PNEO: PAIRWISE NEURAL ENERGY OPERATORS FOR SCALABLE COLLISION DYNAMICS OF 3D DEFORMABLE OBJECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate prediction of 3D deformable-object collision dynamics is important for robotic interaction, world model, and a broad range of engineering simulation problems. Numerical methods are accurate but become increasingly inefficient as the number of interacting objects grows. Learning-based simulators predict faster, but typically require training on large multi-object scenes to scale up, limiting generalisation to unseen object counts. We introduce the *Pairwise Neural Energy Operator* (PNEO), which learns collision dynamics from two-object interactions and composes the same learned interaction law across multi-object interactions. PNEO models the conservative energy exchange and dissipation of each contacting object pair using scalar potential and Rayleigh dissipation terms, which are summed over the contact graph to recover multi-object dynamics. This pair-local formulation makes the learned interaction law independent of scene cardinality. Experiments show that a single PNEO trained only on two-object collisions transfers zero-shot across object count (up to 50), point-cloud resolution, and mechanical properties without retraining or fine-tuning. PNEO outperforms baselines across the main multi-object evaluation settings while maintaining stable trajectory, contact, and deformation accuracy as scene size increases.

1 INTRODUCTION

Predicting 3D dynamics of deformable objects is fundamental to a wide range of domains, including robotic manipulation (Zhang et al., 2025; Li et al., 2025; Chen et al., 2026), physical reasoning (Li et al., 2026; Chen et al., 2025; Wang et al., 2025), and computer graphics (Kim & Eberle, 2022). Accurately simulating the dynamics is challenging due to the coupled effects of deformation, contact, and rapidly changing interaction constraints. Numerical solvers, such as finite elements (Sifakis & Barbic, 2012), the material point method (Stomakhin et al., 2013), and particle discretisations (Bender & Koschier, 2015), address this complexity by discretising the governing mechanics and explicitly enforcing material and contact laws, enabling accurate simulation of the resulting dynamics. However, these methods become increasingly expensive as the number of interacting objects grows.

Learning-based simulators have emerged as a promising alternative to classical numerical methods by learning forward dynamics directly from data to reduce the simulation cost (Greydanus et al., 2019; Lu et al., 2019; Raissi et al., 2019). Among them, graph neural networks (GNNs) have become a predominant formulation for particle- and mesh-based dynamics, as graph representations naturally align with the discretised structure of simulated objects and their interactions. GNN-based simulators represent objects and the environment as graph nodes, with contact or interaction edges encoding inter-object relations (Chen et al., 2026; Pfaff et al., 2021; Li et al., 2019; Yu et al., 2024; Anandkumar et al., 2019). However, their memory and computational costs increase with the number of nodes and interaction edges, limiting scalability during both training and inference.

Recently, RigidFormer (Dou et al., 2026) demonstrated scalable dynamics prediction for scenes containing more than 200 rigid objects. This scalability benefits from the substantially lower-dimensional state space of rigid bodies compared with deformable objects. Moreover, its large-scale predictions rely on training scenes with comparable object counts, resulting in substantial training costs as scene scale increases. For deformable objects, existing neural dynamics models (Sanchez-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

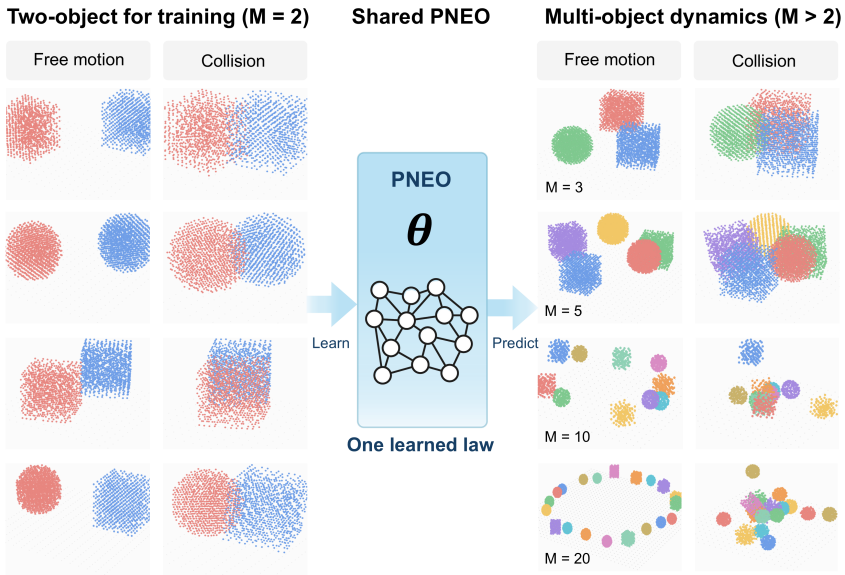

Figure 1: Overview of PNEO.

Gonzalez et al., 2020; Han et al., 2022; Yu et al., 2024) often couple object dynamics and inter-object interactions within a scene-level representation, limiting their ability to generalise to out-of-distribution (OOD) scenes with larger object counts. Neural physics engine (NPE) (Chang et al., 2017) demonstrated that learning shared pairwise interactions and composing them across objects can enable generalisation to unseen scene sizes. However, NPE was limited to simple 2D rigid-body systems and composed latent interaction embeddings to predict object-level motion without modelling the underlying interaction mechanics.

By leveraging the compositional physics, we propose the *Pairwise Neural Energy Operator* (PNEO) for the scalable 3D deformable object dynamics. PNEO learns the interaction laws between object pairs independent of scene cardinality as shown in Figure 1. We decouple multi-object dynamics into object-local free motion and inter-object collisions, and further decompose collisions into pairwise contacts governed by local dynamic states, contact geometry, and mechanical properties. Rather than composing latent interaction features, PNEO explicitly parameterises each pairwise interaction through a scalar contact potential and Rayleigh dissipation function learned from two-object collisions. These pairwise terms are summed over the dynamic contact graph and differentiated to recover multi-object dynamics. As scene size increases, only the contact graph changes, while the same learned pairwise law is reused across object counts. Our contribution are summarized as follows:

- We formulate multi-object deformable collision dynamics as the composition of pairwise scalar potentials and Rayleigh dissipation energy terms, enabling the same learned interaction law to be reused over arbitrary contact graphs.
- We introduce the *Pairwise Neural Energy Operator* (PNEO), which learns pair-local conservative and dissipative collision dynamics independently of scene cardinality.
- We demonstrate that a single model trained only on two-object collisions generalises without retraining to scenes containing up to 50 objects, outperforms competing learned simulators across the main multi-object settings, and transfers across point-cloud resolutions and varied mechanical properties.

2 Related Work

GNN-based neural dynamics. GNN-based methods discretise the simulated geometry into particles or mesh vertices as graph nodes, encode physical interactions between them through graph edges, and predict forward dynamics via message passing (Battaglia et al., 2016; Li et al., 2019;

Belbute-Peres et al., 2020; Rubanova et al., 2021; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Deng et al., 2025b). Subsequent works have leveraged certain symmetries of the physical system, i.e., geometric equivariance (Satorras et al., 2021; Han et al., 2022; Huang et al., 2025; Xu et al., 2024) and mechanical equivariance (Sharma & Fink, 2026) to improve consistency with the underlying mechanics. To enable GNNs to model the complex contact dynamics of deformable bodies, several works introduce contact-aware graph constructions and specialised interaction mechanisms (Pfaff et al., 2021; Dubey et al., 2026; Yu et al., 2024; Chen et al., 2026; Ying et al., 2026). However, these methods remain limited in scalability on 3D, as scaling to larger numbers of 3D deformable objects requires extensive training on large multi-object datasets and introduce substantial training cost due to the increasing complexity of contact dynamics. To improve scalability, Neural Force Field (Li et al., 2026) learns continuous object-level force fields and demonstrates OOD transfer to unseen rigid-body dyanmics with up to nine objects. RigidFormer further scales 3D rigid-body collision prediction to more than 200 objects (Dou et al., 2026), although model relies on training on the dataset with comparable object counts. Both approaches derive their scalability from the low-dimensional dynamics of rigid-body motion. This compact representation does not directly extend to deformable bodies, which require modelling time-varying geometry and internal deformation.

Energy-based neural dynamics. Energy-based neural dynamics provide a physically structured alternative to predicting dynamics. Hamiltonian and Lagrangian neural networks learn scalar mechanical functions and recover dynamics through differentiation, introducing conservation structure that improves long-horizon consistency in conservative systems (Greydanus et al., 2019; Cranmer et al., 2020; Zhong et al., 2019b; Finzi et al., 2020). However, classical Hamiltonian formulations assume conservative systems and cannot directly represent energy dissipation. Subsequent works therefore extend energy-based models with dissipation and external forcing to capture non-conservative dynamics (Zhong et al., 2019a; Desai et al., 2021; Sosanya & Greydanus, 2022; Zhong et al., 2021; Deng et al., 2025a). NEXUS (Ying et al., 2026) extends this energy-based view to contact-rich 3D deformable dynamics using conservative potentials and Rayleigh dissipation over structural and contact graphs, enabling composable physical effects, explicit control, and improved long-horizon predictions. However, NEXUS remains limited to small multi-object scenes, with scalability beyond five objects unexplored.

3 METHOD

3.1 PROBLEM SETUP AND OVERVIEW

Figure 2(a)-(d) summarises the architecture and the composition of pairwise interactions into multi-object dynamics.

Scene representation and mechanical properties. We consider a scene containing M deformable objects interacting with one another within an otherwise static environment. Let $\mathcal{O}$ denote object labels with $|\mathcal{O}| = M$. Object $A \in \mathcal{O}$ is discretised by N_A particles, and the dynamic scene state is represented by the stacked particle positions and velocities $x_t, v_t \in \mathbb{R}^{N\times 3}$ respectively. The environment is represented by a fixed point cloud x_{env} with outward normals n_{env}. Each dynamic particle has mass m_i, and we define the mass matrix as $\mathbf{M} = \text{diag}(m_i) \otimes I_3$. Our method represents each object as a structural graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0, \ell^0)$, where $\mathcal{V}$ denotes its particles, $\mathcal{E}_0$ contains intra-object structural connections, and ℓ^0 stores their rest lengths. The structural graph is constructed from the initial geometry and remains fixed throughout the rollout. Inter-object and object–environment interactions are instead represented by a dynamic contact graph

$$\mathcal{G}_{\text{con}}(x_t) = \mathcal{E}_{\text{o-o}}(x_t) \cup \mathcal{E}_{\text{o-e}}(x_t), \tag{1}$$

where $\mathcal{E}_{\text{o-o}}$ and $\mathcal{E}_{\text{o-e}}$ denote object–object and object–environment contact pairs, respectively, and are rebuilt from the current state during rollout. Figure 2(a) exemplifies the graph construction using a three-object scene. The mechanical properties and controls include object stiffness k, gravity g, and optional external forces $f^{\text{ext}}_{i,t}$. We index the T stored states by $t = 1, \ldots, T$, with (x_1, v_1) denoting the initial state. Given this state, geometry, and mechanical properties, our goal is to predict the remaining rollout $\{x_t, v_t\}_{t=2}^{T}$.

Compositional neural energy dynamics. Our method aims to learn the local energy dynamics of two-body interactions from a only two-body collision dataset with a shared *Pairwise Neural Energy*

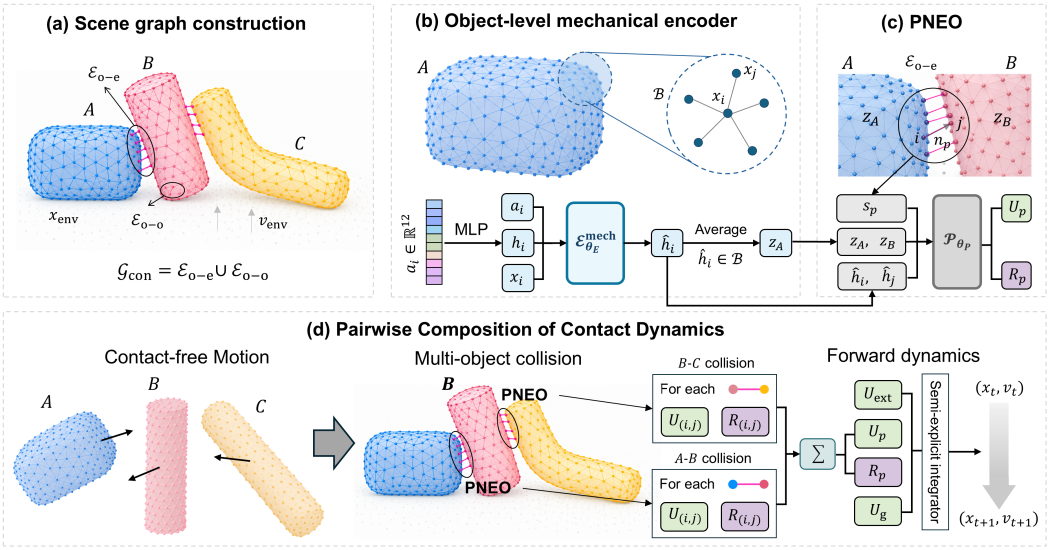

Figure 2: **PNEO architecture for compositional contact dynamics.** (a) Scene graph construction for objects and the environment. (b) Object-local mechanical encoding, comprising pointwise feature mapping and neighbourhood aggregation. (c) PNEO for inferring the contact potential and dissipation between two interacting objects. (d) Composition and integration of multi-object dynamics using a shared interaction law learned from pairwise collisions.

Operator (PNEO), and compose these learned interactions into multi-object dynamics through a collision graph. Given the (x_t, v_t) and mechanical properties of two interacting objects, PNEO predicts the local coefficients governing their conservative and dissipative contact responses, thereby parameterising how energy is exchanged and dissipated during collision. The total interaction potential and Rayleigh dissipation decompose additively over active pairs, allowing multi-object dynamics to be composed by applying the same PNEO to each collision-graph edge and summing the resulting pairwise terms. Although PNEO is learned from two-body interactions, its pair-local and shared parameterisation ensures that increasing M changes only the collision graph and the number of active pairwise evaluations, rather than the learned interaction law itself. Section 3.2 develops the pairwise dynamics and PNEO, Section 3.3 extends them to arbitrary object counts, and Section 3.4 describes integration on forward dynamics and training.

3.2 Pairwise Energy Dynamics

Energy-based formulation. We formulate collision dynamics through scalar potential and Rayleigh dissipation functions rather than directly predicting the dynamics. The energy formulation enables interaction energies and dissipation to decompose additively because they are scalar terms. Differentiation preserves this decomposition because it is a linear operation, allowing pairwise interaction laws to compose into multi-object dynamics. It also provides a unified interface for physical control, as external controls (such as the gravity) can be incorporated as additive scalar potentials and differentiated together with the learned interaction terms. Let θ denote the learnable parameters of the neural models that parameterise the energy and dissipation terms. A potential U_θ and a Rayleigh dissipation function R_θ then define the force field:

$$\mathbf{M}\,\ddot{x} = F_\theta(x, \dot{x}) = -\frac{\partial U_\theta}{\partial x}(x) - \frac{\partial R_\theta}{\partial v}(x, \dot{x}). \tag{2}$$

The conservative and dissipative contributions are

$$U_\theta = U_{\mathrm{g}} + U_{\mathrm{ext}} + U_{\mathrm{str},\theta} + U_{\mathrm{con},\theta}, \qquad R_\theta = R_{\mathrm{str},\theta} + R_{\mathrm{con},\theta}, \tag{3}$$

where U_{g} is the gravitational potential, U_{ext} represents prescribed external forcing, and $U_{\mathrm{str},\theta}, U_{\mathrm{con},\theta}$ and $R_{\mathrm{str},\theta}, R_{\mathrm{con},\theta}$ denote structural/contact potentials and dissipations, respectively. Let $\mathcal{P}(x) =$

$\mathcal{E}_0 \cup \mathcal{E}_{\text{o-o}}(x) \cup \mathcal{E}_{\text{o-e}}(x)$ denote the complete set of interacting pairs, with $p = (i, j)$ an unordered pair. The same dynamics can then be expressed in pairwise form as

$$U_\theta(x) = U_{\text{g}}(x) + U_{\text{ext}}(x) + \sum_{p \in \mathcal{P}(x)} U_p(x), \qquad R_\theta(x, v) = \sum_{p \in \mathcal{P}(x)} R_p(x, v), \tag{4}$$

where U_p and R_p denote the potential and dissipation associated with each structural or contact pair (e.g., $U_{\text{str},\theta} + U_{\text{con},\theta}$ and $R_{\text{str},\theta} + R_{\text{con},\theta}$, respectively).

Pairwise Neural Energy Operator. Consider two distinct objects $A, B \in \mathcal{O}$, with particle sets $\mathcal{V}_A$ and $\mathcal{V}_B$. For each candidate contact pair $p = (i, j)$ with $i \in \mathcal{V}_A$ and $j \in \mathcal{V}_B$, we denote its local contact state by

$$s_p = (\delta_p, n_p, v_p^n), \tag{5}$$

where δ_p, n_p, and v_p^n denote the penetration depth, contact normal, and normal relative velocity, respectively. Their geometric definitions are provided in Appendix A.1. Each particle in $\mathcal{V}$ is represented by the local mechanical field

$$a_i = \left[x_i - x_i^0,\ v_i,\ \log m_i,\ \tilde{k}_i,\ g_{\text{on}},\ f_i^{\text{ext}}\right] \in \mathbb{R}^{12}, \tag{6}$$

where x_i^0 denotes the fixed reference position of particle i, $\tilde{k}_i$ denotes its normalised log stiffness, and $g_{\text{on}} \in \{0, 1\}$ indicates whether gravity is disabled or enabled. For a particle $i \in \mathcal{V}_A$, we define its object-local ball-query neighbourhood by $\mathcal{B}(x_i,\, r) = \{j' \in \mathcal{V}_A : \|x_{j'} - x_i\| \le r\}$, where r is a fixed physical radius and the centre particle i is included. j' represents the indexes of neighbours within object A, whereas j denotes the contact endpoint on object B as shown in Figure 2(b) and (c). As shown in Figure 2(b), each a_i is first projected to a mechanical latent representation h_i by a learnable MLP. The object-level mechanical encoder $\mathcal{E}_{\theta_E}^{\text{mech}}$ (we adapted the encoder of GNO (Kovachki et al., 2023)) with learnable parameters θ_E (the detailed implementation is provided in Appendix A.2) then aggregates over $\mathcal{B}(x_i,\, r)$ and returns encoded particle latents $\hat{h}_i$ together with an object descriptor z_a by averaging on $\hat{h}_i$ from all particles.

$$\left(\{\hat{h}_i\}_{i \in \mathcal{V}_A},\, z_A\right) = \mathcal{E}_{\theta_E}^{\text{mech}}\left(\{(x_i, a_i, h_i)\}_{i \in \mathcal{V}_A}; \{\mathcal{B}(x_i,\, r)\}_{i \in \mathcal{V}_A}\right),\, z_A \quad = \frac{1}{|\mathcal{V}_A|} \sum_{i \in \mathcal{V}_A} \hat{h}_i. \tag{7}$$

$\mathcal{E}_{\theta_E}^{\text{mech}}$ is shared across different objects to compute the mechanical representation (including $\{\hat{h}_j\}_{j \in \mathcal{V}_B}$ and z_B shown in Figure 2(c)). Let $d_p = \|x_i - x_j\|$, we construct the pairwise feature to predict the interaction dynamics of two objects (A and B) as,

$$\xi_p = \left[\hat{h}_i,\ \hat{h}_j,\ z_A,\ z_B,\ d_p n_p,\ d_p,\ \delta_p,\ v_p^n,\ \tilde{k}_i,\ \tilde{k}_j\right]. \tag{8}$$

where ξ_p represents the learned mechanical context of both objects with their instantaneous contact geometry and stiffness properties. A shared pairwise head $\mathcal{P}_{\theta_P}$ then predicts the interaction parameters

$$\begin{bmatrix} z_p^k \\ z_p^c \end{bmatrix} = \mathcal{P}_{\theta_P}(\xi_p), \qquad \theta = \{\theta_{\text{emb}}, \theta_E, \theta_P\}, \tag{9}$$

where θ_P denotes the parameters of $\mathcal{P}_{\theta_P}$ and θ collects all learnable parameters of PNEO. The outputs determine positive and dimensionless stiffness and damping gains g_p^k, g_p^c, which are scaled by the physical reference values (e.g., $\bar{k}_c$, $\bar{c}_c$) to give the contact stiffness k_p and normal damping coefficient c_p:

$$g_p^k = \frac{\text{softplus}(z_p^k)}{\text{softplus}(0)}, \qquad g_p^c = \frac{\text{softplus}(z_p^c)}{\text{softplus}(0)}, \qquad k_p = \bar{k}_c\, g_p^k, \qquad c_p = \bar{c}_c\, g_p^c, \tag{10}$$

where softplus denotes the softplus function (Dugas et al., 2000). The predicted coefficients parameterise the pairwise contact potential and Rayleigh dissipation shown in Figure 2(c),

$$U_p = \tfrac{1}{2}\, w_p k_p \varphi(\delta_p)^2, \qquad R_p = \tfrac{1}{2}\, w_p \mathbf{1}[\delta_p > 0]\, c_p \varphi(-v_p^n)^2, \tag{11}$$

where w_p is a prescribed contact-counting weight, equal to one for each distinct unordered contact pair. $\varphi(u) = \max(u, 0)$ is the positive-part function. The corresponding interaction forces are obtained by differentiation,

$$f_{i,p} = -\frac{\partial U_p}{\partial x_i} - \frac{\partial R_p}{\partial v_i} = -f_{j,p}. \tag{12}$$

The initial projection, $\mathcal{E}^{\text{mech}}_{\theta_E}$, and $\mathcal{P}_{\theta_P}$ are learned from solely two-object collisions and subsequently shared across all objects, contact pairs, and scenes. Since neither their inputs nor parameter dimensions depend on the object count M, the learned pairwise interaction law can be applied directly to scenes with different numbers of objects. Structural interactions are parameterised separately on the fixed intra-object structural graph.

3.3 Composing multi-object collision dynamics

For active object–object contact pairs in the scene, the shared PNEO is applied to each edge $p = (i, j)$ using ξ_p, with the object labels in equation 8 referring to the two endpoint objects of that edge. The resulting scalar contributions are summed as in equation 4, as illustrated in Figure 2(d). At time t, a particle $i \in \mathcal{V}_A$ can simultaneously contact particles from several other objects. Let $\Omega_A = \mathcal{O} \setminus \{A\}$ denote the set of all objects other than A. For instance, in a scene containing objects A, B, and C, we have $\Omega_A = \{B, C\}$. For each $\sigma \in \Omega_A$, let $\mathcal{C}_i^\sigma(t)$ denote the set of particles of object σ in active contact with i:

$$\mathcal{C}_i^\sigma(t) = \left\{ j \in \mathcal{V}_\sigma : (i, j) \in \mathcal{G}_{\text{con}}(x_t),\ \delta_{(i,j)}(t) > 0 \right\}. \tag{13}$$

Here Ω_A contains object labels, while $\mathcal{C}_i^\sigma(t)$ contains particle indices and is empty when i has no active contact with object σ. For each detected contact edge $p = (i, j)$, PNEO constructs the $U_{(i,j)}$ and $R_{(i,j)}$ values from its predicted coefficients at the common scene state (x_t, v_t). By differentiating the sum of these pairwise contributions, the total contact force on i is:

$$f_i^{\text{o-o}}(t) = -\frac{\partial}{\partial x_i} \sum_{\sigma \in \Omega_A} \sum_{j \in \mathcal{C}_i^\sigma(t)} U_{(i,j)} - \frac{\partial}{\partial v_i} \sum_{\sigma \in \Omega_A} \sum_{j \in \mathcal{C}_i^\sigma(t)} R_{(i,j)}. \tag{14}$$

Our method combines the contact forces with the remaining contributions and uses a numerical integrator to roll out the coupled forward dynamics of all M objects in the scene.

3.4 Forward integration and training

We perform forward dynamics over the full M-object system given (x_1, v_1) using semi-explicit Euler integration with contact detection as shown in Figure 2(d). At each contact substep, contact detection updates the graph from the current state, and the shared PNEO pairwise head is re-evaluated using these cached embeddings and the current contact geometry and relative velocities to obtain new $U_{i,j}$ and $R_{i,j}$ terms. These terms are added to the remaining energy and dissipation components following equation 14 and equation 3, differentiated to obtain the net force, and integrated to update all object states. Appendix A.3 provides the full rollout algorithm, integration details, and training objective.

4 Experiments

We evaluate whether PNEO, trained only on two-object interactions, generalises to larger and OOD settings. Section 4.1 introduces the benchmark, metrics, and baselines. Section 4.2 compares PNEO with competing methods on multi-object rollout prediction. Section 4.3 evaluates zero-shot generalisation across object counts, mechanical properties, and input resolutions. Finally, Section 4.4 examines the rationale for training exclusively on two-object interactions.

4.1 Experimental setup

Simulation benchmark. We construct scenes of deformable objects undergoing internal deformation, object–object collision, and floor contact under gravity, with optional local external forces. Each object is represented by 128 sampled particles, while the floor is represented by 1024 static

Table 1: Comparison with baselines trained at different M. Our model is trained only at $M = 2$ and evaluated zero-shot at $M \in \{3, 5\}$. Baselines are trained or fine-tuned on the corresponding target count.

	$M=2$					$M=3$					$M=5$				
Method	RE@6	RE@12	RE@24	RE@48	Pen@24	RE@6	RE@12	RE@24	RE@48	Pen@24	RE@6	RE@12	RE@24	RE@48	Pen@24
Ours (M=2)	**0.0003**	**0.0056**	**0.0633**	0.1892	**0.0066**	**0.0003**	**0.0084**	**0.0800**	0.2135	**0.0071**	**0.0014**	**0.0164**	**0.0913**	**0.2236**	**0.0075**
SBC	0.0168	0.0432	0.1894	0.3621	0.0773	0.0131	0.0553	0.2678	0.5308	0.0774	0.0159	0.0552	0.2125	0.4292	0.0768
EGNO	0.1006	0.2116	0.1928	**0.1760**	0.0763	0.1147	0.2405	0.2222	**0.1880**	0.0790	0.1172	0.2091	0.1786	48.28	0.0792
MeshGraphNets	0.0423	0.0967	0.1997	0.4064	0.0769	0.0161	0.0593	0.2533	0.5375	0.0765	0.0390	0.0857	0.1831	0.4135	0.0784
GNS	0.0160	0.0487	0.2306	0.3873	0.0775	0.0151	0.0690	0.2863	0.6302	0.0773	0.0301	0.0895	0.2529	0.5515	0.0781
EqCollide	0.0736	0.1222	0.2578	0.4649	0.0774	0.1327	0.2058	0.3029	0.6243	0.0796	0.0567	0.1541	0.2901	0.5076	0.0788
SGNN	0.0447	0.0951	0.2671	0.5032	0.0795	0.0288	0.0990	0.3505	0.7483	0.0794	0.0467	0.1132	0.2985	0.6364	0.0783
TIE	0.0661	0.1530	0.2704	0.4238	0.0743	0.0836	0.1575	0.3428	0.6164	0.0721	0.0833	0.1742	0.3400	0.5744	0.0795
DPI-Net	0.1346	0.1569	0.2937	0.4054	0.0706	0.1580	0.2056	0.3772	0.6287	0.0767	0.0645	0.1541	0.2844	0.4649	0.0771

points. Each trajectory contains 48 states at 24 Hz and is generated using 32 simulator substeps per frame. Object stiffness is sampled from $k \in \{10, 20, 50, 100, 200, 500\}$, with both homogeneous and heterogeneous assignments across objects. Further details of the dataset generation and simulator parameters are provided in Appendix B.1.

Metrics. All methods are rolled out autoregressively from the same initial states under identical prescribed controls. For a system containing N dynamic particles, the rollout position error at horizon H is defined as

$$\mathrm{RE@}H = \frac{1}{N} \sum_{i=1}^{N} \left\| \hat{x}_{h,i} - x^{\star}_{h,i} \right\|_2, \qquad h = \min(H + 1, T), \tag{15}$$

where $x^{\star}_{h,i}$ is the ground-truth position of particle i at frame h, and $T = 48$ denotes the number of stored states, indexed from the initial frame $t = 1$ to the final frame $t = 48$. H denotes the prediction horizon and h the corresponding frame index. We report RE at $H \in \{6, 12, 24, 48\}$, averaged over all evaluation episodes, in normalised object-length units. To assess contact quality, Pen@24 measures the maximum inter-object particle penetration over the first 24 predicted frames ($t = 2, \ldots, 25$): $\max(0, 2r_p - \|\hat{x}_{t,i} - \hat{x}_{t,j}\|_2)$ for particles belonging to different objects, with $r_p = 0.04$. Lower values indicate better performance for both metrics. Strain error (S@H) is the mean absolute difference between predicted and ground-truth edge strains, averaged over the inferred structural graph and the $h - 1$ predicted frames $t = 2, \ldots, h$; lower is better.

Baselines. We compare against SBC (Dubey et al., 2026), EqCollide (Chen et al., 2026), EGNO (Xu et al., 2024), MeshGraphNets (Pfaff et al., 2021), GNS (Sanchez-Gonzalez et al., 2020), SGNN (Han et al., 2022), TIE (Shao et al., 2022), and DPI-Net (Li et al., 2019). All methods use the same simulation setup and are evaluated on the same episodes at each object count. For EqCollide, training on $M = 2$ followed by fine-tuning at $M \in \{3, 5\}$ yielded lower errors than training from scratch at the corresponding counts; we therefore report the better fine-tuned results.

4.2 Multi-object rollout prediction

Table 1 compares a single PNEO trained only at $M = 2$ with baselines trained at the corresponding target count, except for EqCollide as discussed in Section 4.1. PNEO achieves the lowest reported RE@6, RE@12, RE@24, and Pen@24 at every object count. We additionally report strain errors in Table 6 of Appendix B.2, where PNEO demonstrated the best mechanical fidelity across all reported horizons. The only exceptions in position error are at RE@48 for $M = 2$ and $M = 3$, where EGNO outperforms PNEO with a small margin (0.1760 and 0.1880, versus our 0.1892 and 0.2135) in the reported RE metrics. However, such dynamic gains at endpoints are accompanied by larger errors at earlier horizons and worse strain prediction, indicating that the lower RE@48 does not reflect uniformly better rollout dynamics. This distinction becomes clear at $M = 5$, where EGNO diverges at the final horizon, with RE@48 increasing to 48.28 and S@48 to 8.599. In contrast, PNEO remains stable, demonstrating substantially more robust long-horizon behaviour as scene cardinality increases. Table 7 in Appendix B.2 provides additional zero-shot object-count comparisons across baselines, where all methods are trained at $M = 2$ and evaluated at larger unseen object counts. Table 8 additionally evaluates collision-aligned errors at and after contact, highlighting that PNEO improved collision-dynamics prediction over other methods.

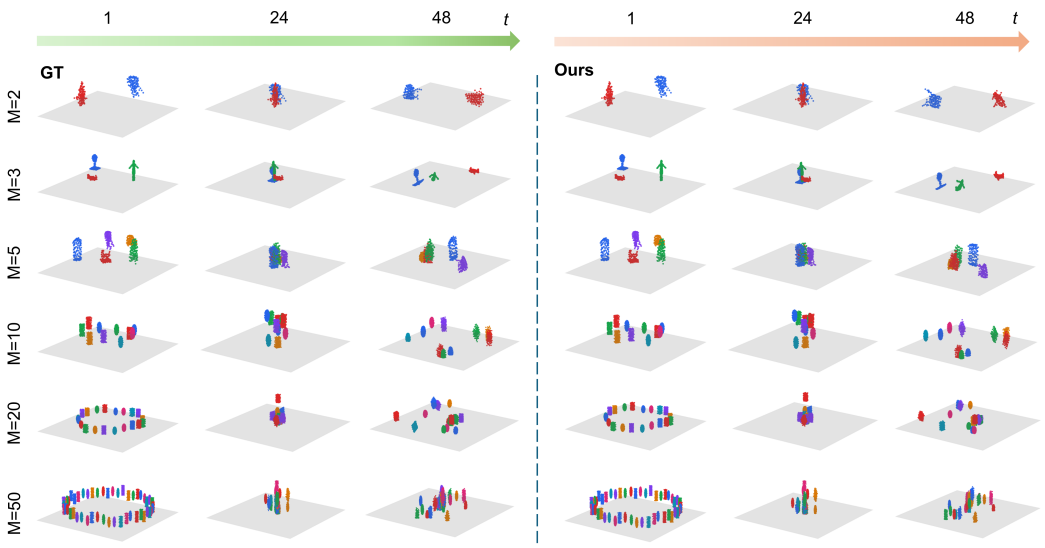

Figure 3: Qualitative comparison of ground-truth (GT) and PNEO rollouts at $t = 1, 24, 48$ for scenes with $M = 2, 3, 5, 10, 20, 50$ objects.

4.3 Scalability and Generalisation

Table 2: Zero-shot object-count scalability. A single PNEO trained only at $M = 2$ is evaluated without retraining up to $M = 50$.

	RE@H				Pen	S
M	6	12	24	48	@24	@24
2	0.0003	0.0056	0.0633	0.1892	0.0066	0.0199
3	0.0003	0.0084	0.0800	0.2135	0.0071	0.0198
5	0.0014	0.0164	0.0913	0.2236	0.0075	0.0234
10	0.0001	0.0042	0.0838	0.2159	0.0073	0.0178
20	0.0002	0.0029	0.0762	0.2187	0.0079	0.0169
50	0.0004	0.0043	0.0825	0.1268	0.0061	0.0091

Table 3: Stiffness generalisation at $M = 2$. RE@24 is reported for PNEO trained on $k \in \{10, 20, 50, 100, 200, 500\}$ and evaluated on seen and unseen stiffness values.

	Stiffness k					
Model	10	100	500	2000	20000	50000
Ours	**0.0746**	**0.0407**	**0.1235**	**0.1307**	**0.2133**	**0.1374**
MeshGraphNets	0.1316	0.1385	0.2059	0.2710	0.5144	0.4558
GNS	0.2034	0.2302	0.3302	0.4528	0.6872	0.6141
SGNN	0.2237	0.2310	0.2509	0.2501	0.4234	0.4490
TIE	0.2033	0.2169	0.3024	0.3545	0.5365	0.5464
EGNO	0.1550	0.1575	<u>0.1845</u>	<u>0.2021</u>	<u>0.3679</u>	<u>0.3107</u>
DPI-Net	0.3002	0.3481	0.3783	0.3778	0.5036	0.5709
SBC	0.1577	<u>0.1267</u>	0.2333	0.3495	0.6055	0.5356
EqCollide	<u>0.1304</u>	0.1901	0.3114	0.4274	0.6279	0.5542

Zero-shot scalability across object counts. Figure 3 qualitatively shows the zero-shot rollouts of the same PNEO trained only at $M = 2$ when applied to progressively larger multi-object scenes. Table 2 reports the corresponding quantitative results. Despite the substantial increase in scene cardinality, RE@24 remains within $[0.0633, 0.0913]$ and shows no monotonic growth with increasing M. The same trend holds for longer-horizon, contact, and deformation errors, with RE@48, Pen@24, and S@24 remaining stable across the tested object counts. These results demonstrate zero-shot rollout scalability well beyond the training scene size, without retraining or fine-tuning. Additional rollouts under stronger randomisation of the initial object positions, together with their RE values, are provided in Appendix B.3, further demonstrating robustness to scene initialisation.

Generalisation across mechanical properties. We evaluate PNEO trained on stiffnesses $k \in \{10, 20, 50, 100, 200, 500\}$ on $M = 2$ scenes with test stiffnesses ranging from 10 to 50,000 respectively. As shown in Table 3, PNEO achieves the lowest RE@24 at all six tested values. At the unseen stiffnesses $k \in \{2000, 20000, 50000\}$, PNEO reduces RE@24 significantly relative to other baselines. These results demonstrate generalisation of PNEO w.r.t different mechanical properties.

Table 4: Zero-shot point-cloud resolution generalisation. PNEO is trained at M=2 with 128 particles per object.

Particles per object	RE@6	RE@12	RE@24	RE@48	Pen@24	Relative RE@24
64	0.0003	0.0045	0.1306	0.3052	0.0046	2.06
128	0.0003	0.0056	0.0633	0.1892	0.0066	1.00
256	0.0003	0.0027	0.0670	0.1747	0.0064	1.06
512	0.0003	0.0036	0.1118	0.2504	0.0076	1.77
1024	0.0003	0.0047	0.1875	0.3292	0.0077	2.96

Zero-shot generalisation across point-cloud resolutions. We evaluate a PNEO trained with 128 particles per object on inputs with different point-cloud resolutions, ranging from 64 to 1024 particles per object, to assess robustness to input resolution (Table 4). Performance remains stable under moderate resolution shifts. The largest degradation occurs at the highest tested density of 1024 particles (resolution grows almost 10 times), where RE@24 rises to 0.1875 and Pen@24 to 0.0077. Overall, PNEO exhibits strong zero-shot generalisation across point-cloud resolutions, with reduced accuracy only under the largest resolution increment.

4.4 Ablation studies

Table 5: Comparison of training strategies for multi-object generalisation. Separate PNEO models are trained only at $M \in \{2, 3, 5\}$ respectively and evaluated across larger object counts in terms of RE@24.

	Evaluation object count				
Object counts	$M = 2$	$M = 3$	$M = 5$	$M = 10$	$M = 20$
$M = 2$ only	**0.0633**	**0.0800**	**0.0913**	**0.0838**	**0.0762**
$M = 3$ only	0.0668	0.0951	0.0953	0.0843	0.0784
$M = 5$ only	0.0680	0.0802	0.0949	0.0884	0.0872

Rationale for two-object training. Table 5 compares PNEO models trained exclusively on scenes with $M = 2$, $M = 3$, or $M = 5$ objects and evaluated across larger scene sizes. Training with more objects does not consistently improve multi-object prediction: the model trained only at $M = 2$ achieves the lowest RE@24 at every evaluation count. This result supports our choice of two-object training, showing that the minimal setting required to observe object–object contact is sufficient for learning an interaction law that can be composed in larger multi-object scenes, without requiring more complex multi-object training data. Additional ablations on neighbourhood selection and contact-candidate count are reported in Tables 9 and 10 (e.g., Appendix B.4), respectively.

5 Conclusion

We introduced the PNEO for scalable collision dynamics of 3D deformable objects. PNEO learns energy exchange and disspation on interaction dynamics from two-object collisions and composes the same pairwise energy law over the contact graph of larger scenes. Experiments show that a single model trained only at $M = 2$ generalises zero-shot to scenes with up to 50 objects without retraining, while outperforming competing neural simulators across the main multi-object settings. PNEO further generalises across mechanical properties and point-cloud resolutions, supporting the pairwise energy formulation as a scalable representation for multi-object deformable collision dynamics.

Limitations and future work. Our current study focuses on collision-dominated dynamics with penalty-based normal contact in a spring–mass benchmark. Future work will extend PNEO to richer contact behaviours, such as friction, sticking, rolling, and sustained static contact. In addition, the spring–mass model provides a simplified approximation of real-world deformable mechanics and may not capture more complex constitutive behaviour. Future work will therefore consider richer mechanical models and investigate the sim-real gap through validation on real deformable-object interactions.

AI USE STATEMENT

In this work, large language models (LLMs) were used to assist with the implementation of parts of the research code, the preparation of visualisations, the exploration and refinement of aspects of the methodology, and the editing of the manuscript for clarity and readability. All methodological decisions, implementations, visualisations, experimental results, analyses, and final manuscript content were reviewed and approved by the authors, who take full responsibility for the work, including material developed with LLM assistance.

REFERENCES

Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019. URL https://openreview.net/forum?id=fg2ZFmXFO3.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016. URL https://arxiv.org/pdf/1612.00222.

Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pp. 2402–2411. PMLR, 2020. URL https://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html.

Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 147–155, 2015. URL https://dl.acm.org/doi/abs/10.1145/2786784.2786796.

Michael Chang, Tomer Ullman, Antonio Torralba, and Joshua Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Bkab5dqxe.

Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. Physgen3d: Crafting a miniature interactive world from a single image. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6178–6189, 2025. doi: 10.1109/CVPR52734.2025.00579.

Qianyi Chen, Tianrun Gao, Chenbo Jiang, and Tailin Wu. Eqcollide: Equivariant and collision-aware deformable objects neural simulator. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 10623–10634, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3818862. URL https://doi.org/10.1145/3770855.3818862.

Yunuo Chen, Yafei Hu, Lingfeng Sun, Tushar Kusnur, Laura Herlant, and Chenfanfu Jiang. EMPM: Embodied MPM for Modeling and Simulation of Deformable Objects. *IEEE Robotics and Automation Letters*, 11(4):4179–4186, January 2026. doi: 10.1109/LRA.2026.3664610.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020. URL https://arxiv.org/abs/2003.04630.

Congyue Deng, Brandon Y Feng, Cecilia Garraffo, Alan Garbarz, Robin Walters, William T Freeman, Leonidas Guibas, and Kaiming He. Denoising hamiltonian network for physical reasoning. *arXiv preprint arXiv:2503.07596*, 2025a.

Huayu Deng, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Evomesh: adaptive physical simulation with hierarchical graph evolutions. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025b.

Shaan A Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physical Review E*, 104(3):034312, 2021.

Zhiyang Dou, Minghao Guo, Haixu Wu, Doug Roble, Tuur Stuyck, and Wojciech Matusik. Rigidformer: Learning rigid dynamics using transformers. *arXiv preprint arXiv:2605.09196*, 2026. URL https://arxiv.org/abs/2605.09196.

Vijay K Dubey, Collin E Haese, Osman Gültekin, David Dalton, Manuel K Rausch, and Jan Fuhg. Graph neural network surrogates for contacting deformable bodies with necessary and sufficient contact detection. *Computer methods in applied mechanics and engineering*, 448:118413, 2026.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13, 2000. URL https://dl.acm.org/doi/10.5555/3008751.3008817.

Marc Finzi, Ke Alexander Wang, and Andrew G Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *Advances in neural information processing systems*, 33:13880–13889, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9f655cc8884fda7ad6d8a6fb15cc001e-Paper.pdf.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. *arXiv e-prints*, art. arXiv:1906.01563, June 2019. doi: 10.48550/arXiv.1906.01563.

Jiaqi Han, Wenbing Huang, Hengbo Ma, Jiachen Li, Josh Tenenbaum, and Chuang Gan. Learning physical dynamics with subequivariant graph neural networks. *Advances in Neural Information Processing Systems*, 35:26256–26268, 2022. URL https://arxiv.org/pdf/2210.06876.

Bingwei Huang, Chenxi Zhang, Fengge Wu, and Junsuo Zhao. Subequivariant heterogeneous graph network for physical dynamics: Modeling and control. In *Advanced Intelligent Computing Technology and Applications: 21st International Conference, ICIC 2025, Ningbo, China, July 26–29, 2025, Proceedings, Part IX*, pp. 248–259, Berlin, Heidelberg, 2025. Springer-Verlag. ISBN 978-981-96-9883-7. doi: 10.1007/978-981-96-9884-4_21. URL https://doi.org/10.1007/978-981-96-9884-4_21.

Theodore Kim and David Eberle. Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH 2022 Courses*, pp. 1–259. Association for Computing Machinery, 2022. URL https://dl.acm.org/doi/abs/10.1145/3532720.3535628.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL https://www.jmlr.org/papers/v24/21-1524.html.

Hongyu Li, Lingfeng Sun, Yafei Hu, Duy Ta, Jennifer Barry, George Konidaris, and Jiahui Fu. Novaflow: Zero-shot manipulation via actionable flow from generated videos. *arXiv preprint arXiv:2510.08568*, 2025. URL https://arxiv.org/abs/2510.08568.

Shiqian Li, Ruihong Shen, Yaoyu Tao, Chi Zhang, and Yixin Zhu. Neural force field: Few-shot learning of generalized physical reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wNtdVoqEmr.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgbSn09Ym.

Lu Lu, Peng-Zhan Jin, Guo-Fei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218 – 229, 2019. URL https://api.semanticscholar.org/CorpusID:233822586.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Yulia Rubanova, Alvaro Sanchez-Gonzalez, Tobias Pfaff, and Peter Battaglia. Constraint-based graph network simulator. *arXiv preprint arXiv:2112.09161*, 2021. URL https://arxiv.org/abs/2112.09161.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. Pmlr, 2020. URL https://arxiv.org/pdf/2002.09405.

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

Yidi Shao, Chen Change Loy, and Bo Dai. Transformer with implicit edges for particle-based physics simulation. In *European conference on computer vision*, pp. 549–564. Springer, 2022. URL https://arxiv.org/abs/2207.10860.

Vinay Sharma and Olga Fink. A physics-informed graph neural network conserving linear and angular momentum for dynamical systems. *Nature Communications*, 17(1):1045, 2026.

Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pp. 1–50. Association for Computing Machinery, 2012. URL https://doi.org/10.1145/2343483.2343501.

Andrew Sosanya and Sam Greydanus. Dissipative hamiltonian neural networks: Learning dissipative and conservative dynamics separately. *arXiv preprint arXiv:2201.10085*, 2022. URL https://arxiv.org/abs/2201.10085.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4), July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461948. URL https://doi.org/10.1145/2461912.2461948.

Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AHEKhff4Oa.

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 *International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Qizhen Ying, Guangming Wang, Yangchen Pan, Victor Adrian Prisacariu, Brian Sheil, and Yixiong Jing. Nexus: Neural energy fields for physically consistent contact-rich 3d object dynamics. *arXiv preprint arXiv:2606.15015*, 2026. URL https://arxiv.org/abs/2606.15015.

Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, ChangSeung Woo, ILHO KIM, SeokWoo Lee, Joon Young Yang, SOOYOUNG YOON, and Noseong Park. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=90yw2uM6J5.

Kaifeng Zhang, Shuo Sha, Hanxiao Jiang, Matthew Loper, Hyunjong Song, Guangyan Cai, Zhuo Xu, Xiaochen Hu, Changxi Zheng, and Yunzhu Li. Real-to-sim robot policy evaluation with gaussian splatting simulation of soft-body interactions. *arXiv preprint arXiv:2511.04665*, 2025. URL https://arxiv.org/abs/2511.04665.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative symODEN: Encoding hamiltonian dynamics with dissipation and control into deep learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019a. URL https://openreview.net/forum?id=knjWFNx6CN.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019b. URL https://arxiv.org/abs/1909.12077.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Extending lagrangian and hamiltonian neural networks with differentiable contact models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=pZQrKCkbas.

A SUPPLEMENTARY METHODOLOGY AND IMPLEMENTATION DETAILS

A.1 CONTACT GEOMETRY

Object–object contact geometry. For an unordered object–object particle pair $p = (i, j)$, we define the penetration depth as

$$\delta_p = 2r_p\alpha - \|x_i - x_j\|, \tag{16}$$

where r_p is the particle radius and α controls the contact margin. Positive δ_p therefore indicates an active compliant contact.

The contact normal is defined along the relative displacement,

$$n_p = \frac{x_i - x_j}{\|x_i - x_j\|}, \tag{17}$$

and the corresponding normal relative velocity is

$$v_p^n = (v_i - v_j)^\top n_p. \tag{18}$$

Negative v_p^n corresponds to an approaching pair and activates the contact-dissipation term in equation 11.

A.2 OBJECT-LOCAL MECHANICAL ENCODER

The mechanical representation module introduced in Section 3.2 consists of a pointwise particle encoder $\mathcal{E}_{\theta_E}^{\text{mech}}$ inspired by GNO (Anandkumar et al., 2019). The module maps the mechanical field of each particle to a context-aware particle representation and an object-level mechanical descriptor.

Particle embedding. The mechanical field a_i defined in equation 6 is first projected into a d-dimensional latent space:

$$h_i = \text{SiLU}\left(W_{\text{emb}}a_i + b_{\text{emb}}\right) \in \mathbb{R}^d. \tag{19}$$

The embedding parameters are $\theta_{\text{emb}} = \{W_{\text{emb}}, b_{\text{emb}}\}$. We initialise the initial embedding as $q_i^{(0)} = h_i$.

Object-local GNO encoding. The encoder propagates mechanical information over $\mathcal{B}(x_i,\, r)$ defined in Section 3.2. For each neighbouring pair (i, j') belonging to the same object, the kernel input combines their relative geometry and original mechanical fields:

$$\eta_{ij'} = [x_{j'} - x_i,\ a_i,\ a_{j'}] \in \mathbb{R}^{27}. \tag{20}$$

At layer $\ell = 0, \ldots, L-1$, we parameterise the matrix-valued kernel as

$$\kappa^{(\ell)}(\eta_{ij'}) = B^{(\ell)}\,\text{diag}\left(\phi^{(\ell)}(\eta_{ij'})\right) A^{(\ell)}, \tag{21}$$

where $A^{(\ell)}, B^{(\ell)} \in \mathbb{R}^{d\times d}$ are learnable bias-free linear maps. The kernel network $\phi^{(\ell)}$ is an MLP with widths $27 \to d \to d \to d$, with SiLU activations after the first two affine layers.

For each particle, neighbouring latent features are transformed by the kernel and mean-aggregated within the object-local neighbourhood:

$$u_i^{(\ell)} = \frac{1}{|\mathcal{B}(x_i,\, r)|} \sum_{j' \in \mathcal{B}(x_i,\, r)} \kappa^{(\ell)}(\eta_{ij'}) q_{j'}^{(\ell)}. \tag{22}$$

The aggregate is then combined with a pointwise transformation through a residual update:

$$q_i^{(\ell+1)} = \text{LayerNorm}^{(\ell)}\left(q_i^{(\ell)} + \text{SiLU}\left(W^{(\ell)}q_i^{(\ell)} + b^{(\ell)} + u_i^{(\ell)}\right)\right). \tag{23}$$

After L layers, the context-aware particle representation is

$$\hat{h}_i = q_i^{(L)}. \tag{24}$$

Object-level representation. For each object A, we obtain an object-level mechanical descriptor by mean-pooling its encoded particle features:

$$z_A = \frac{1}{|\mathcal{V}_A|} \sum_{i \in \mathcal{V}_A} \hat{h}_i. \tag{25}$$

For a contact pair with $i \in \mathcal{V}_A$ and $j \in \mathcal{V}_B$, the PNEO head therefore receives the endpoint representations $\hat{h}_i, \hat{h}_j$ together with the corresponding object descriptors z_A, z_B.

Implementation details. We use $L = 2$ for $\mathcal{E}^{\mathrm{mech}}_{\theta_E}$ with latent dimension $d = 128$ and $r = 0.25$. The object-local neighbourhood graph and the original mechanical fields are constructed once and reused across all encoder layers within each evaluation. Although each layer aggregates only within r, stacking multiple layers enlarges the effective receptive field through successive local interactions. Both neighbourhood aggregation and object-level pooling are restricted to particles belonging to the same object. Object identity is used only to define these aggregation domains and is not provided as an input feature. The encoder is therefore object-local, while all learnable parameters are shared across objects and scenes.

A.3 Forward integration and training

Algorithm 1 performs forward dynamics at frame interval Δt using contact detection. All forward dynamics are integrated using the *semi-explicit Euler method*. Each training window uses local frame indices $t = 1, \ldots, H + 1$, where $H \leq T - 1$ is the number of prediction steps and the initial state is at $t = 1$.

Contact detection and integration branch selection. At the beginning of each frame, $\mathrm{Detect}(x_t)$ constructs the contact map $\mathcal{A}_{\mathrm{con},t}$ over M objects and the static environment, and returns $\chi = \mathbf{1}[\mathcal{A}_{\mathrm{con},t} \neq \varnothing]$. We identify candidate pairs using k-NN searches between distinct objects and between objects and the environment, then apply the geometric contact criterion. For object–object pairs, an active contact satisfies $\delta_p > 0$ with zero additional detection margin; object–environment contact uses the normal gap to the environment surface. A nearest neighbour is therefore a candidate, not necessarily an active contact. The same detection procedure refreshes the contact map at each contact substep. The frame-start detection result directly selects the integration branch. If contact is detected ($\chi = 1$), we enter the contact branch and use PNEO to compute the pairwise contact potential and Rayleigh dissipation for the active object–object pairs, as detailed below. If no contact is detected ($\chi = 0$), we advance (x_t, v_t) directly to (x_{t+1}, v_{t+1}) by integrating equation 2 with $U_{\mathrm{con},\theta} = R_{\mathrm{con},\theta} = 0$ over one frame using S_{free} Euler steps of size $\Delta t / S_{\mathrm{free}}$, denoted by $\mathrm{Euler}_{S_{\mathrm{free}}}$ in the algorithm.

Contact substeps. Contact dynamics are integrated using the same semi-explicit Euler scheme with S substeps of size $\Delta\tau = \Delta t / S$. We initialise the substep state and contact map as $(x^0, v^0, \mathcal{A}^0_{\mathrm{con}}) = (x_t, v_t, \mathcal{A}_{\mathrm{con},t})$. At the beginning of each frame in the contact branch, a_i is first projected by a shared MLP and then processed by $\mathcal{E}^{\mathrm{mech}}_{\theta_E}$. The resulting $\hat{h}_i$ and z_A ($A \in \mathcal{O}$) are computed once and cached for all S substeps. At substep q, the current contact map $\mathcal{A}^q_{\mathrm{con}}$ determines the active contact pairs. For each active object–object pair, we construct ξ^q_p according to equation 8. We use $\mathrm{PNEO}_\theta(\xi^q_p)$ to denote the mapping in Eqs. equation 9–equation 11 from ξ^q_p to the corresponding pairwise potential U_p and Rayleigh dissipation R_p. This mapping is re-evaluated at every substep, while $\hat{h}_i$ and z_A are not recomputed. The active pairwise terms are then summed with the remaining contributions in equation 3, differentiated at the common state (x^q, v^q), and used in the Euler update. The contact map is rebuilt from the updated geometry after each substep, and the final state (x^S, v^S) defines (x_{t+1}, v_{t+1}). When differentiating the potential and dissipation functions with respect to the physical state, the predicted coefficients k_p and c_p are frozen within that force evaluation. Their dependence on θ is retained for training, and the coefficients are re-evaluated from the updated contact state at the next substep.

Training objective. Training is performed on H-step two-object rollout windows with ground-truth states $(x^\star_t, v^\star_t)$ and perturbations applied to the initial state at $t = 1$. The accumulated rollout loss is used to update the model parameters with AdamW. Let $\ell_t = \ell(x_{t+1}, v_{t+1}; x^\star_{t+1}, v^\star_{t+1})$ denote

Algorithm 1 PNEO rollout and training

Input: Initial state (x_1, v_1); reference positions $\{x_i^0\}$; structural graph $\mathcal{G}_0$; environment positions/normals $(x_{\text{env}}, n_{\text{env}})$; mass matrix $\mathbf{M}$; stiffness k; gravity g; external forces $\{f_{i,t}^{\text{ext}}\}$; model parameters θ; frame interval Δt; prediction steps H; contact substeps S; contact-free Euler steps per frame S_{free}; target positions/velocities $\{(x_t^\star, v_t^\star)\}_{t=2}^{H+1}$

Output: Predicted states $\{(x_t, v_t)\}_{t=2}^{H+1}$; training loss $\mathcal{L}_\theta$; updated training parameters θ

$\mathcal{L}_\theta \leftarrow 0, \quad \Delta\tau \leftarrow \Delta t / S$
for $t = 1, \dots, H$ **do**
  $(\mathcal{A}_{\text{con},t}, \chi) \leftarrow \text{Detect}(x_t)$ ▷ k-NN candidates and geometric contact test
  if $\chi = 1$ **then** ▷ contact detected: evaluate PNEO and integrate
    $(x^0, v^0, \mathcal{A}_{\text{con}}^0) \leftarrow (x_t, v_t, \mathcal{A}_{\text{con},t})$ ▷ frame-start state/map at substep $q = 0$
    for all $A \in \mathcal{O}$ **do**
      $h_i \leftarrow \text{MLP}(a_i)$ for $i \in \mathcal{V}_A$ ▷ pointwise lifting; Eq. 19
      $\big(\{\hat{h}_i\}_{i\in\mathcal{V}_A}, z_A\big) \leftarrow \mathcal{E}_{\theta_E}^{\text{mech}}\big(\{(x_i, a_i, h_i)\}_{i\in\mathcal{V}_A}; \{\mathcal{B}(x_i, r)\}_{i\in\mathcal{V}_A}\big)$ ▷ Eq. 7
    end for
    for $q = 0, \dots, S-1$ **do**
      for all object–object pairs $p \in \mathcal{A}_{\text{con}}^q$ **do**
        Construct ξ_p^q from cached representations and (x^q, v^q) ▷ Eq. 8
        $(U_p, R_p) \leftarrow \text{PNEO}_\theta(\xi_p^q)$ ▷ refresh pairwise terms at every substep
      end for
      Sum pair scalars with structural, environment, and control terms into U_θ, R_θ ▷ Eq. 3
      $F_\theta^q \leftarrow \big[-\partial_x U_\theta - \partial_v R_\theta\big]_{(x^q, v^q)}$ ▷ total force; Eq. 2
      $v^{q+1} \leftarrow v^q + \Delta\tau\, \mathbf{M}^{-1} F_\theta^q, \quad x^{q+1} \leftarrow x^q + \Delta\tau\, v^{q+1}$
      $(\mathcal{A}_{\text{con}}^{q+1}, \chi) \leftarrow \text{Detect}(x^{q+1})$ ▷ repeat contact detection for the next substep
    end for
    $(x_{t+1}, v_{t+1}) \leftarrow (x^S, v^S)$ ▷ final state after S contact substeps
  else
    $(x_{t+1}, v_{t+1}) \leftarrow \text{Euler}_{S_{\text{free}}}\big((x_t, v_t); \Delta t\big)$ ▷ free motion with $U_{\text{con},\theta} = R_{\text{con},\theta} = 0$; Eq. 2
  end if
  $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \dfrac{\omega_t}{Z}\ell\big(x_{t+1}, v_{t+1}; x_{t+1}^\star, v_{t+1}^\star\big)$ ▷ Eq. 26
end for
$\theta \leftarrow \text{AdamW}(\theta, \nabla_\theta \mathcal{L}_\theta)$

the loss between the predicted and target states at frame $t+1$. The training objective combines position, velocity, structural strain, and penetration losses,

$$\begin{aligned} \mathcal{L}_\theta &= \frac{1}{Z}\sum_{t=1}^{H} \omega_t \ell_t, \qquad \omega_t = \gamma^{t-1}\big(1 + \lambda_{\text{con}}\chi_{t+1}^\star\big), \qquad Z = \sum_{t=1}^{H}\omega_t, \\ \ell_t &= \lambda_x \mathcal{L}_{x,t} + \lambda_v \mathcal{L}_{v,t} + \lambda_s \mathcal{L}_{\text{str},t} + \lambda_{\text{pen}} \mathcal{L}_{\text{pen},t}. \end{aligned} \tag{26}$$

where γ controls horizon decay, i.e., choosing $0 < \gamma < 1$ downweights later frames to reduce the influence of accumulated rollout errors. The λ terms denote loss weights, and $\chi_{t+1}^\star$ equals one if inter-object contact is detected in the target state $x_{t+1}^\star$ and zero otherwise, increasing the weight of contact frames. This ground-truth indicator excludes environment contacts; the algorithm's contact indicator χ uses the simulated state and includes object–environment contacts. Let $\mathcal{H}_\beta$ denote the mean smooth-L_1 loss with transition parameter β. The state losses are

$$\mathcal{L}_{x,t} = \mathcal{H}_1\left(\frac{x_{t+1} - x_{t+1}^\star}{s_x}\right), \qquad \mathcal{L}_{v,t} = \mathcal{H}_1\left(\frac{v_{t+1} - v_{t+1}^\star}{s_v}\right), \tag{27}$$

where s_x and s_v are data-derived normalisation scales. The structural loss applies $\mathcal{H}_{0.25}$ to the predicted–target differences in edge strain

$$\epsilon_{ij}(x) = \frac{\|x_i - x_j\|}{\ell_{ij}^0} - 1, \qquad (i,j) \in \mathcal{E}_0.$$

The penetration loss penalises inter-object penetration exceeding the corresponding target value. For this loss, $\delta_p(x)$ is evaluated using equation 16 with $\alpha = 1$, giving a contact margin of $2r_p$. Let $\mathcal{S}_A \subseteq \mathcal{V}_A$ be the particles sampled from object A for loss evaluation. For each pair of distinct objects $A, B \in \mathcal{O}$ with $A < B$ under a fixed ordering of object labels (so each unordered object pair

is counted once), let $\mathcal{C}_{t+1}^{AB}$ denote the set of sampled particle pairs $p = (i, j)$ (definition is similar to equation 13 but on two objects), with $i \in \mathcal{S}_A$ and $j \in \mathcal{S}_B$, that are in active contact in either the predicted state x_{t+1} or the target state $x_{t+1}^{\star}$ at frame $t + 1$:

$$\mathcal{C}_{t+1}^{AB} = \left\{ p = (i, j) \in \mathcal{S}_A \times \mathcal{S}_B : \delta_p(x_{t+1}) > 0 \text{ or } \delta_p(x_{t+1}^{\star}) > 0 \right\}, \qquad A < B.$$

The normalised mean excess penetration is then

$$\mathcal{L}_{\text{pen},t} = \sum_{A<B} \frac{\sum_{p \in \mathcal{C}_{t+1}^{AB}} \varphi\big(\varphi(\delta_p(x_{t+1})) - \varphi\big(\delta_p(x_{t+1}^{\star})\big)\big)}{2r_p \, \max\big(1, |\mathcal{C}_{t+1}^{AB}|\big)}. \tag{28}$$

B Supplementary experimental results

B.1 Dataset and Simulation Benchmark

We construct a multi-object point-cloud benchmark for deformable-body dynamics using a spring–mass simulator. Object geometries are retrieved from the PhysCtrl (Wang et al., 2025) asset library, downsampled to 128 particles using farthest-point sampling, and scaled to unit diameter. Each rollout contains M dynamic objects interacting with a ground plane represented by 1,024 points. We record per-frame particle positions and velocities, object identities, ground geometry, object stiffness values, and control signals. The benchmark contains 12,032 / 7,200 / 7,200 rollouts for $M = 2/3/5$, respectively, with stiffness values spanning $k \in \{10, 20, 50, 100, 200, 500\}$. Each trajectory consists of 48 frames at 24 FPS.

Simulator. Each deformable object is represented by a particle system connected through a hidden spring graph comprising symmetric 18-neighbour edges, a minimum spanning tree, and eight randomly sampled long-range springs per particle. We use rest-length-normalised strain energy so that the prescribed stiffness k remains independent of particle sampling density. Object–object and object–ground contacts are modelled using a penalty-based contact law. For object–object contact, forces are activated when the distance between a particle pair falls below $2r_p$, with $r_p = 0.04$, and normal contact damping uses a damping ratio of 0.2. Gravity is set to $(0, -4, 0)$. All trajectories are integrated using semi-implicit Euler with 32 substeps per frame.

Scene generation and control. Objects are initially placed along a circular ring above the ground, with random yaw rotations and positional perturbations of up to 30% of the nominal inter-object spacing for the spacial randomness. They are assigned inward velocities with magnitudes sampled from $v \sim \mathcal{U}[1.0, 1.8]$, producing trajectories that include object–object collisions, ground contact, deformation, and eventual settling. Object stiffnesses are assigned either homogeneously and heterogeneously, where the later involves varying stiffness values across different springs of the same object. In 50% of episodes, one object additionally receives a local pushing force applied to approximately 10% of its particles for 3–8 frames, with force magnitude sampled from $\mathcal{U}[2, 8]$. The resulting per-particle force is provided to all models as an input control. To evaluate generalisation to unseen object counts, we additionally generate independent test scenes with geometries not used for training, including scenes with $M = 10$, 20, and 50 objects.

Dataset split and evaluation. For training, we use a deterministic 90/10 episode-level split and a rollout curriculum with a maximum training horizon of 24 steps. We index the 48 stored states by $t = 1, \ldots, 48$. At evaluation time, all methods are rolled out autoregressively from the initial state at $t = 1$ under identical initial conditions and controls. We report performance at requested prediction horizons $H \in \{6, 12, 24, 48\}$, using the frame index $h = \min(H + 1, 48)$ defined in equation 15; the final stored frame is reached after 47 prediction steps.

B.2 Supplementary results for multi-object rollout prediction

This section provides supplementary results that complement Section 4.2 with strain accuracy measurements, zero-shot transfer, collision-aligned rollout errors, and scene motion statistics.

Strain accuracy. Table 6 evaluates deformation accuracy on the inferred structural graph. PNEO achieves the lowest strain error at every reported horizon and object count. At S@24, even the strongest baseline exhibits higher error than PNEO, complementing the trajectory-level comparison in Table 1. Notably, although EGNO achieves slightly lower RE@48 at $M = 2$ and $M = 3$, its substantially larger strain errors indicate lower mechanical fidelity in the predicted deformation. At $M = 5$, this discrepancy becomes more pronounced, with EGNO's S@48 increasing sharply to 8.599, consistent with the long-horizon instability observed in position space.

Table 6: Strain error for the comparison in Table 1. S@H averages the absolute edge-strain error over predicted frames $t = 2, \ldots, h$, where $h = \min(H + 1, T)$ and $T = 48$, as in equation 15.

	$M = 2$				$M = 3$				$M = 5$			
Method	S@6	S@12	S@24	S@48	S@6	S@12	S@24	S@48	S@6	S@12	S@24	S@48
Ours (M=2)	**0.0000**	**0.0017**	**0.0199**	**0.0289**	**0.0000**	**0.0025**	**0.0198**	**0.0265**	**0.0006**	**0.0053**	**0.0234**	**0.0291**
SBC	0.0127	0.0237	0.0733	0.1393	0.0144	0.0292	0.0910	0.1494	0.0157	0.0308	0.0779	0.1475
EGNO	0.0784	0.1330	0.1998	0.2164	0.0789	0.1527	0.2196	0.2417	0.0998	0.1665	0.2209	8.599
MeshGraphNets	0.0199	0.0253	0.0593	0.0839	0.0106	0.0207	0.0495	0.0506	0.0177	0.0285	0.0521	0.0610
GNS	0.0126	0.0239	0.0814	0.1524	0.0129	0.0284	0.0814	0.1270	0.0231	0.0481	0.1006	0.1689
EqCollide	0.0073	0.0160	0.0606	0.0998	0.0073	0.0188	0.0595	0.0941	0.0091	0.0243	0.0675	0.1015
SGNN	0.0281	0.0603	0.1263	0.1943	0.0197	0.0424	0.0877	0.1358	0.0252	0.0499	0.1028	0.1627
TIE	0.0305	0.0698	0.1784	0.2587	0.0243	0.0601	0.1607	0.2214	0.0258	0.0752	0.1874	0.2448
DPI-Net	0.0504	0.1070	0.2019	0.2751	0.0786	0.1512	0.2381	0.2901	0.0284	0.0673	0.1330	0.1864

Zero-shot transfer to unseen object counts. Table 7 compares methods trained only at $M = 2$ and evaluated on the same rollouts at larger M counts, without retraining or fine-tuning. EqCollide and DPI-Net are omitted because their implementations do not support evaluation at unseen 3D object counts using weights trained only at $M = 2$. The final column reports the ratio between RE@24 at $M = 20$ and $M = 2$, where values closer to one indicate stronger object-count transfer. PNEO achieves the lowest RE@24 at every evaluated count and a ratio of 1.20 at $M = 20$, compared with 2.88–8.06 for the transferable baselines.

Table 7: Zero-shot comparison across object counts. All methods are trained only at $M = 2$ and evaluated without retraining; entries report RE@24.

	RE@24 ↓ by evaluation object count					
Method	2	3	5	10	20	M=20/M=2 (→ 1)
Ours (PNEO)	**0.0633**	**0.0800**	**0.0913**	**0.0838**	**0.0762**	**1.20**
SBC	0.1894	0.2749	0.2524	0.2919	0.8003	4.23
EGNO	0.1928	0.2475	div.[a]	0.5644	0.8148	4.23
MeshGraphNets	0.1997	0.2626	0.2491	0.4349	1.6094	8.06
GNS	0.2306	0.3359	0.3425	0.3257	0.7830	3.40
SGNN	0.2671	0.3922	0.3829	0.4570	0.7682	2.88
TIE	0.2704	0.3726	0.3913	0.3388	0.8617	3.19

Rollout accuracy at and after first contact. To better assess collision-response accuracy separately from error accumulated during preceding free motion, we additionally evaluate each model around its first predicted object–object contact. Table 8 reports RE at the predicted contact frame t_c and at $t_c + 10$ and $t_c + 20$, comparing predicted and ground-truth positions at the same absolute frames. PNEO achieves the lowest error at all three collision-aligned horizons for every evaluated object count. In particular, the error remains low at contact onset and grows substantially more slowly after contact than for the competing methods, including under zero-shot evaluation at $M = 3$ and $M = 5$. These results further support the accuracy of the learned pairwise interaction law during and after collision.

Table 8: Rollout error aligned to the first object–object contact frame t_c of each episode: RE at t_c, t_c+10 and t_c+20. Our model is trained only at $M = 2$ and evaluated zero-shot at $M \in \{3, 5\}$; baselines are trained at the evaluated object count.

	$M = 2$			$M = 3$			$M = 5$		
Method	t_c	t_c+10	t_c+20	t_c	t_c+10	t_c+20	t_c	t_c+10	t_c+20
Ours	**0.0012**	**0.0424**	**0.0884**	**0.0003**	**0.0444**	**0.1049**	**0.0001**	**0.0176**	**0.0794**
SBC	0.0281	0.1384	0.2352	0.0181	0.1717	0.3118	0.0058	0.0572	0.1914
EGNO	0.1933	0.1651	0.1707	0.1853	0.2111	0.2049	0.0419	0.2060	0.1581
MeshGraphNets	0.0803	0.1585	0.2540	0.0227	0.1704	0.3054	0.0102	0.0885	0.1683
GNS	0.0248	0.1785	0.2765	0.0284	0.1807	0.3514	0.0086	0.0905	0.2299
EqCollide	0.1044	0.2169	0.3016	0.1645	0.2268	0.3522	0.0427	0.1515	0.2867
SGNN	0.0880	0.2086	0.3159	0.0459	0.2285	0.4260	0.0100	0.1162	0.2689
TIE	0.1166	0.2374	0.3184	0.1134	0.2632	0.3750	0.0183	0.1723	0.3204
DPI-Net	0.1611	0.2550	0.3198	0.1909	0.2917	0.4187	0.0151	0.1539	0.2686

B.3 Rollout prediction under randomised initial positions

Figure 4 shows additional rollouts for $M \in \{2, 3, 5, 10, 20\}$ under stronger randomisation of the initial object positions compared to those in Figure 3. The same PNEO trained only at $M = 2$ is evaluated without retraining. The annotated RE values quantify rollout accuracy under these more varied initial configurations.

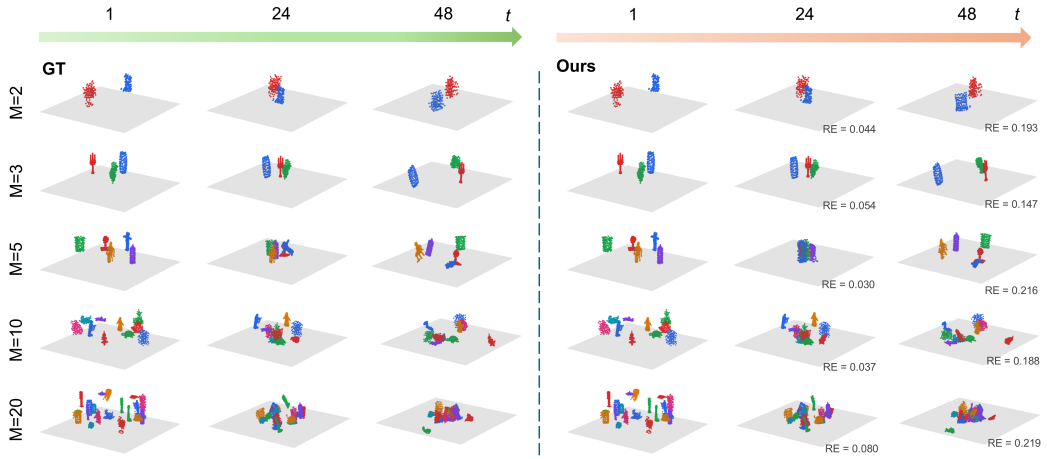

Figure 4: Supplementary rollouts at $t = 1, 24, 48$ for $M = 2, 3, 5, 10, 20$, with RE values annotated for the predicted states at $t = 24$ and $t = 48$.

B.4 SUPPLEMENTARY RESULTS FOR ABLATION STUDIES

Comparison of neighbourhood selection methods. We compare two neighbourhood selection methods for $\mathcal{E}^{\text{mech}}_{\theta_E}$: a fixed-radius ball query with $r = 0.25$ and k-nearest neighbours (k-*NN*) with $k = 12$. In our training dataset, which contains 128 particles per object, the ball-query neighbourhoods contain 12.4 neighbours on average. We therefore set $k = 12$ to approximately match the average neighbourhood size for the k-NN method. Both methods restrict aggregation to particles within the same object. As shown in Table 9, the ball-query neighbourhood achieves lower RE@24 at every evaluated object count. The largest improvement occurs at $M = 3$ (0.0800 versus 0.1937), while the gap narrows at $M = 50$ (0.0825 versus 0.0844). These results support the use of fixed-radius neighbourhoods for object-count generalisation.

Table 9: Comparison of fixed-radius and k-*NN* neighbourhoods for $\mathcal{E}^{\text{mech}}_{\theta_E}$. RE@24 is reported across object counts using 128 particles per object.

	Number of objects M					
Encoder neighbourhood	2	3	5	10	20	50
k-*NN*, k=12	0.1045	0.1937	0.1661	0.1079	0.1140	0.0844
Ball query, r=0.25	**0.0633**	**0.0800**	**0.0913**	**0.0838**	**0.0762**	**0.0825**

Comparison on the number of contact candidates. We further evaluate the number of cross-object contact candidates considered for each particle during contact detection, comparing the nearest particle with the four nearest particles of the other object. As shown in Table 10, the two settings yield very similar RE@24 across most evaluated object counts. This is because candidate neighbours are included in the dynamics only when they satisfy the penetration criterion. Therefore, even when four candidates are queried, typically only one or two become active contact edges. The results therefore indicate that PNEO is relatively insensitive to the number of contact candidates used during detection within the tested range.

Table 10: **Contact-candidate count ablation.** Zero-shot RE@24 across object counts using 1 or 4 cross-object nearest-neighbour candidates per particle.

	Number of objects M					
Contact-candidate count	2	3	5	10	20	50
4 (Ours)	0.0633	0.0800	0.0913	0.0838	0.0762	0.0825
1	0.0614	0.0800	0.0912	0.0837	0.0765	0.0562