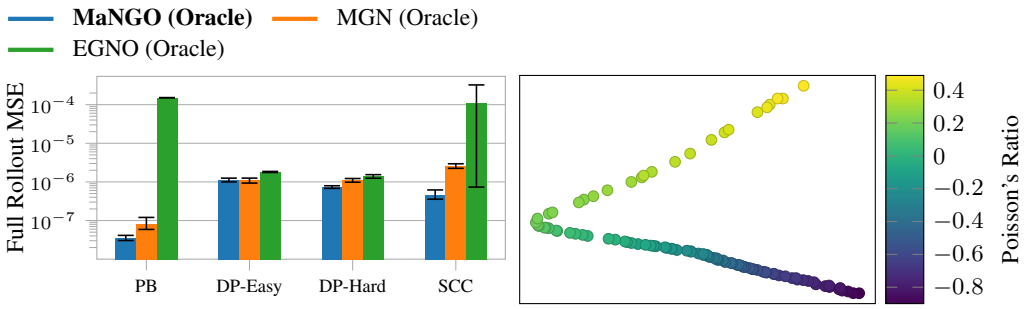

Figure 5: **Left:** Comparison of different GNS decoders with oracle information. MaNGO outperforms both MGN and EGNO, with the performance gap being more visible in the *Sphere Cloth Coupling* task due to its highly complex underlying dynamics. Additionally, EGNO fails to learn in the *Planar Bending* task, a phenomenon further analyzed in Appendix B. **Right:** Visualization of the 2D latent space for Deformable Plate (Easy), with points color-coded by Poisson's ratio. The structured separation (at Poisson value 0) shows that $\mathbf{r}$ effectively captures the underlying material properties.

5 Experiments

We validate MaNGO on four different simulation datasets derived from three distinct simulation platforms. All tasks are normalized to $[0, 1]^3$. The first dataset is a 2D *Deformable Plate* (DP) task [12], which has two variants: *DP-easy* and *DP-hard*. In *DP-easy*, the material property of interest is Poisson's ratio [58]. To increase complexity, *DP-hard* additionally varies Young's modulus and the initial velocity across trials within the same task. Next, we introduce a novel *Planar Bending* (PB) dataset, which simulates the bending of a 2D sheet subjected to two external forces perpendicular to the sheet. In this task, Young's modulus is the property of interest. The final dataset is a new *Sphere Cloth Coupling* (SCC) task, inspired by the Physion++ dataset [59], which involves a coupling system consisting of a sphere and a cloth. In this task, spheres of varying sizes are dropped onto the cloth. Their density is varied to influence the cloth's deformation upon contact.

The length of each simulation trial varies across datasets: 52 time steps for DP, 50 for PB, and 100 time steps for SCC to account for the increased complexity of simulating elastic behaviors. Further details about preprocessing steps and ground-truth simulators can be found in Appendix D.

5.1 Training setup and baselines

In this section, we present all methods evaluated in our experiments. Each model variant, whether meta-learning or non-meta-learning, has approximately three million trainable parameters, ensuring a fair comparison across architectures.

Meta-learning. We first describe the setup used for our meta-learning framework. Each task dataset $\mathcal{D}_l$ contains 16 different simulation trials, where each trial shares the same material properties ρ (e.g. Poisson's ratio or Young's modulus) but varies in initial conditions (e.g., collider position and size, or force position). During training, we randomly select a subset of S_l of a size between 1 to 8 to serve as the context dataset. This context is then used to predict a random trial[3] using different decoder methods. We compare our proposed MaNGO decoder against the EGNO architecture and a step-based MGN simulator. For testing, we split $\mathcal{D}_*$ into two distinct subsets. The first subset is used as the context dataset, while the second subset provides the initial conditions for predicting the full trajectory. We evaluate various context sizes to assess adaptation capability.

Two-Stage Training Setup. Since we assume access to the simulation parameters during training, we compare the meta-learning methods against a two-stage training scheme. In the first stage, we train an encoder to predict the simulation parameters given a simulation trial. In the second stage, we freeze the encoder and train a decoder that uses the predicted simulation parameters from the encoder as an additional input. Therefore, this baseline predicts an explicit representation rather than a latent representation of the simulation parameters. We refer to this approach as MaNGO/MGN/EGNO (Two-Stage).

[3]We empirically find that predicting a random trial per batch instead of all trials improves performance.

Vanilla GNS. We compare our method against non-meta-learning baselines, including EGNO, and MGN, and our proposed MaNGO decoder without context information. To assess the upper bound, we additionally train an oracle method using the MaNGO decoder which has access to the simulation parameters during training *and test time*. We refer to it as MaNGO-Oracle. For these baselines, the full training set containing all 16 simulation trials is provided. Additionally, for the MGN baseline, we follow the approach of Pfaff et al. [19], adding noise during training to mitigate error accumulation and stabilize rollouts at inference. We tune the input noise level for each task to maximize MGN's performance. The full experimental protocol, along with computational budget details, is available in Section F.

5.2 Results

Main evaluation. We compare the mean-squared error (MSE) between the ground truth and the predicted simulation averaged over all time steps. Overall, Figure 4 shows that meta-learning approaches consistently outperform non-meta-learning baselines, achieving a relatively low MSE close to the oracle method, MaNGO-Oracle. The two-stage training scheme is only competitive in the simplest environment, Deformable Plate (Easy), and across all environments it requires a larger context size to achieve its best performance. In general, within the meta-learning setting, performance improves with larger context sizes. Furthermore, we emphasize that strong performance is achieved with as few as 2 to 3 context samples (even under the high-complexity DP-hard dataset), highlighting the ability of our meta-learning approach to adapt quickly with minimal context data. All non-meta-learning baselines perform similarly. Interestingly, EGNO struggled to learn in the Planar Bending task, a phenomenon we further analyze in Appendix B. Qualitative visualizations of all methods and tasks are provided in Section G.

To directly compare our proposed MaNGO decoder with existing architectures, we evaluate it in the oracle setting, where material properties are known. As shown in Figure 5 on the left side, MaNGO is outperforming other baselines, confirming the effectiveness of our approach. We suspect that for MGN, auto-regressive prediction suffers from accumulated errors,

an issue particularly evident in the Sphere Cloth Coupling dataset, where highly nonlinear contact dynamics [60, 59] lead to MSE an order of magnitude higher than MaNGO. As for EGNO, while equivariance reduces the amount of required training data, it can also be overly restrictive, as real-world physics is not strictly E(3)-equivariant [61, 62]. Various factors, such as friction and gravity, can break this symmetry, leading to suboptimal generalization.

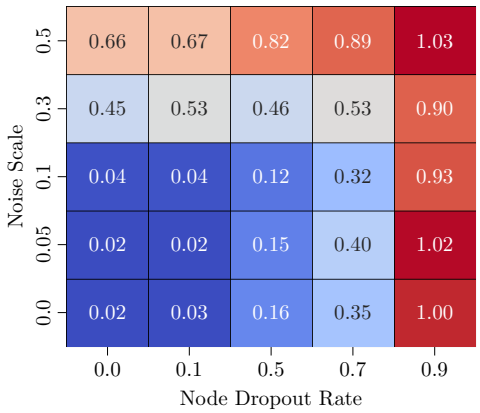

Figure 6: Robustness analysis of MaNGO on the Deformable Plate task under varying levels of Gaussian noise and node dropout, a setup mimicking real-world conditions. The normalized MSE is reported. Our method remains stable with up to 10% noise (relative to the width of the mesh) and 10% node dropout.

Latent visualization. To understand how the learned latent representation correlates with simulation parameters ρ, we visualize the 2D latent space with $\dim(\mathbf{r}) = 2$ for the Deformable Plate task, color-coded by Poisson's ratio. Figure 5 (right) shows a strong correlation between $\mathbf{r}$ and Poisson's ratio, with two linear trends emerging: one from 0 to 0.49 and the other from -0.9 to 0. This separation reflects the underlying material behavior — plates with a positive Poisson's ratio expand on contact, while those with a negative Poisson's ratio contract. The learned representation captures this distinction, indicating that the model encodes meaningful physical properties.

Robustness under sparse and noisy observations. Inspired by the setup in [31], we evaluate the robustness of MaNGO on the Deformable Plate task by introducing Gaussian noise into the observational context data and reducing the number of observed nodes at test time. To this end, we also introduce a small noise level of 0.05 for the context split during training to enhance the robustness of the encoder. We report the normalized MSE in the range $[0, 1]$, where the minimum is set by MaNGO-Oracle and the maximum by the non-meta-learning MaNGO approach (as shown in

Figure 4). As shown in Figure 6, the lower-left region of the matrix—corresponding to a 10% noise level and 10% node dropout—demonstrates near-optimal performance. Even with 50% of nodes unobserved, the performance drop remains around 15%, highlighting the model's ability to handle sparse and noisy observations. These results confirm that our encoder is robust under such conditions, reflecting real-world scenarios.

Runtime Efficiency and Memory Consumption A key advantage of MaNGO is its ability to predict entire trajectories in a single forward pass, enabling efficient batched inference over time. This design leads to substantial improvements in inference speed compared to traditional autoregressive next-step simulators. On the Sphere-Cloth Coupling benchmark, the CNP encoder requires approximately 6 ms to compute the latent representation, and the MaNGO decoder simulates the full trajectory in only 13 ms. In contrast, the MGN next-step simulator takes about 500 ms, as it predicts one step at a time, making MaNGO over an order of magnitude faster than the already efficient autoregressive neural simulator.

However, the batched inference and training scheme comes with increased memory requirements. GPU memory consumption scales approximately linearly with the number of predicted time steps, since the model retains intermediate activations across the temporal dimension. In our experiments, we observe that

$$\text{Memory(MaNGO)} \approx \text{Memory(MGN)} \times \text{Number of predicted steps.}$$

6 Conclusion

In this work, we explored data-driven adaptation of graph network simulators (GNS) via meta-learning. Specifically, we investigated the setting where simulation parameters are unknown at test time which would require retraining or labor-intensive data collection for existing methods. In a series of experiments, we demonstrated the potential of meta-learning for GNS, where our approach achieves accuracy on unseen material properties comparable to that of an oracle model. We view this work as a first stepping-stone towards the next generation of data-driven simulators, that are fast, differentiable, and capable of adapting to a wide variety of simulation settings. A discussion of the broader impact of this work is provided in Section A.

Limitations and Future Work One limitation of our current approach is its focus on a single data modality at test time. In practical scenarios, however, test-time data may come in diverse formats, such as point clouds captured by cameras. Our method does not yet support such modalities, which constrains its applicability in real-world settings. Addressing this limitation would require the development of new architectures capable of handling heterogeneous data.

A further limitation is that MaNGO assumes a fixed graph topology and requires substantial GPU memory when predicting full trajectories in a single forward pass. These constraints limit its scalability and applicability to systems with dynamic connectivity. A promising direction for future work is to adopt a hybrid decoding strategy that predicts shorter temporal segments while dynamically updating the graph structure after each segment. Such an approach could reduce memory demands and enable the modeling of systems with evolving topologies, bridging the gap between efficiency and flexibility.

Acknowledgements

This work is part of the DFG AI Resarch Unit 5339 regarding the combination of physics-based simulation with AI-based methodologies for the fast maturation of manufacturing processes. The financial support by German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) is gratefully acknowledged. The authors acknowledge support by the state of Baden-Württemberg through bwHPC, as well as the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the German Federal Ministry of Education and Research. We thank Philipp Becker for suggesting the title acronym used in this work.

References

[1] Abdelaziz Yazid, Nabbou Abdelkader, and Hamouine Abdelmadjid. A state-of-the-art review of the x-fem for computational fracture mechanics. *Applied Mathematical Modelling*, 33(12): 4269–4282, 2009.

[2] Olek C Zienkiewicz and Robert Leroy Taylor. *The finite element method for solid and structural mechanics*. Elsevier, 2005.

[3] Eva Stanova, Gabriel Fedorko, Stanislav Kmet, Vieroslav Molnar, and Michal Fabian. Finite element analysis of spiral strands with different shapes subjected to axial loads. *Advances in engineering software*, 83:45–58, 2015.

[4] TJ Chung. Finite element analysis in fluid dynamics. *NASA STI/Recon Technical Report A*, 78: 44102, 1978.

[5] Olek C Zienkiewicz, Robert Leroy Taylor, and Perumal Nithiarasu. *The finite element method for fluid dynamics*. Butterworth-Heinemann, 2013.

[6] Jerome J Connor and Carlos Alberto Brebbia. *Finite element techniques for fluid flow*. Newnes, 2013.

[7] Jian-Ming Jin. *The finite element method in electromagnetics*. John Wiley & Sons, 2015.

[8] Anastasis C Polycarpou. *Introduction to the finite element method in electromagnetics*. Springer Nature, 2022.

[9] CJ Reddy. *Finite element method for eigenvalue problems in electromagnetics*, volume 3485. NASA, Langley Research Center, 1994.

[10] Paul Maria Scheikl, Eleonora Tagliabue, Balázs Gyenes, Martin Wagner, Diego Dall'Alba, Paolo Fiorini, and Franziska Mathis-Ullrich. Sim-to-real transfer for visual reinforcement learning of deformable object manipulation for robot-assisted surgery. *IEEE Robotics and Automation Letters*, 8(2):560–567, 2022.

[11] Liman Wang and Jihong Zhu. Deformable object manipulation in caregiving scenarios: A review. *Machines*, 11(11):1013, 2023.

[12] Jonas Linkerhägner, Niklas Freymuth, Paul Maria Scheikl, Franziska Mathis-Ullrich, and Gerhard Neumann. Grounding graph network simulators using physical sensor observations. In *The Eleventh International Conference on Learning Representations*, 2023.

[13] Susanne C Brenner and L Ridgway Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[14] Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[15] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939738. URL `https://doi.org/10.1145/2939672.2939738`.

[16] Ying Da Wang, Martin J Blunt, Ryan T Armstrong, and Peyman Mostaghimi. Deep learning in pore scale imaging and modeling. *Earth-Science Reviews*, 215:103555, 2021.

[17] Jichao Li, Xiaosong Du, and Joaquim RRA Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134:100849, 2022.

[18] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

[19] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03409.

[20] Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kimberly Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. *Conference on Robot Learning (CoRL).*, 2022.

[21] Kelsey R Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William F Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[22] Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dccRCYmL5x.

[23] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

[24] Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. In *Robotics: Science & Systems*, 2021.

[25] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[26] Victor Isakov. *Inverse problems for partial differential equations*, volume 127. Springer, 2006.

[27] Tetsuya Takahashi and Ming C. Lin. Video-guided real-to-virtual parameter transfer for viscous fluids. *ACM Trans. Graph.*, 38(6), November 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356551. URL https://doi.org/10.1145/3355089.3356551.

[28] Pingchuan Ma, Tao Du, Joshua B. Tenenbaum, Wojciech Matusik, and Chuang Gan. RISP: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uSE03demja.

[29] Berthy T. Feng, Alexander C. Ogren, Chiara Daraio, and Katherine L. Bouman. Visual vibration tomography: Estimating interior material properties from monocular video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16210–16219, 2022. doi: 10.1109/CVPR52688.2022.01575.

[30] Bin Wang, Longhua Wu, Kangkang Yin, Libin Liu, and Hui Huang. Deformation capture and modeling of soft objects. *ACM Transactions on Graphics(Proc. of SIGGRAPH 2015)*, 34(4): 94:1–94:12, 2015.

[31] Qingqing Zhao, David B. Lindell, and Gordon Wetzstein. Learning to solve pde-constrained inverse problems with graph networks. 2022.

[32] Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VLw8ZyKfcm.

[33] Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural pde solvers. In *International Conference on Learning Representations*, 2022.

[34] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. *International Conference on Machine Learning*, 2018.

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. *Advances in Neural Information Processing Systems*, 2017.

[36] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer. In *International Conference on Machine Learning*, pages 3408–3430. PMLR, 2023.

[37] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[38] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

[39] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[40] Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=6niwHlzh10U.

[41] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, jun 2019. doi: 10.1007/s00466-019-01740-0. URL https://doi.org/10.1007%2Fs00466-019-01740-0.

[42] Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, Chang-Seung Woo, Ilho Kim, Seok-Woo Lee, Joon-Young Yang, et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. *arXiv preprint arXiv:2312.12467*, 2023.

[43] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[44] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[45] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[46] Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion, 2024. URL https://arxiv.org/abs/2411.12633.

[47] Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. *arXiv preprint arXiv:2403.19578*, 2024.

[48] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017.

[49] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas L. Griffiths. Recasting Gradient-Based Meta-Learning as Hierarchical Bayes. *International Conference on Learning Representations*, 2018.

[50] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic Model-Agnostic Meta-Learning. *Advances in Neural Information Processing Systems*, 2018.

[51] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian Model-Agnostic Meta-Learning. *Advances in Neural Information Processing Systems*, 2018.

[52] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo Jimenez Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[53] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *International Conference on Learning Representations*, 2019.

[54] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E. Turner. Meta-Learning Probabilistic Inference for Prediction. *International Conference on Learning Representations*, 2019.

[55] Christos Louizos, Xiahan Shi, Klamer Schutte, and M. Welling. The functional neural process. *Advances in Neural Information Processing Systems*, 2019.

[56] Michael Volpp, Fabian Flürenbrock, Lukas Großberger, Christian Daniel, and Gerhard Neumann. Bayesian context aggregation for neural processes. *International Conference on Learning Representations*, 2021.

[57] Michael Volpp, Philipp Dahlinger, Philipp Becker, Christian Daniel, and Gerhard Neumann. Accurate bayesian meta-learning by accurate task posterior inference. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=sb-IkS8DQw2`.

[58] Teik-Cheng Lim. *Auxetic Materials and Structures*. Springer Singapore, 01 2015. ISBN 978-981-287-274-6. doi: 10.1007/978-981-287-275-3. URL `https://doi.org/10.1007/978-981-287-275-3`.

[59] Fish Tung, Mingyu Ding, Zhenfang Chen, Daniel M. Bear, Chuang Gan, Joshua B. Tenenbaum, Daniel L. K. Yamins, Judith Fan, and Kevin A. Smith. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. *arXiv*, 2023.

[60] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=HyTqHL5xg`.

[61] Rui Wang, Elyssa Hofgard, Han Gao, Robin Walters, and Tess Smidt. Discovering symmetry breaking in physical systems with relaxed group convolution. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=59oXyDTLJv`.

[62] Putri A Van der Linden, Alejandro García Castellanos, Sharvaree Vadgama, Thijs P. Kuipers, and Erik J Bekkers. Learning symmetries via weight-sharing with doubly stochastic tensors. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=44WWOW4GPF`.

[63] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[64] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer, June 2012. doi: 10.1007/8415_2012_125. URL `https://hal.inria.fr/hal-00681539`.

[65] NVIDIA. Isaac sim, 2022. URL https://developer.nvidia.com/isaac-sim.

[66] NVIDIA. Nvidia physx sdk, 2022. URL https://developer.nvidia.com/physx-sdk.

[67] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are fully supported by our method section, as well as in the qualitative and quantitative results in the experiments section. The appendix provides more detailed results where required.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses current limitations of the approach, including the scope of the paper and assumptions made.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We proof that EGNN [63] is restricted to lower-dimensional predictions for certain underlying data manifolds in Appendix B

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We detail our method in Sections3 3.1, and provide additional information in the appendix where required. We detail our experimental setup and datasets used in the experiments section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide documented and run-able code in the supplement, and will release data upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide a high-level overview in Section 5, and detail hyperparameters, train/test splits and further information in Appendices D, E, F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report bootstrapped confidence intervals over five random seeds for all experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report compute resources in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics. We made sure that our research complies to the Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We include a discussion on broader impact in Appendix A.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not use pretrained language models, image generators or similar high-risk models in our approach, and do not scrape datasets. We still discuss potential cases for miss-use of our learned simulator in Appendix A.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We adapt the DeformablePlate dataset of Linkerhägner et al. [12], which we cite at the corresponding part of the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: At time of submission, we do not release new assets. We will open-source all used datasets, including documentation, after submission.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Braoder Impact Statement

Our proposed Graph Network Simulator, with its ability to adapt to new material properties through a small context set, offers significant advancements across fields that rely on computational modeling and simulation. By reducing the need for extensive simulation data and recalibration, it can lower computational costs while maintaining high accuracy. This adaptability could benefit industries ranging from materials science to robotics, enabling more efficient design and testing of novel materials.

However, this flexibility in simulating a wide range of material properties could also be misused in contexts where precise material behavior is critical, such as in the development of advanced weaponry or other high-risk technologies. While the primary intent is to advance scientific and engineering applications, ethical considerations must be taken into account to prevent unintended harmful uses.

B Limitations of the Equivariant Graph Neural Network

In this section, we analyze the limitations of the Equivariant Graph Neural Network (EGNN) [63], which serves as the GNN backbone of the Equivariant Graph Neural Operator (EGNO) baseline [22]. During our experiments, we observed that EGNO fails to predict any deformation in the Planar Bending task.

In this task, all neural operators receive an initial input configuration $\mathbf{p}_0$, representing a completely flat plane as the positions of the mesh. Consequently, all points in this initial graph lie within a plane E embedded in three-dimensional space. In this appendix, we demonstrate that the spatial output of any trained EGNN will also remain confined to a plane. This implies that an EGNN cannot solve the Planar Bending task, as the required deformed state does not lie within a single plane.

We establish this result by first proving that an EGNN maps all points in the specific plane

$$\boldsymbol{Z}_{=0} = \{(x, y, z) \in \mathbb{R}^3 \mid z = 0\}$$

back onto the same plane $\boldsymbol{Z}_{=0}$. To see this, consider any node $v \in \mathcal{G}$ with an initial position $\mathbf{p}_v^0 = (x, y, 0) \in \boldsymbol{Z}_{=0}$. Applying a single layer of EGNN to update the node positions, we obtain

$$\mathbf{p}_v^1 = \mathbf{p}_v^0 + C \sum_{w \in \mathcal{N}(v)} (\mathbf{p}_v^0 - \mathbf{p}_w^0) \varphi_x(\mathbf{m}_{vw}).$$

Examining the z-coordinate, we note that the message term $\varphi_x(\mathbf{m}_{vw})$ is scaled by $(\mathbf{p}_v^0 - \mathbf{p}_w^0)$. Since all $\mathbf{p}_v^0$ lie in $\boldsymbol{Z}_{=0}$, their z-coordinates are zero, meaning $(\mathbf{p}_v^0 - \mathbf{p}_w^0)$ has no z-component. Consequently, the updated z-coordinate in $\mathbf{p}_v^1$ remains zero, ensuring that $\mathbf{p}_v^1 \in \boldsymbol{Z}_{=0}$. By induction, this property holds for all subsequent layers, proving that EGNN maps $\boldsymbol{Z}_{=0}$ onto itself.

Next, we extend this result to any arbitrary plane E in three-dimensional space. Since E is a plane, there exists a transformation $T \in \mathrm{SE}(3)$ such that

$$T(E) = \boldsymbol{Z}_{=0}.$$

By the equivariance property of EGNN, there exists another transformation $S \in \mathrm{SE}(3)$ satisfying

$$\mathrm{EGNN}(\underbrace{T(\mathcal{G})}_{\subset \boldsymbol{Z}_{=0}}) = S(\mathrm{EGNN}(\mathcal{G})).$$

From our previous result, the left-hand side remains confined to $\boldsymbol{Z}_{=0}$. Thus, applying S^{-1} yields

$$\mathrm{EGNN}(\mathcal{G}) \subset S^{-1}\boldsymbol{Z}_{=0},$$

which is another plane in three-dimensional space. This establishes the key result: an EGNN always maps planar inputs to planar outputs, rendering it incapable of solving the Planar Bending task.

A similar argument extends to the full EGNO architecture since its equivariant temporal convolution layer in Fourier space is also equivariant and it preserves the property of mapping $\boldsymbol{Z}_{=0}$ onto itself. Thus, EGNO, like EGNN, is inherently unable to capture the required deformations in the Planar Bending task.

C Feature creation for the MaNGO decoder

This appendix provides a detailed explanation of the input processing for the MaNGO simulator. First, similar to the Spatiotemporal encoder, we create a sequence of graphs $\mathcal{G}_t$ over time. However, unlike the encoder, we repeat the initial positions and velocities of all deformable nodes across time, as the future positions—our target variable—are unknown and need to be estimated. In $\mathcal{G}_t$, we define the node features for a node v as the tensor $[\mathbf{r}, \mathbf{h}_v, \mathbf{v}_t, \mathrm{TE}(t)]$ consisting of the latent system identification $\mathbf{r}$, the node features $\mathbf{h}_v$, the velocity $\mathbf{v}_t$ and a time embedding of the time step t. Note that the position is not part of the node features. Instead, following [19], we compute for each edge e the relative position $\mathbf{p}^{\mathrm{rel}}_{e,t}$ to ensure translation invariance. The complete edge features are given as $[\mathbf{e}_e, \mathbf{p}^{\mathrm{rel}}_{e,t}, \mathrm{TE}(t)]$ consisting of the (time-independent) edge features, the relative position and a time step embedding.

D Datasets and Preprocessing information

In this section, we provide detailed information about the datasets used in our experiments. Table 1 summarizes the key characteristics of each dataset, including the dataset splits, simulation length, and the number of nodes used for prediction.

For brevity, we use the following abbreviations throughout the paper:

- **PB**: *Planar Bending*.
- **DP**: *Deformable Plate*, with two variants: *DP-easy* and *DP-hard*.
- **SCC**: *Sphere Cloth Coupling*.

Table 2 further details the training setup for each dataset, specifying the material properties considered and the variations in initial conditions. These variations influence the dynamics of each task, ensuring diverse training scenarios that test the generalization capabilities of our method.

Table 1: Dataset descriptions

Name	Train/Val/Test Splits	Number of Steps	Number of Nodes for Prediction
PB	460/50/50	50	225
DP-easy	600/100/100	52	81
DP-hard	600/100/100	52	81
SCC	600/100/100	100	400 (cloth) + 98 (sphere)

Table 2: Training setup for each dataset

Name	Material Properties	Initial Condition Variations
PB	Young's modulus	Two forces: (x, y) position, $\{-1, 1\}$ direction, constant magnitude
DP-easy	Poisson's ratio	Collider's x position, size, constant initial velocity
DP-hard	Young's modulus, Poisson's ratio	Collider's x position, size, **varied** initial velocity
SCC	Sphere's density	Sphere's size, same initial position

Planar Bending. We uniformly select Young's modulus from 10 to 500, from a very deformable to an almost stiff sheet. The boundary nodes of the sheet are kept in place.

Deformable Plate. The original task was introduced in [12], generated using Simulation Open Framework Architecture (SOFA) [64]. We extended to meta-learning setting by sampling Poisson's ratios between -0.9 to 0.49, under different trapezoidal meshes. We further increase the difficulty of this dataset by also randomizing the Young's modulus within a range from 500 to 10000 using Log-Uniform distribution.

Sphere Cloth Coupling. Each trajectory in this dataset is generated by selecting a sphere radius from the range $[0.2, 0.8]$. The material property of interest is the sphere's density, which varies between $[2.0, 100.0]$. The cloth is initialized in a stable state, remaining consistent across all tasks and trials. This dataset is created using NVIDIA Isaac Sim [65], which leverages PhysX 5.0 [66] to simulate position-based dynamics (PBD) particle interactions.

Table 3: **Left:** Training setup for each dataset. **Right:** Noise-scale per task for Auto-regressive methods

Parameter	Value
Node feature dimension	128
Latent representation dimension	128
Decoder hidden dimension	128
Message passing blocks	15
GNN Aggregation function	Mean
GNN Activation function	Leaky ReLU
Learning rate (Auto-regressive methods)	5.0×10^{-4}
Learning rate	1.0×10^{-4}
Optimizer	AdamW [67]
Min Context Size (Training)	1
Max Context Size (Training)	8
MaNGO-CNN Decoder Kernel size	7
CNN-Deepset Encoder Kernel size	3
Latent representation aggregation	Maximum

Task	Value
PB	5.0×10^{-4}
DP-easy	7.0×10^{-4}
DP-hard	7.0×10^{-4}
SCC	1.0×10^{-3}

For meta-learning setup, we assume a set S_i containing 16 different simulation trials, where each trial shares the same material properties (e.g., density, Poisson's ratio, Young's modulus) but varies in initial conditions (e.g., sphere size, force positions, and magnitudes).

E MGN Decoder for Conditional Neural Processes

In step-based prediction tasks, batches are typically shuffled to include different simulations and time steps. However, when using a Conditional Neural Process (CNP), only data from the same task can be used for each batch, which can impact performance. During hyper parameter optimization, we tested a modified version of the MGN where we reduced batch shuffling to mimic the CNP approach. This modification resulted in poorer performance compared to the standard MGN. This difference in performance may partially explain why Meta-MGN underperforms relative to MaNGO.

F Experimental Protocol

In order to promote reproducibility, we provide details of our experimental methodology. Table 2 presents the hyperparameters used in our experiments. For a comprehensive description of the creation of all datasets, please refer to Appendix D.

The training took place on an NVIDIA A100 GPU, with each method given the same computation budget of 48 hours. Consequently, the number of epochs varied, as the batching differed significantly between the meta-learning methods and the step-based MeshGraphNet (MGN). In total, generating the results presented in this paper required approximately 8,500 GPU hours.

We conducted a multi-staged grid-based hyperparameter search for the learning rate, input noise, and other hyperparameters as residual connections and layer norms. We did not use the test data for this, but tuned all hyperparameters on a separate validation split. This split was also used to determine the best epoch checkpoint to mitigate any overfitting effects. Hyperparameter tuning required an additional computational budget of approximately 6,000 GPU hours.

For MGN, we included velocity features of the current step.

G Visualizations

In this section, we present qualitative results for all tasks and methods discussed in the main paper.

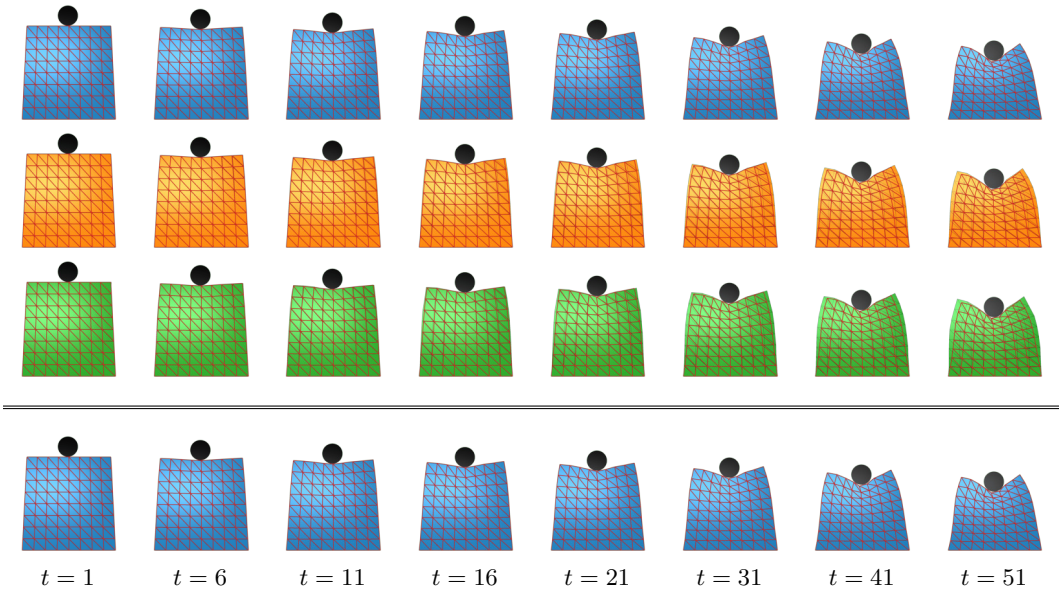

Figure 7: Simulation over time of an exemplary test trajectory from the **DP-easy** task. The figure compares predictions from **MaNGO**, MGN, and EGNO. The last row, MaNGO-Oracle, is separated by a horizontal line and represents predictions using oracle information. The **context set size** is set to 4. All visualizations show the colored **predicted mesh**, with a **wireframe** representing the ground-truth simulation. **MaNGO** accurately predicts the correct material properties, leading to a highly accurate simulation.

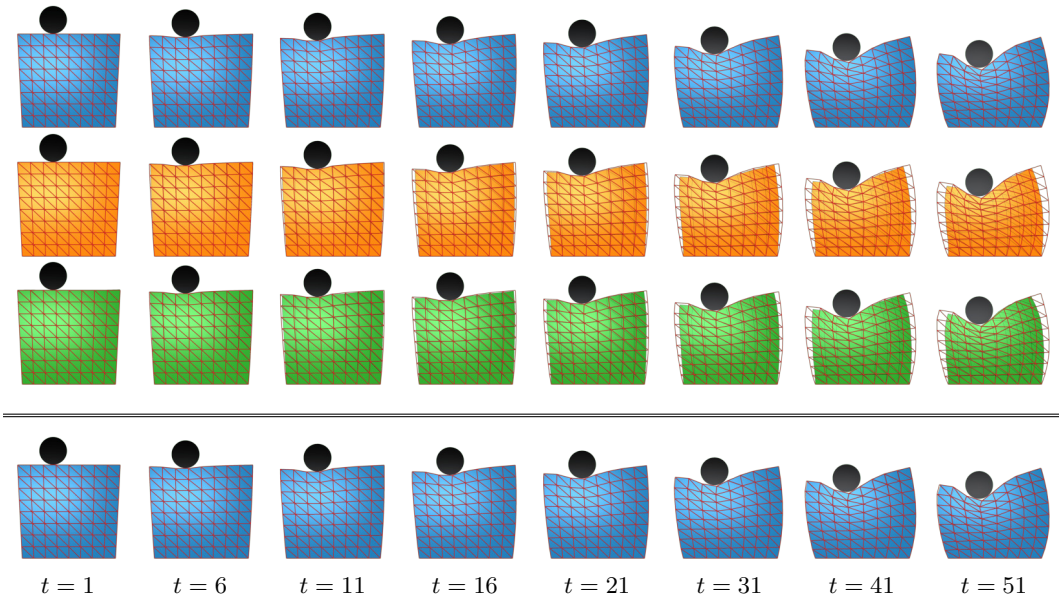

Figure 8: Simulation over time of an exemplary test trajectory from the **DP-hard** task. The figure compares predictions from **MaNGO**, MGN, and EGNO. The last row, MaNGO-Oracle, is separated by a horizontal line and represents predictions using oracle information. The **context set size** is set to 4. All visualizations show the colored **predicted mesh**, with a **wireframe** representing the ground-truth simulation. **MaNGO** accurately predicts the correct material properties, leading to a highly accurate simulation.

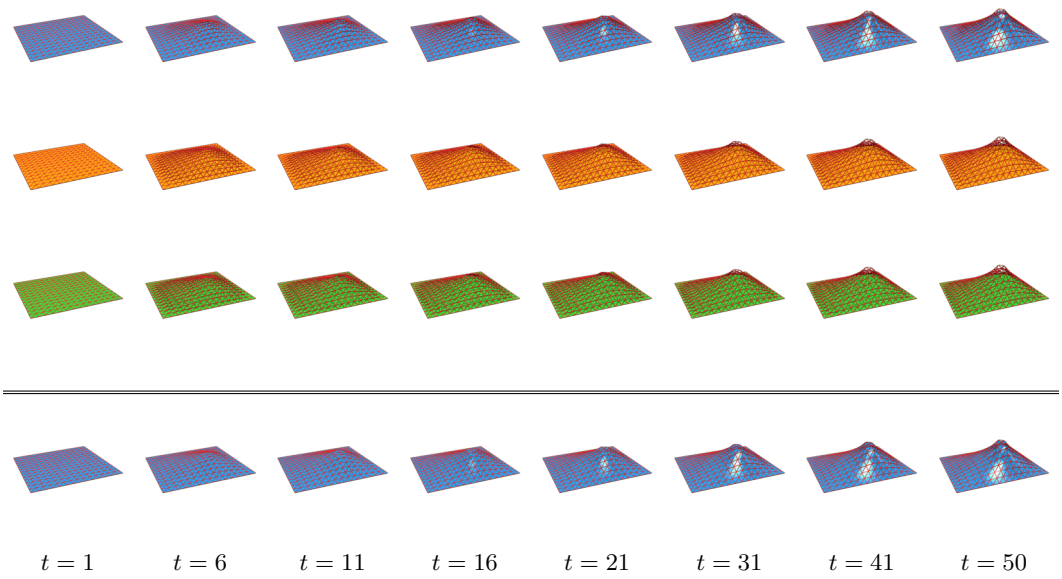

Figure 9: Simulation over time of an exemplary test trajectory from the **PB-easy** task. The figure compares predictions from **MaNGO**, MGN, and EGNO. The last row, MaNGO-Oracle, is separated by a horizontal line and represents predictions using oracle information. The **context set size** is set to 4. All visualizations show the colored **predicted mesh**, with a **wireframe** representing the ground-truth simulation. **MaNGO** accurately predicts the correct material properties, leading to a highly accurate simulation.

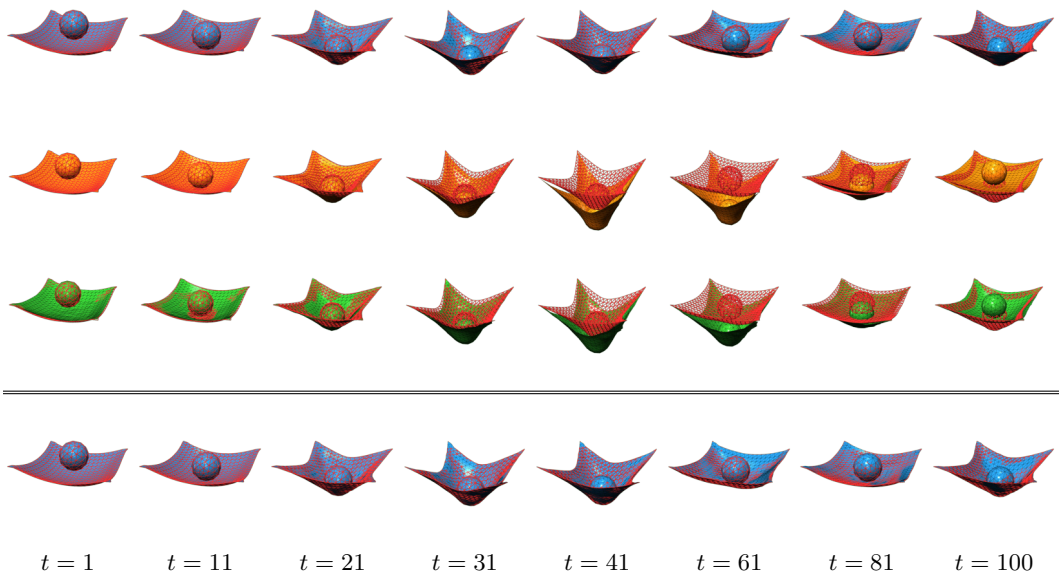

Figure 10: Simulation over time of an exemplary test trajectory from the **Sphere Cloth Coupling** task. The figure compares predictions from **MaNGO**, MGN, and EGNO. The last row, MaNGO-Oracle, is separated by a horizontal line and represents predictions using oracle information. The **context set size** is set to 4. All visualizations show the colored **predicted mesh**, with a **wireframe** representing the ground-truth simulation. **MaNGO** accurately predicts the correct material properties, leading to a highly accurate simulation.