Improving Long-Range Interactions in Graph Neural Simulators via Hamiltonian Dynamics

Tai Hoang[†, 1] **Alessandro Trenta**[*, 2] **Alessio Gravina**[*, 2]
Niklas Freymuth[1] **Philipp Becker**[1] **Davide Bacciu**[2] **Gerhard Neumann**[1]
[1]Karlsruhe Institute of Technology [2]University of Pisa

Abstract

Learning to simulate complex physical systems from data has emerged as a promising way to overcome the limitations of traditional numerical solvers, which often require prohibitive computational costs for high-fidelity solutions. Recent Graph Neural Simulators (GNSs) accelerate simulations by learning dynamics on graph-structured data, yet often struggle to capture long-range interactions and suffer from error accumulation under autoregressive rollouts. To address these challenges, we propose Information-preserving Graph Neural Simulators (IGNS), a graph-based neural simulator built on the principles of Hamiltonian dynamics. This structure guarantees preservation of information across the graph, while extending to port-Hamiltonian systems allows the model to capture a broader class of dynamics, including non-conservative effects. IGNS further incorporates a warmup phase to initialize global context, geometric encoding to handle irregular meshes, and a multi-step training objective to reduce rollout error. To evaluate these properties systematically, we introduce new benchmarks that target long-range dependencies and challenging external forcing scenarios. Across all tasks, IGNS consistently outperforms state-of-the-art GNSs, achieving higher accuracy and stability under challenging and complex dynamical systems.

1 Introduction

Simulating physical systems is vital across scientific fields. These systems, governed by partial differential equations (PDEs), often require computationally expensive numerical solvers for accurate solutions [1–3]. Neural simulators, particularly Graph Neural Simulators (GNSs), offer a faster alternative by learning dynamics from data [4, 5]. By leveraging Graph Neural Networks (GNNs) [6] to capture local interactions, GNSs achieve efficiency and generalization across diverse applications [7]. Despite their promise, GNSs face challenges with long-range dependencies and error accumulation during rollouts. Existing methods, such as denoising objectives [4, 8], mitigate local noise but fail to address long-term drift. Multi-step objectives [9] improve stability but are limited by message-passing inefficiencies [10]. To overcome these limitations, we introduce Information-preserving Graph Neural Simulators (IGNS), a neural simulator that preserves information flow using port-Hamiltonian dynamics [11]. This design enhances long-range dependency handling and reduces rollout errors. IGNS incorporates a warmup phase for global context initialization and geometric encoding for irregular meshes, ensuring stability and accuracy. Experimentally, IGNS outperforms state-of-the-art GNSs on benchmarks targeting long-range interactions and complex dynamics.

Contributions: we **(i)** introduce Information-preserving Graph Neural Simulators (IGNS), a graph neural simulator based on port-Hamiltonian formalism that supports information-preserving rollouts with high accuracy, **(ii)** analyze its theoretical properties, showing why IGNS maintains information

[†]Correspondence to `tai.hoang@kit.edu`. [*]Equal contribution.

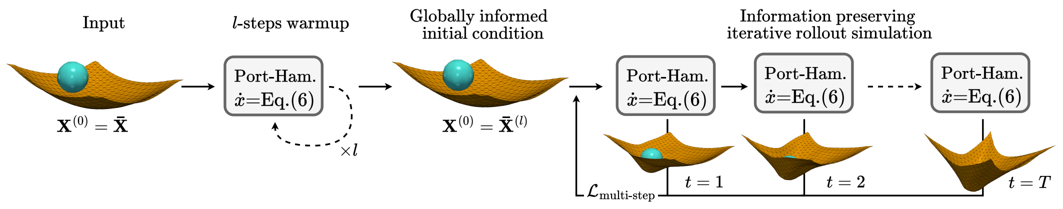

Input
$\mathbf{X}^{(0)} = \bar{\mathbf{X}}$
l-steps warmup
Port-Ham.
$\dot{x}$=Eq.(6)
$\times l$
Globally informed initial condition
$\mathbf{X}^{(0)} = \bar{\mathbf{X}}^{(l)}$
Information preserving iterative rollout simulation
Port-Ham.
$\dot{x}$=Eq.(6)
Port-Ham.
$\dot{x}$=Eq.(6)
Port-Ham.
$\dot{x}$=Eq.(6)
$t = 1$
$t = 2$
$t = T$
$\mathcal{L}_{\text{multi-step}}$

Figure 1: A high-level overview of the proposed IGNS. The model takes as input the initial node state $\bar{\boldsymbol{X}}$ and performs an l-step warmup phase (left), enabling each node to incorporate broader spatial context before the rollout begins. The enriched state is then used to initialize the rollout $\boldsymbol{X}^{(0)} = \bar{\boldsymbol{X}}^{(l)}$ (middle). During the simulation phase (right), the system evolves according to the port-Hamiltonian dynamics of Eq. (3), while the multi-step loss $\mathcal{L}_{\text{multi-step}}$ supervises all intermediate predictions, ensuring stable and accurate simulations.

flow across the spatial domain and can capture a broad class of dynamics, **(iii)** train IGNS with multi-step objective, which yields stable rollouts and reduces error accumulation compared to standard GNSs, and **(iv)** propose new tasks including Plate Deformation, Sphere Cloth, and Wave Balls, specifically designed to test long-range propagation and oscillatory dynamics under external forcing.

2 Background

Many physical systems are governed by PDEs that describe how quantities evolve over space and time. A general time-dependent PDE is

$$\partial_t u = F(u, \nabla u, \nabla^2 u, \dots; \boldsymbol{x}, t), \quad (\boldsymbol{x}, t) \in \Omega \times (0, T], \tag{1}$$

where u is the unknown field, Ω is the spatial domain, and F is a (possibly nonlinear) function of u and its spatial derivatives. Traditional numerical methods, such as finite differences and finite elements, discretize space and yield an ODE system for time integration [12]: $\dot{\boldsymbol{u}}(t) = \boldsymbol{f}(t, \boldsymbol{u}(t))$. These solvers can be computationally expensive for complex systems or fine meshes.

Graph Neural Simulators (GNSs) provide a data-driven alternative by representing the discretized domain as a graph $G = (V, E)$. The state update at time t is

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \Delta t \cdot \boldsymbol{g}_\theta^L\left(\boldsymbol{X}^{(t)}; \boldsymbol{E}; G\right), \tag{2}$$

where $\boldsymbol{g}_\theta^L$ denotes L stacked message-passing layers [8]. GNSs are typically trained with a one-step loss and, at inference, are rolled out autoregressively. While effective, standard GNSs often accumulate errors over long horizons, which limits accuracy.

3 Information-preserving Graph Neural Simulators (IGNS)

Problem statement. Given a time series of node states $\{\boldsymbol{X}^{(t)}\}_{t=0}^T$ on an irregular mesh or graph $G = (V, E)$, our goal is to learn and simulate the evolution of each node $\boldsymbol{x}_i^{(t)}$ over time. To address this, we propose the Information-preserving Graph Neural Simulators (IGNS), a GNN-based simulator that uses port-Hamiltonian dynamics as its latent evolution core. Fig. 1 shows the high-level overview of our method. IGNS is built on four key components: port-Hamiltonian formalism, warmup phase, geometric encoding, and multi-step objective.

Port-Hamiltonian formalism. We start by expressing the underlying dynamics using the port-Hamiltonian formalism [13–15]. Specifically, we parameterize the Hamiltonian $H_\theta(t, \boldsymbol{X})$ and evolve the joint state according to

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}\nabla_{\boldsymbol{x}_i} H_\theta(t, \boldsymbol{X}) - \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{D}_\theta \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{X}) \end{bmatrix}}_{\text{damping}} + \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{r}_\theta(t, \boldsymbol{X}) \end{bmatrix}}_{\text{forcing \& residual}}, \quad \boldsymbol{J} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix}. \tag{3}$$

Here, we omit the time index t for clarity, and denote $\boldsymbol{x}_i = [\boldsymbol{q}; \boldsymbol{p}]_i^T$ with $\boldsymbol{q} \in \mathbb{R}^{n \times \frac{d}{2}}$ and $\boldsymbol{p} \in \mathbb{R}^{n \times \frac{d}{2}}$ to represent the generalized coordinates and momenta, respectively. Without forcing and damping, Eq. (3) reduces to the classical Hamiltonian dynamics, which conserves the energy H_θ over time. However, most real-world physical systems are non-conservative, where energy can be dissipated (e.g., due to friction) or injected (e.g., due to external forces). To account for this, two additional terms are introduced: $\boldsymbol{D}_\theta \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{X})$ and $\boldsymbol{r}_\theta(t, \boldsymbol{X})$, for damping and external forcing, respectively. Next, we parameterize the Hamiltonian H_θ as

$$H_\theta(t, \boldsymbol{X}) = \sum_{i \in \mathcal{V}} \tilde{\sigma}\left(\boldsymbol{W}(t)\boldsymbol{x}_i + \sum_{j \in \mathcal{N}_i} \boldsymbol{V}(t)\boldsymbol{x}_j \right)^T \cdot \mathbf{1}_d, \tag{4}$$

where $\tilde{\sigma}$ is an anti-derivative of a non-linear activation function σ, $\mathbf{1}_d$ is a row vector of ones of dimension d, and $\boldsymbol{W}(t)$ and $\boldsymbol{V}(t)$ are block diagonal matrices (to ensure the separation into the $\boldsymbol{q}$ and $\boldsymbol{p}$ components), i.e., $\boldsymbol{W}(t) = \mathrm{diag}([\boldsymbol{W}_{\boldsymbol{q}}(t), \boldsymbol{W}_{\boldsymbol{p}}(t)])$ and $\boldsymbol{V}(t) = \mathrm{diag}([\boldsymbol{V}_{\boldsymbol{q}}(t), \boldsymbol{V}_{\boldsymbol{p}}(t)])$. The matrices $\boldsymbol{W}(t)$ and $\boldsymbol{V}(t)$ can be either shared across time (i.e., $\boldsymbol{W}(t) = \boldsymbol{W}$ and $\boldsymbol{V}(t) = \boldsymbol{V}\ \forall\, t$) or time dependent, as further discussed in Section B.

To justify this choice of dynamics, we provide a more theoretical discussion in Section 4, showing that Eq. (3) allows stable long-range information propagation across space via energy conservation property. Hence, it is well-suited for modeling complex physical systems.

Warmup phase. In graph-based simulations, information typically spreads only to direct neighbors in each update, which can delay the emergence of global dynamics. To address this, we introduce a warmup phase that performs l rounds of message passing before the simulation starts. This process initializes the system by propagating information across the graph, enabling each node to incorporate signals from a larger neighborhood. The initial state of the simulation is set to $\boldsymbol{X}^{(0)} = \bar{\boldsymbol{X}}^{(l)}$, where $\bar{\boldsymbol{X}}^{(l)}$ is the state after l warmup steps. This ensures that the model begins with a globally informed latent state, improving accuracy and stability from the start of the rollout.

Geometrical information encoding. Modeling physical systems on irregular meshes requires encoding geometry to capture interactions based on relative positions. We compute edge features $\boldsymbol{e}_{ij}$ as:

$$\phi_{\text{node}}(\cdot) = \sigma\left(\boldsymbol{W}_{\text{node}}\boldsymbol{q}_i + \sum_{j \in \mathcal{N}_i} \boldsymbol{e}'_{ij} \right), \quad \boldsymbol{e}'_{ij} = \phi_{\text{edge}}(\cdot) = \boldsymbol{W}_{\text{edge}}(\boldsymbol{q}_j - \boldsymbol{q}_i) + \boldsymbol{e}_{ij}, \tag{5}$$

where $\boldsymbol{e}_{ij} = [\boldsymbol{s}_{ij}, \boldsymbol{d}_{ij}]$ combines displacement vectors $\boldsymbol{s}_{ij}$ and distances $\boldsymbol{d}_{ij}$. This formulation separates world-space edges $(\boldsymbol{q}_j - \boldsymbol{q}_i)$ from mesh-space edges $(\boldsymbol{e}_{ij})$, providing useful context while reducing dependence on explicit geometry.

Multi-step loss. We train IGNS using a multi-step loss to improve trajectory consistency and reduce error accumulation. Given a window $(\boldsymbol{q}^{(t)}, \ldots, \boldsymbol{q}^{(t+K)})$, the loss is defined as:

$$\mathcal{L}_{\text{multi-step}} = \sum_{\tau=1}^{K} \left(\|\widehat{\boldsymbol{q}}^{(t+\tau)} - \boldsymbol{q}^{(t+\tau)}\|_2^2 + \|\widehat{\boldsymbol{p}}^{(t+\tau)} - \boldsymbol{p}^{(t+\tau)}\|_2^2 \right), \tag{6}$$

where $\widehat{\boldsymbol{q}}$ and $\widehat{\boldsymbol{p}}$ are predictions, and $\boldsymbol{q}$ and $\boldsymbol{p}$ are ground truth. This loss leverages IGNS's ability to preserve signals over time, making gradients from distant steps effective. Including $\boldsymbol{p}$ further enhances performance by capturing velocity information.

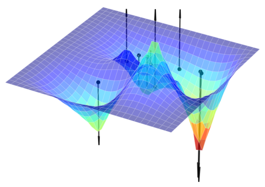
4 Theoretical Properties

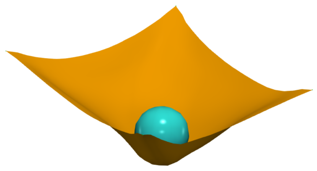
The ability to propagate information effectively is crucial for physical simulations. In the absence of damping and external forces, IGNS reduces to a pure Hamiltonian system, adhering to conservation laws [16, 17, 15]. This ensures divergence-free vector fields, preserving information during propagation.

To analyze IGNS's propagation capability, we study the sensitivity matrix $\|\partial \boldsymbol{x}(t)/\partial \boldsymbol{x}(s)\|$, as formalized below:

Theorem 1 (Theorem 2.3 of Heilig et al. [15])**.** *If $\boldsymbol{x}(t)$ solves $\dot{\boldsymbol{x}}_i = \boldsymbol{J}\nabla_{\boldsymbol{x}_i} H_\theta(t, \boldsymbol{X})$ (Hamiltonian dynamics in Eq.* (3) *without damping and forcing), then $\left\| \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{x}(s)} \right\| \geq 1$ for any $0 \leq s \leq t$.*

This theorem guarantees non-vanishing gradients, enabling long-range information propagation, unlike GCNs [18], which suffer from exponential decay [19]. This property also supports IGNS's


(a) Plate Deformation


(b) Sphere Cloth

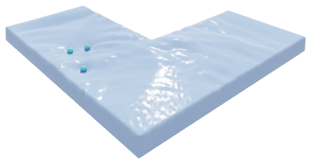

(c) Wave Balls

Figure 2: Three novel tasks introduced in this work to evaluate long-range interactions and complex dynamics under external forcing.

Table 1: Per–task test MSE (mean ± std, lower is better). Methods are grouped by operator class, *(I)* Graph Conv., *(II)* Eff. Propagation and *(III)* **Port-Ham. (Ours)**. Results are averaged over 4 seeds. Best and second best results are highlighted in **orange** and **teal**, respectively.

	Method	**Plate Def.** MSE	**Impact Plate** MSE	**Sphere Cloth** MSE ($\times 10^{-3}$)	**Wave Balls** MSE ($\times 10^{-3}$)	**Cylinder Flow** MSE ($\times 10^{-3}$)	**KS** MSE ($\times 10^{-3}$)
(I)	GCN+LN	3.98 ± 0.66	3997.98 ± 199.83	26.45 ± 0.23	1.85 ± 0.17	11.71 ± 1.03	3.10 ± 0.78
	MGN	1.27 ± 0.06	3095.75 ± 908.58	32.07 ± 2.45	1.78 ± 0.14	12.08 ± 2.60	10.76 ± 9.16
	MP-ODE	–	–	25.75 ± 1.32	87.28 ± 2.30	18.17 ± 1.15	59.55 ± 14.82
(II)	A-DGN	1.99 ± 0.76	3974.56 ± 309.98	30.99 ± 0.80	2.19 ± 0.30	14.72 ± 0.94	0.97 ± 0.89
	GraphCON	1.20 ± 0.02	2279.09 ± 720.25	29.00 ± 0.44	1.90 ± 0.06	7.32 ± 0.05	0.49 ± 0.02
(III)	**IGNS$_{\text{ti}}$**	1.35 ± 0.06	343.09 ± 156.74	27.99 ± 0.64	0.49 ± 0.03	8.39 ± 0.09	0.82 ± 0.09
	IGNS	1.34 ± 0.07	266.35 ± 89.44	23.46 ± 0.58	0.46 ± 0.07	7.91 ± 0.04	1.40 ± 0.25

warmup phase (Section 3). While Hamiltonian dynamics ensure robust propagation, dissipation and external forcing may counteract this. However, as shown in Rusch et al. [20], oscillatory behavior prevents exponential gradient decay, even with these terms. Specifically, for small Δt, gradients remain independent of the number of updates, alleviating the vanishing gradient problem.

5 Experiments

Datasets and Evaluation Methods. We evaluate our methods on six benchmark tasks: *Plate Deformation*, *Impact Plate*, *Sphere Cloth*, *Wave Balls*, *Cylinder Flow* and *Kuramoto-Sivashinsky* (KS). These tasks are designed to test long-range propagation, oscillatory behavior, and error accumulation under autoregressive training. Full dataset specifications are in Section D. We compare our proposed *port-Hamiltonian* models (IGNS$_{\text{ti}}$ and time-varying IGNS) against baselines grouped into *Graph Convolution* including GCN with LayerNorm (GCN+LN), MGN, MP-ODE (a MGN variant applied high-order ODE solver) [21] and *Effective Propagation* (e.g., ADGN [22], GraphCON [20]).

Results. Table 1 summarizes the main results across the six tasks. IGNS$_{\text{ti}}$ and IGNS consistently outperform the main competitor MGN and other graph-convolution baselines. Notably, GraphCON with the geometric encoding also performs strongly on static graph tasks (Plate Def., Cylinder Flow, KS) yet fails severely on the remaining ones. This suggests that while the oscillatory bias in GraphCON improves over standard graph convolution methods, its formulation limits its expressiveness for more complex dynamics. Furthermore, introducing time-variation in the weight matrices (IGNS) produces further gains on three tasks. Further insights, including ablations and comparisons with additional baselines, are detailed in Section F.

6 Conclusion

In this work, we introduced IGNS, a graph-based neural simulator leveraging port-Hamiltonian dynamics to enhance long-range information propagation and reduce error accumulation in physical modeling. By integrating warmup initialization, geometric encoding, and multi-step training, IGNS consistently outperforms baselines across diverse tasks, including benchmarks for long-range and complex dynamics. Theoretical analysis highlights IGNS's Hamiltonian structure for preserving gradient flow and its port-Hamiltonian extension for approximating a wide range of dynamics. While currently limited to open-loop settings, IGNS's ability to handle long-range dependencies opens avenues for closed-loop control, inverse design, and optimal action identification in future work.

References

[1] J.R. Dormand. *Numerical Methods for Differential Equations: A Computational Approach*. Engineering Mathematics. Taylor & Francis, 1996. ISBN 9780849394331.

[2] Uri M. Ascher. *Numerical Methods for Evolutionary Differential Equations*. Society for Industrial and Applied Mathematics, USA, 2008. ISBN 0898716527.

[3] Lawrence C Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, Providence, RI, 2 edition, March 2010.

[4] 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`.

[5] 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. URL `https://openreview.net/forum?id=jsZsEd8VEY`.

[6] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[7] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[8] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=vSix3HPYKSU`.

[9] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. In *Conference on Robot Learning*, pages 642–660. PMLR, 2023.

[10] Álvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in gnns: Bridging recurrent and graph learning. *arXiv preprint arXiv:2502.10818*, 2025.

[11] Arjan Van der Schaft. *L2-gain and passivity techniques in nonlinear control*. Springer, third edition, 2017.

[12] Heiner Igel. *Computational Seismology: A Practical Introduction*. Oxford University Press, 1. edition, 2016. ISBN 9780198717409. URL `https://global.oup.com/academic/product/computational-seismology-9780198717409?cc=de&lang=en&`.

[13] Arjan van der Schaft and Dimitri Jeltsema. *Port-Hamiltonian Systems Theory: An Introductory Overview*. 2014. doi: 10.1561/2600000002.

[14] Shaan A. Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J. Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Phys. Rev. E*, 104:034312, Sep 2021. doi: 10.1103/PhysRevE.104.034312. URL `https://link.aps.org/doi/10.1103/PhysRevE.104.034312`.

[15] Simon Heilig, Alessio Gravina, Alessandro Trenta, Claudio Gallicchio, and Davide Bacciu. Port-Hamiltonian Architectural Bias for Long-Range Propagation in Deep Graph Networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=03EkqSCKuO`.

[16] Clara Lucia Galimberti, Liang Xu, and Giancarlo Ferrari Trecate. A unified framework for hamiltonian deep neural networks. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A.;Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 275–286. PMLR, 07 – 08 June 2021. URL `https://proceedings.mlr.press/v144/galimberti21a.html`.

[17] Clara Lucia Galimberti, Luca Furieri, Liang Xu, and Giancarlo Ferrari-Trecate. Hamiltonian deep neural networks guaranteeing nonvanishing gradients by design. *IEEE Transactions on Automatic Control*, 68(5):3155–3162, 2023. doi: 10.1109/TAC.2023.3239430.

[18] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

[19] Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *The 39th Annual AAAI Conference on Artificial Intelligence*, 2025.

[20] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[21] Marten Lienen and Stephan Günnemann. Learning the dynamics of physical systems from sparse observations with finite element networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=HFmAukZ-k-2`.

[22] Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=J3Y7cgZOOS`.

[23] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. URL `https://arxiv.org/abs/1806.01261`.

[24] 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`.

[25] Youn-Yeol Yu, Jeongwhan Choi, Jaehyeon Park, Kookjin Lee, and Noseong Park. PIORF: Physics-informed ollivier-ricci flow for long–range interactions in mesh graph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=qkBBHixPow`.

[26] 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`.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[28] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf`.

[29] Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time {pde}s from sparse data with graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=aUX5Plaq7Oy.

[30] Dai Shi, Andi Han, Lequan Lin, Yi Guo, and Junbin Gao. Exposition on over-squashing problem on GNNs: Current Methods, Benchmarks and Challenges, 2023.

[31] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A Survey on Oversmoothing in Graph Neural Networks. *arXiv preprint arXiv:2303.10993*, 2023.

[32] Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 35:1–35:23. PMLR, 27–30 Nov 2024.

[33] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[34] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9266–9275, 2019. doi: 10.1109/ICCV.2019.00936.

[35] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=xkljKdGe4E.

[36] Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 34, pages 3836–3849. Curran Associates, Inc., 2021.

[37] Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

[38] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

[39] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Liò, and Michael Bronstein. On over-squashing in message passing neural networks: the impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[40] Johannes Gasteiger, Stefan Weiß enberger, and Stephan Günnemann. Diffusion Improves Graph Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[41] Federico Barbero, Ameya Velingker, Amin Saberi, Michael M. Bronstein, and Francesco Di Giovanni. Locality-aware graph rewiring in GNNs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4Ua4hKiAJX.

[42] Vladimir Igorevich Arnol'd. *Mathematical methods of classical mechanics*, volume 60. Springer Science & Business Media, 2013.

[43] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[44] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxmb1rKDS.

[45] Samuel Lanthaler, T. Konstantin Rusch, and Siddhartha Mishra. Neural oscillators are universal. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46786–46806. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/923285deb805c3e14e1aeebc9854d644-Paper-Conference.pdf`.

[46] T. Konstantin Rusch and Daniela Rus. Oscillatory state-space models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=GRMfXcAAFh`.

[47] Ernst Hairer, Marlis Hochbruck, Arieh Iserles, and Christian Lubich. Geometric numerical integration. *Oberwolfach Reports*, 3(1):805–882, 2006.

[48] Michael Smith. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp, United States, 2009.

[49] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.

[50] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

[51] 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`.

[52] David Zwicker. py-pde: A python package for solving partial differential equations. *Journal of Open Source Software*, 5(48):2158, 2020. doi: 10.21105/joss.02158. URL `https://doi.org/10.21105/joss.02158`.

A Related Work

Graph Neural Simulators. Graph Neural Networks have been widely adopted in machine learning for physical simulation tasks [23, 4]. Notable examples include modeling particle interactions [7, 24, 7], fluid dynamics [4, 25], and deformable materials [26, 5]. Here, given the observations at the current time step, one can predict the next states by applying a few iterations of message-passing steps to incorporate neighborhood information, and output the next state. This underlying message-passing step is often implemented with graph convolutional layers [18] or attention-based methods [27], or more generally, message-passing framework [23]. During rollout, the model can generate the whole trajectories in an autoregressive manner. However, these models often struggle with long-term accuracy due to error accumulation over long time horizons. In contrast, another line of work focuses on directly learning the underlying continuous dynamics using neural ODEs [28], which helps mitigate the error accumulation problem by enabling the model to output whole trajectories in a single forward pass and better capture the continuous nature of physical systems. Recent works extend this idea for graph-based architectures [29, 21]. These models are often trained with multi-step objectives, showing accuracy improvement and with the additional ability to handle irregular time series and improve generalization. However, these models are often limited to short-term predictions due to the challenges in propagating information over both long spatial and temporal horizons.

Effective propagation in Graph Neural Networks. Effectively propagating information across the graph, and thus effectively modeling long-range dependencies, remain a critical challenge for GNNs [30–32]. Stacking many GNN layers to capture long-range dependencies often leads to issues such as over-smoothing [33, 31], when node states become indistinguishable as the number of layers increases. To address this, methods like residual connections [34], normalization techniques [35], and advanced architectures [20, 36] have been proposed. Another issue that prevent effective long-range propagation is the over-squashing phenomenon [37–39], which arises from topological bottlenecks that squash exponentially increasing amounts of information into node states, causing loss of information between distant nodes. To overcome this, a variety of strategies have been proposed. For example, graph rewiring methods [38, 40, 41] modify the graph topology to facilitate communication. In physical system simulation, hierarchical and rewiring methods, including HCMT [26] and PIORF [25], improve the global information flow by introducing shortcuts or multi-scale pooling. However, these approaches can make models overly tied to the mesh structure, which can result in geometric overfitting. More recently, [10] demonstrated that both over-smoothing and over-squashing problems are closely linked to the problem of vanishing gradients in message passing, and works like [36, 22, 19, 15, 10] have explored this perspective to design GNNs that preserve the gradient flow over many layers. However, most of these methods focus on static graphs and node classification tasks, and it is still unclear how to extend them to physical simulation tasks.

Neural ODEs and Hamiltonian Dynamics. Neural Ordinary Differential Equations (ODEs) have been widely used to model continuous-time dynamics [28], offering flexibility in time integration and improved generalization. There has been significant interest in extending neural ODEs to capture physical systems by drawing inspiration from Hamiltonian dynamics [42]. Notable examples include Hamiltonian Neural Networks (HNNs) [43] and Symplectic ODE-Net [44], which achieve energy conservation and stable long-term predictions by learning a parameterized Hamiltonian function. This conservation property has also been explored as a way to mitigate gradient vanishing in deep neural networks [17, 45, 46], and it has inspired new architectures for enhancing long-range propagation in GNNs [15] as well as long-horizon forecasting in state-space-model [46]. However, these works remain different from ours, as they focus either on static graphs or purely temporal sequences.

In this work, we propose a novel GNS that leverages dynamical systems theory to preserve the gradient flow across distant nodes in spatial domains. Trained with multi-step loss, this model can mitigate error accumulation and hence, enable accurate long-term physical simulation.

B Time-varying Weight Matrices

To further enhance the model's expressiveness, we propose the following parameterization to make $\boldsymbol{W}(t)$ and $\boldsymbol{V}(t)$ in Eq. (4) time-varying, allowing the Hamiltonian to adapt over time, which is particularly useful for modeling more complex, non-stationary dynamics.

Here, we consider only the case of $\boldsymbol{W}(t)$, as $\boldsymbol{V}(t)$ can be treated similarly. Assuming $\boldsymbol{W}(t)$ is a square matrix of size $d \times d$, we decompose it into symmetric and skew-symmetric parts, i.e., $\boldsymbol{W}(t) = \boldsymbol{S}(t) + \boldsymbol{A}(t)$, where

$$\boldsymbol{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1d} \\ s_{12} & s_{22} & \cdots & s_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1d} & s_{2d} & \cdots & s_{dd} \end{bmatrix}, \qquad \boldsymbol{A} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1d} \\ -a_{12} & 0 & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{1d} & -a_{2d} & \cdots & 0 \end{bmatrix},$$

with $s_{ij} = s_{ji}$, $a_{ij} = -a_{ji}$, and $a_{ii} = 0$. We then factor these parts through time-invariant and time-varying components as

$$\boldsymbol{S}(t) = \boldsymbol{U}_s \operatorname{diag}(\gamma_\theta(t))\, \boldsymbol{U}_s^\top, \qquad \boldsymbol{A}(t) = \boldsymbol{U}_a \operatorname{diag}(\tau_\theta(t))\, \boldsymbol{P}_a^\top - \boldsymbol{P}_a \operatorname{diag}(\tau_\theta(t))\, \boldsymbol{U}_a^\top,$$

where $\boldsymbol{U}_s, \boldsymbol{U}_a, \boldsymbol{P}_a$ are learned (time-invariant) bases, and $\gamma_\theta(t), \tau_\theta(t)$ are two time-varying coefficient vectors output by MLPs with parameters θ. This two-step construction helps keeping all the learned parameters unconstrained while reducing the parameter counts to $\mathcal{O}(d)$, compared to the naive approach of letting a MLP output all the matrix entries, which leads to d^2 parameters. In practice, we implement $\gamma_\theta(t), \tau_\theta(t)$ as small MLPs that take time-embedding t as input and output the corresponding coefficient vectors.

C Symplectic Euler Integrator

To ensure the energy-conserving property of the Hamiltonian dynamics, we employ a symplectic integrator to numerically solve Eq. (3). Specifically, we choose the symplectic Euler method [47], which updates the state as follows:

$$\begin{aligned} \boldsymbol{p}_i^{(t+1)} &= \boldsymbol{p}_i^{(t)} + \Delta t\big(- \nabla_{\boldsymbol{q}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t)}) - \boldsymbol{D}_\theta \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t)}) + \boldsymbol{r}_\theta(t, \boldsymbol{X}^{(t)})\big), \\ \boldsymbol{q}_i^{(t+1)} &= \boldsymbol{q}_i^{(t)} + \Delta t \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t+1)}). \end{aligned} \tag{7}$$

Here, Δt is the time step size, and the gradients of the Hamiltonian are computed with respect to the generalized coordinates and momenta.

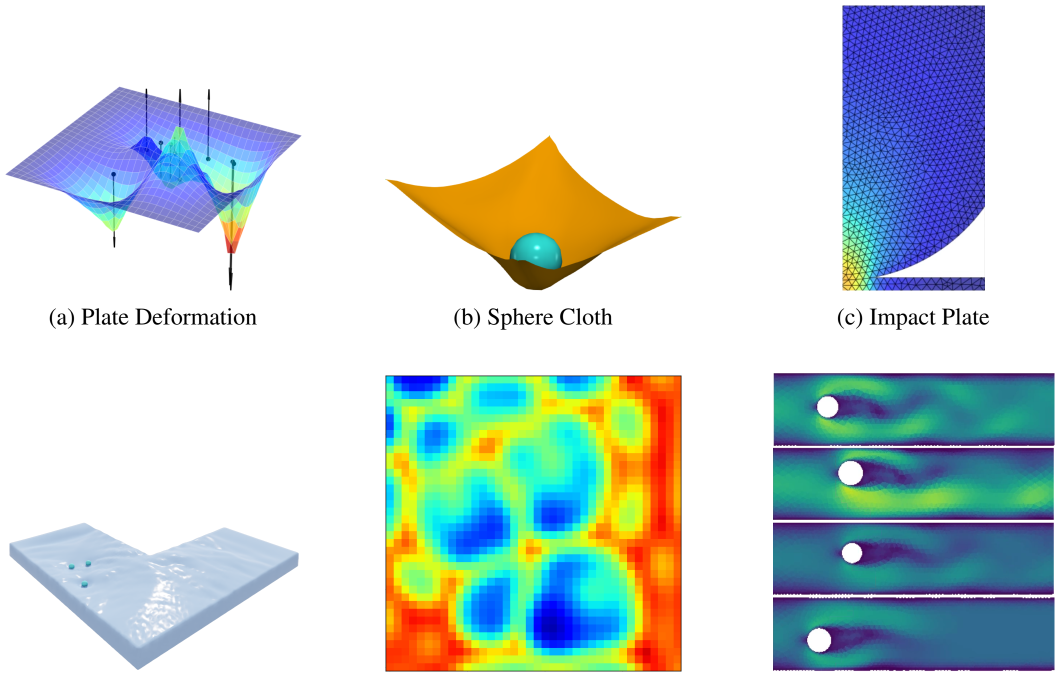
D Datasets Descriptions

Table 2: Dataset summary: average number of nodes, average number of edges, total horizon T, sub-horizon length (window size), and train/val/test split (number of trajectories).

Dataset	Avg. Nodes	Avg. Edges	Horizon T	Window	Train/Val/Test
Plate Deformation	851	3279	48	48	800/100/100
Sphere Cloth	401	3020	49	49	800/100/100
Impact Plate	2201	13031	51	51	2000/200/200
Wave Balls	1596	6048	50	50	250/100/50
Cylinder Flow	1885	10843	180	30	200/100/100
Kuramoto-Sivashinsky	1600	6240	300	100	250/100/50

Table 2 reports the dataset details, including the average number of nodes/edges, as well as horizon length and window size. For the full-rollout *Plate Deformation* task variant used in the ablation in Section F, we predict the whole trajectory with length $T = 48$. In addition, for *Cylinder Flow* and *Kuramoto-Sivashinsky*, we follow the suggestion from [21] to only split the sub-trajectories into equal-size, non-overlapping windows. Compared to the *Cylinder Flow* used in the [21] paper, we use the window size of 30 instead of 10 to further increase the complexity.

Next, we report the node-level inputs and outputs used for each task in Table 3. For Lagrangian systems (Plate Deformation, Sphere Cloth, Impact Plate), the state consists of displacement $\boldsymbol{q}_i$ and velocity $\dot{\boldsymbol{q}}_i$, together with static properties $\boldsymbol{n}_i$ such as material parameters, forces magnitude (for the Plate Deformation task) and node types. For these tasks, methods trained with the multi-step objective additionally receive the initial absolute position $\boldsymbol{q}_i^0$, which represents the rest configuration.


(a) Plate Deformation (b) Sphere Cloth (c) Impact Plate

(d) Wave Balls (e) Kuramoto-Sivashinsky (f) Cylinder FLow

Figure 3: Dataset overview. **(a) Plate Deformation**: a flat plate deformed subject to varying numbers and magnitudes of point forces, simulated with linear elasticity. **(b) Sphere Cloth**: a cloth mesh impacted by a falling sphere, producing elastic deformation with contact dynamics. **(c) Impact Plate**: an elastic plate under impact with a rigid ball. **(d) Wave Balls**: surface waves on water generated by three balls moving linearly from different initial positions. **(e) Kuramoto-Sivashinsky**: chaotic spatio-temporal evolution of a scalar field governed by the KS equation. **(f) Cylinder Flow**: incompressible fluid flow around cylindrical obstacles with vortex shedding.

Table 3: Task Inputs/Outputs used in our experiments. $\boldsymbol{q}_i$: displacement, $\dot{\boldsymbol{q}}_i$: velocity, $\boldsymbol{n}_i$: static properties (e.g., node-type), ρ_i: density/scalar field, $\boldsymbol{v}_i$: velocity field (for Cylinder Flow). For methods training with multi-step loss ([MS]), they also receive initial absolute positions $\boldsymbol{x}_i^0$ (notated inline).

Tasks	Type	Node Inputs	Node Outputs
Plate Deformation	Lagrangian	$\boldsymbol{n}_i$	$\boldsymbol{q}_i$
Sphere Cloth	Lagrangian	$\boldsymbol{n}_i,\ \dot{\boldsymbol{q}}_i, (\boldsymbol{q}_i^{(0)}$ [MS])	$\boldsymbol{q}_i,\ \dot{\boldsymbol{q}}_i$
Impact Plate	Lagrangian	$\boldsymbol{n}_i,\ \dot{\boldsymbol{q}}_i, (\boldsymbol{q}_i^{(0)}$ [MS])	$\boldsymbol{q}_i,\ \dot{\boldsymbol{q}}_i$
Wave Balls	Eulerian	$\boldsymbol{n}_i,\ \rho_i$	$\rho_i,\ \dot{\rho}_i$
Cylinder Flow	Eulerian	$\boldsymbol{n}_i,\ \boldsymbol{v}_i,\ \dot{\boldsymbol{v}}_i$	$\boldsymbol{v}_i,\ \dot{\boldsymbol{v}}_i$
Kuramoto-Sivashinsky	Eulerian	$\boldsymbol{n}_i,\ \rho_i$	$\rho_i,\ \dot{\rho}_i$

For Eulerian systems (Wave Balls, Kuramoto-Sivashinsky), the state is represented by a scalar density field ρ_i, and the outputs include both ρ_i and its temporal derivative $\dot{\rho}_i$. In Cylinder Flow, the state instead contains velocity $\boldsymbol{v}_i$, with $\dot{\boldsymbol{v}}_i$ provided to capture the temporal evolution of the velocity field.

In the following, we give the full details for each dataset, with examples given in Fig. 3:

Solid mechanics. This category includes Plate Deformation, Sphere Cloth, and Impact Plate (c.f. Yu et al. [26]).

- *Plate Deformation:* The task is to predict the final state (or full horizon) of an initially flat plate subjected to external forces of varying positions and magnitudes. These forces are encoded as special nodes connected locally to the sheet within a small radius, and their number ranges from 1 to 19. This setup primarily evaluates the model's ability to capture long-range spatial interactions. Ground-truth simulations are obtained using linear-elastic dynamics with shell elements in the commercial Abaqus software [48].

- *Sphere Cloth:* A ball is dropped from random positions and heights onto a cloth mesh of size 19×19, and the system is simulated for $T = 50$ steps. To ensure fair training and compatibility with dissipative models, we further enhance the cloth mesh connectivity by introducing edges between the ball and every fourth cloth node. The dynamics are governed by elasticity, and the initial ball position strongly influences the outcome. Simulations are performed using multi-body physics in NVIDIA Isaac Sim [49]. We also consider a more challenging variant of this task, used to ablate the importance of the time-varying component, with a larger cloth size of 29×29 and rollout length $T = 100$.
- *Impact Plate:* This is a standard benchmark from the HCMT paper [26] to test long-range interaction, where the model must predict both stress and displacement propagation from distant nodes. The ground-truth simulation is Ansys.

Fluid dynamics. This category includes Wave Balls, Cylinder Flow, and the Kuramoto-Sivashinsky (KS) equation.

- *Cylinder Flow:* This task is based on the standard MeshGraphNets task [4], but we use only $T = 180$ time steps (vs. 600) and 200 trajectories (vs. 1000), windowed into non-overlapping segments of length 30. After 180 steps, the system converges to a periodic steady state. This setup highlights error accumulation under autoregressive training, explaining the lower MGN performance in our results.
- *Wave Balls:* We let three balls, starting with random positions, move from left to right linearly along a water surface in various grid shapes (cross, L, U, T). The dynamics are governed by a second-order hyperbolic PDE with external forcing generated by those balls:

  $$\partial_{tt}u - c^2\,\nabla^2 u = f_{\text{external}}(x, t),$$

  The simulation is implemented in NVIDIA Warp [50]. An extended version of this task, used in the ablation study, runs for $T = 100$ steps. For the first 50 steps the balls move across the surface, and for the next 50 steps they stop. The waves, however, keep oscillating since there is no damping. This creates an interesting case where the model must learn not only the forced response but also continue the free oscillation, and figure out which phase the system is in.
- *Kuramoto-Sivashinsky (KS):* A 4th-order PDE commonly used in neural operator benchmarks [51], generating chaotic behavior. Instead of random initial states, we use three Gaussian sources on a 40×40 grid and apply Neumann boundary conditions (not periodic) to focus on local rather than global behaviors. Simulated with the `py-pde` library [52]. The KS equation in 2D is given by

  $$\partial_t u = -\frac{1}{2}|\nabla u|^2 - \nabla^2 u - \nabla^4 u.$$

E Hyperparameters and Training Time

Table 4 summarizes the shared and task-specific hyperparameters. Most settings are common across models, with feature sizes fixed at 128 and Adam as the optimizer. For *MGN*, we adopt mean aggregation with Leaky ReLU, use 15 message-passing blocks by default (50 for Plate Deformation), and apply Gaussian training noise with $\sigma = 1 \times 10^{-2}$. In contrast, *MS* methods use ReLU activations in the encoder and decoder, while the graph-convolution blocks rely on $\tanh$ in A-DGN, IGNS and IGNS$_{\mathbf{ti}}$. Training noise is generally disabled for *MS* methods, but we enable it with $\sigma = 1 \times 10^{-2}$ in Cylinder Flow and Kuramoto–Sivashinsky, where the initial conditions vary. In all cases, we normalize inputs and unnormalize outputs for every training batch, which stabilizes optimization and allows us to use the same learning rate and training noise across tasks.

Table 5 (a) shows the training time allocated to each task. To ensure fairness, all methods were trained for the same wall-clock time on a single NVIDIA A100 GPU. For the default settings, the budgets were chosen based on validation curves, where extending training further did not yield noticeable improvements. For the extended version of Sphere Cloth and Wave Balls tasks, we instead fixed a

Table 4: Hyperparameters used in all experiments. "AR" = autoregressive methods (e.g., MGN). "MS" = multi-step methods (others).

Hyperparameter	**Common**	**AR**	**MS**
Node feature dimension	128	–	–
Latent representation dimension	128	–	–
Decoder hidden dimension	128	–	–
Optimizer	Adam	–	–
Learning rate	5×10^{-4}	–	–
Message-passing blocks	–	$15^{\dagger}$	Window size
Training noise (std)	–	1×10^{-2}	none§
Task-specific overrides (MS unless noted):			
Plate Deformation	message blocks = 48.		
Sphere Cloth	Warmup $= 20$.		
Impact Plate	Warmup $= 15$.		
Wave Balls	Warmup $= 0$.		
Kuramoto–Sivashinsky	Warmup $= 10$; MS training noise $= 1 \times 10^{-2}$.		
Cylinder Flow	Warmup $= 10$; MS training noise $= 1 \times 10^{-2}$.		

† MGN uses 15 message-passing blocks by default; Plate Deformation uses 50.
§ For MS methods we disable training noise except on Cylinder Flow and Kuramoto–Sivashinsky, where we use 1×10^{-2} due to varying initial conditions.

Table 5: (a) Training time budget for each task, where all methods were trained with the same wall-clock time. (b) Parameter counts (measured on the Kuramoto-Sivashinsky task).

Task	Default (hours)	Long (hours)
Plate Deformation	12	–
Sphere Cloth	12	24
Impact Plate	33	–
Wave Balls	16	48
Cylinder Flow	24	–
Kuramoto–Sivashinsky	24	–

(a) Training time

Method	#Parameters
IGNS	216,000
IGNS$_{\text{ti}}$	80,100
GCN+LN	79,700
MP-ODE	101,000
GraphCON	67,800
A-DGN	100,000
MGN (15 MP)	1,800,000

(b) Parameter counts

hard budget to ensure sufficient training for fair comparison in the ablation studies, independent of validation plateaus. In Table 5 (b), we report the number of parameters for each method. It is clear that the standard MGN with a non-shared processor requires by far the largest number of parameters.

F Additional Results

Table 6: Comparisons with additional strong baselines on Plate Deformation, Impact Plate and Kuramoto-Sivashinsky tasks. Best results are highlighted in orange.

Method	MSE
MGN-rewiring	3.72 ± 0.15
MGN-shared	8.21 ± 1.98
MGN	1.27 ± 0.06
IGNS$_{\text{ti}}$	1.35 ± 0.06
IGNS	1.34 ± 0.07

(a) Plate Deformation

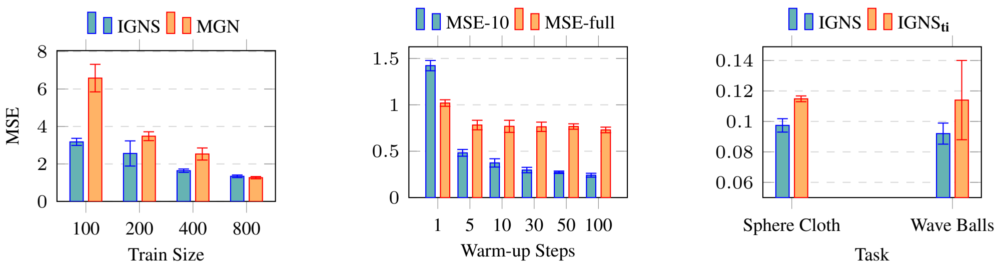
Method	MSE
HCMT	646.81 ± 27.87
MGN	3095.75 ± 908.58
IGNS$_{\text{ti}}$	343.09 ± 156.74
IGNS	266.35 ± 89.44

(b) Impact Plate

Method	MSE ($\times 10^{-3}$)
FNO-RNN	6.69 ± 0.46
MGN	10.76 ± 9.16
IGNS$_{\text{ti}}$	0.82 ± 0.09
IGNS	1.40 ± 0.25

(c) Kuramoto-Sivashinsky

Long-range propagation. In Plate Deformation (final-state prediction only), the port-Hamiltonian-based models (IGNS$_{\text{ti}}$, IGNS) and GraphCON achieve low error, while A-DGN and GCN+LN perform substantially worse. Interestingly, MGN also performs well on this task. To better understand



(a) Data efficiency. (b) Varying warm-up steps. (c) Longer horizon.

Figure 4: Ablations: *(a)* data efficiency on **final-state-only** Plate Deformation task; *(b)* warm-up steps on **full-roll-out** Plate Deformation task; *(c)* Sphere Cloth vs. Wave Balls with longer horizon ($T = 100$) (Wave Balls MSE $\times 100$ for visibility). All results are computed using 4 seeds.

this result, we further evaluate two MGN variants: MGN-shared, which uses a shared processor across steps, and MGN-rewiring, which connects each force node to every second node in the mesh. As shown in Table 6 (a), both variants yield higher error than standard MGN, suggesting that MGN's strong performance is primarily due to its non-shared processor. Although this design increases the parameter count significantly ($\times$#MP_Layers), it allows the model to overfit to geometric details. Additional visualizations supporting this observation are provided in Section F.

For Impact Plate, only our proposed models and the baseline HCMT achieve strong performance, with IGNS attaining the lowest error. This result highlights that the energy-preserving property of Hamiltonian dynamics enables effective long-range transmission of elastic energy between distant nodes, without requiring specialized hierarchical architectures. Notably, we find that GraphCON's loss consistently diverges after a certain number of training iterations across all seeds, indicating that its fixed oscillatory bias may be too restrictive to capture the dynamics present in this task. Note that we omit MP-ODE results on these two tasks, as the method was not proposed for static graphs, and for Impact Plate, it fails to converge during training.

Complex dynamics analysis. We now analyze performance across the remaining tasks in Table 1, that involve complex dynamics and oscillatory behavior. For Sphere Cloth, IGNS achieves the lowest error, while the other models yield errors of similar magnitude, with MGN exhibiting the largest error. In Wave Balls, our proposed models (IGNS, IGNS$_{\text{ti}}$) significantly outperform all baselines, suggesting that the port-Hamiltonian structure, viewed as a generalization of wave equations (**??**), is particularly well-suited for this type of wave dynamics.

For Cylinder Flow, both port-Hamiltonian-based and GraphCON models perform comparably, while A-DGN, GCN+LN, and MP-ODE show larger errors under the causal-only setup used here. Notably, in the original paper [4], MGN performs better due to longer autonomous portions in the dataset; our causal-focused split reduces that advantage (Section D). Finally, for Kuramoto-Sivashinsky, the chaotic dynamics heavily penalize autoregressive MGN (resulting in large error), while GraphCON, IGNS$_{\text{ti}}$/IGNS, and A-DGN remain comparatively strong; GCN+LN struggles to represent the emergence of high-frequency modes towards the end of the evolution. Although Fourier Neural Operators (FNO-RNN) [51] often perform well on this type of dynamic, they underperform significantly compared to graph-based models, as shown in Table 6 (c). This is likely because, unlike the usual setup with uniformly random initial states, here we initialize with three Gaussian sources randomly placed in the grid to encourage local propagation. Further discussion and visualizations for all tasks are provided in Section F.

Ablation Studies. (a) Data efficiency. We compare IGNS and MGN on Plate Deformation with varying training set sizes (100, 200, 400 and 800 samples). As shown in Fig. 4a, IGNS consistently outperforms MGN across all dataset sizes, with the performance gap widening as the training set decreases. This suggests that the inductive bias from port-Hamiltonian dynamics helps IGNS generalize better from limited data, while MGN lead to geometric overfitting when data is scarce. **(b) Warm-up steps.** Next, we study the effect of varying the number of warm-up steps l used during training for IGNS on Plate Deformation with full-rollout prediction. Fig. 4b shows two metrics: cumulative MSE over the first 10 steps (MSE-10) and full-rollout validation loss. Both improve as l increases. Moving from $l = 1$ to $l = 5$ yields the largest gain (since at $t = 1$, the task already requires global propagation across the plate); after $l = 30$ the full-rollout loss largely plateaus, while MSE-10 continues to improve with larger l, indicating larger warm-up reduces early error directly, while smaller l forces the model to learn dynamics that recover from early mistakes later (visualizations are shown in Fig. 5). **(c) Longer horizon.** Finally, we compare IGNS and IGNS$_{\mathbf{ti}}$ on Sphere Cloth and Wave Balls with a longer rollout horizon of $T = 100$ (vs. $T = 50$ in the main experiments). In Sphere Cloth, we additionally increase the number of nodes to 29×29 (vs. 19×19 in the main experiment) to increase complexity. As shown in Fig. 4c, IGNS outperforms IGNS$_{\mathbf{ti}}$ on both tasks, with lower std. This suggests that making weighted matrices time-varying helps increase the model's expressiveness and therefore, enables it to capture more complex dynamics.

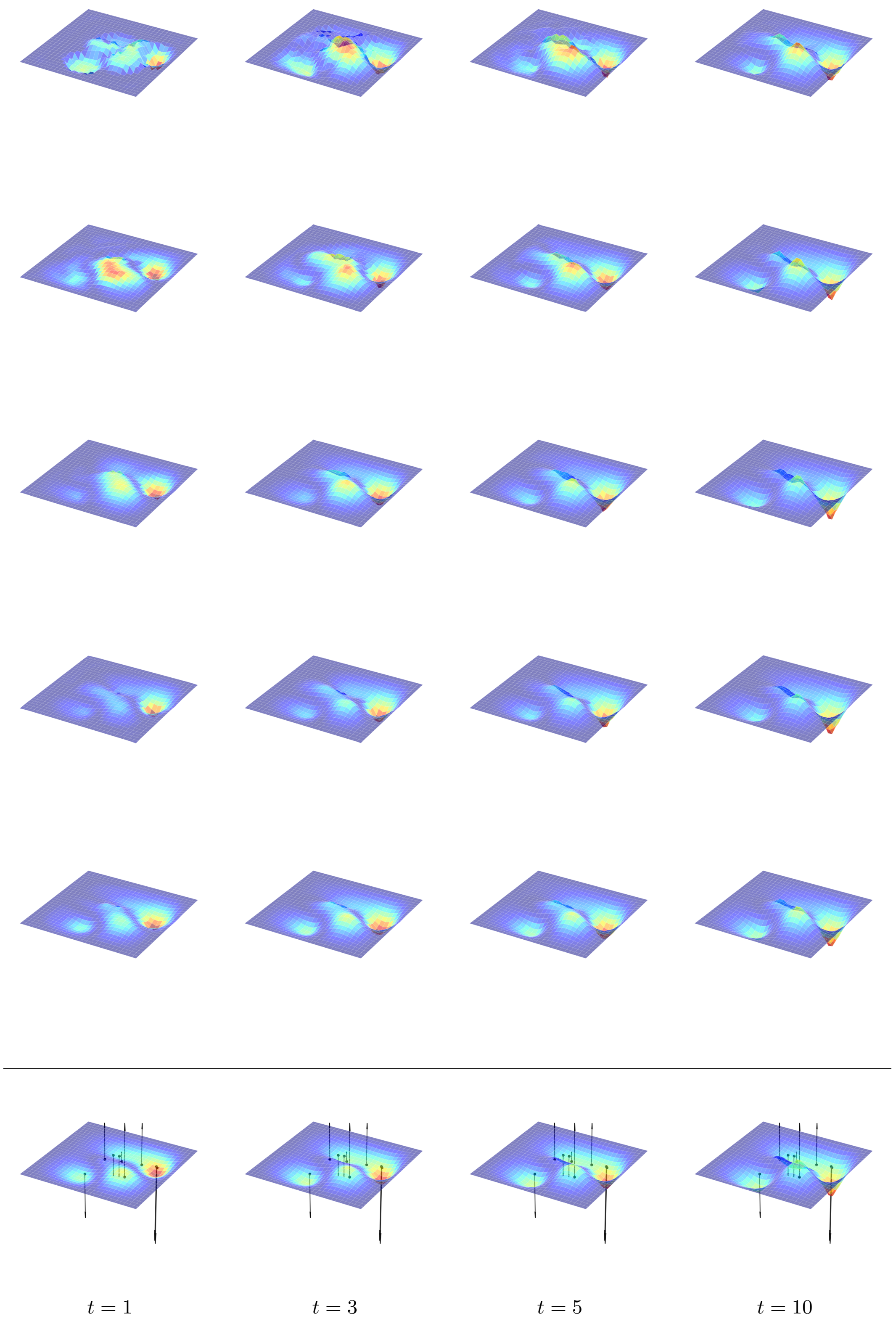


Figure 5: Comparison between different warmup steps l from top to bottom $l = \{1, 5, 10, 30, 50\}$ with ground truh in the last row, on Plate Deformation task.