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 complex physical systems is central to many scientific domains. These systems are typically governed by partial differential equations (PDEs), which are rarely solvable in closed form. To approximate the solution, classical numerical solvers (Dormand, 1996; Ascher, 2008; Evans, 2010) discretize the domain, obtaining a mesh in the case of the finite element method (Brenner & Scott, 2008). However, most solvers are tied to a specific kind of discretization, and sufficiently accurate simulations require fine-grained meshes, which quickly become prohibitively expensive.

On the other hand, neural simulators directly learn system dynamics from data (Guo et al., 2016; Bhatnagar et al., 2019; Li et al., 2019b; Pfaff et al., 2021), enabling simulations that can be several orders of magnitude faster than numerical methods. In particular, Graph Neural Simulators (GNSs) (Pfaff et al., 2021; Linkerhägner et al., 2023; Yu et al., 2024) leverage Graph Neural Networks (GNNs) (Wu et al., 2020; Gravina & Bacciu, 2024) to capture interactions between entities, such as mesh elements, through message passing (Gilmer et al., 2017). By encoding local geometry in graph edges, they achieve accurate and efficient simulations that generalize across meshes in applications ranging from fluid to elastic dynamics (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Yu et al., 2025).

GNSs are commonly trained with a one-step loss to predict the next state from the current observation, and then rolled out autoregressively to generate long trajectories during inference. While being effective over short horizons, even small global errors accumulate rapidly. Implicit (Pfaff et al., 2021; Brandstetter et al., 2022) and explicit (Würth et al., 2025) denoising objectives alleviate this to some extent by reducing local noise, but the loss still fails to capture low-frequency drift that emerges only after several steps. Multi-step objectives have been proposed to overcome the shortcomings of this one-step training (Shi et al., 2023b; Lienen & Günnemann, 2022). In practice, how-

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

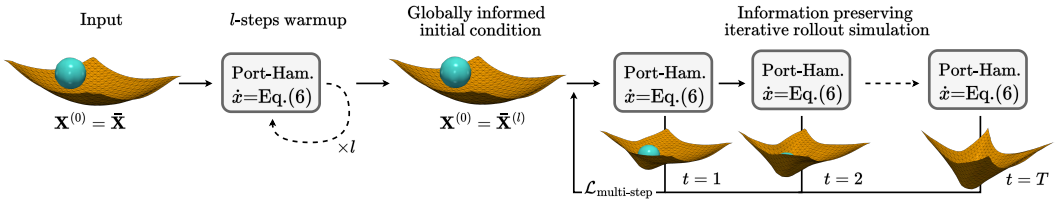


Figure 1: A high-level overview of the proposed IGNS. The model takes as input the initial node state $\bar{\boldsymbol{X}}$ and performs an l-step warmup phase (left), enabling each node to incorporate broader spatial context before the rollout begins. The enriched state is then used to initialize the 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. (6), while the multi-step loss $\mathcal{L}_{\text{multi-step}}$ supervises all intermediate predictions, ensuring stable and accurate simulations.

ever, they only work over short horizons, since message passing quickly becomes unstable and loses information after a few steps (Arroyo et al., 2025), making direct long-horizon training infeasible. This limitation is fundamental to existing GNSs, as their GNN architectures struggle with long-range dependencies due to over-smoothing (Cai & Wang, 2020; Rusch et al., 2023), over-squashing (Alon & Yahav, 2021; Di Giovanni et al., 2023), and more generally gradient degradation (Arroyo et al., 2025).

We propose Information-preserving Graph Neural Simulators (IGNS), a neural simulator that learns continuous evolution in a latent space and enables effective information propagation across the graph for accurate predictions. A schematic of IGNS is shown in Fig. 1; unlike existing approaches with dissipative updates that quickly degrade information as message passing depth increases, IGNS structures information flow through port-Hamiltonian dynamics (Van der Schaft, 2017; Galimberti et al., 2021; Lanthaler et al., 2023; Heilig et al., 2025). This design helps preserve long-range dependencies between distant nodes better and, when combined with multi-step training, reduces error accumulation during rollout. In addition, IGNS incorporates a warmup phase to initiate global context before rollout and a geometric encoding module to capture local spatial relations on irregular meshes. Together, these components enhance stability and generalization, enabling accurate simulation across diverse physical systems while maintaining a physically consistent inductive bias. Experimentally, IGNS outperforms state-of-the-art GNSs on long-range and complex dynamics benchmarks, yielding more accurate predictions and stable rollouts.

To summarize, 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 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

In this section, we provide an overview of PDEs for modeling physical systems, followed by a brief discussion of message-passing-based GNSs for learning such systems. A in-depth discussion of related works about GNSs, effective propagation, as well as Hamiltonian-inspired neural networks is provided in Section A.

2.1 Partial Differential Equations for Physical Modeling

Many physical phenomena can be described by PDEs, which relate the rates of change of a quantity with respect to both space and time. A general form of a time-dependent PDE is given by

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

where $u : \Omega \times [0, T] \to \mathbb{R}$ is the unknown function (without loss of generality, assumed to be a scalar field), $\Omega \subseteq \mathbb{R}^d$ is the spatial domain, and F is a (possibly nonlinear) function of u and its spatial derivatives.

To solve Eq. (1) numerically, one typically discretizes the spatial domain using methods such as finite differences or finite elements, resulting in a system of ordinary differential equations (ODEs) that can be integrated over time using various time-stepping schemes (Igel, 2016). This is often referred to as the method of lines, where spatial derivatives are approximated at discrete points, leading to a system of ODEs of the form

$$\dot{\boldsymbol{u}}_i(t) = \boldsymbol{f}_i(t, \boldsymbol{u}(t)), \quad i = 1, \ldots, n, \quad \boldsymbol{u}(0) = \bar{\boldsymbol{u}}, \tag{2}$$

where $\boldsymbol{u}(t) = [\boldsymbol{u}_1(t), \ldots, \boldsymbol{u}_n(t)]^\top$ represents the state of the system at discrete spatial points, and $\boldsymbol{f}_i$ encapsulates the spatial discretization of F at point i. The initial condition $\bar{\boldsymbol{u}}$ is derived from the initial state of the PDE.

Remark. While first-order systems as in Eq. (1) are common, many physical phenomena are better captured by second-order dynamical systems (Evans, 2010), which naturally arise in the modeling of oscillatory behavior like propagation of waves and vibrations. A general second-order PDE with damping and external forcing can be written as

$$\partial_{tt} u + \mathcal{B}\, \partial_t u + \mathcal{L}\, u = g(\boldsymbol{x}, t), \qquad (\boldsymbol{x}, t) \in \Omega \times (0, T],$$

where $\mathcal{L}$ and $\mathcal{B}$ are linear operators on $L^2(\Omega)$, and g is a prescribed forcing term. A detailed derivation of applying the method of lines to this equation is provided in Section B.2, leading to a system of second-order ODEs.

2.2 Message Passing for Model Learning

Under the model learning settings, the goal is to learn the underlying dynamics (i.e., the RHS of Eq. (1)) from data. Similar to traditional numerical methods, we first need to discretize the spatial domain into a set of points or elements. This discretization can be naturally represented as a graph, $G = (V, E)$ where V is the set of n nodes (i.e., the mesh entities) and $E \subseteq V \times V$ is the set of m edges capturing their interactions. Then, following the method of lines, we obtain the following system of ODEs:

$$\dot{\boldsymbol{x}}_i^{(t)} = \boldsymbol{f}_i(t, \boldsymbol{X}^{(t)}; \boldsymbol{E}; G), \quad i = 1, \ldots, n, \quad \boldsymbol{X}^{(0)} = \bar{\boldsymbol{X}}, \tag{3}$$

where $\boldsymbol{x}_i^{(t)} \in \mathbb{R}^d$ is the state vector of node i at time t, $\boldsymbol{X}(t) = [\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_n^{(t)}]^\top \in \mathbb{R}^{n \times d}$ is the node-state matrix, $\boldsymbol{E} = [\boldsymbol{e}_{ij}, \ldots]^T \in \mathbb{R}^{m \times c}$ is the edge-feature matrix, and $\boldsymbol{f}_i$ is the unknown function that captures the interactions between nodes based on the graph structure G. Here, we also assume the initial condition $\bar{\boldsymbol{X}}$ is given.

Solving this requires integrating Eq. (3) over time, which can be done using standard ODE solvers (Hairer et al., 1993). However, standard Graph Neural Simulators (GNS) simplify this process by further discretizing time using a fixed step size Δt and applying the message-passing paradigm to compute the node states at the next time step $\boldsymbol{x}^{(t+1)}$, leading to the following update rule

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \Delta t\, \boldsymbol{g}_\theta^L(\boldsymbol{X}^{(t)}; \boldsymbol{E}; G). \tag{4}$$

Here, $\boldsymbol{g}_\theta^L$ denotes the function obtained by stacking L message-passing layers, which aggregate information from neighboring nodes to update the state of each node, defined as

$$\boldsymbol{x}_i' = \phi_{\text{node};}\left(\boldsymbol{x}_i, \sum_{j \in \mathcal{N}_i} \boldsymbol{e}_{ij}'\right), \quad \boldsymbol{e}_{ij}' = \phi_{\text{edge}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{e}_{ij}). \tag{5}$$

Here, ϕ_{edge} is the message function, ϕ_{node} is the node update function, and $\mathcal{N}_i$ is the set of neighbors of node i. Both ϕ_{edge} and ϕ_{node} are often implemented as MLPs. We can see that under this setting, $\boldsymbol{f}_i$ in Eq. (3) is approximated by a stack of L message-passing layers, i.e., $\boldsymbol{f}_i(t, \boldsymbol{X}^{(t)}; \boldsymbol{E}; G) \approx \boldsymbol{g}_\theta^L(\boldsymbol{X}^{(t)}; \boldsymbol{E}; G)$. In practice, one can obtain the whole trajectory by iteratively applying Eq. (4) for T time steps, starting from the initial condition $\bar{\boldsymbol{X}}$. However, this approach often suffers from error accumulation over long time horizons, limiting long-term accuracy (Brandstetter et al., 2022).

In this work, guided by the theory of dynamical systems, we propose a novel GNS that directly models the underlying continuous dynamics in Eq. (3), built upon the port-Hamiltonian formalism. Under this framework, information is maintained more effectively across the entire space. Combined with the multi-step loss, the learned model demonstrates that it can mitigate the error accumulation issue and achieve greater accuracy on long rollouts. We discuss them in detail in the next section.

3 THE INFORMATION-PRESERVING GRAPH NEURAL SIMULATORS

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*: by parameterizing a Hamiltonian H and adding damping and external forcing terms, IGNS captures both energy-conserving and non-conservative behavior.
- *Warmup*: the conservation property is further exploited through the use of a *warmup phase* at initialization, enabling the model to obtain a more global context from the start of the rollout.
- *Geometric encoding*: by embedding geometric features directly into the edge attributes, IGNS can operate on irregular meshes and exploit spatial structure for more accurate local interactions.
- *Multi-step objective*: it supervises the entire rollout windows, reducing error accumulation, improving data efficiency, and better aligning training with long-horizon prediction.

3.1 PORT-HAMILTONIAN FORMALISM

We start by expressing the dynamics on the RHS of the Eq. (3) using the port-Hamiltonian formalism (van der Schaft & Jeltsema, 2014; Desai et al., 2021; Heilig et al., 2025), which is a generalization of Hamiltonian systems that integrate both conservative and non-conservative dynamics based on energy principles. Specifically, we parameterize the Hamiltonian $H_\theta(t, \boldsymbol{X})$ and evolve the joint state according to

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}\nabla_{\boldsymbol{x}_i} H_\theta(t, \boldsymbol{X}) - \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{D}_\theta \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{X}) \end{bmatrix}}_{\text{damping}} + \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{r}_\theta(t, \boldsymbol{X}) \end{bmatrix}}_{\text{forcing \& residual}}, \quad \boldsymbol{J} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix}. \tag{6}$$

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. (6) 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

$$H_\theta(t, \boldsymbol{X}) = \sum_{i \in \mathcal{V}} \tilde{\sigma}\left(\boldsymbol{W}(t)\boldsymbol{x}_i + \sum_{j \in \mathcal{N}_i} \boldsymbol{V}(t)\boldsymbol{x}_j \right)^T \cdot \mathbf{1}_d, \tag{7}$$

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 C.

To justify this choice of dynamics, we provide a more theoretical discussion in Section 4, showing that Eq. (6) is universal, and via energy conservation, it allows stable long-range information propagation across space. Hence, it is well-suited for modeling complex physical systems.

Symplectic Integrator. To ensure the energy-conserving property of the Hamiltonian dynamics, we employ a symplectic integrator to numerically solve Eq. (6). Specifically, we choose the symplectic Euler method (Hairer et al., 2006), which updates the state as follows:

$$\begin{aligned}\boldsymbol{p}_i^{(t+1)} &= \boldsymbol{p}_i^{(t)} + \Delta t\big(- \nabla_{\boldsymbol{q}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t)}) - \boldsymbol{D}_\theta \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t)}) + \boldsymbol{r}_\theta(t, \boldsymbol{X}^{(t)})\big), \\ \boldsymbol{q}_i^{(t+1)} &= \boldsymbol{q}_i^{(t)} + \Delta t \nabla_{\boldsymbol{p}_i} H_\theta(t, \boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t+1)}).\end{aligned} \tag{8}$$

Here, the gradients of the Hamiltonian are computed with respect to the generalized coordinates and momenta.

3.2 WARMUP PHASE

Under the message-passing framework, a single update propagates information only to direct neighbors. This locality is a fundamental limitation when simulating physical systems, where long-range interactions often play a crucial role from the very first timestep. For example, in mesh-based simulations of fluids or elastic materials, the evolution of a node may depend not only on its immediate surroundings but also on distant regions of the mesh, such as boundary conditions or sources of external forces. If the model starts from a completely local representation, it must accumulate long-range information gradually over several rollout steps, which delays the emergence of globally consistent dynamics and may degrade accuracy in the early stages of the simulation.

To alleviate this issue, we introduce a *warmup phase* applied once at initialization. Starting from the initial state, we perform l additional rounds of message passing without advancing time, as visually summarized on the left of Fig. 1. More formally, the initial condition of the simulation is set to $\boldsymbol{X}^{(0)} = \bar{\boldsymbol{X}}^{(l)}$, where $\bar{\boldsymbol{X}}^{(l)}$ denotes the state after l warmup iterations. This strategy broadcast information across the graph up to a radius l from each node, so that for sufficiently large l, each node effectively receives signals from increasingly larger neighborhoods. Thanks to the energy-conserving core of IGNS, this globally informed latent state is preserved throughout the rollout, rather than being dissipated. The effect of l is further analyzed in Section 5.1.

3.3 GEOMETRICAL INFORMATION ENCODING

Accurately encoding geometry is essential for modeling physical systems on irregular meshes, where interactions depend on relative positions (i.e., in the mesh space) rather than absolute states (i.e., in the physical world space). To capture this relationship, we model the external force to drive the system's evolution based on geometrical information, thereby allowing the dynamics to be influenced by the spatial structure. We design the external force following an edge-level message-passing scheme inspired by Pfaff et al. (2021), where each edge feature $\boldsymbol{e}_{ij}$ encodes both the displacement vectors ($\boldsymbol{s}_{ij} = \text{pos}_j - \text{pos}_i$) and distances ($\boldsymbol{d}_{ij} = \|\text{pos}_j - \text{pos}_i\|_2$), with pos_i indicating the position of node i in the mesh space. More formally, we formulate the external force as $\boldsymbol{r}_\theta(t, \boldsymbol{X}) = \boldsymbol{g}_\theta^L(\cdot)$ where each message-passing step is defined as

$$\phi_{\text{node}}(\cdot) = \sigma\left(\boldsymbol{W}_{\text{node}} \boldsymbol{q}_i + \sum_{j \in \mathcal{N}_i} \boldsymbol{e}'_{ij}\right), \quad \boldsymbol{e}'_{ij} = \phi_{\text{edge}}(\cdot) = \boldsymbol{W}_{\text{edge}}(\boldsymbol{q}_j - \boldsymbol{q}_i) + \boldsymbol{e}_{ij} \tag{9}$$

with $\boldsymbol{e}_{ij}$ as the concatenation between displacement vectors and distance values, i.e., $\boldsymbol{e}_{ij} = [\boldsymbol{s}_{ij}, \boldsymbol{d}_{ij}]$. This formulation explicitly separates the edge in the world space ($\boldsymbol{q}_j - \boldsymbol{q}_i$) and the edge in the mesh space ($\boldsymbol{e}_{ij}$) in the edge update. Unlike Pfaff et al. (2021), we do not update edge messages at every time step. Combined with the multi-step loss (discussed below), the edge information serves instead as a static prior that weighs neighbor messages. This makes the model less dependent on explicit geometry, helps prevent overfitting to specific meshes, and still provides sufficiently useful context. We empirically study this hypothesis in Section G.

3.4 MULTI-STEP LOSS

We train our models through a multi-step objective. Given a window $(\boldsymbol{q}^{(t)}, \ldots, \boldsymbol{q}^{(t+K)})$, we roll out from $\boldsymbol{q}^{(t)}$ and optimize both the predicted $\widehat{\boldsymbol{q}}^{(t+\tau)}$ and $\widehat{\boldsymbol{p}}^{(t+\tau)}$ to match its corresponding ground truth fields $\boldsymbol{q}^{(t+\tau)}$ and velocities $\boldsymbol{p}^{(t+\tau)}$, $\forall \tau \in [1, K]$,

$$\mathcal{L}_{\text{multi-step}} = \sum_{\tau=1}^{K} \Big(\|\widehat{\boldsymbol{q}}^{(t+\tau)} - \boldsymbol{q}^{(t+\tau)}\|_2^2 + \|\widehat{\boldsymbol{p}}^{(t+\tau)} - \boldsymbol{p}^{(t+\tau)}\|_2^2 \Big). \tag{10}$$

This multi-step loss enhances the consistency between the prediction and ground-truth data on a trajectory level. In contrast, one-step training followed by autoregressive rollout is prone to suffering from the error accumulation problem, especially when the time-series data exhibits a strong temporal correlation, which we analyze in Section 5.1. The multi-step loss also fits IGNS well because the non-dissipative core preserves signals across time; therefore, the gradients from distant steps do not vanish, and the model can make use of the whole trajectory. Standard first-order message passing instead tends to oversmooth, gradients decay over many steps, and the benefit of a long horizon objective is limited. Finally, including $\boldsymbol{p}$ helps improve performance further; when momenta are not explicit in the state, we approximate them by finite differences on the ground truth $\boldsymbol{q}^{(t)}$ and $\boldsymbol{q}^{(t+\tau)}$. A visual exemplification of this multi-step loss strategy is summarized on the right of Fig. 1.

4 Theoretical Properties

In this section, we discuss the theoretical benefits of our IGNS, focusing on universality and effective information propagation during the simulation, underpinning its ability to effectively simulate complex physical systems. A more in-depth discussion is provided in Section D.

Universality. We start by showing that our IGNS, with state updates in Eq. (6) and Hamiltonian in Eq. (7) yields a universal model. Therefore, IGNS can approximate any given functional $\Phi : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ that maps the initial condition to the solution at time τ.

Theorem 1 (Universality of IGNS)**.** *Let Ψ_θ be the functional that maps the initial condition $\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0$ to the solution at time τ to the ODE defined Eq.* (6) *and Eq.* (7)*. In other words, Ψ_θ represents IGNS. Then, for any map $F : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, which represents the evolution of the system after time τ, there exist a set of parameters θ such that $\|\Psi_\theta(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0) - F(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0)\| \leq \epsilon$.*

The proof is reported in Section D.1, along with an in-depth discussion and interpretation for physical simulations. As a consequence of the theorem and its proof, IGNS is capable of approximating any functional that maps the time-dependent signal of external forces and geometrical information into the dynamics of the physical system, making it suitable for accurately modeling complex interactions and dynamics in arbitrary graph-structured systems.

Long-Range Information Propagation. Our next focus is on the information propagation capability of IGNS, a critical aspect for physical simulations, since reliable modeling hinges on the ability to capture long-range interactions in both time and space.

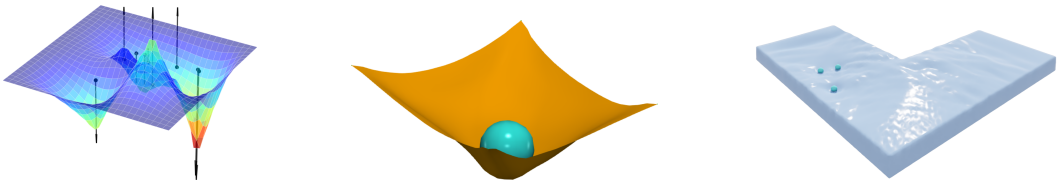
In the absence of damping and external forces, our IGNS reduces to a pure Hamiltonian systems and thus adheres to the corresponding conservation laws (Galimberti et al., 2021; 2023; Heilig et al., 2025). Consequently, the vector field induced by the conservative IGNS is divergence-free, ensuring information preservation throughout propagation. In other words, under this setting, the system dynamics can be interpreted as purely rotational, with no information dissipation.

Following Arroyo et al. (2025), which demonstrates that the inability to perform effective propagation is closely tied to the vanishing gradient problem, we investigate the behavior of IGNS with the Hamiltonian proposed in Eq. (7) through a sensitivity analysis. Thus, we study the quantity $\|\partial \boldsymbol{x}(t)/\partial \boldsymbol{x}(s)\|$, as formalized in the following theorem.

Theorem 2. *If $\boldsymbol{x}(t)$ is the solution to the ODE $\dot{\boldsymbol{x}}_i = \boldsymbol{J}\nabla_{\boldsymbol{x}_i} H_\theta(t, \boldsymbol{X})$, that is, the Hamiltonian dynamic defined in Eq.* (6) *without dampening and external forcing, then $\left\| \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{x}(s)} \right\| \geq 1$ for any $0 \leq s \leq t$.*

The proof is discussed in Section D.2. This theorem shows that the sensitivity matrix is lower bounded by a constant, ensuring non-vanishing gradients and thereby enabling effective long-range information propagation; unlike GCNs (Kipf & Welling, 2017), which suffer from an exponential decay in propagation rate over time (Gravina et al., 2025). This non-dissipative behavior also enables IGNS to propagate and preserve a broader context across nodes during the warmup phase (see Section 3.2).

Although Hamiltonian dynamics provide theoretical guarantees for propagation behavior, additional terms, such as dissipation and external forcing, may counteract this effect. To this end, we recall the discussion in Rusch et al. (2022); there, they showed that the oscillatory behavior of the system



(a) Plate Deformation (b) Sphere Cloth (c) Wave Balls

Figure 2: Three novel tasks. *(a)* **Plate Deformation**: a flat plate deformed subject to varying numbers and magnitudes of point forces. *(b)* **Sphere Cloth**: a cloth mesh impacted by a falling sphere, producing elastic deformation with contact dynamics. *(c)* **Wave Balls**: surface waves on water generated by three balls moving linearly from different initial positions.

allows the gradients not to decay exponentially even in the presence of these additional terms. More specifically, in the limit of small Δt, the gradient of the loss function with respect to any learnable weight parameter in the forcing term is independent of the number of updates. Additionally, as long as the dependence between Δt and the number of iterations N is polynomial $\Delta t \sim N^{-s}$, gradients do not decay exponentially in N, alleviating the vanishing gradient problem.

5 Experiments

Datasets. We evaluate our methods on six benchmark datasets in two categories: Lagrangian systems (*Plate Deformation*, *Impact Plate*, *Sphere Cloth*) and Eulerian systems (*Wave Balls*, *Cylinder Flow*, *Kuramoto-Sivashinsky*). First, we introduce ***Plate Deformation*** (Plate Def.) task to evaluate long-range propagation effects on spatial domain, ignoring the evolution over time. Here, given only force positions and magnitudes, the model must learn to predict the final deformed state of an initially flat sheet. Next, the *Impact Plate* task, proposed in HCMT (Yu et al., 2024), tests the model's ability to capture rapid stress propagation across a large mesh over time. We then introduce two additional new tasks, ***Sphere Cloth*** and ***Wave Balls***, both designed to evaluate oscillatory behavior under external forcing. Finally, we include two complex dynamics tasks, *Cylinder Flow* (Pfaff et al., 2021) and the fourth-order *Kuramoto-Sivashinsky* (KS) equation, each slightly modified to probe error accumulation under autoregressive training. Fig. 2 highlights the *three novel tasks* we introduced. Full dataset specifications and simulator details are in Section E (Table 3).

Evaluating Methods. We evaluate a broad set of methods, grouped into standard graph-convolution (Graph Conv.), effective propagation (Eff. Propagation), and our proposed port-Hamiltonian (Port-Ham.) models. **Graph Conv.:** Our main competitor, *MGN* (MeshGraphNets) (Pfaff et al., 2021), is a standard message passing GNN trained autoregressively with explicit Euler integration. For the remaining baselines, we use the multi-step loss (Eq. (10)). *MP-ODE* (Iakovlev et al., 2021; Lienen & Günnemann, 2022) applies neural ODE integration each step; here, we choose the RK4 integrator (instead of `dopri5`) to ensure a fair training budget compared to other baselines; additionally, we define the message function on nodal points instead of cell elements as in Lienen & Günnemann (2022). Next, *GCN+LN* is a graph convolutional network with LayerNorm, a common technique to mitigate oversmoothing in deep message passing (Luo et al., 2024). **Eff. Propagation**: *ADGN* (Gravina et al., 2023) adopted an anti-symmetric matrix in the graph convolution to avoid dissipative behavior, *GraphCON* (Rusch et al., 2022) is an oscillatory graph network, originally proposed to avoid oversmoothing effects. **Port-Ham. (Ours)**: *IGNS*$_{ti}$ (time independent) and *IGNS* (time varying) are port-Hamiltonian-based models that follow the state update in Eq. (6). For fairness, all baselines, except *MGN* which updates edge features at each message-passing step, use the same proposed geometric encoding (Section 3.3). Section F provides details on the choice of hyperparameters and training budget.

5.1 Results and Discussions

Table 1 summarizes the main results across all six tasks, reporting mean and standard deviation of test MSE over 4 random seeds. Overall, IGNS$_{\mathbf{ti}}$ and IGNS consistently outperform the main competitor MGN, as well as other baselines, across all tasks. Notably, GraphCON with the geometric encoding also performs strongly on static graph tasks (Plate Def., Cylinder Flow, KS) yet

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

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. To gain further insight, we provide an analysis of the relationship between GraphCON and our models in Section B. Furthermore, introducing time-variation in the weight matrices (IGNS) produces further gains, suggesting that flexible, time-dependent parameterizations better capture complex, nonstationary dynamics. We now analyze per-task behavior below.

Table 2: 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

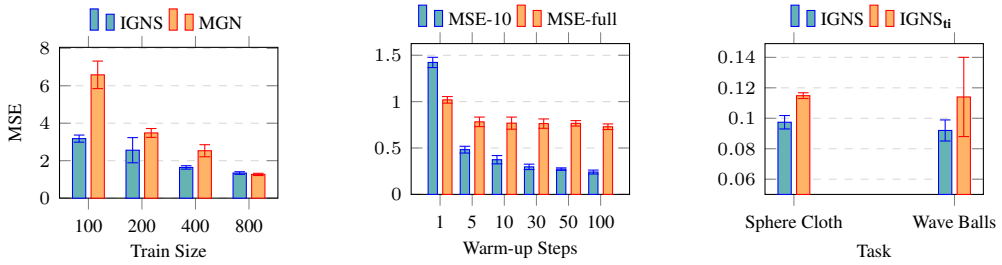
Method	MSE ($\times 10^{-3}$)
FNO-RNN	6.69 ± 0.46
MGN	10.76 ± 9.16
IGNS$_{\text{ti}}$	0.82 ± 0.09
IGNS	1.40 ± 0.25

(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 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 2 (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 G.

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 (Section B), is particularly well-suited for this type of wave dynamics.



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

Figure 3: 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 (SC) vs. Wave Balls (WB) with $T = 100$ (Wave Balls MSE $\times 100$ for visibility). All results are computed using 4 seeds.

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 (Pfaff et al., 2021), MGN performs better due to longer autonomous portions in the dataset; our causal-focused split reduces that advantage (Section E). Finally, for Kuramoto-Sivashinsky, the chaotic dynamics heavily penalize autoregressive MGN (resulting in large error), while GraphCON, IGNS$_{\mathbf{ti}}$/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) Li et al. (2021) often perform well on this type of dynamic, they underperform significantly compared to graph-based models, as shown in Table 2 (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 G.

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. 3a, 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. 3b 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. 10). **(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. 3c, 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.

6 Conclusion

In this work, we presented IGNS, a graph-based neural simulator that leverages port-Hamiltonian dynamics to improve long-range information propagation and reduce error accumulation in physical modelling problems. By integrating warmup initialization, geometric encoding, and multi-step training, IGNS shows consistent improvements over existing baselines across six tasks, including new benchmarks designed to evaluate long-range and complex dynamics. Our theoretical analysis explains that the Hamiltonian structure of IGNS helps preserve gradient flow over extended spatial and temporal horizons, while the port-Hamiltonian extension allows the model to approximate a wider range of dynamics via universality results. These findings, along with the empirical evi-

dence, indicate that IGNS is a promising approach for more accurate and stable neural simulation of complex physical systems.

Limitations and Future Work. Our method currently operates in an open-loop, causal setting: actions are specified at the start, and the model predicts forward. However, many applications require closed-loop control, where the model adapts based on feedback. Extending IGNS in this direction is therefore a natural next step. Moreover, the ability of IGNS to propagate information over long ranges makes it possible to specify actions in abstract forms (e.g., force nodes in Plate Deformation). This opens the door to inverse design (Allen et al., 2022) and control tasks (Hoang et al., 2025), where the goal is to identify optimal actions that achieve a desired outcome. Exploring these directions represents a promising path for future work.

REFERENCES

Kelsey R Allen, Tatiana Lopez-Guavara, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, Jessica B Hamrick, and Tobias Pfaff. Inverse design for fluid-structure interactions using graph network simulators. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=HaZuqj0Gvp2.

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.

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

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.

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

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.

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.

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, Aug 2019. ISSN 1432-0924. doi: 10.1007/s00466-019-01740-0. URL https://doi.org/10.1007/s00466-019-01740-0.

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.

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

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

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 (eds.), *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.

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.

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.

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

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, pp. 3836–3849. Curran Associates, Inc., 2021.

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

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 (eds.), *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pp. 275–286. PMLR, 07 – 08 June 2021. URL https://proceedings.mlr.press/v144/galimberti21a.html.

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.

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.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *ICML'17*, pp. 1263–1272. JMLR.org, 2017.

Alessio Gravina and Davide Bacciu. Deep Learning for Dynamic Graphs: Models and Benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2024. doi: 10.1109/TNNLS.2024.3379735.

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.

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.

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

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

Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

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

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.

Tai Hoang, Huy Le, Philipp Becker, Vien Anh Ngo, and Gerhard Neumann. Geometry-aware RL for manipulation of varying shapes and deformable objects. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7BLXhmWvwF.

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.

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&.

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.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46786–46806. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/923285deb805c3e14e1aeebc9854d644-Paper-Conference.pdf.

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)*, pp. 9266–9275, 2019a. doi: 10.1109/ICCV.2019.00936.

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*, 2019b. URL https://openreview.net/forum?id=rJgbSn09Ym.

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.

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.

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.

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.

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).

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.

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.

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*, pp. 35:1–35:23. PMLR, 27–30 Nov 2024.

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.

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

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

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.

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

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*, pp. 642–660. PMLR, 2023b.

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

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.

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

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

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.

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.

Tobias Würth, Niklas Freymuth, Gerhard Neumann, and Luise Kärger. Diffusion-based hierarchical graph neural networks for simulating nonlinear solid mechanics. *arXiv preprint arXiv:2506.06045*, 2025.

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.

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.

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.

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 (Battaglia et al., 2018; Pfaff et al., 2021). Notable examples include modeling particle interactions (Sanchez-Gonzalez et al., 2020; Li et al., 2019b; Sanchez-Gonzalez et al., 2020), fluid dynamics (Pfaff et al., 2021; Yu et al., 2025), and deformable materials (Yu et al., 2024; Linkerhägner et al., 2023). 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 (Kipf & Welling, 2017) or attention-based methods (Veličković et al., 2018), or more generally, message-passing framework (Battaglia et al., 2018). 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 (Chen et al., 2018), 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 (Iakovlev et al., 2021; Lienen & Günnemann, 2022). 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 (Shi et al., 2023a; Rusch et al., 2023; Roth & Liebig, 2024). Stacking many GNN layers to capture long-range dependencies often leads to issues such as over-smoothing (Cai & Wang, 2020; Rusch et al., 2023), when node states become indistinguishable as the number of layers increases. To address this, methods like residual connections (Li et al., 2019a), normalization techniques (Luo et al., 2024), and advanced architectures (Rusch et al., 2022; Eliasof et al., 2021) have been proposed. Another issue that prevent effective long-range propagation is the over-squashing phenomenon (Alon & Yahav, 2021; Topping et al., 2022; Di Giovanni et al., 2023), 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 (Topping et al., 2022; Gasteiger et al., 2019; Barbero et al., 2024) modify the graph topology to facilitate communication. In physical system simulation, hierarchical and rewiring methods, including HCMT (Yu et al., 2024) and PI-ORF (Yu et al., 2025), 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, Arroyo et al. (2025) demonstrated that both over-smoothing and over-squashing problems are closely linked to the problem of vanishing gradients in message passing, and works like Eliasof et al. (2021); Gravina et al. (2023; 2025); Heilig et al. (2025); Arroyo et al. (2025) 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 (Chen et al., 2018), 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 (Arnol'd, 2013). Notable examples include Hamiltonian Neural Networks (HNNs) (Greydanus et al., 2019) and Symplectic ODE-Net (Zhong et al., 2020), 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 (Galimberti et al., 2023; Lanthaler et al., 2023; Rusch & Rus, 2025), and it has inspired new architectures for enhancing long-range propagation in GNNs (Heilig et al., 2025) as well as long-horizon forecasting in state-space-model (Rusch & Rus, 2025). 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 DERIVATIONS

In this section, we establish the connection between the port-Hamiltonian formalism in Eq. (6) and the PDE wave equation defined on a space–time domain. In particular, we show that both can be expressed in the form of a mass–spring–damper system. These two links make clear that our port–Hamiltonian formalism induces a second–order inductive bias, which can be interpreted as a weak form of an underlying wave equation.

B.1 IGNS AS MASS-SPRING-DAMPER ODE

To begin with, we first consider the following Hamiltonian function:

$$H(t, \boldsymbol{q}, \boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T \boldsymbol{M}^{-1}\boldsymbol{p} + \frac{1}{2}\boldsymbol{q}^T \boldsymbol{K}\boldsymbol{q}$$

where we define the generalized coordinates $\boldsymbol{q}$ and momenta $\boldsymbol{p} = \boldsymbol{M}\dot{\boldsymbol{q}}$, and let $\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{p}]^T$. In fact, this definition is consistent with the Hamiltonian defined in Eq. (7) if we set $\tilde{\sigma}(x) = \frac{1}{2}x^2$ and $\boldsymbol{W} = \mathrm{diag}\left([\boldsymbol{K}^{1/2}, \boldsymbol{M}^{-1/2}]\right)$ and $\boldsymbol{V} = \boldsymbol{0}$. The update of $\boldsymbol{q}$ and $\boldsymbol{p}$ is then given by:

$$\dot{\boldsymbol{q}}(t) = \nabla_{\boldsymbol{p}} H(t, \boldsymbol{q}, \boldsymbol{p}) = \boldsymbol{M}^{-1}\boldsymbol{p}(t)$$
$$\dot{\boldsymbol{p}}(t) = -\nabla_{\boldsymbol{q}} H(t, \boldsymbol{q}, \boldsymbol{p}) = -\boldsymbol{K}\boldsymbol{q}(t).$$

We now eliminate $\boldsymbol{p}(t)$ by substituting $\boldsymbol{p}(t) = \boldsymbol{M}\,\dot{\boldsymbol{q}}(t)$, and add the damping term $\boldsymbol{D}\,\dot{\boldsymbol{q}}(t)$ and the external force $\boldsymbol{r}(t, \boldsymbol{q})$, yielding:

$$\boldsymbol{M}\,\ddot{\boldsymbol{q}}(t) + \boldsymbol{D}\,\dot{\boldsymbol{q}}(t) + \boldsymbol{K}\,\boldsymbol{q}(t) = \boldsymbol{r}(t, \boldsymbol{q}). \tag{11}$$

This recovers the mass-spring-damper system.

For a more general case, consider a Hamiltonian $H(t, \boldsymbol{q}, \boldsymbol{p}) \in \mathcal{C}^2$ in all arguments. Taking the time derivative of $\dot{\boldsymbol{q}} = \nabla_{\boldsymbol{p}} H(t, \boldsymbol{q}, \boldsymbol{p})$ gives

$$\ddot{\boldsymbol{q}} = \nabla^2_{\boldsymbol{pp}} H\,\dot{\boldsymbol{p}} + \nabla^2_{\boldsymbol{pq}} H\,\dot{\boldsymbol{q}} + \partial_t \nabla_{\boldsymbol{p}} H.$$

Assuming $\nabla^2_{\boldsymbol{pp}} H$ exists and is invertible (which holds for certain activation functions, e.g. $\sigma(\cdot) = \tanh(\cdot)$ and regularity conditions for invertibility), define $\boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p}) := \left(\nabla^2_{\boldsymbol{pp}} H(t, \boldsymbol{q}, \boldsymbol{p})\right)^{-1}$. Solving for $\dot{\boldsymbol{p}}$,

$$\dot{\boldsymbol{p}} = \boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p})\,\left[\ddot{\boldsymbol{q}} - \nabla^2_{\boldsymbol{pq}} H\,\dot{\boldsymbol{q}} - \partial_t \nabla_{\boldsymbol{p}} H\right].$$

Substitute this into the second Hamiltonian equation $\dot{\boldsymbol{p}} = -\nabla_{\boldsymbol{q}} H - \boldsymbol{D}\,\dot{\boldsymbol{q}} + \boldsymbol{r}(t, \boldsymbol{q})$ and rearrange, we arrive at the general second-order form:

$$\underbrace{\boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p})\,\ddot{\boldsymbol{q}}}_{\text{mass}} + \underbrace{\left[\boldsymbol{D} - \boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p})\,\nabla^2_{\boldsymbol{pq}} H\right]\dot{\boldsymbol{q}}}_{\text{damping}} + \underbrace{\nabla_{\boldsymbol{q}} H}_{\text{stiffness}} = \underbrace{\boldsymbol{r}(t, \boldsymbol{q}) + \boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p})\partial_t \nabla_{\boldsymbol{p}} H}_{\text{external force \& residual}}. \tag{12}$$

This analysis demonstrates that, for general Hamiltonians, the resulting second-order dynamics exhibit state- and time-dependent mass, damping, and stiffness terms. In particular, for separable Hamiltonians, we have $\nabla^2_{\boldsymbol{pq}} H = 0$, and for time-invariant Hamiltonians, $\partial_t \nabla_{\boldsymbol{p}} H = 0$. In these cases, the dynamics reduce to the form of Eq. (11), with $\boldsymbol{M}(t, \boldsymbol{q}, \boldsymbol{p}) = (\nabla^2_{\boldsymbol{pp}} H)^{-1}$ and $\boldsymbol{K}(t, \boldsymbol{q}, \boldsymbol{p})\boldsymbol{q}(t) = \nabla_{\boldsymbol{q}} H(t, \boldsymbol{q}, \boldsymbol{p})$.

Comparison with GraphCON. We can further observe that the baseline GraphCON (Rusch et al., 2022) is in fact a simpler instantiation of Eq. (11). Its update takes the form

$$\ddot{\boldsymbol{q}}(t) + \alpha\dot{\boldsymbol{q}}(t) + \gamma\boldsymbol{q}(t) = \sigma(\boldsymbol{R}_\theta(t, \boldsymbol{q})),$$

where $\boldsymbol{R}_\theta$ is implemented as a standard graph convolution or attention operator. This corresponds directly to a mass–spring–damper system with $\boldsymbol{M} = \boldsymbol{I}$, $\boldsymbol{D} = \alpha\boldsymbol{I}$, $\boldsymbol{K} = \gamma\boldsymbol{I}$, and external forcing $\boldsymbol{r}(\cdot; G) = \sigma(\boldsymbol{R}_\theta(t, \boldsymbol{q}))$. Notably, the mass matrix $\boldsymbol{M}$ is the identity, meaning that coupling between nodes occurs only through the external forcing term $\boldsymbol{r}$, whereas in our formulation coupling also appears on the left-hand side through $\boldsymbol{M}$, $\boldsymbol{D}$, and $\boldsymbol{K}$.

B.2 MASS–SPRING–DAMPER ODE AS A SECOND–ORDER PDE

In this subsection, we show how a second-order PDE with damping and forcing reduces to a finite-dimensional mass-spring-damper ODE via Galerkin projection. Interested readers are referred to Igel (2016) for further details.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and consider the following damped, forced wave equation

$$\partial_{tt} u + \mathcal{B}\,\partial_t u + \mathcal{L}\,u = g(\boldsymbol{x}, t), \qquad (\boldsymbol{x}, t) \in \Omega \times (0, T], \tag{13}$$

where the solution is $u : \Omega \times (0, T] \to \mathbb{R}$, i.e., a scalar field. We set $\mathcal{L} = -c^2\Delta$ with constant wave speed $c > 0$, where Δ is the spatial Laplacian, and define $\mathcal{B}$ as a damping operator (e.g., αI) acting on the time-derivative term $\partial_t u$.

Let $V \subset H^1(\Omega)$ be the test space. To obtain the weak form of Eq. (13), we take a test function $v \in V$, multiply the strong equation by v, and integrate over Ω:

$$\int_\Omega \partial_{tt} u\, v\, d\boldsymbol{x} \;+\; \int_\Omega \mathcal{B}\,\partial_t u\, v\, d\boldsymbol{x} \;-\; c^2 \int_\Omega \Delta u\, v\, d\boldsymbol{x} = \int_\Omega g(\cdot, t)\, v\, d\boldsymbol{x}.$$

The first two terms already appear in weak form. For the Laplacian term, we apply integration by parts and assume that $u(t)$ vanishes at the boundary. This gives the following bilinear form:

$$a(u, v) := -c^2 \int_\Omega \Delta u\, v\, d\boldsymbol{x} = c^2 \int_\Omega \nabla u \cdot \nabla v\, d\boldsymbol{x}.$$

Altogether, the weak form reads: for all $v \in V$,

$$\langle \partial_{tt} u, v\rangle + \langle \mathcal{B}\,\partial_t u, v\rangle + a(u, v) = \langle g(\cdot, t), v\rangle, \tag{14}$$

where $\langle f, g\rangle = \int_\Omega f(\boldsymbol{x})\, g(\boldsymbol{x})\, d\boldsymbol{x}$ denotes the $L^2(\Omega)$ inner product.

Choose a finite-dimensional subspace $V_h = \operatorname{span}\{\phi_1, \ldots, \phi_k\} \subset V$ and rewrite the solution as

$$u_h(\boldsymbol{x}, t) = \sum_{j=1}^{k} q_j(t)\phi_j(\boldsymbol{x}), \tag{15}$$

where $q_j(t)$ are time-dependent coefficients to be determined. Since Eq. (14) holds for all $v = \phi_i$, $i = 1, \ldots, k$, rewriting it using the Eq. (15) yields

$$\sum_{j=1}^{k} \langle \phi_j, \phi_i\rangle\, \ddot{q}_j(t) + \langle \mathcal{B}\,\phi_j, \phi_i\rangle\, \dot{q}_j(t) + a(\phi_j, \phi_i)\, q_j(t) = \langle g(\cdot, t), \phi_i\rangle. \tag{16}$$

To show the equivalent mass-spring-damper form, we now define the following matrices and vectors:

$$\begin{aligned}
\boldsymbol{M}_{ij} &:= \langle \phi_j, \phi_i\rangle, && \text{(mass matrix)}\\
\boldsymbol{D}_{ij} &:= \langle \mathcal{B}\,\phi_j, \phi_i\rangle, && \text{(damping matrix)}\\
\boldsymbol{K}_{ij} &:= a(\phi_j, \phi_i), && \text{(stiffness matrix)}\\
\boldsymbol{r}_i(\cdot, t) &:= \langle g(\cdot, t), \phi_i\rangle, && \text{(external forcing)}
\end{aligned}$$

with vector $\boldsymbol{q} = [q_j]_{j=1}^k$, thus Eq. (16) becomes

$$\boldsymbol{M}\,\ddot{\boldsymbol{q}}(t) + \boldsymbol{D}\,\dot{\boldsymbol{q}}(t) + \boldsymbol{K}\,\boldsymbol{q}(t) = \boldsymbol{r}(\cdot, t).$$

Interpretation. Because our latent dynamics (Eqs. (6) and (12)) has the same form as the system above, IGNS can be seen as learning *effective* mass, damping, and stiffness operators, consistent with a weak-form second-order PDE (in a P1 nodal basis, coefficients equal nodal values, i.e., $u(\boldsymbol{x}_j, t) = \boldsymbol{q}_j(t)$). This method-of-lines view is related to Finite Element Networks (FEN) Lienen & Günnemann (2022), which were the first to introduce the connection between message passing and the finite element method. However, while FEN focuses on *diffusion/heat* dynamics (parabolic, first order in time), here we model a more general *wave* equation with forcing and damping, yielding a more general second-order hyperbolic PDE.

C 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. (7) 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

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

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.

D PROOFS OF THE THEORETICAL STATEMENTS AND FURTHER DISCUSSION

In this section, we prove the theoretical statements in this paper together with a more in-depth discussion.

D.1 NEURAL OSCILLATORS AND UNIVERSALITY

We start this section by recalling the definition of neural oscillators and their universality property (Lanthaler et al., 2023). The general setting involves learning an operator $\Phi : \boldsymbol{u} \to \Phi(\boldsymbol{u})$, that maps a time-dependent input signal $\boldsymbol{u}(t)$ to an output function $\Phi(\boldsymbol{u})(t) \in \mathbb{R}^q$. Here, Φ is defined as a continuous operator (w.r.t. the L^∞ norm) that maps the space

$$C_0([0,T];\mathbb{R}^p) = \{\boldsymbol{u} : [0,T] \to \mathbb{R}^p | t \mapsto u(t) \text{ is continuous and } \boldsymbol{u}(0) = 0\} \tag{17}$$

into itself. The assumption $\boldsymbol{u}(0) = 0$ is not restrictive, as the main paper showed that the proposed neural oscillators can operate even in the case $\boldsymbol{u}(0)$ is non-zero. Here, by adding an arbitrarily small time interval to "warm-up", the oscillator starting from the rest state can synchronize with this non-zero input signal. The only remaining assumption is that Φ needs to be *causal*, in the sense that the value of $\Phi(u)(t)$ does not depend on future values $\{u(\tau)|\tau > t\}$, which matches our setup considered in this paper well. Formally speaking, Φ is causal if giving two input signals and assuming that $u|_{[0,t]} = v|_{[0,t]}$, then $\Phi(u)(t) = \Phi(v)(t)$.

Then, we recall the general form of the neural oscillator, given in the following

$$\begin{cases} \ddot{\boldsymbol{y}}(t) = \sigma\left(\boldsymbol{A}\boldsymbol{y}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{c}\right), \\ \boldsymbol{y}(0) = \dot{\boldsymbol{y}}(0) = \boldsymbol{0}, \\ \boldsymbol{z}(t) = \boldsymbol{D}\boldsymbol{y}(t) + \boldsymbol{e}, \end{cases} \tag{18}$$

where $\boldsymbol{y} \in \mathbb{R}^d$ and σ is a $C^\infty(\mathbb{R})$ activation function with $\sigma(0) = 0$ and $\sigma'(0) = 1$, e.g., $\tanh$ or $\sin$. Here, Eq. (18) defines an input-output mapping $\boldsymbol{u}(t) \mapsto \boldsymbol{z}(t) \in \mathbb{R}^q$ as a solution of a second-order dynamics. Furthermore, Lanthaler et al. (2023) (Section 2) also considers another form of neural oscillator, which is given by

$$\begin{cases} \ddot{\boldsymbol{y}}(t) = \sigma\left(\boldsymbol{A}\boldsymbol{y}(t) + \boldsymbol{c}\right) + \boldsymbol{r}(t), \\ \boldsymbol{y}(0) = \dot{\boldsymbol{y}}(0) = \boldsymbol{0}, \\ \boldsymbol{z}(t) = \boldsymbol{D}\boldsymbol{y}(t) + \boldsymbol{e}. \end{cases} \tag{19}$$

In this case, the input is provided via $\boldsymbol{r}(t)$, an external forcing term which in their setting is a linear transformation of the inputs. For the universality result presented shortly below, we will consider the neural oscillator in Eq. (19), which is the closest setting to ours. Here, the universality result remains the same, as the original proof does not change if the forcing is inside or outside the activation (see the discussion at the end of Section 2 in Lanthaler et al. (2023)).

The main result of Lanthaler et al. (2023) can be summarized in the following statement:

Theorem 3 (Theorem 3.1 of Lanthaler et al. (2023))**.** *Let $\Phi : C_0([0,T];\mathbb{R}^p) \to \Phi : C_0([0,T];\mathbb{R}^p)$ be a causal and continuous operator and let $K \subset C_0([0,T];\mathbb{R}^p)$ be compact. Then, for any $\epsilon > 0$, there exists a latent dimension d, weights $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}$, and biases $\boldsymbol{c}, \boldsymbol{e}$ such that the output of the multi-layer neural oscillator satisfies:*

$$\sup_{t\in[0,T]} |\Phi(u)(t) - z(t)| \leq \epsilon, \quad \forall u \in K. \tag{20}$$

While this theorem is stated for maps between time-dependent signals $\boldsymbol{u}(t)$ and $\boldsymbol{z}(t)$, Lanthaler et al. (2023) provides an additional result for continuous functions $F : \mathbb{R}^p \to \mathbb{R}^q$, i.e., neural oscillators can learn to approximate to arbitrary precision any such map F between *vector spaces*. The full discussion is provided in Appendix A of Lanthaler et al. (2023).

To show that our model is universal in either sense, we need to cast our model, defined by Eq. (6) and Eq. (7), into the definition in Eq. (19). First, we provide a proof of the universality of our model, cast into the neural oscillator form, and then we provide an interpretation of our results.

Theorem (Universality of IGNS, Theorem 1 of Section 4)**.** *Let Ψ_θ be the functional that maps the initial condition $\boldsymbol{x}_0 = [\boldsymbol{q}_0, \boldsymbol{p}_0]$ to the solution at time τ to the ODE defined Eq.* (6) *and Eq.* (7)*. In other words, Ψ_θ represents IGNS. Then, for any $F : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, which represents the evolution of the system after time τ, there exist a set of parameters θ such that $\|\Psi_\theta(\boldsymbol{x}_0) - F(\boldsymbol{x}_0)\| \leq \epsilon$.*

Proof. Our goal for the proof is to reconcile our definition of the port-Hamiltonian system in Eq. (6) with the definition of the oscillatory model in Eq. (19). We start from the derivation of IGNS as a mass-spring-damper system in Eq. (12). Here, we consider the separable Hamiltonian as in Eq. (7), hence $\nabla^2_{\boldsymbol{pq}} H = 0$. Without loss of generality, we consider $\boldsymbol{D} = 0$. Then, Eq. (12) can be written as

$$\ddot{\boldsymbol{q}} = -\boldsymbol{M}^{-1} \left(\nabla_{\boldsymbol{q}} H + \boldsymbol{r}(t, \boldsymbol{q})\right) + \partial_t \nabla_{\boldsymbol{p}} H. \tag{21}$$

For this proof, we consider the time-invariant case, while the general case is very similar, as $\partial_t \nabla_{\boldsymbol{p}} H$ plays the same role as the external forcing $\boldsymbol{r}(t, \boldsymbol{q})$. By using the definition of the Hamiltonian in Eq. (7), we rewrite Eq. (21) for each node as

$$\ddot{\boldsymbol{q}}_i = -\boldsymbol{M}^{-1} \left(\boldsymbol{W}_{\boldsymbol{q}} \sigma \left(\boldsymbol{W}_{\boldsymbol{q}} \boldsymbol{q}_i + \sum_{j\in\mathcal{N}(i)} \boldsymbol{V}_{\boldsymbol{q}} \boldsymbol{q}_j + \boldsymbol{b} \right) + \boldsymbol{r}(t, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q})_i \right) \tag{22}$$

where $\boldsymbol{g}(\boldsymbol{q})_i = \sum_{j\in\mathcal{N}(i)} \boldsymbol{V}_{\boldsymbol{q}} \sigma(\boldsymbol{W}_{\boldsymbol{q}} \boldsymbol{q}_j + \sum_{k\in\mathcal{N}(j)} \boldsymbol{V}_{\boldsymbol{q}} \boldsymbol{q}_k + \boldsymbol{b})$ accounts for the terms in which node i is a neighbor of node j (there exists a $k \in \mathcal{N}(j)$ for which $k = i$). Stacking all nodes, we arrive at the form

$$\ddot{\boldsymbol{q}} = -\boldsymbol{M}^{-1} \left(\tilde{\boldsymbol{W}}_{\boldsymbol{q}} \sigma \left(\tilde{\boldsymbol{W}}_{\boldsymbol{q}} \boldsymbol{q} + \tilde{\boldsymbol{b}} \right) + \tilde{\boldsymbol{r}}(t, \boldsymbol{q}) \right), \tag{23}$$

where $\tilde{\boldsymbol{b}} = [\boldsymbol{b}, \ldots, \boldsymbol{b}]^\top$ contains one copy of $\boldsymbol{b}$ for each node and $\tilde{\boldsymbol{W}}$ is a block matrix, where each diagonal block has a copy of $\boldsymbol{W}$ and block i, j has a copy of $\boldsymbol{V}$ iff $j \in \mathcal{N}(i)$. In fact, $\tilde{\boldsymbol{W}}$ absorbs the effects of node aggregation into one bigger block matrix.

Since the definition from Lanthaler et al. (2023) has a null initial condition, we adopt a trick with the forcing terms to reconcile the two forms finally. Let $\bar{\boldsymbol{q}}$ be an auxiliary function, solution to $\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q}, \boldsymbol{p})\tilde{\boldsymbol{r}}(t, \boldsymbol{q})$, with initial conditions $\boldsymbol{q}_0$ and $\dot{\boldsymbol{q}}_0 = \boldsymbol{M}(\boldsymbol{q}_0, \boldsymbol{p}_0)\boldsymbol{p}_0$. Then, the function $\boldsymbol{y} = \boldsymbol{q} - \bar{\boldsymbol{q}}$ evolves through the following equations

$$\begin{aligned} \ddot{\boldsymbol{y}} &= -\boldsymbol{M}^{-1}(\boldsymbol{y} + \bar{\boldsymbol{q}}, \dot{\boldsymbol{y}} + \bar{\boldsymbol{p}}) \tilde{\boldsymbol{W}}_{\boldsymbol{q}} \sigma \left(\tilde{\boldsymbol{W}}_{\boldsymbol{q}} \boldsymbol{y} + \tilde{\boldsymbol{W}}_{\boldsymbol{q}} \bar{\boldsymbol{q}} + \tilde{\boldsymbol{b}} \right) \\ &\quad - \boldsymbol{M}^{-1}(\boldsymbol{y} + \bar{\boldsymbol{q}}, \dot{\boldsymbol{y}} + \bar{\boldsymbol{p}}) \tilde{\boldsymbol{r}}(t, \boldsymbol{y} + \bar{\boldsymbol{q}}) - \boldsymbol{M}^{-1}(\bar{\boldsymbol{q}}, \bar{\boldsymbol{p}}) \tilde{\boldsymbol{r}}(t, \bar{\boldsymbol{q}}), \\ \boldsymbol{y}(0) &= \boldsymbol{0}, \\ \dot{\boldsymbol{y}}(0) &= \boldsymbol{0}. \end{aligned} \tag{24}$$

Here, we can see that Eq. (24) is indeed in a similar form as Eq. (19). The only difference is in the additional matrix $\boldsymbol{M}^{-1} = \nabla^2_{pp} H$, which depends only on $\boldsymbol{W_p}$ (since H is separable in $\boldsymbol{p}, \boldsymbol{q}$), and makes Eq. (24) more general than Eq. (19). Finally, we define $\boldsymbol{z}(t)$ as either $\boldsymbol{z}(t) = \boldsymbol{y}(t)$ or as obtained via a decoder $\boldsymbol{z}(t) = \text{dec}(\boldsymbol{y}(t))$. Hence, IGNS is a universal neural oscillator by Theorem 3. To conclude our proof, we need two extra observations. First, since $\boldsymbol{p}(t) = \boldsymbol{M}\dot{\boldsymbol{q}}(t)$, IGNS is universal also in terms of $\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{p}]^T$. Second, this result shows that IGNS is an operator that can convert an input signal $\boldsymbol{r}(t, \boldsymbol{q})$ into any possible evolution, and it can be further extended to any map between the space of our interest $\mathbb{R}^{n\times d}$ to itself. For this, it is sufficient to consider the proof given in Appendix A of Lanthaler et al. (2023) and recall the definition of $\bar{\boldsymbol{q}}$, which adds the information on the initial condition through $\tilde{\boldsymbol{r}}(t, \bar{\boldsymbol{q}})$. □

Remark 1. *This proof also shows that IGNS can learn any possible transformation between the forcing terms and the dynamics of the physical system. Specifically, the Hamiltonian in Eq.* (7) *can learn any transformation between continuous functions in* $C([0, T], \mathbb{R}^{n\times d})$.

There is an interesting interpretation of this result, which links the definition of IGNS in Eq. (6) and the one of neural oscillators. In particular, looking at the formulation in Eq. (24), we see that all the terms that are outside the activation function play the same role as the input signal $\boldsymbol{r}(t)$, defined in Eq. (19). In our case, these terms contain the geometrical information and the external or driving forces acting on the system, as well as the initial condition. Hence, IGNS learns to transform these inputs into the desired dynamics. In this sense, the Hamiltonian learns how to propagate through the mesh the information from the initial conditions, geometry, and external forcing.

Another interpretation comes directly from Eq. (22) by looking at the state of each node $\boldsymbol{q}_i$. In particular, we can interpret the state of neighboring nodes $\boldsymbol{q}_j$, with $j \in \mathcal{N}(i)$, as the input function $\boldsymbol{u}(t)$. Hence, IGNS learns how each node should behave in the simulation as a consequence of the influence of the neighboring nodes, dissipation, and forcing.

D.2 Vanishing gradients and Long-range Interactions

The radius of propagation exploited by the GNN has a large influence on its performance (Li et al., 2019a; Cai & Wang, 2020; Alon & Yahav, 2021; Rusch et al., 2023; Di Giovanni et al., 2023; Roth & Liebig, 2024), as it is often related to over-smoothing, which causes node representations collapse, and over-squashing, which causes information dissipation. Recently, it has been observed that these behaviors are strongly related to the sensitivity of the Jacobians $\partial \boldsymbol{x}^{(l+1)}/\partial \boldsymbol{x}^{(l)}$ (or $\partial \boldsymbol{x}_i^{(l)}/\partial \boldsymbol{x}_j^{(0)}$) of the GNN updates and their norms (Arroyo et al., 2025), as low sensitivity values can cause a contractive behavior that induces information loss. Hence, it is necessary to employ models that provide non-contractive updates to avoid these issues. Additionally, real-world simulations often require modeling long-range interactions, as physical information can propagate over long distances. In this discussion, we aim to show that all these problems can be solved or limited by adopting an oscillatory dynamic at the core of the model. Particularly, the Hamiltonian formalism provides further guarantees on vanishing gradients (Galimberti et al., 2021; 2023) and long-range interactions (Heilig et al., 2025), strengthening performance. From a dynamical system perspective, a key aspect of oscillatory systems lies in their purely imaginary eigenvalues:

Theorem 4 (Eigenvalues of an Oscillatory System)**.** *Consider a second-order system of the form* $\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = 0$*, where* $\boldsymbol{M}$ *and* $\boldsymbol{K}$ *are positive definite. Then, the eigenvalues of the system are pure imaginary.*

Proof. Since $\boldsymbol{M}$ is positive definite, it is invertible, and we can consider the following system instead:

$$\ddot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = 0. \tag{25}$$

We decompose this second-order system into a system of first-order ODEs:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{v}, \\ \dot{\boldsymbol{v}} &= -\boldsymbol{K}\boldsymbol{x}. \end{aligned} \tag{26}$$

Calling $\boldsymbol{y} = [\boldsymbol{x}, \boldsymbol{v}]^\top$, the system can be represented as

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \boldsymbol{y} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{y}. \tag{27}$$

That is, the evolution is determined by a matrix which is the product of an antisymmetric matrix and a positive definite one. Hence, as per Lemma 2 of Galimberti et al. (2021), the eigenvalues of the system are all imaginary. □

In terms of sensitivity matrices, a similar property has been derived for general model dynamics defined by $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t))$ (Heilig et al., 2025; Galimberti et al., 2023; 2021). The starting point is understanding how the sensitivity matrix $\partial \boldsymbol{x}(T)/\partial \boldsymbol{x}(T-t)$ evolves through time, measuring the effect of past states on the current one:

Lemma 1 (Lemma 1 of Galimberti et al. (2021))**.** *Let $\boldsymbol{x}$ be the solution to the first-order ODE $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t))$. Then, calling $\phi(T, T-t) = \frac{\partial \boldsymbol{x}(T)}{\partial \boldsymbol{x}(T-t)}$, it holds that*

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi(T, T-t) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(T-t)} \phi(T, T-t) \tag{28}$$

Hence, the sensitivity matrix and, most importantly, the vanishing gradient effect are strictly related to the eigenvalues of $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ which, for oscillatory models, are purely imaginary. In reality, Heilig et al. (2025) shows a much stronger property for $\text{IGNS}_{\textbf{ti}}$. In particular:

Theorem (Theorem 2 of Section 4 and Theorem 2.3 of Heilig et al. (2025))**.** *If $\boldsymbol{x}(t)$ is the solution to the ODE $\dot{\boldsymbol{x}}_i = J\nabla_{\boldsymbol{x}_i} H_\theta(t, \boldsymbol{X})$, that is, the Hamiltonian dynamic with Eq.* (6) *without dampening and external forcing, then $\left\| \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{x}(s)} \right\| \geq 1$ for any $0 \leq s \leq t$.*

This result represents a lower bound on the sensitivity matrix norm for any time t. This ensures that gradients never vanish, even over extended periods. We emphasize that not all oscillatory dynamics exhibit the same desirable properties as IGNS. For instance, while GraphCON (Rusch et al., 2022) models a mass-spring-damper system (see Section B.1) and alleviates some of the propagation issues presented. A Hamiltonian-like dynamic (Heilig et al., 2025), as adopted in our IGNS, provides stronger theoretical guarantees on long-range interactions and sensitivity matrices. These advantages also translate into improved empirical performance, as demonstrated in Section 5. GraphCON adds node coupling only in its forcing term, while employing diagonal matrices in the left-hand side of Eq. (11). See Section B.1 for additional details on the interpretation of these models as mass-spring-damper systems.

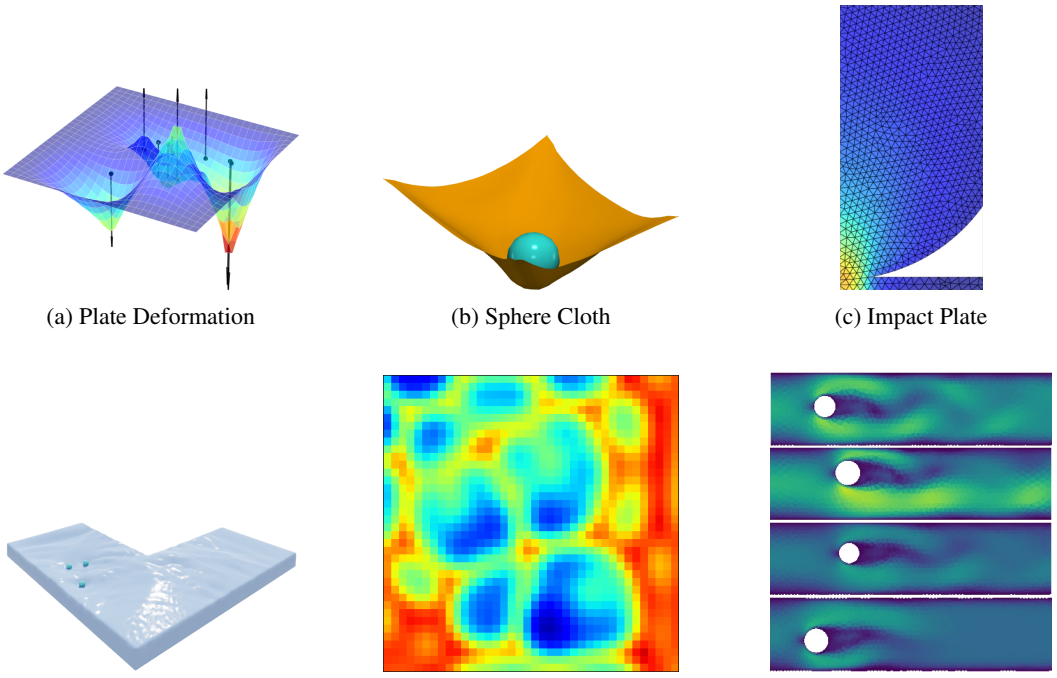
First-order models and the Heat Equation. Without inductive biases introduced in oscillatory and second-order models, there is no theoretical guarantee of preventing vanishing gradients. On the contrary, there are both practical (Li et al., 2019b) and theoretical (Arroyo et al., 2025) insights that show how GNNs are prone to these exponentially decaying gradients. Similarly, first-order differential-equation-inspired GNNs are based on the heat equation $\partial_t u - \Delta u = 0$, which leads again to exponentially decaying gradient and information propagation (Gravina et al., 2025), provable also through the lenses of over-smoothing (Eliasof et al., 2021).

E Datasets Descriptions

Table 3: 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 3 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



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

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

Figure 4: 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 4: 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	$\boldsymbol{n}_i,\ \rho_i$	$\rho_i,\ \dot{\rho}_i$

Section 5.1, we predict the whole trajectory with length $T = 48$. In addition, for *Cylinder Flow* and *Kuramoto-Sivashinsky*, we follow the suggestion from Lienen & Günnemann (2022) to only split the sub-trajectories into equal-size, non-overlapping windows. Compared to the *Cylinder Flow* used in the Lienen & Günnemann (2022) 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 4. 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. 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. 4:

Solid mechanics. This category includes Plate Deformation, Sphere Cloth, and Impact Plate (c.f. Yu et al. (2024)).

- *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 (Smith, 2009).
- *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 (Mittal et al., 2023). 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 (Yu et al., 2024) 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 (Pfaff et al., 2021), 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 (Macklin, 2022). 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 (Li et al., 2021), 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 (Zwicker, 2020). The KS equation in 2D is given by

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

F Hyperparameters and Training Time

Table 5 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

Table 5: 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[†]	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.

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 $\text{IGNS}_{\textbf{ti}}$. 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 6: (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
$\textbf{IGNS}_{\textbf{ti}}$	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

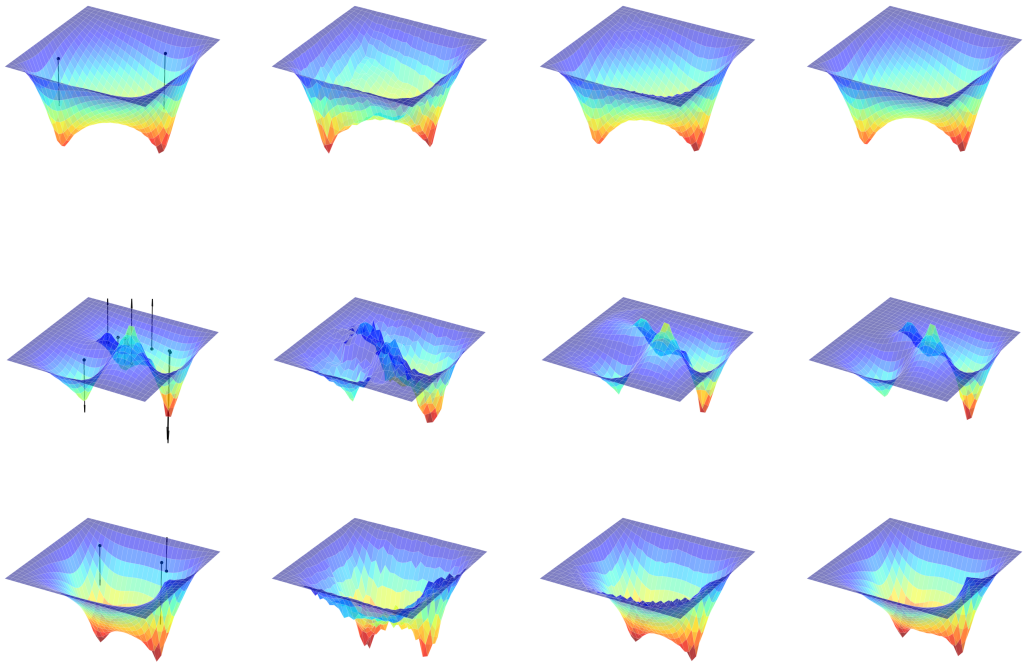
Table 6 (a) shows the training time allocated to each task. To ensure fairness, all methods were trained for the same wall-clock time on a single NVIDIA A100 GPU. For the default settings, the budgets were chosen based on validation curves, where extending training further did not yield noticeable improvements. For the extended version of Sphere Cloth and Wave Balls tasks, we instead fixed a hard budget to ensure sufficient training for fair comparison in the ablation studies, independent of validation plateaus. In Table 6 (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.

G Additional Results

Plate Deformation. Fig. 5 shows qualitative results on three test samples, comparing IGNS, MGN, and a rewired variant of MGN. In MGN-rewiring, we improve connectivity by directly link-

Table 7: Comparison of train and test losses, shown as mean ± std, on Plate Deformation. Best results are highlighted with **orange**.

Method	Test loss	Train loss
MGN (m=50)	1.27 ± 0.06	0.09 ± 0.01
MGN-rewiring (m=5)	3.72 ± 0.15	0.20 ± 0.01
IGNS	1.34 ± 0.07	0.62 ± 0.02



Ground Truth MGN-Rewiring MGN IGNS

Figure 5: Plate Deformation qualitative comparison on three test samples with predictions from *MGN-rewiring*, MGN and IGNS. The arrows indicate the forces (with magnitude propotion to its length) applied to the plate at specific postions.

ing the force nodes to the entire mesh (through every second node), a common technique to mitigate over-squashing. We also reduce the number of message passing steps from 50 (standard MGN) to 5 (MGN-rewiring). The results highlight clear differences: only IGNS produces smooth and physically consistent surfaces, while MGN predictions are less smooth despite lower error, and MGN-rewiring generates the roughest outputs. Quantitative results in Table 7 further confirm this trend, showing that both MGN variants severely overfit by relying too heavily on geometric features, while IGNS achieves a better trade-off between train and test performance.

Impact Plate. Table 8 reports quantitative results on the Impact Plate task, comparing IGNS and IGNS against the HCMT model from Yu et al. (2024) and MGN. Here, we take the results from Yu et al. (2024) for both HCMT and MGN, and report with the same metrics RMSE for position and stress. Both IGNS and $\text{IGNS}_{\textbf{ti}}$ significantly outperform the baselines in both position and stress prediction, with IGNS achieving the best performance overall. This demonstrates the effectiveness of our approach in capturing long-range interactions and complex dynamics in this challenging task.

Table 8: Impact Plate quantitative results. Reported are root mean squared errors (RMSE) for position and stress, shown as mean ± std ($\times 10^{-3}$). Best results are highlighted with **orange**.

Method	Position RMSE	Stress RMSE ($\times 10^3$)
HCMT	20.71 ± 0.57	14.74 ± 5.02
MGN	40.73 ± 2.94	35.87 ± 11.89
IGNS$_{\text{ti}}$	17.33 ± 4.43	4.60 ± 1.29
IGNS	15.59 ± 2.84	3.88 ± 0.39

Table 9: Sphere Cloth quantitative results. Reported are mean squared errors (MSE) for overall loss, position, and velocity, shown as mean ± std ($\times 10^{-3}$). Best results are highlighted with **orange**.

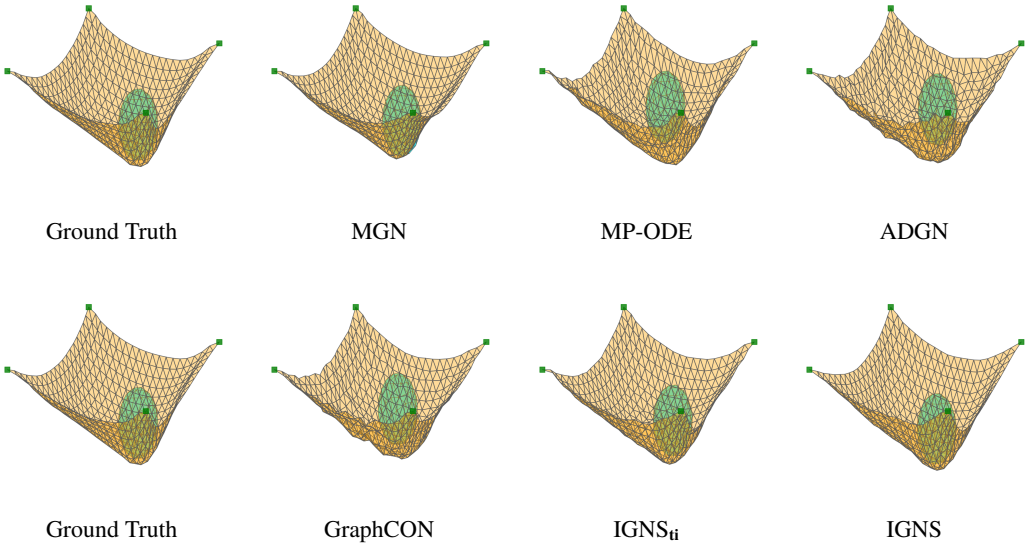
Model	Loss	Position	Velocity
GCN+LN	26.45 ± 0.23	4.14 ± 0.04	22.31 ± 0.20
MGN	32.07 ± 2.45	6.53 ± 0.76	25.54 ± 1.73
MP-ODE	25.75 ± 1.32	4.26 ± 0.26	21.49 ± 1.06
ADGN	30.99 ± 0.80	4.57 ± 0.25	26.42 ± 0.66
GraphCON	29.00 ± 0.44	4.19 ± 0.07	24.82 ± 0.40
IGNS$_{\text{ti}}$	27.99 ± 0.64	3.78 ± 0.10	24.21 ± 0.57
IGNS	23.46 ± 0.58	3.55 ± 0.12	19.92 ± 0.47

Sphere Cloth. In this task, the model must jointly predict the motion of a rigid sphere (represented only by its center position) and a deformable cloth, creating strong coupling dependencies. Figure 6 illustrates two representative test samples. In the top sample, MGN produces wrong predictions due to error accumulation, while in the bottom sample it manages to produce a more reasonable result. This highlights the instability of first order methods, in contrast to IGNS, which consistently yields results closest to the ground truth. The quantitative results in Table 9 confirm this trend: although IGNS$_{\text{ti}}$ achieves slightly lower position error, its surfaces remain rougher, while MGN is prone to error accumulation and overfitting. Other baselines (GraphCON, ADGN, and GCN) fail to generate smooth surfaces.

Wave Balls. Fig. 7 compares IGNS with the baselines MGN, GraphCON, and A-DGN. The absolute error maps show that IGNS is the only method producing consistently reasonable results. MGN is able to track the ball positions but overestimates wave amplitude, leading to large errors. GraphCON captures the wave pattern but mislocalizes the balls, causing errors near the sources. A-DGN fails to learn this task.

Kuramoto-Sivashinsky. To illustrate performance, we show predictions and absolute errors for FNO-RNN, MGN, GraphCON, and IGNS on two test samples (Figs. 8 and 9). The first sample reflects an early stage where local behavior dominates; the second shows a later, more chaotic regime. Two patterns emerge: FNO-RNN produces more globally distributed responses, while the graph-based models remain more localized. In both samples, MGN shows larger absolute errors than the second-order models (GraphCON and IGNS). In Fig. 8, MGN also misses one mode, consistent with error accumulation under autoregressive rollout. Overall, GraphCON performs best on this task, with IGNS and IGNS$_{\text{ti}}$ close competitors.

Warmup Phase. Figure 10 illustrates the effect of different warmup steps l on the Plate Deformation task over a full rollout. With almost no warmup ($l = 1$), the model produces sharp local patterns that gradually recover towards the correct global shape, matching the ground truth only after about $t = 10$. Increasing l accelerates this recovery: with $l = 5$, the deformation aligns well already by $t = 5$, and with $l = 10$ by $t = 3$. For larger values, $l = 30$ and $l = 50$, the global shape is already close to the ground truth at $t = 1$. These results highlight the role of the warmup phase in seeding long-range information before rollout, enabling the model to capture the dynamics more effectively from the very first step.



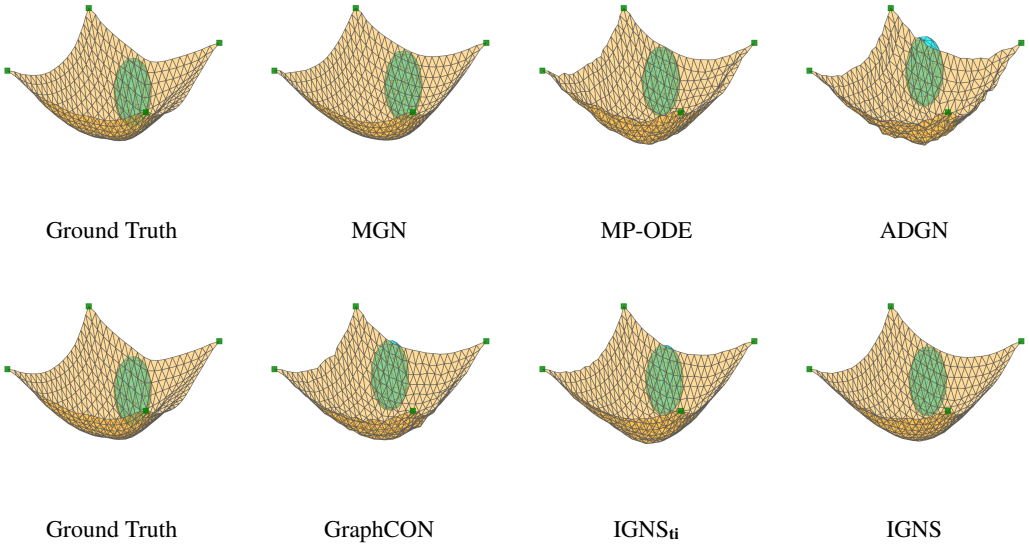


Figure 6: Sphere Cloth qualitative comparison on two test samples (separated by a horizontal line) with predictions at the final step $t = T$ from first-order models (MGN, MP-ODE) and second-order models (ADGN, GraphCON, **IGNS$_{ti}$** and **IGNS**).

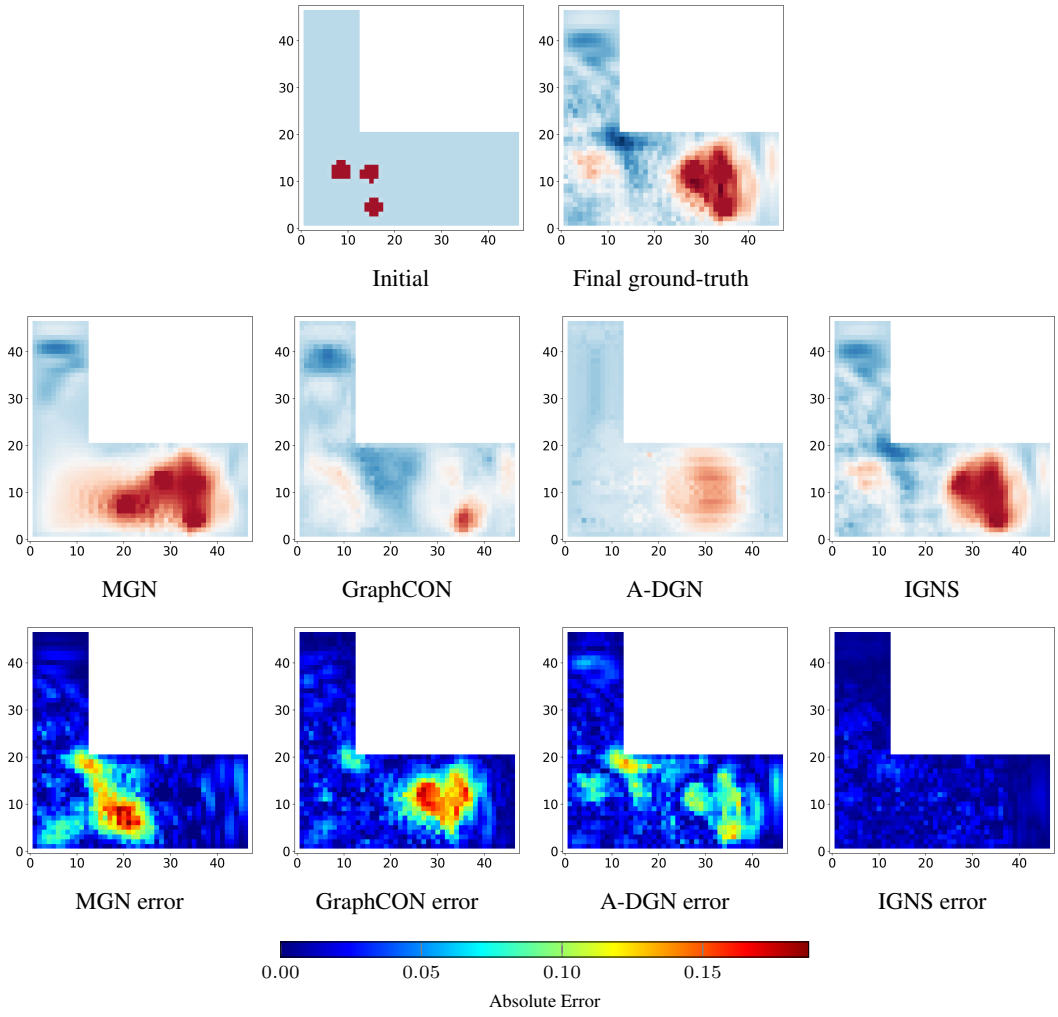


Figure 7: Wave Balls qualitative comparison on one test sample. Top: initial condition and ground-truth final state at time $t{=}T$. Middle: predictions at $t{=}T$ from MGN, GraphCON, A-DGN and **IGNS**. Bottom: absolute error maps, $|u_{\text{pred}} - u_{\text{gt}}|$. The horizontal colorbars indicate the error magnitude in the bottom row.

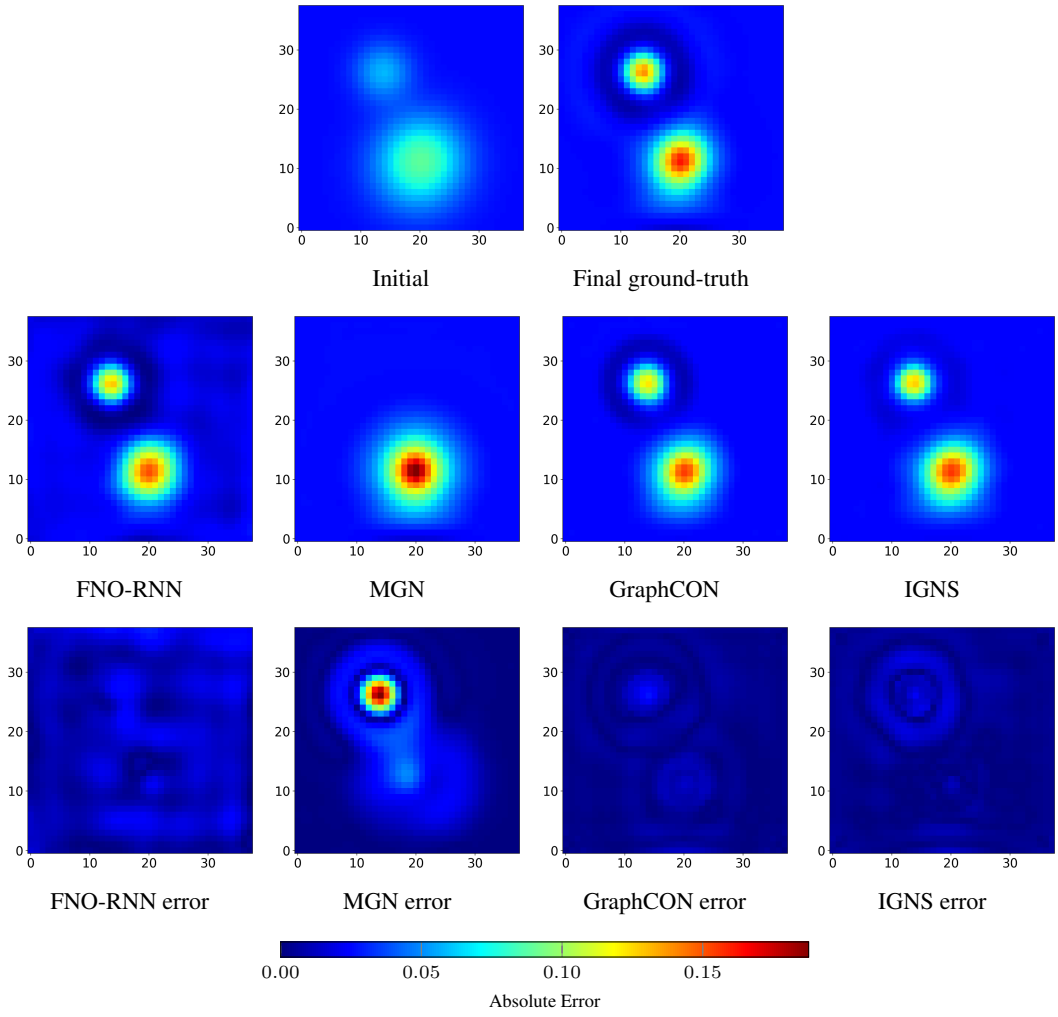


Figure 8: Kuramoto-Sivashinsky (KS) qualitative comparison on one test sample. Top: initial condition and ground-truth final state at time $t{=}T$. Middle: predictions at $t{=}T$ from FNO-RNN, MGN, GraphCON, and **IGNS**. Bottom: absolute error maps, $|u_{\text{pred}} - u_{\text{gt}}|$. The horizontal colorbars indicate the error magnitude in the bottom row.

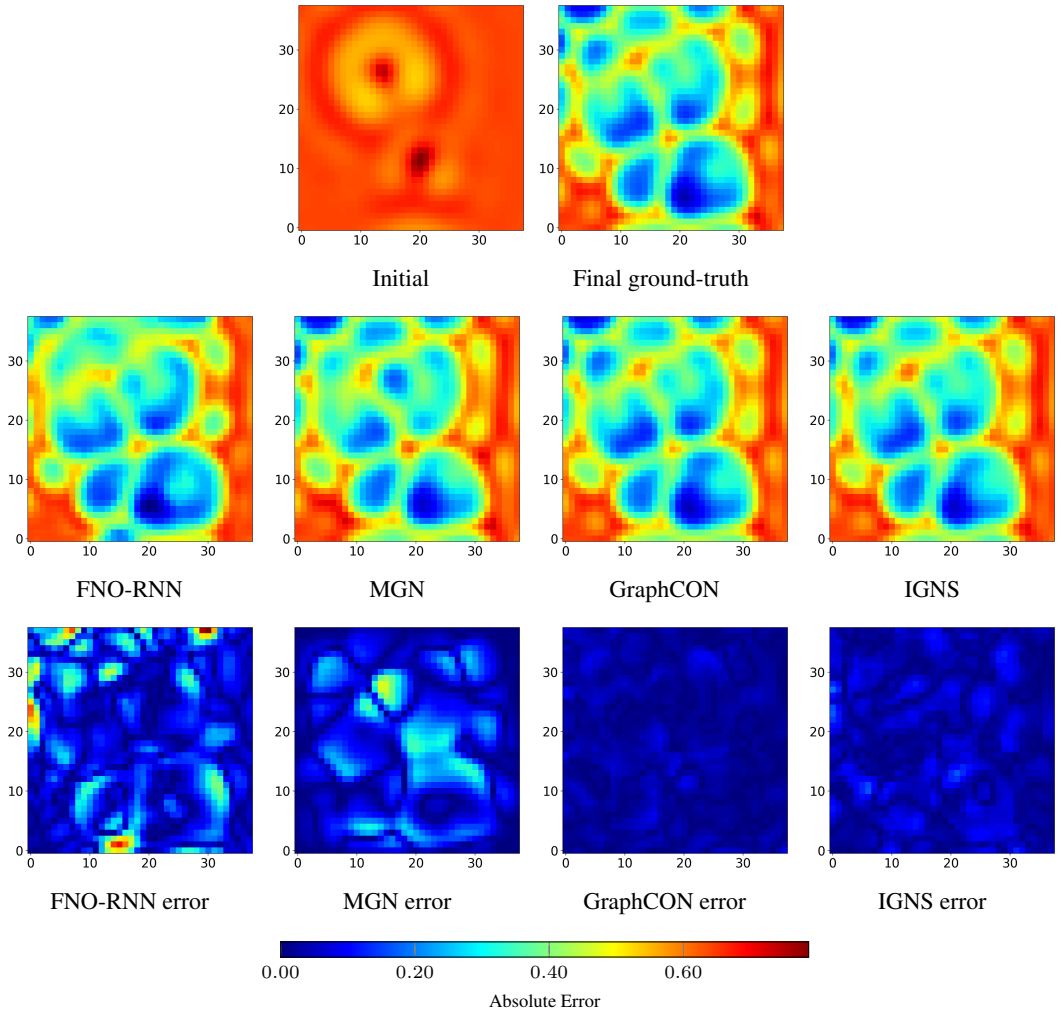
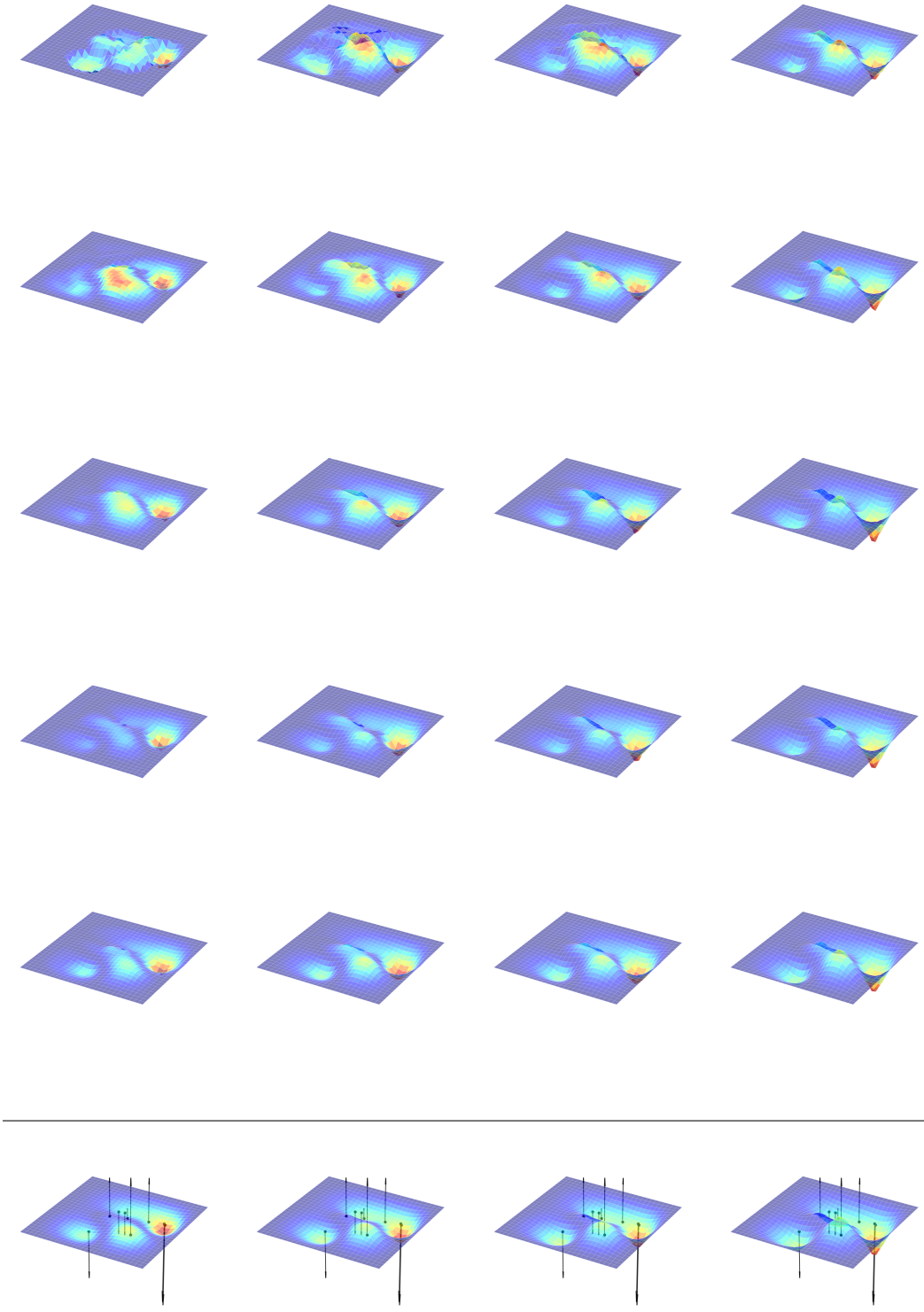


Figure 9: Kuramoto-Sivashinsky (KS) qualitative comparison on one test sample. Top: initial condition and ground-truth final state at time $t{=}T$. Middle: predictions at $t{=}T$ from FNO-RNN MGN, GraphCON, and **IGNS**. Bottom: absolute error maps, $|u_{\text{pred}} - u_{\text{gt}}|$. The horizontal colorbars indicate the error magnitude in the bottom row.



$t = 1$ $t = 3$ $t = 5$ $t = 10$

Figure 10: 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.