WHEN TO AGGREGATE: DELAY-AWARE MESSAGE SCHEDULING WITH PHYSICS-MOTIVATED OPERATOR DECOMPOSITION FOR DYNAMICS PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks have become a core paradigm for dynamics prediction by explicitly modeling spatial interactions among entities. Most existing approaches rely on synchronous message passing, in which all graph components are updated at every propagation layer. Although effective, this paradigm couples propagation depth with interaction-update frequency: increasing the processor depth not only expands the receptive field but also forces every local interaction to undergo additional transformations. This coupling can be undesirable for autoregressive dynamics, where different interactions may evolve at different rates. In this paper, we propose an asynchronous graph propagation scheme that regulates message aggregation by activating only a subset of edges at each layer. This edge-level mechanism enables information to propagate in a staggered manner, decoupling update frequency from propagation depth without modifying the underlying model architecture. It is model-agnostic and can be integrated into existing message-passing neural networks with minimal implementation effort. We further reinterpret graph propagation as the numerical integration of edge-local and node-local physics-motivated generators and decompose the latent dynamics into complementary operators. Extensive experiments across diverse benchmarks show that asynchronous propagation and operator decomposition consistently improve forecasting accuracy.

1 INTRODUCTION

Dynamics prediction on graph-structured data has attracted increasing attention because of its broad applications in traffic forecasting (Bai et al., 2020; Bui et al., 2022), climate modeling (Kochkov et al., 2024), power systems (Wu et al., 2025), and sensor networks (Du et al., 2026). By explicitly representing spatial dependencies among interacting entities, graph neural networks (GNNs) (Gilmer et al., 2017; Corso et al., 2024) have become a fundamental component of modern dynamics prediction frameworks. Most existing graph models adopt *synchronous message passing*, where all nodes update their representations simultaneously at every propagation layer. Although this paradigm facilitates implementation and parallel computation, it implicitly assumes that all interactions require the same update frequency and should be repeatedly transformed at every layer.

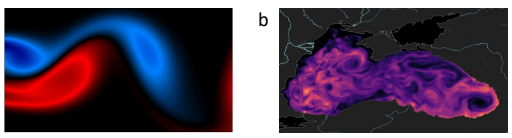

Figure 1: Representative dynamics scenarios. **a**, vortex evolution under complex boundary conditions. **b**, ocean currents in the Black Sea.

As illustrated in Figure 1, real-world dynamical systems are often observed on irregular spatial structures and exhibit heterogeneous interaction strengths and delayed dependencies (Yanchuk & Giacomelli, 2017; Ren et al., 2025). Under these conditions, synchronously updating every interaction at every layer couples spatial propagation depth with local update frequency. Increasing the number of graph layers therefore not only enlarges the receptive field, but also repeatedly transforms all local interactions, including weak or slowly varying ones. For autoregressive dynamics prediction, such uniform transformation frequencies may alter the evolution of latent interactions and exacerbate error accumulation over long rollouts. This motivates separating *how far information propagates from how frequently individual interactions are updated*. Existing studies have improved graph-based dynamics prediction through enhanced

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

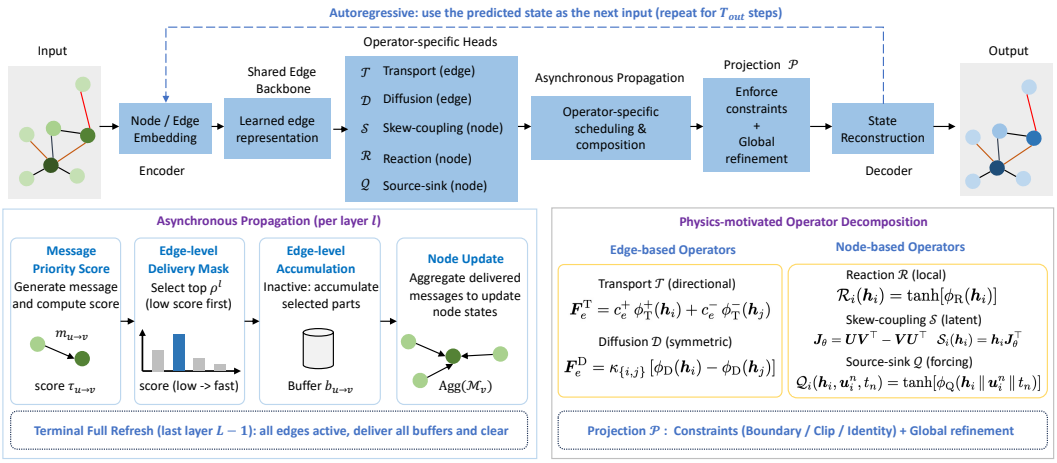

Figure 2: Overview of our method. Notations refer to the main text and supplementary material.

temporal modeling (Yang et al., 2026) or more expressive aggregation functions (Corso et al., 2020), but most retain fully synchronous propagation. Recent asynchronous GNNs provide an alternative by regulating node update orders, cross-layer aggregation rates, event-triggered recomputation, or distributed execution schedules. In contrast, this work focuses on *edge-level deferred aggregation* for autoregressive dynamics prediction. We further introduce an element-wise retention mechanism that determines which components of an inactive message are preserved in a persistent edge buffer for subsequent propagation.

We propose an *asynchronous graph propagation* framework for dynamics prediction, as illustrated in Figure 2. Rather than aggregating all edge messages at every layer, our method selectively activates edges according to state-dependent priorities, while inactive increments are retained and delivered later through persistent buffers. A terminal full refresh operation ensures that all deferred messages are incorporated before the final graph state is produced. This design reduces redundant transformations while maintaining complete graph coverage and compatibility with standard message-passing architectures. The scheduling rule is model-agnostic and leaves the graph topology unchanged. The percentile-based scheduler itself introduces no additional learnable parameters; learnable retention gates, when used for deferred accumulation, belong to the accumulation module and are analyzed separately. We further formulate graph propagation as the integration of physics-motivated latent operators, including transport $\mathcal{T}$, diffusion $\mathcal{D}$, reaction $\mathcal{R}$, skew-symmetric coupling $\mathcal{S}$, and source–sink forcing $\mathcal{Q}$, followed by projection $\mathcal{P}$. This formulation unifies asynchronous information delivery with physics-motivated operator integration within a single predictive framework. Our main contributions are summarized as follows:

- We introduce an edge-level deferred scheduling mechanism that decouples interaction-update frequency from propagation depth through selective activation, persistent accumulation, and terminal synchronization.
- We couple asynchronous scheduling with a physics-motivated operator decomposition, allowing transport and diffusion interactions to follow distinct state-dependent priority clocks instead of a single generic update schedule.
- Through controlled comparisons across four dynamical systems, we show that the gains persist under matched graph backbones, activation budgets, alternative priority policies, and alternative deferred-accumulation mechanisms.

2 Related Work

Graph Networks: Graph neural networks (GNNs) have been widely used for dynamics prediction because of their ability to model spatial dependencies among interacting entities. Early studies (Narayan & Roe, 2018; Diao et al., 2019) combine graph convolution with recurrent architectures to capture temporal evolution, while later methods incorporate temporal convolution (Liu et al., 2024) or attention mechanisms (Vaswani et al., 2017) to improve forecasting accuracy. By jointly learning spatial and temporal representations, graph-based models have achieved strong performance

across a variety of physical and spatiotemporal benchmarks (Pfaff et al., 2021; Brandstetter et al., 2022). Despite their architectural diversity, most existing methods (Du et al., 2022; Cini et al., 2023) adopt synchronous message passing, in which all edges or nodes are updated at every propagation layer. This convention is largely inherited from standard GNNs, yet its implications for long-horizon prediction stability remain insufficiently studied.

Neural Operators: Neural operators provide a complementary perspective by learning mappings between infinite-dimensional function spaces. Representative methods, including DeepONet (Lu et al., 2021), the Fourier Neural Operator (Li et al., 2021), and its factorized variants (Tran et al., 2023), have demonstrated strong performance on regular-grid partial differential equation benchmarks. Transformer-based neural solvers further extend operator learning to irregular geometries and large spatial domains through global token interactions (Wu et al., 2024). However, global mixing does not explicitly account for heterogeneous local update rates, nor does it distinguish among physically different processes such as transport, diffusion, and reaction.

Graph Propagation Strategies: A growing body of research has investigated more efficient and expressive graph propagation strategies, including adaptive aggregation (Huang et al., 2025), dynamic graph construction (Ye et al., 2021), sparse message passing (Baranwal et al., 2023), residual connections (Liu et al., 2021), and normalization techniques (Chen et al., 2022). These methods primarily modify *how* messages are constructed, aggregated, or transformed. In comparison, the propagation schedule—that is, *when* individual messages should be delivered and accumulated during multi-layer computation—has received considerably less attention.

Asynchronous Update Strategies: Asynchronous computation has been studied in distributed optimization (Assran et al., 2020), parallel training (Yang et al., 2021), and event-driven systems (Zhao et al., 2025), where partial or delayed updates improve scalability or efficiency. Related graph models include random or selective updates (Teke & Vaidyanathan, 2019), multi-rate gating and delayed propagation (Rusch et al., 2023; Gutteridge et al., 2023; Azabou et al., 2023), message- or node-level asynchronous and selective communication (Chen et al., 2024; Faber & Wattenhofer, 2024; Finkelshtein et al., 2024; Bose & Das, 2025), event-driven recomputation (Schaefer et al., 2022; Wang et al., 2021), and robustness to stale execution (Solodova et al., 2024). Subgraph-based methods additionally update only selected graph components during training or inference, mainly to reduce computational or memory costs (Alsentzer et al., 2020). However, these schedules are generally topology-, event-, or communication-driven and do not distinguish physical interaction types.

3 METHODOLOGY

Problem: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed computational graph with node positions $\boldsymbol{x}_i$, static attributes $\boldsymbol{s}_i$, and geometric edge attributes $\boldsymbol{g}_{ij}$, including the distance $d_{ij} = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2$. When a physical interaction is undirected, $\mathcal{E}_{\mathrm{u}}$ contains one fixed representative orientation, chosen once during graph construction and preserved throughout training and inference. Given the observed state $\boldsymbol{U} = \{\boldsymbol{u}_i\}_{i\in\mathcal{V}}$ at the physical time t, we predict T_{out} future states autoregressively from the single initial observation:

$$\widehat{\mathbf{U}}_{t_{k+1}} = \mathbf{U}_{t_k} + \mathrm{Dec}_\theta\big(\boldsymbol{H}^L\big), \quad \mathbf{U}_{t_{k+1}} \leftarrow \widehat{\mathbf{U}}_{t_{k+1}}. \tag{1}$$

Overview: The encoder initializes nodes and a shared backbone produces the edge representation,

$$\boldsymbol{h}_i^0 = \mathrm{Enc}_\theta\left(\boldsymbol{u}_i \,\|\, \boldsymbol{s}_i \,\|\, t\right), \quad \boldsymbol{e}_{ij}^\ell = \phi_{\mathrm{sh}}\big(\boldsymbol{h}_i^\ell \,\|\, \boldsymbol{h}_j^\ell \,\|\, (\boldsymbol{h}_j^\ell - \boldsymbol{h}_i^\ell) \,\|\, \boldsymbol{g}_{ij}\big). \tag{2}$$

Operator-specific heads map this shared representation to transport and diffusion increments. Node-local heads model reaction, skew-symmetric coupling, and forcing. Each micro-step first composes the delivered edge increments and node-local operators, followed by the projection block $\mathcal{P}$. After L such micro-steps, the decoder produces the next physical state.

3.1 PHYSICS-MOTIVATED OPERATOR DECOMPOSITION

We model the latent continuous-time vector field as

$$\dot{\boldsymbol{H}} = \mathcal{T}(\boldsymbol{H}) + \mathcal{D}(\boldsymbol{H}) + \mathcal{R}(\boldsymbol{H}) + \mathcal{S}(\boldsymbol{H}) + \mathcal{Q}(\boldsymbol{H}, \boldsymbol{U}, t), \tag{3}$$

followed by the projection block $\boldsymbol{H} \leftarrow \mathcal{P}(\dot{\boldsymbol{H}})$. These operator names denote architectural inductive biases and do not imply that every latent channel corresponds to a directly measurable physical quantity.

Transport $\mathcal{T}$: For an oriented edge $(i, j) \in \mathcal{E}_{\mathrm{u}}$, the transport head predicts a signed coefficient and an upwind-type latent flux

$$c_e^{\ell} = \phi_c\big(\boldsymbol{e}_{ij}^{\ell}, \boldsymbol{g}_{ij}\big), \quad \boldsymbol{F}_e^{\mathrm{T}} = c_e^{+}\, \phi_{\mathrm{T}}^{+}(\boldsymbol{h}_i^{\ell}) + c_e^{-}\, \phi_{\mathrm{T}}^{-}(\boldsymbol{h}_j^{\ell}). \tag{4}$$

where $c_e^{+} = \max(c_e, 0)$ and $c_e^{-} = \min(c_e, 0)$. The sign selects the upstream endpoint and permits directional behavior that cannot be represented by a symmetric diffusivity. A reversal of the stored edge orientation is accompanied by the corresponding reversal of the signed coefficient.

Diffusion $\mathcal{D}$: The diffusion head predicts nonnegative directed candidates and symmetrizes each reciprocal pair. Then the diffusive flux is

$$\widetilde{\kappa}_{ij}^{\ell} = \mathrm{softplus}\big(\phi_{\kappa}(\boldsymbol{e}_{ij}^{\ell}, \boldsymbol{g}_{ij})\big), \quad \kappa_{ij}^{\ell} = \tfrac{1}{2}\left(\widetilde{\kappa}_{ij}^{\ell} + \widetilde{\kappa}_{ji}^{\ell}\right) \geq 0, \quad \boldsymbol{F}_e^{\mathrm{D}} = \kappa_{ij}\left[\phi_{\mathrm{D}}(\boldsymbol{h}_i^{\ell}) - \phi_{\mathrm{D}}(\boldsymbol{h}_j^{\ell})\right]. \tag{5}$$

Thus $\kappa_{\{i,j\}} = \kappa_{\{j,i\}}$, and scattering the same flux with opposite incidence signs gives pairwise conservative smoothing in the transformed latent coordinates.

Node-Local Operators: Reaction is a bounded pointwise nonlinearity,

$$\mathcal{R}_i(\boldsymbol{h}_i^{\ell}) = \tanh\big[\phi_{\mathrm{R}}(\boldsymbol{h}_i^{\ell})\big], \tag{6}$$

which captures local production, decay, phase interaction, or other effects that do not require spatial communication. To represent non-dissipative mixing among latent channels, we parameterize

$$\boldsymbol{J}_{\theta} = \boldsymbol{U}\boldsymbol{V}^{\top} - \boldsymbol{V}\boldsymbol{U}^{\top}, \quad \mathcal{S}_i(\boldsymbol{h}_i) = \boldsymbol{h}_i^{\ell}\boldsymbol{J}_{\theta}^{\top}. \tag{7}$$

Since $\boldsymbol{J}_{\theta}^{\top} = -\boldsymbol{J}_{\theta}$, we have $\langle \boldsymbol{h}_i, \mathcal{S}_i(\boldsymbol{h}_i)\rangle = 0$ in continuous time. This motivates the term *skew-symmetric latent coupling*; it does not imply exact Hamiltonian conservation after discrete integration.

Forcing or controls are modeled by a conditional source–sink/local correction operator,

$$\mathcal{Q}_i(\boldsymbol{h}_i^{\ell}, \boldsymbol{u}_i, t) = \tanh\big[\phi_{\mathrm{Q}}(\boldsymbol{h}_i^{\ell} \,\|\, \boldsymbol{u}_i \,\|\, t)\big]. \tag{8}$$

Projection $\mathcal{P}$: Each integration step is followed by a learnable global refinement and, when available, an explicit constraint projection,

$$\boldsymbol{H}^{\ell+1} = \underbrace{\Pi_{\mathcal{C}}}_{\text{constraint}} \circ \underbrace{\mathcal{G}_{\theta}}_{\text{refinement}} (\dot{\boldsymbol{H}}^{\ell}) \equiv \mathcal{P}(\dot{\boldsymbol{H}}^{\ell}). \tag{9}$$

Appendix Sec. I specifies the choices used for each benchmark.

3.2 Operator-Specific Scheduling

The edge operators decompose into local generators,

$$\mathcal{T} = \sum\nolimits_{e \in \mathcal{E}_{\mathrm{u}}} \mathcal{T}_e, \quad \mathcal{D} = \sum\nolimits_{e \in \mathcal{E}_{\mathrm{u}}} \mathcal{D}_e. \tag{10}$$

Transport and diffusion evolve on different scales with separate activation masks a_{T}^{ℓ} and a_{D}^{ℓ}.

Candidate Increments and Priority Clocks: Let $\Delta t_{\ell} > 0$ be the duration assigned to micro-step ℓ, with $\sum_{\ell=0}^{L-1} \Delta t_{\ell} = t_{k+1} - t_k$. The operator heads form candidate edge increments

$$\Delta \boldsymbol{m}_{\square}^{\ell} = \Delta t_{\ell}\, \boldsymbol{F}_e^{\square}(\boldsymbol{H}^{\ell}), \quad \square \in \{\mathrm{T}, \mathrm{D}\}. \tag{11}$$

We rank candidate increments using operator-specific state-dependent priority clocks

$$\tau_{\mathrm{T}}^{\ell} = \frac{d_e}{\|\boldsymbol{F}_e^{\mathrm{T}}\|_2 + \varepsilon_T}, \quad \tau_{\mathrm{D}}^{\ell} = \frac{d_e^2}{\|\boldsymbol{F}_e^{\mathrm{D}}\|_2 + \varepsilon_D}. \tag{12}$$

The factors d_e and d_e^2 are motivated by the characteristic length scaling of advective and diffusive processes, respectively, while the instantaneous latent flux magnitudes make the priorities state dependent. Accordingly, τ should be interpreted as a physics-scaled scheduling score, rather than a calibrated physical travel time. For an activation ratio $\rho_{\square}^{\ell} \in (0, 1]$, define

$$\gamma_{\square}^{\ell} = \mathrm{Quantile}_{\rho_{\square}^{\ell}}\left(\{\tau_{\square}^{\ell} : e \in \mathcal{E}_{\mathrm{u}}\}\right), \quad a_{\square}^{\ell} = \mathbb{I}[\tau_{\square}^{\ell} \leq \gamma_{\square}^{\ell}]. \tag{13}$$

Smaller scores are delivered earlier. Quantile ties are handled by activating the entire tie group (or by a permutation-invariant randomization), avoiding dependence on stored edge indices.

Deferred Accumulation and Delivery: Each edge and operator maintains a buffer $\boldsymbol{b}_{\square}^{\ell}$, initialized to zero. A componentwise gate $\boldsymbol{z}_{\square}^{\ell} \in [0,1]^{d_h}$ determines which components of an inactive increment are retained. Delivery and buffer updates are

$$\boldsymbol{d}_{\square}^{\ell} = a_{\square}^{\ell}\left(\Delta \boldsymbol{m}_{\square}^{\ell} + \boldsymbol{b}_{\square}^{\ell}\right), \quad \boldsymbol{b}_{\square}^{\ell+1} = (1 - a_{\square}^{\ell})\left(\boldsymbol{b}_{\square}^{\ell} + \boldsymbol{z}_{\square}^{\ell} \odot \Delta \boldsymbol{m}_{\square}^{\ell}\right). \tag{14}$$

Hence an active edge delivers its current and deferred increments and then clears its buffer; an inactive edge stores only the gated components. Hard gates can be trained with a straight-through estimator, while sigmoid gates give a fully differentiable alternative. Let $\boldsymbol{B} \in \{-1,0,1\}^{|\mathcal{V}|\times|\mathcal{E}_{\mathrm{u}}|}$ be the oriented incidence matrix, with $B_e = -1$ at the edge tail and $B_e = +1$ at the edge head. Delivered edge increments are scattered to nodes and a first-order multirate update is

$$\Delta \boldsymbol{h}_{\square}^{\ell} = \sum_{e\in\mathcal{E}_{\mathrm{u}}} B_e \boldsymbol{d}_{\square}^{\ell}, \quad \boldsymbol{h}^{\ell+1} = \boldsymbol{h}^{\ell} + \Delta \boldsymbol{h}_{\mathrm{T}}^{\ell} + \Delta \boldsymbol{h}_{\mathrm{D}}^{\ell} + \Delta t_{\ell}\left(\mathcal{R}(\boldsymbol{h}^{\ell}) + \mathcal{S}(\boldsymbol{h}^{\ell}) + \mathcal{Q}(\boldsymbol{h}^{\ell}, \boldsymbol{u}, t_{\ell})\right). \tag{15}$$

Higher-order compositions in Appendix Sec. E may replace Eq. 15 without changing the scheduler.

Terminal Full Refresh: The last micro-step activates every edge for both edge operators:

$$a_{\square}^{L-1} = 1, \quad \square \in \{\mathrm{T}, \mathrm{D}\}. \tag{16}$$

Eq. 14 flush all deferred increments before projection. The model is asynchronous only within a physical interval; it does not permit unbounded message staleness across autoregressive steps.

3.3 Theoretical Properties

The following results establish structural properties of the asynchronous integrator, including synchronous consistency, permutation equivariance, bounded deferred accumulation, and terminal synchronization. The final bound characterizes autoregressive error amplification but does not assume that asynchronous scheduling is universally more stable. Proofs and assumptions are provided in Appendix Sec. H.

Proposition 3.1 (Synchronous consistency)**.** *If $a_{\square}^{\ell} = 1$ for every edge, operator, and micro-step, and $\boldsymbol{b}_{\square}^{0} = \mathbf{0}$, then Eqs. 14–15 reduce to the corresponding synchronous operator-decomposed update.*

Theorem 3.2 (Permutation equivariance)**.** *Suppose all node, edge, gate, and operator maps share parameters; aggregation is permutation invariant; geometry is relabeled with the graph; and score ties are resolved equivariantly. Then one complete asynchronous forecasting step, including terminal refresh and an equivariant projection, is permutation equivariant.*

Proposition 3.3 (Finite deferred accumulation)**.** *Assume $\left\|\Delta \boldsymbol{m}_{\square}^{\ell}\right\|_2 \le M_{\square}$ and $\left\|\boldsymbol{z}_{\square}^{\ell}\right\|_{\infty} \le 1$. If $\ell_{\square}^{\mathit{last}}$ is the most recent delivery step before ℓ, then*

$$\left\|\boldsymbol{b}_{\square}^{\ell}\right\|_2 \le (\ell - \ell_{\square}^{\mathit{last}} - 1) M_{\square}. \tag{17}$$

In particular, the buffer is finite for finite L.

Proposition 3.4 (Terminal synchronization)**.** *Under Eq. 16, $\boldsymbol{b}_{\square}^{L} = \mathbf{0}$ for every edge and edge operator.*

For completeness, let Ψ_A be one full asynchronous step and F the true discrete evolution. Assume

$$\|\Psi_A(x) - \Psi_A(y)\| \le K_A \|x - y\|, \quad \|\Psi_A(x) - F(x)\| \le \varepsilon_A. \tag{18}$$

Corollary 3.5 (Autoregressive rollout bound)**.** *After T recursive steps,*

$$e_{t_{k+T}} \le K_A^T e_{t_k} + \varepsilon_A \sum\nolimits_{r=0}^{T-1} K_A^r. \tag{19}$$

If $K_A < 1$, the second term is at most $\varepsilon_A/(1 - K_A)$.

Corollary 3.5 identifies K_A as the effective sensitivity controlling autoregressive error amplification. A tighter rollout bound than that of a synchronous counterpart therefore requires a smaller effective sensitivity together with comparable one-step approximation error; the corollary itself does not assume or guarantee this ordering.

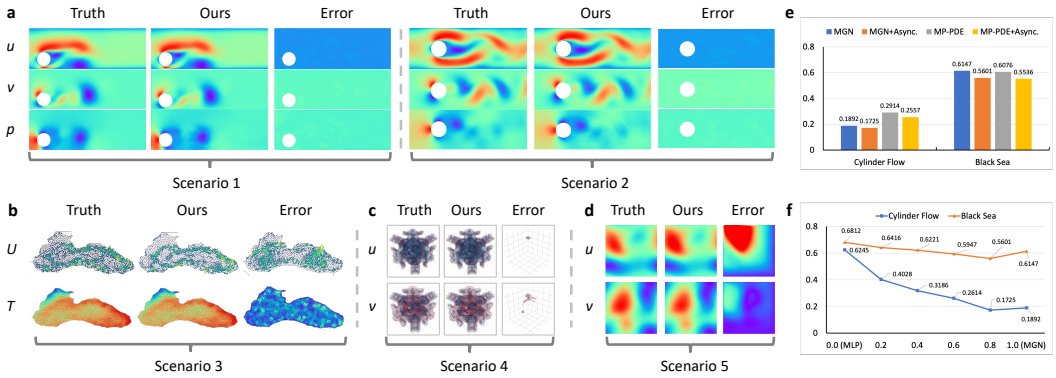

Figure 3: Visualized results. **a–d**, snapshots of ground-truth states, predictions, and prediction errors on four datasets. **e–f**, visualizations for Tables 3–4. More results are shown in Appendix Figure S1.

Special Cases: Setting every mask to one gives a synchronous operator-decomposed graph solver. Sharing one generic edge state, one mask, and one intensity-aware distance score recovers conventional edge-level asynchronous percentile scheduling. The complete model instead separates transport and diffusion, assigns them distinct clocks, and composes their increments over a physical interval.

Computational Characteristics: Let d_h be the latent width. If the shared backbone and all operator heads are evaluated on every edge before ranking, the dominant cost per micro-step remains $\mathcal{O}(|\mathcal{E}|d_h^2)$ plus $\mathcal{O}(|\mathcal{E}|)$ / $\mathcal{O}(|\mathcal{E}|\log(|\mathcal{E}|))$ scheduling cost, depending on the quantile implementation. Thus masking alone changes information evolution but does not guarantee computational savings. In a computation-sparse implementation, a cheap priority proxy is computed from geometry, previous edge states, and node differences before expensive operator heads are executed only on active edges. The head cost then scales approximately as $\mathcal{O}((\rho_{\mathrm{T}}^{\ell} + \rho_{\mathrm{D}}^{\ell})|\mathcal{E}|d_h^2)$, up to shared-backbone, priority, buffering, and terminal-refresh costs. Accordingly, we describe the principal benefit as asynchronous information propagation unless sparse execution is implemented and benchmarked explicitly.

4 Experiments

Datasets: We evaluate our framework on four public benchmarks spanning both simulated PDE systems and real-world dynamics: (i) 2D Cylinder Flow (CF), which models incompressible flow around varying cylinder; (ii) 2D Black Sea (BS), which captures ocean circulation dynamics; (iii) the 3D Gray–Scott system, a representative reaction–diffusion process; and (iv) the 2D Burgers equation, a canonical nonlinear advection–diffusion benchmark. More details are listed in Appendix Table S4.

Baselines: We compare against representative operator-learning methods, including DeepONet (Lu et al., 2021), FNO (Li et al., 2021), and FFNO (Tran et al., 2023); general-purpose neural architectures, including MLP and vanilla Transformer (Vaswani et al., 2017); the geometry-aware neural solver Transolver (Wu et al., 2024); and graph-based physical simulators, including MGN (Pfaff et al., 2021), MP-PDE (Brandstetter et al., 2022), and CiGNN (Mi et al., 2025). Additional details are provided in Appendix Sec. K.

Implementation Details: All models are optimized using the Adam optimizer. Performance is evaluated using root mean squared error (RMSE) over node states. For controlled synchronous-versus-asynchronous comparisons, the number of propagation layers, hidden dimension, and training hyperparameters are kept fixed. All models are repeated using 3 independent random seeds, except for MLP. Unless otherwise specified, we report the mean performance; for the main results, we report the mean $\pm$ standard deviation. Training is conducted on NVIDIA A100 GPU (80GB) and Intel(R) Xeon(R) Platinum 8380 CPU.

4.1 Overall Performance

Table 1 compares autoregressive forecasting performance across four benchmarks spanning both 2D and 3D dynamical systems. Our model achieves the lowest RMSE on all datasets, outperforming both graph-based simulators and operator-learning baselines, as illustrated in Figures 3**a**–**d**. The

Table 1: Performance Summary. "Imp." is the relative RMSE reduction of the **best** over second-best.

Model	2D Cylinder Flow	2D Black Sea	3D Gray-Scott	2D Burgers
MLP	0.6245	0.6812	0.2914	0.1872
DeepONet	0.9109 ± 0.0068	2.8696 ± 0.0187	0.1395 ± 0.0019	0.3775 ± 0.0048
FNO	–	–	0.1716 ± 0.0015	0.0575 ± 0.0009
FFNO	–	–	0.0359 ± 0.0006	0.0334 ± 0.0007
Transformer	0.3782 ± 0.0032	0.5583 ± 0.0057	0.1178 ± 0.0014	0.0374 ± 0.0028
Transolver	0.4610 ± 0.0471	0.8199 ± 0.0286	0.1520 ± 0.0121	0.1742 ± 0.0128
MGN	0.1892 ± 0.0017	0.6147 ± 0.0058	0.0193 ± 0.0003	0.0118 ± 0.0013
MP-PDE	0.2914 ± 0.0028	0.6076 ± 0.0062	0.0652 ± 0.0011	0.0177 ± 0.0015
CiGNN	0.1591 ± 0.0014	0.6035 ± 0.0053	0.0093 ± 0.0001	0.0066 ± 0.0002
Ours	**0.1426** ± 0.0011	**0.5191** ± 0.0063	**0.0089** ± 0.0002	**0.0044** ± 0.0003
Imp. (%)	10.4	7.02	4.30	33.3

improvements are particularly pronounced on 2D CF, 2D Black Sea, and 2D Burgers, while remaining consistent on 3D Gray–Scott. These results indicate that asynchronous propagation and operator-specific modeling generalize across distinct dynamical regimes.

4.2 Ablation Study

Table 2: Ablations measured by test RMSE.

Variant	CF	BS	GS	Burgers
Full	0.1426	0.5191	0.0089	0.0044
w/o $\mathcal{T}$	0.1459	0.5290	0.0090	0.0047
w/o $\mathcal{D}$	0.1453	0.5297	0.0090	0.0048
w/o $\mathcal{R}$	0.1457	0.5281	0.0089	0.0048
w/o $\mathcal{S}$	0.1452	0.5296	0.0089	0.0048
w/o $\mathcal{Q}$	0.1453	0.5306	0.0090	0.0047
w/o $\mathcal{P}$	0.1579	0.5700	0.0093	0.0062
w/o Ope.	0.1725	0.5601	0.0186	0.0051
w/o Async.	0.1579	0.5477	0.0098	0.0061
w/o Both	0.1892	0.6147	0.0193	0.0118

To further isolate the contribution of each component, we conduct ablation studies across all benchmarks, with the results summarized in Table 2. We first ablate each operator individually and then consider three coarse-grained variants: removing the asynchronous propagation mechanism ("w/o Async."), removing the operator-specific modeling scheme ("w/o Ope."), and removing both components ("w/o Both"). Removing either component consistently degrades prediction accuracy, while removing both leads to the largest performance drop. These results demonstrate that both asynchronous propagation and operator-specific modeling contribute substantially to accurate dynamics prediction. Moreover, the consistent gap between the full model and all ablated variants suggests that the two components provide complementary benefits when integrated within a unified framework.

Effect of Asynchronous Scheduling: To evaluate the effectiveness of asynchronous propagation, we compare synchronous and asynchronous variants of representative graph-based models, including MGN (Pfaff et al., 2021) and MP-PDE (Brandstetter et al., 2022). For each baseline, the propagation schedule is the only modified component, while the model architecture and capacity remain unchanged, ensuring a controlled and fair comparison. Table 3 reports quantitative results across four benchmarks spanning both 2D and 3D dynamical systems. Asynchronous propagation consistently improves prediction accuracy across all datasets and model backbones.

Table 3: Synchronous graph models and their asynchronous counterparts. Lower is better.

Model	CF	BS	GS	Burgers
MGN	0.1892	0.6147	0.0193	0.0118
+ Async.	0.1725	0.5601	0.0186	0.0051
MP-PDE	0.2914	0.6076	0.0652	0.0177
+ Async.	0.2557	0.5536	0.0628	0.0099

Effect of Scheduling Rate: We further investigate the effect of percentile-based scheduling in asynchronous propagation. In our method, edges are activated at each propagation layer according to the percentile ranks of their message priority scores, which controls the fraction of edge-level messages aggregated at each layer. The "MLP" row is a no-message-passing reference rather than a scheduler setting; terminal refresh is not applied in this endpoint. The "MGN" row corresponds to the synchronous setting. As shown in Table 4,

Table 4: Effect of the scheduling rate.

Rate ρ	CF	BS	GS	Burgers
0(MLP)	0.6245	0.6812	0.2914	0.1872
0.2	0.4028	0.6416	0.0910	0.0693
0.4	0.3186	0.6221	0.0586	0.0373
0.6	0.2614	0.5947	0.0395	0.0185
0.8	**0.1725**	**0.5601**	**0.0186**	**0.0051**
1(MGN)	0.1892	0.6147	0.0193	0.0118

moderate scheduling rates consistently yield the best performance. Excessively small rates lead to sparse message aggregation and insufficient information flow, while excessively large rates weaken the benefits of asynchronous propagation and introduce more redundant updates. These results indicate that effective asynchronous propagation requires a balanced scheduling rate that reduces unnecessary message updates while maintaining sufficient information exchange.

Masked Deferred Accumulation: The hard-mask formulation is obtained as a special case by restricting the gate values to $\{0, 1\}^{d_m}$. The ablations in Table 5 disentangle the effects of deferred message accumulation and selective retention. The random and shuffled variants preserve the same gating architecture and parameter count, but replace or permute the learned gate outputs during evaluation. Masked deferred accumulation achieves the lowest RMSE across all four datasets and consistently outperforms the variant without the accumulation buffer. Moreover, the relatively small gap between full and masked accumulation suggests that preserving deferred information accounts for most of the performance gain, while the learned gate provides an additional benefit through selective filtering.

Table 5: Ablation of the deferred-message buffer and retention gate. Lower is better.

Variant	CF	BS	GS	Burgers
No Buffer	0.1845	0.6048	0.0189	0.0100
Latest Message	0.1809	0.5876	0.0189	0.0084
Random Mask	0.1829	0.5960	0.0191	0.0091
Shuffled Mask	0.1814	0.5977	0.0189	0.0087
Full Accum.	0.1782	0.5763	0.0186	0.0061
Masked Accum.	0.1725	0.5601	0.0186	0.0051

Comparison of Scheduling Policies: To examine whether the observed gains can be attributed merely to generic sparsification, we compare different scheduling policies under the same intermediate-layer activation rate in Table 6. Random scheduling activates the same number of edges as the proposed policy, whereas the static schedule is fixed before rollout. We further consider representative priority functions based on distance, message magnitude, and the proposed state-dependent criterion (see Appendix Sec. M). Since edges with smaller priority scores are activated earlier, the magnitude-based policy is defined to prioritize messages with larger norms. Synchronous propagation serves as the full-update reference. We also include a no-terminal-refresh variant as a mechanism ablation; unlike the policy comparisons above, this variant has a smaller total aggregation budget unless the final-layer activation rate is compensated. The proposed scheduling policy achieves the lowest error on all four datasets, yielding lower mean RMSE than random and single-factor policies. The particularly large improvement on Burgers suggests that state-dependent message delivery may be especially beneficial for transport-dominated dynamics.

Table 6: Comparison of scheduling policies. Asynchronous priority policies use the same intermediate activation rate.

Scheduling policy	CF	BS	GS	Burgers
Synchronous	0.1892	0.6147	0.0193	0.0118
Random	0.1854	0.6065	0.0189	0.0099
Static Distance	0.1797	0.5937	0.0189	0.0085
Message Magnitude	0.1770	0.5849	0.0187	0.0066
No terminal refresh	0.1774	0.5847	0.0187	0.0072
Proposed Priority	0.1725	0.5601	0.0186	0.0051

Message-Retention and Staleness Diagnostics: We analyze message retention and edge coverage to characterize the behavior of asynchronous propagation in Table 7. Despite aggregating only about 85% of edge updates, the pre-terminal edge coverage remains above 91% across all datasets, indicating that the scheduler dynamically rotates the active edge subset rather than repeatedly selecting a fixed set of edges. Burgers exhibits the highest retention ratio, indicating that the retention gate preserves a larger fraction of deferred message components, whereas GS applies more aggressive filtering. More definitions and statistics are in Appendix Sec. M.

Table 7: Statistics of asynchronous propagation and deferred accumulation.

Statistic	CF	BS	GS	Burgers
Retention ratio	0.619	0.688	0.539	0.721
Aggregated edges (%)	85	85	85	85
Pre-terminal edges (%)	96.7	94.9	91.3	97.7
Mean waiting depth	1.17	1.33	1.47	1.12
Maximum waiting depth	3	3	3	3

Alternative Asynchronous Mechanisms: We compare several representative partial-update mechanisms under the same message-passing backbone, including random edge activation, node-batch updates, delayed aggregation, and multi-rate gating.

The proposed scheduler achieves the best performance across the tested benchmarks in Table 8, with its result on GS matching multi-rate gating at four-decimal precision. Compared with multi-rate gating, the proposed scheduler further reduces RMSE on CF, BS, and Burgers, suggesting that state-dependent prioritization combined with explicit deferred accumulation is more effective than modulating the update rate alone.

Table 8: Alternative asynchronous mechanisms using a common MPNN backbone and a matched update budget.

Method	CF	BS	GS	Burgers
Synchronous MPNN	0.1892	0.6147	0.0193	0.0118
Random Edge Update	0.1854	0.6065	0.0189	0.0099
Node-Batch Async	0.1860	0.6076	0.0191	0.0104
Delayed Aggregation	0.1782	0.5763	0.0186	0.0061
Multi-Rate Gating	0.1752	0.5663	0.0186	0.0056
MGN + Async.	0.1725	0.5601	0.0186	0.0051

Asynchronous Efficiency: Asynchronous aggregation does not necessarily yield proportional computational savings, since scheduling may still require message representations to be computed before edge selection. We report inference latency and memory consumption together with the fractions of computed and aggregated edge messages in Table 9. All measurements are conducted under identical hardware and model parameters (e.g., L=8).

Table 9: Efficiency under a controlled setting.

Model	MGN	MGN + Async.
Params (M)	$\sim$ 1.3M	$\sim$ 1.3M
Train (s/epoch)	44.2	46.8
Infer. (s/step)	0.0079	0.0085
Memory (GB)	64.1	65.3
Computed edges (%)	100	100
Aggregated edges (%)	100	82.5

Operator-Specific Scheduling Sensitivity: We extend the scheduling rate analysis to the full operator-specific model by varying $(\rho_{\mathrm{T}}, \rho_{\mathrm{D}})$ while keeping the micro-step count L fixed at its validation-selected value. Table 10 reports representative balanced and mixed scheduling settings. The balanced setting $(0.8, 0.8)$ achieves the lowest error across all four datasets. CF, BS, and Burgers favor more frequent transport updates, whereas GS slightly favors more frequent diffusion updates. These results suggest that the preferred relative update frequency can vary across dynamical systems, although the differences between the two mixed settings are modest. In contrast, the setting $(0.6, 0.6)$ performs worse than the fully synchronous operator model on CF, BS, and Burgers, indicating that excessive sparsification can offset the benefits of asynchronous scheduling.

Table 10: Sensitivity to the transport and diffusion activation ratios.

Dataset	ρ_{T}	ρ_{D}	RMSE	Configuration
CF	0.6	0.6	0.1596	sparse balanced
CF	0.8	0.6	0.1463	higher-T rate
CF	0.6	0.8	0.1512	higher-D rate
CF	0.8	0.8	0.1426	our model
CF	1.0	1.0	0.1579	synchronous only
BS	0.6	0.6	0.5649	sparse balanced
BS	0.8	0.6	0.5336	higher-T rate
BS	0.6	0.8	0.5440	higher-D rate
BS	0.8	0.8	0.5191	our model
BS	1.0	1.0	0.5477	synchronous only
GS	0.6	0.6	0.0094	sparse balanced
GS	0.8	0.6	0.0091	higher-T rate
GS	0.6	0.8	0.0090	higher-D rate
GS	0.8	0.8	0.0089	our model
GS	1.0	1.0	0.0098	synchronous only
Burgers	0.6	0.6	0.0068	sparse balanced
Burgers	0.8	0.6	0.0050	higher-T rate
Burgers	0.6	0.8	0.0054	higher-D rate
Burgers	0.8	0.8	0.0044	our model
Burgers	1.0	1.0	0.0061	synchronous only

5 Conclusion

In this work, we propose an asynchronous graph propagation framework for dynamics prediction. The method selectively delivers edge messages according to state-dependent priority scores, stores deferred increments in edge buffers, and performs a terminal full refresh to ensure complete information exchange. We further decompose latent dynamics into transport, diffusion, reaction, skew-symmetric coupling, and source–sink forcing, followed by projection, with operator-specific schedules for heterogeneous physical processes. Theoretical analysis establishes synchronous consistency, permutation equivariance, finite deferred accumulation, terminal synchronization, and an autoregressive rollout bound. Experiments on four dynamical systems show that asynchronous scheduling and operator decomposition consistently improve long-horizon forecasting accuracy across different graph backbones. A current limitation is that the scheduling ratios and operator forms remain manually specified, motivating future work on adaptive, data-driven scheduling and extensions to larger-scale and more complex dynamical systems.

REFERENCES

Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 33:8017–8029, 2020.

Mahmoud Assran, Arda Aytekin, Hamid Reza Feyzmahdavian, Mikael Johansson, and Michael G Rabbat. Advances in asynchronous parallel and distributed optimization. *Proceedings of the IEEE*, 108(11):2013–2031, 2020.

Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Veličković, and Eva L Dyer. Half-hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning (ICML)*, pp. 1341–1360. PMLR, 2023.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems (NIPS)*, 33: 17804–17815, 2020.

Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Optimality of message-passing architectures for sparse graphs. *Advances in Neural Information Processing Systems (NIPS)*, 36: 40320–40341, 2023.

Kushal Bose and Swagatam Das. Asynchronous message passing for addressing oversquashing in graph neural networks. *arXiv preprint arXiv:2509.06777*, 2025.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. In *International Conference on Learning Representations (ICLR)*, 2022.

Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*, 52(3):2763–2774, 2022.

Jialong Chen, Tianchi Liao, Chuan Chen, and Zibin Zheng. Improving message-passing gnns by asynchronous aggregation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 228–238, 2024.

Yihao Chen, Xin Tang, Xianbiao Qi, Chun-Guang Li, and Rong Xiao. Learning graph normalization for graph neural networks. *Neurocomputing*, 493:613–625, 2022.

Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable spatiotemporal graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7218–7226, 2023.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems (NIPS)*, 33:13260–13271, 2020.

Gabriele Corso, Hannes Stark, Stefanie Jegelka, Tommi Jaakkola, and Regina Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(1):17, 2024.

Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 890–897, 2019.

Shangjie Du, Hui Wei, Dong Yoon Lee, Zhizhang Hu, and Shijia Pan. Graph-based physics-guided urban pm2. 5 air quality imputation with constrained monitoring data. *ACM Transactions on Sensor Networks*, 22(4):1–23, 2026.

Yuanqi Du, Xiaojie Guo, Hengning Cao, Yanfang Ye, and Liang Zhao. Disentangled spatiotemporal graph generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6541–6549, 2022.

Lukas Faber and Roger Wattenhofer. Gwac: Gnns with asynchronous communication. In *Learning on Graphs Conference*, volume 231, pp. 8:1–8:20. PMLR, 2024.

R. Farina, S. Dobricic, A. Storto, S. Masina, and S. Cuomo. A revised scheme to compute horizontal covariances in an oceanographic 3d-var assimilation system. *Journal of Computational Physics*, 284:631–647, 2015.

Ben Finkelshtein, Xingyue Huang, Michael M. Bronstein, and Ismail Ilkan Ceylan. Cooperative graph neural networks. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 13633–13659. PMLR, 2024.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pp. 1263–1272. Pmlr, 2017.

Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning (ICML)*, pp. 12252–12267. PMLR, 2023.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

Xinlei Huang, Zhiqi Ma, Dian Meng, Yanran Liu, Shiwei Ruan, Qingqiang Sun, Xubin Zheng, and Ziyue Qiao. Praga: prototype-aware graph adaptive aggregation for spatial multi-modal omics analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 326–333, 2025.

Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066, 2024.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Huaiyuan Liu, Donghua Yang, Xianzhang Liu, Xinglei Chen, Zhiyu Liang, Hongzhi Wang, Yong Cui, and Jun Gu. Todynet: temporal dynamic graph neural network for multivariate time series classification. *Information Sciences*, 677:120914, 2024.

Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems (NIPS)*, 34: 9720–9733, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Yuan Mi, Pu Ren, Hongteng Xu, Hongsheng Liu, Zidong Wang, Yike Guo, Ji-Rong Wen, Hao Sun, and Yang Liu. Conservation-informed graph learning for spatiotemporal dynamics prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1056–1067, 2025.

Apurva Narayan and Peter H. O'N Roe. Learning graph dynamics using deep neural networks. *Ifac-Papersonline*, 51(2):433–438, 2018.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Pu Ren, Jialin Song, Chengping Rao, Qi Wang, Yike Guo, Hao Sun, and Yang Liu. Learning spatiotemporal dynamics from sparse data via a high-order physics-encoded network. *Computer Physics Communications*, 312:109582, 2025.

T. Konstantin Rusch, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. Gradient gating for deep multi-rate learning on graphs. In *International Conference on Learning Representations (ICLR)*, 2023.

Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12371–12381, 2022.

Olga Solodova, Nick Richardson, Deniz Oktay, and Ryan P Adams. Graph neural networks gone hogwild. *arXiv preprint arXiv:2407.00494*, 2024.

Oguzhan Teke and Palghat P Vaidyanathan. Random node-asynchronous updates on graphs. *IEEE Transactions on Signal Processing*, 67(11):2794–2809, 2019.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *International Conference on Learning Representations (ICLR)*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NIPS)*, 30, 2017.

Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 international conference on management of data*, pp. 2628–2638, 2021.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning (ICML)*, 2024.

Ying Wu, Zhiyuan Zang, Xitao Zou, Wentao Luo, Ning Bai, Yi Xiang, Weiwei Li, and Wei Dong. Graph attention and kolmogorov–arnold network based smart grids intrusion detection. *Scientific Reports*, 15(1):8648, 2025.

Serhiy Yanchuk and Giovanni Giacomelli. Spatio-temporal phenomena in complex systems with time delays. *Journal of Physics A: Mathematical and Theoretical*, 50(10):103001, 2017.

Bowen Yang, Jian Zhang, Jonathan Li, Christopher Ré, Christopher Aberger, and Christopher De Sa. Pipemare: Asynchronous pipeline parallel dnn training. *Proceedings of Machine Learning and Systems*, 3:269–296, 2021.

Shiyu Yang, Qunyong Wu, and Mengmeng Li. Decoupled multi-spatio-temporal fusion graph convolutional recurrent network for traffic prediction. *Engineering Applications of Artificial Intelligence*, 163:112956, 2026.

Rui Ye, Yuqing Hou, Te Lei, Yunxing Zhang, Qing Zhang, Jiale Guo, Huaiwen Wu, and Hengliang Luo. Dynamic graph construction for improving diversity of recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pp. 651–655, 2021.

Xiuyuan Zhao, Yongbin Yang, Jingyun Yang, and Jiaying Chen. Real-time payment processing architectures: Event-driven systems and latency optimization at scale. *Journal of Banking and Financial Dynamics*, 9(12):10–21, 2025.

APPENDIX

AI USE STATEMENT

Generative AI tools were used to assist with language editing, improving readability, and refining the presentation of the manuscript. They were not used to develop the proposed methodology, formulate mathematical claims, conduct experiments, or interpret experimental results. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the content, claims, and conclusions of this work.

ETHICS STATEMENT

This work develops general-purpose machine learning methods for modeling physical dynamical systems. The experiments do not involve human subjects, personally identifiable information, or sensitive personal data. The datasets used in this study are either publicly available benchmarks or numerical simulation data. We are not aware of any direct ethical concerns arising from the methodology or experiments presented in this work.

REPRODUCIBILITY STATEMENT

We have taken several steps to facilitate reproducibility. The proposed model architecture, asynchronous message-passing mechanism, and training objectives are described in the main paper, with additional algorithmic and implementation details provided in the appendix. The appendix further reports dataset preprocessing, model configurations, hyperparameters, training protocols, baseline settings, and additional experimental results. The theoretical properties are stated with their assumptions and corresponding derivations or proofs. We will release the implementation and experimental configurations necessary to reproduce the reported results.

A SCOPE AND NOTATION

This appendix provides the complete scheduling algorithm, precise model definitions, numerical-integration variants, assumptions, and proofs supporting the methodology in the main paper. The notation follows the main paper throughout and is summarized in Table S1. In particular, ℓ indexes processor micro-steps within one physical interval. Vectors are treated as row vectors unless stated otherwise, and the rows of $\boldsymbol{H}^{\ell}$ contain the node-level latent states. Norms without an explicit subscript denote the norm appropriate to the corresponding space. All learnable node, edge, gate, and operator maps share parameters across nodes, edges, physical forecasting steps, and processor micro-steps unless stated otherwise.

B SYNCHRONOUS GRAPH PROPAGATION

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node is associated with a time-dependent state vector $\mathbf{u}_i^t \in \mathbb{R}^d$. Let h_i^{ℓ} denote the node embedding at layer l and $\mathcal{N}(i) = \{j \mid (i,j) \in \mathcal{E}\}$ denote the set of neighbors of node i. In a standard synchronous graph propagation layer, the message from node i to j is defined as

$$m_{j\to i}^{\ell+1} = \phi_e\left(h_j^{\ell} \| h_i^{\ell} \| m_{j\to i}^{\ell}\right), \tag{S1}$$

where ϕ_e is the edge-type function. Through applying a fixed aggregation operator, we compute

$$h_i^{\ell+1} = \phi_v\Big(h_i^{\ell} \,\|\, \mathrm{Agg}\big(\{m_{j\to i}^{\ell+1} \mid j \in \mathcal{N}(i)\}\big)\Big), \tag{S2}$$

where ϕ_v is a node-type function and $j \in \mathcal{N}(i)$. This synchronous mechanism (Gilmer et al., 2017) enforces all nodes to participate in every propagation layer, regardless of whether meaningful information exchange is required at that step. Such a design implicitly assumes uniform update frequency and homogeneous information flow across the graph, which may be suboptimal in scenarios involving irregular sampling, heterogeneous interactions, or delayed dependencies.

Table S1: Core notation for the operator-decomposed and asynchronous schemes.

Symbol	Meaning
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Directed computational graph with node set $\mathcal{V}$ and directed edge set $\mathcal{E}$.
$\mathcal{E}_{\mathrm{u}}$	One fixed orientation $e = (i, j)$ representing each reciprocal physical edge pair in edge-flux operators.
$\boldsymbol{B} \in \{-1, 0, 1\}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{E}_{\mathrm{u}}\rvert}$	Oriented node–edge incidence matrix: $B_{i,e} = -1$ at the tail and $B_{j,e} = +1$ at the head.
t_k	Physical time.
ℓ	Processor micro-step index, $\ell \in \{0, \dots, L-1\}$.
$\boldsymbol{U}, \boldsymbol{H}^{\ell}$	Observed physical state and node-latent matrix, respectively.
$\square$	Edge-operator index, $\square \in \{\mathrm{T}, \mathrm{D}\}$.
$\boldsymbol{e}_{ij}^{\ell}, \boldsymbol{e}_{ij,\square}^{\ell}$	Shared edge feature and its operator-specific adaptation.
$\boldsymbol{\xi}_{\square}^{\ell}$	Optional recurrent state of edge e for operator $\square$; distinct from the deferred buffer.
$\boldsymbol{F}_e^{\square}$	Oriented latent flux generated by edge operator $\square$.
$\Delta \boldsymbol{m}_{\square}^{\ell}$	Candidate latent increment for one edge, operator, and micro-step.
$\tau_{\square}^{\ell}$	Relative priority clock; a smaller value has higher delivery priority.
$\gamma_{\square}^{\ell}$	Quantile threshold used to construct the delivery mask for operator $\square$.
$a_{\square}^{\ell}$	Binary delivery indicator.
$\boldsymbol{z}_{\square}^{\ell}$	Componentwise deferred-accumulation gate.
$\boldsymbol{b}_{\square}^{\ell}$	Deferred increment buffer before micro-step ℓ.
$\mathcal{P} = \Pi_{\mathcal{C}} \circ \mathcal{G}_{\theta}$	Global refinement and optional explicit projection block.
$\boldsymbol{d}_{\square}^{\ell}$	Increment delivered at micro-step ℓ.
$\rho_{\square}^{\ell}$	Target activation fraction for operator $\square$.
Δt_{ℓ}	Physical duration represented by micro-step ℓ.
$\boldsymbol{D}_{\square}^{\ell}$	Matrix stacking delivered increments $\boldsymbol{d}_{\square}^{\ell}$ over $e \in \mathcal{E}_{\mathrm{u}}$.
$\mathcal{T}, \mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{Q}, \mathcal{P}$	Transport, diffusion, reaction, skew-symmetric coupling, source–sink forcing, and projection.

C Asynchronous Scheduling

Unlike conventional synchronous message passing where all edges participate in every propagation layer, we propose a masked asynchronous propagation scheme that regulates the timing and content of message aggregation. The proposed mechanism consists of four components: (i) message priority score estimation, (ii) edge-level delivery scheduling, (iii) feature-level deferred accumulation, and (iv) terminal full refresh. Note that, in this part, the symbols are same in the main body for ease of understanding and comparison.

Message Priority Score: For each directed edge $(i, j) \in \mathcal{E}$, the edge network first generates a message representation:

$$m_{j\to i}^{\ell+1} = \phi_e \left(h_j^{\ell} \| h_i^{\ell} \| m_{j\to i}^{\ell} \right), \tag{S3}$$

where ϕ_e denotes the edge update function and $m_{j\to i}^{\ell}$ is the historical message state. The propagation priority of each edge is determined by an interaction-aware delay score:

$$\tau_{j\to i}^{\ell} = \frac{d_e}{\left\| m_{j\to i}^{\ell+1} \right\|_2 + \varepsilon}, \tag{S4}$$

where d_e denotes the distance between nodes j and i, and ε is a small constant for numerical stability.

A smaller value of $\tau_{j\to i}^{\ell}$ indicates a higher priority for message delivery. Intuitively, messages associated with shorter distances or stronger interactions are propagated earlier, while weaker or distant interactions can be temporarily deferred. Importantly, $\tau_{j\to i}^{\ell}$ serves as a relative scheduling score rather than a physical travel time, since the latent message representation does not necessarily possess physical units.

Edge-level Delivery Mask: Based on the propagation scores, edges are selectively activated at each intermediate propagation layer. Specifically, the delivery mask is defined as $a_{j\to i}^{\ell} \in \{0, 1\}$, where $a_{j\to i}^{\ell} = 1$ indicates that the message carried by edge (j, i) is aggregated at layer l. The active edge

set is constructed according to the percentile rank of the arrival scores:

$$\mathcal{E}^{\ell} = \left\{(j,i) \in \mathcal{E} \mid \mathrm{rank}(\tau^{\ell}_{j\to i}) \le \rho^{\ell}\right\}, \tag{S5}$$

where ρ^{ℓ} controls the proportion of activated edges. The delivery mask can therefore be written as

$$a^{\ell}_{j\to i} = \mathbb{I}\left[(j,i) \in \mathcal{E}^{\ell}\right]. \tag{S6}$$

Compared with synchronous propagation, the proposed scheduling strategy allows different edges to communicate at different propagation layers, thereby reducing unnecessary repeated updates. For an activated edge, the delivered message is computed as

$$m^{\ell+1}_{j\to i} \leftarrow m^{\ell+1}_{j\to i} + b^{\ell}_{j\to i}, \tag{S7}$$

where $b^{\ell}_{j\to i}$ denotes the deferred message buffer.

Feature-level Accumulation Mask: Based on the delivery mask, we employ an edge-level accumulation mask to selectively preserve informative message components. Specifically, for each deferred message, an accumulation mask is generated: $z^{\ell+1}_{j\to i} \in \{0,1\}^d$, where element $z^{\ell+1}_{j\to i}$ determines whether the corresponding edge component participates in deferred accumulation.

For an inactive edge $a^{\ell}_{j\to i} = 0$, the message buffer is updated as

$$b^{\ell+1}_{j\to i} = b^{\ell}_{j\to i} + z^{\ell+1}_{j\to i} \odot m^{\ell+1}_{j\to i}. \tag{S8}$$

For activated edges, after message delivery, the corresponding buffer is cleared:

$$b^{\ell+1}_{j\to i} = 0, \quad a^{\ell}_{j\to i} = 1. \tag{S9}$$

Therefore, the node representation is finally updated by

$$h^{\ell+1}_{v} = \phi_v\left(h^{\ell}_{v} \| \ \mathrm{Agg}\left(\{m^{\ell+1}_{j\to i} | (j,i) \in \mathcal{E}^{\ell}\}\right)\right), \tag{S10}$$

where ϕ_v denotes the node update function.

Terminal Full Refresh: Although intermediate propagation layers adopt selective edge activation, the final propagation layer performs a complete message refresh to ensure that all spatial interactions are incorporated before generating the final prediction. Specifically, the final active edge set is defined as $\mathcal{E}^{L-1} = \mathcal{E}$, which is equivalent to

$$a^{L-1}_{j\to i} = 1, \quad \forall (j,i) \in \mathcal{E}. \tag{S11}$$

Therefore, all deferred messages accumulated during intermediate layers are delivered before the final node representation is produced.

This terminal synchronization mechanism distinguishes the proposed method from fully asynchronous systems with indefinite message staleness. The proposed scheme only modifies the intermediate information arrival order while preserving complete information exchange at the end of propagation.

Relation to Other Asynchronous Methods: Existing asynchronous or asynchrony-related GNNs can be broadly classified according to the role of asynchrony. Logical propagation methods allow nodes or messages to be updated at different propagation steps (Chen et al., 2024; Bose & Das, 2025; Faber & Wattenhofer, 2024), whereas multi-rate and delayed-propagation methods regulate the update rate, communication path, or effective propagation speed (Rusch et al., 2023; Gutteridge et al., 2023; Azabou et al., 2023; Finkelshtein et al., 2024). Event-driven methods instead exploit sparse temporal events to reduce recomputation or inference latency (Schaefer et al., 2022; Wang et al., 2021). A separate line of research studies robustness to uncontrolled execution orders and stale states in distributed systems (Solodova et al., 2024).

Our method belongs to the first category but introduces a distinct deferred-aggregation mechanism. It jointly uses an edge-level delivery mask to determine *when* a message is communicated and a feature-level accumulation mask to determine *what* is retained while that message is deferred. The selected components are stored in an explicit edge-wise buffer and are released upon the next activation; a terminal full refresh ensures that no buffered information remains at the end of each forecasting step. Thus, the proposed asynchrony describes a controlled logical propagation schedule on the original physical graph, rather than asynchronous hardware execution. Its primary purpose is to regulate information evolution during autoregressive physical forecasting.

D COMPLETE MODEL SPECIFICATION

Autoregressive State Transition: The encoder consumes one observed state, static attributes, and optional time information:

$$\boldsymbol{h}_i^0 = \mathrm{Enc}_\theta(\boldsymbol{u}_i \,\|\, \boldsymbol{s}_i \,\|\, t). \tag{S12}$$

After L micro-steps and projections,

$$\widehat{\mathbf{U}}_{t_{k+1}} = \mathbf{U}_{t_k} + \mathrm{Dec}_\theta(\boldsymbol{H}^L), \quad \mathbf{U}_{t_{k+1}} \leftarrow \widehat{\mathbf{U}}_{t_{k+1}}. \tag{S13}$$

The residual connection is omitted if state semantics favor direct decoding. During training, scheduled sampling can interpolate between the ground truth and prediction supplied to the next rollout step; at evaluation time the transition is fully autoregressive.

For each directed representation (i, j), define

$$\boldsymbol{e}_{ij}^\ell = \phi_{\mathrm{sh}}\big(\boldsymbol{h}_i^\ell \,\|\, \boldsymbol{h}_j^\ell \,\|\, (\boldsymbol{h}_j^\ell - \boldsymbol{h}_i^\ell) \,\|\, \boldsymbol{g}_{ij}\big)\,. \tag{S14}$$

Lightweight adapters produce

$$\boldsymbol{e}_{ij,\square}^\ell = \phi_\square(\boldsymbol{e}_{ij}^\ell), \quad \square \in \{\mathrm{T}, \mathrm{D}\}. \tag{S15}$$

Transport and Diffusion Heads: For a representative edge $(i, j) \in \mathcal{E}_{\mathrm{u}}$, transport uses

$$c_e^\ell = \phi_c(\boldsymbol{e}_{ij,\mathrm{T}}^\ell, \boldsymbol{g}_{ij}), \quad \boldsymbol{F}_e^{\mathrm{T}} = c_e^+ \phi_{\mathrm{T}}^+(\boldsymbol{h}_i^\ell) + c_e^- \phi_{\mathrm{T}}^-(\boldsymbol{h}_j^\ell), \tag{S16}$$

where $c^+ = \max(c, 0)$ and $c^- = \min(c, 0)$. Implementations that store one representative per reciprocal pair keep this orientation fixed throughout scheduling and incidence scattering. If both directions are stored, the reverse flux must be constructed consistently rather than independently assumed to be the negative of the forward flux: Eq. S16 alone does not imply antisymmetry when $\phi_{\mathrm{T}}^+ \neq \phi_{\mathrm{T}}^-$.

Diffusion uses reciprocal candidates:

$$\widetilde{\kappa}_{ij}^\ell = \mathrm{softplus}\big[\phi_\kappa(\boldsymbol{e}_{ij,\mathrm{D}}^\ell, \boldsymbol{g}_{ij})\big]\,, \quad \kappa_{ij}^\ell = \tfrac{1}{2}(\widetilde{\kappa}_{ij}^\ell + \widetilde{\kappa}_{ji}^\ell), \quad \boldsymbol{F}_e^{\mathrm{D}} = \kappa_{ij}^\ell \left[\phi_{\mathrm{D}}(\boldsymbol{h}_i^\ell) - \phi_{\mathrm{D}}(\boldsymbol{h}_j^\ell)\right]. \tag{S17}$$

Consequently $\kappa_{ij} \geq 0$ and the same edge flux is scattered to its endpoints with opposite incidence signs. Computing κ_{ij} requires both directed candidates; a genuinely directed graph must therefore add the reciprocal candidate for diffusion or define diffusion on an associated undirected graph.

Node-Local Operators: The three local generators are

$$\mathcal{R}_i(\boldsymbol{h}_i^\ell) = \tanh[\phi_{\mathrm{R}}(\boldsymbol{h}_i^\ell)], \tag{S18}$$

$$\mathcal{S}_i(\boldsymbol{h}_i^\ell) = \boldsymbol{h}_i^\ell \boldsymbol{J}_\theta^\top, \quad \boldsymbol{J}_\theta = \boldsymbol{U}\boldsymbol{V}^\top - \boldsymbol{V}\boldsymbol{U}^\top, \tag{S19}$$

$$\mathcal{Q}_i(\boldsymbol{h}_i^\ell, \boldsymbol{u}_i, t) = \tanh[\phi_{\mathrm{Q}}(\boldsymbol{h}_i^\ell \,\|\, \boldsymbol{u}_i \,\|\, t)]. \tag{S20}$$

The rank of $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{d_h \times r}$ controls the cost of skew coupling. Since $\boldsymbol{J}_\theta^\top = -\boldsymbol{J}_\theta$, $\langle \boldsymbol{h}_i, \mathcal{S}_i(\boldsymbol{h}_i)\rangle = 0$ in continuous time; this is a structural latent-space property and does not by itself establish Hamiltonian conservation of the decoded physical state. The source operator is disabled when observable forcing or controls are unavailable.

Specifically, the skew-coupling operator $\mathcal{S}$ in Eq. S19 is integrated through the additive Euler update in Eq. S27. Although the continuous-time generator satisfies $\langle \boldsymbol{h}, \mathcal{S}(\boldsymbol{h})\rangle = 0$, explicit Euler does not guarantee exact discrete norm preservation. We therefore interpret $\mathcal{S}$ as a skew-symmetric latent coupling rather than claiming exact Hamiltonian or norm conservation.

Operator-Specific Priority Clocks: Candidate increments are

$$\Delta \boldsymbol{m}_\square^\ell = \Delta t_\ell \boldsymbol{F}_e^\square(\boldsymbol{H}^\ell), \quad \sum_{\ell=0}^{L-1} \Delta t_\ell = t_{n+1} - t_n. \tag{S21}$$

Our default scores are

$$\tau_{\mathrm{T}}^\ell = \frac{d_e}{\|\boldsymbol{F}_e^{\mathrm{T}}\|_2 + \varepsilon_T}, \quad \tau_{\mathrm{D}}^\ell = \frac{d_e^2}{\|\boldsymbol{F}_e^{\mathrm{D}}\|_2 + \varepsilon_D}. \tag{S22}$$

The advective and diffusive distance powers encode different characteristic scaling. These quantities define relative order only; they should not be interpreted as physical time unless the latent features have been dimensionally calibrated.

For a prescribed fraction $\rho_{\square}^{\ell}$, let

$$\gamma_{\square}^{\ell} = \text{Quantile}_{\rho_{\square}^{\ell}}(\{\tau_{\square}^{\ell} : e \in \mathcal{E}_{\text{u}}\}), \quad a_{\square}^{\ell} = \mathbb{I}[\tau_{\square}^{\ell} \le \gamma_{\square}^{\ell}]. \tag{S23}$$

Eq. S23 defines the operator-specific activation rule. At micro-step ℓ, $\gamma_{\square}^{\ell}$ is the $\rho_{\square}^{\ell}$-quantile of the priority scores for operator $\square \in \{\text{T}, \text{D}\}$. Since a smaller score indicates higher delivery priority, edges satisfying $\tau_{\square}^{\ell} \le \gamma_{\square}^{\ell}$ are activated. Thus, approximately the lowest-scoring $\rho_{\square}^{\ell}$ fraction of edges is delivered, while the remaining edges are deferred. Ties at the threshold may cause the realized activation fraction to slightly exceed $\rho_{\square}^{\ell}$.

The threshold rule is invariant to the stored edge order. If an exact cardinality is required, ties may be resolved using random keys sampled independently of node labels and carried along under relabeling. Activating the complete tie group is not the deterministic default.

Accumulation Gate, Buffer, and Delivery: A soft gate can be generated as

$$\boldsymbol{z}_{\square}^{\ell} = \text{sigmoid}\big[\phi_{z,\square}\big(\boldsymbol{e}_{\square}^{\ell} \,\|\, \boldsymbol{b}_{\square}^{\ell}\big)\big] \in (0,1)^{d_h}. \tag{S24}$$

For hard component selection, threshold the logits in the forward pass and use either a straight-through estimator or a continuous relaxation in the backward pass.

With $\boldsymbol{b}_{\square}^{0} = \boldsymbol{0}$, the complete state transition is

$$\boldsymbol{d}_{\square}^{\ell} = a_{\square}^{\ell}(\Delta \boldsymbol{m}_{\square}^{\ell} + \boldsymbol{b}_{\square}^{\ell}), \quad \boldsymbol{b}_{\square}^{\ell+1} = (1 - a_{\square}^{\ell})(\boldsymbol{b}_{\square}^{\ell} + \boldsymbol{z}_{\square}^{\ell} \odot \Delta \boldsymbol{m}_{\square}^{\ell}). \tag{S25}$$

Thus selected components of an inactive increment are retained; unselected components are discarded. For an active edge, the current increment and all retained history are delivered, after which the buffer is exactly zero.

Terminal Refresh: At the last micro-step,

$$a_{\square}^{L-1} = 1 \quad \text{for all } e \in \mathcal{E}_{\text{u}},\ \square \in \{\text{T}, \text{D}\}. \tag{S26}$$

Terminal refresh prevents buffers from crossing a physical forecasting boundary. If cross-step state is useful, it should be represented by the explicit recurrent edge state $\boldsymbol{\xi}$, not by an uncleared increment buffer.

E Numerical Integration and Operator Composition

Additive First-Order Update: Let $\boldsymbol{D}_{\square}^{\ell}$ stack delivered edge increments by edge. Since $\boldsymbol{B}\boldsymbol{D}_{\square}$ already includes Δt_{ℓ}, a first-order update is

$$\dot{\boldsymbol{H}}^{\ell} = \boldsymbol{H}^{\ell} + \boldsymbol{B}\boldsymbol{D}_{\text{T}}^{\ell} + \boldsymbol{B}\boldsymbol{D}_{\text{D}}^{\ell} + \Delta t_{\ell}\left[\mathcal{R}(\boldsymbol{H}^{\ell}) + \mathcal{S}(\boldsymbol{H}^{\ell}) + \mathcal{Q}(\boldsymbol{H}^{\ell}, \boldsymbol{U}, t_{\ell})\right]. \tag{S27}$$

This is the simplest implementation and the reference used by Algorithm 1.

Optional Operator Composition: Let $\Phi_{\square}^{\Delta t}$ denote a numerical flow for operator $\square$. A Lie composition is

$$\dot{\boldsymbol{H}}^{\ell} = \Phi_{\text{Q}}^{\Delta t_{\ell}} \circ \Phi_{\text{S}}^{\Delta t_{\ell}} \circ \Phi_{\text{R}}^{\Delta t_{\ell}} \circ \widetilde{\Phi}_{\text{D}}^{\Delta t_{\ell}} \circ \widetilde{\Phi}_{\text{T}}^{\Delta t_{\ell}}(\boldsymbol{H}^{\ell}), \tag{S28}$$

where the tildes indicate that edge flows use the increments delivered by the asynchronous scheduler. A symmetric composition may be used when the subflows support it, for example

$$\Phi_{\text{sym}}^{\Delta t} = \Phi_{\text{R}}^{\Delta t/2} \circ \Phi_{\text{D}}^{\Delta t/2} \circ \Phi_{\text{T}}^{\Delta t} \circ \Phi_{\text{D}}^{\Delta t/2} \circ \Phi_{\text{R}}^{\Delta t/2}, \tag{S29}$$

with $\mathcal{S}$ and $\mathcal{Q}$ inserted according to the application. A claimed order of accuracy requires the masks to be held fixed over the paired sub-stages; state-dependent rescheduling can otherwise break the standard splitting-order argument.

F Complete Algorithm

The complete procedure is summarized in Algorithm 1. It combines shared edge encoding, operator-specific priority scheduling, deferred accumulation, terminal full refresh, numerical integration, projection, and autoregressive decoding within one physical prediction step.

Algorithm 1 Operator-Decomposed Asynchronous Step

Input: $\mathcal{G}$; $\mathbf{U}_{t_k}$; $\boldsymbol{u}_i$; L; Δt_ℓ; ρ^ℓ_{T}; ρ^ℓ_{D};
Output: $\widehat{\mathbf{U}}_{t_{k+1}}$

$\boldsymbol{H}^0 \leftarrow \mathrm{Enc}_\theta(\mathbf{U}, \boldsymbol{S}, t)$
for all $e \in \mathcal{E}_{\mathrm{u}}$ and $\square \in \{\mathrm{T}, \mathrm{D}\}$ **do**
  $\boldsymbol{b}^0_\square \leftarrow \mathbf{0}$
end for
for $\ell = 0, \ldots, L-1$ **do**
  Build shared features $\boldsymbol{e}^\ell_{ij}$ using Eq. S14
  for all $e \in \mathcal{E}_{\mathrm{u}}$ **do**
    Evaluate $\boldsymbol{F}^{\mathrm{T}}_e$ and $\boldsymbol{F}^{\mathrm{D}}_e$ using Eq. S16 and Eq. S17
    for all $\square \in \{\mathrm{T}, \mathrm{D}\}$ **do**
      $\Delta\boldsymbol{m}^\ell_\square \leftarrow \Delta t_\ell \boldsymbol{F}^\square_e$; Evaluate $\tau^\ell_\square$ and $\boldsymbol{z}^\ell_\square$
    end for
  end for
  if $\ell = L-1$ **then**
    $a^\ell_\square \leftarrow 1$ for every $\square$
  else
    for all $\square \in \{\mathrm{T}, \mathrm{D}\}$ **do**
      $\gamma^\ell_\square \leftarrow \mathrm{Quantile}_{\rho^\ell_\square}(\{\tau^\ell_\square\}_{e \in \mathcal{E}_{\mathrm{u}}})$; $a^\ell_\square \leftarrow \mathbb{I}[\tau^\ell_\square \le \gamma^\ell_\square]$ for every e
    end for
  end if
  for all $e \in \mathcal{E}_{\mathrm{u}}$ and $\square \in \{\mathrm{T}, \mathrm{D}\}$ **do**
    $\boldsymbol{d}^\ell_\square \leftarrow a^\ell_\square(\Delta\boldsymbol{m}^\ell_\square + \boldsymbol{b}^\ell_\square)$; $\boldsymbol{b}^{\ell+1}_\square \leftarrow (1 - a^\ell_\square)(\boldsymbol{b}^\ell_\square + \boldsymbol{z}^\ell_\square \odot \Delta\boldsymbol{m}^\ell_\square)$
  end for
  $\Delta\boldsymbol{H}^\ell_\square \leftarrow \boldsymbol{B}\boldsymbol{D}^\ell_\square$ for $\square \in \{\mathrm{T}, \mathrm{D}\}$; Advance $\boldsymbol{H}^\ell \mapsto \dot{\boldsymbol{H}}^\ell$ using Eq. S27; $\boldsymbol{H}^{l+1} \leftarrow \mathcal{P}(\dot{\boldsymbol{H}}^l)$
end for
$\widehat{\mathbf{U}}_{t_{k+1}} \leftarrow \mathbf{U}_{t_k} + \mathrm{Dec}_\theta(\boldsymbol{H}^L)$
Return $\widehat{\mathbf{U}}_{t_{k+1}}$

G Assumptions, Restrictions, and Claim Scope

Table S2 consolidates the conditions required by the theoretical statements. They are restrictions of the corresponding claims, not properties that automatically hold for every dataset or implementation.

Scope of the stability statement. The rollout corollary is conditional on A7 and does not establish that asynchronous scheduling has a smaller Lipschitz factor than a synchronous model. Likewise, the skew-symmetric generator is norm-neutral in continuous time, whereas the implemented explicit Euler update does not preserve the latent norm exactly. Therefore, no exact discrete conservation property is claimed for $\mathcal{S}$.

H Theoretical Results and Proofs

H.1 Synchronous Consistency

Proposition H.1 (Synchronous consistency)**.** *Suppose $a^\ell_\square = 1$ for all $\square, \ell$ and $\boldsymbol{b}^0_\square = \mathbf{0}$. Then the asynchronous edge update equals the corresponding synchronous operator-decomposed update at every micro-step.*

Proof. When $a^\ell_\square = 1$, Eq. S25 gives $\boldsymbol{b}^{\ell+1}_\square = \mathbf{0}$, irrespective of the accumulation gate. Induction from the zero initial buffer yields $\boldsymbol{b}^\ell_\square = \mathbf{0}$ for all ℓ.

Eq. S25 therefore becomes $\boldsymbol{d}^\ell_\square = \Delta\boldsymbol{m}^\ell_\square$ on every edge. Scattering all candidate increments and applying the same local operators is exactly the synchronous update.

Table S2: Assumptions, structural conditions, and claim restrictions used by the theoretical results.

ID	Assumption, condition, or restriction	Dependent result
A1	The processor depth L is finite, and every deferred buffer is initialized as $\boldsymbol{b}_{\square}^{0} = \mathbf{0}$ at the beginning of each physical forecasting step.	Finite-horizon buffer accounting and prevention of cross-step message staleness.
A2	Candidate increments are uniformly bounded on the rollout domain, $\left\|\Delta \boldsymbol{m}_{\square}^{\ell}\right\|_2 \leq M_{\square}$, and the accumulation gate satisfies $\left\|\boldsymbol{z}_{\square}^{\ell}\right\|_{\infty} \leq 1$.	Finite-buffer bound.
A3	The encoder, node, edge, gate, and operator maps share parameters across nodes and edges; aggregation is permutation invariant; graph attributes and external inputs are relabeled consistently.	Permutation equivariance.
A4	Priority ties are resolved by an equivariant rule; recurrent edge states and buffers are permuted with their corresponding edges; and the global projection and decoder commute with node relabeling.	Permutation equivariance.
A5	For the synchronous special case, $a_{\square}^{\ell} = 1$ for every edge, edge-based operator, and micro-step, with $\boldsymbol{b}_{\square}^{0} = \mathbf{0}$.	Synchronous consistency.
A6	The terminal micro-step activates every edge for every edge-based operator: $a_{\square}^{L-1} = 1$.	Terminal synchronization.
A7	On a common rollout domain, the learned one-step map Ψ_A is K_A-Lipschitz and satisfies the uniform one-step approximation bound $\|\Psi_A(x) - F(x)\| \leq \varepsilon_A$. The recursive trajectories remain inside this domain.	Autoregressive rollout bound.
A8	The incidence matrix $\boldsymbol{B}$ represents internal pairwise edges, with exactly one -1 and one $+1$ in each column.	Zero-sum scattering of delivered edge increments.
A9	Operator names describe architectural inductive biases. Exact physical conservation is claimed only when it is justified by the represented quantity, boundary treatment, projection, and numerical integrator. In particular, the skew-symmetric generator is norm-neutral in continuous time, but the implemented Euler update does not guarantee exact discrete norm preservation.	Interpretation of transport, diffusion, skew-symmetric coupling, and edge-level conservation.

Hence the equality holds at every micro-step and, by composition with the same projection and decoder, for the complete forecasting step. □

H.2 Permutation Equivariance

Definition H.2 (Equivariant tie rule)**.** A tie rule is equivariant if relabeling nodes and the induced edges relabels the selected edge set without changing membership otherwise. Thresholding at a multiset quantile and activating an entire tie group has this property.

Theorem H.3 (Permutation equivariance)**.** *Assume that (i) all learnable node and edge maps share parameters, (ii) edge geometry and external inputs are relabeled with their incident nodes, (iii) aggregation is permutation invariant, (iv) the scheduling tie rule is equivariant, and (v) $\mathcal{P}$ commutes with node relabeling, as do the encoder and decoder. Then the complete asynchronous forecasting step is permutation equivariant.*

Proof. Let $\boldsymbol{\Pi}$ be a node permutation matrix and $\boldsymbol{\Omega}$ its induced permutation on $\mathcal{E}_{\mathrm{u}}$. Relabeling gives $\boldsymbol{H}' = \boldsymbol{\Pi}\boldsymbol{H}$ and $\boldsymbol{B}' = \boldsymbol{\Pi}\boldsymbol{B}\boldsymbol{\Omega}^{\top}$. Because the encoder and shared maps are applied with shared parameters, their node and edge outputs transform as $\boldsymbol{H}'^{0} = \boldsymbol{\Pi}\boldsymbol{H}^{0}$ and $\boldsymbol{R}' = \boldsymbol{\Omega}\boldsymbol{R}$.

If recurrent edge states are used, they are carried with the same edge permutation, $\boldsymbol{\Xi}'_{\square} = \boldsymbol{\Omega}\boldsymbol{\Xi}_{\square}$. The transport orientation data are carried with the edge permutation, and diffusion is pairwise symmetrized, so candidate fluxes and increments satisfy $\Delta\boldsymbol{M}'_{\square} = \boldsymbol{\Omega}\Delta\boldsymbol{M}_{\square}$.

Distances, vector norms, and the multiset quantile are unchanged by reordering. The equivariant tie rule therefore gives $\boldsymbol{a}'_{\square} = \boldsymbol{\Omega}\boldsymbol{a}_{\square}$. The shared gate map likewise gives $\boldsymbol{Z}'_{\square} = \boldsymbol{\Omega}\boldsymbol{Z}_{\square}$. Assuming the induction hypotheses $\boldsymbol{b}'_{\square} = \boldsymbol{\Omega}\boldsymbol{b}_{\square}$ and $\boldsymbol{H}' = \boldsymbol{\Pi}\boldsymbol{H}$, Eq. S25 implies $\boldsymbol{D}'_{\square} = \boldsymbol{\Omega}\boldsymbol{D}_{\square}$ and $\boldsymbol{b}'^{+}_{\square} = \boldsymbol{\Omega}\boldsymbol{b}^{+}_{\square}$. Consequently,

$$\boldsymbol{B}'\boldsymbol{D}'_{\square} = \boldsymbol{\Pi}\boldsymbol{B}\boldsymbol{\Omega}^{\top}\boldsymbol{\Omega}\boldsymbol{D}_{\square} = \boldsymbol{\Pi}\boldsymbol{B}\boldsymbol{D}_{\square}. \tag{S30}$$

The shared node-local maps commute with $\mathbf{\Pi}$, so the induction continues over all micro-steps. Terminal refresh is a constant all-one mask and is therefore equivariant. Finally, the assumed equivariance of projection and decoder yields

$$\Psi_A(\mathbf{\Pi}\boldsymbol{X}, \pi(\mathcal{G})) = \mathbf{\Pi}\Psi_A(\boldsymbol{X}, \mathcal{G}). \tag{S31}$$

□

Remark H.4. Index-based deterministic top-K tie breaking is not generally permutation equivariant. This is why the tie rule is an explicit assumption rather than an implementation detail.

H.3 Finite Deferred Accumulation

Proposition H.5 (Finite buffer)**.** *Suppose* $\left\|\Delta \boldsymbol{m}_{\square}^{\ell}\right\|_2 \le M_{\square}$ *and* $\left\|\boldsymbol{z}_{\square}^{\ell}\right\|_{\infty} \le 1$. *If* $r < \ell$ *is the latest micro-step at which edge* e *delivered operator* $\square$*, then*

$$\left\|\boldsymbol{b}_{\square}^{\ell}\right\|_2 \le (\ell - r - 1)M_{\square}. \tag{S32}$$

If no previous delivery exists, take $r = -1$*, which gives the bound* $\ell M_{\square}$.

Proof. Delivery at step r sets $\boldsymbol{b}_{\square}^{r+1} = \mathbf{0}$. For every subsequent inactive step q, Eq. S25, the triangle inequality, and $\|\boldsymbol{z} \odot \Delta\boldsymbol{m}\|_2 \le \|\boldsymbol{z}\|_{\infty} \|\Delta\boldsymbol{m}\|_2$ give

$$\left\|\boldsymbol{b}_{\square}^{q+1}\right\|_2 \le \left\|\boldsymbol{b}_{\square}^{q}\right\|_2 + M_{\square}. \tag{S33}$$

There are $\ell - r - 1$ such updates before the buffer at step ℓ, which proves Eq. S32. The no-delivery case follows from the zero initialization at $\ell = 0$.

Since L is finite, the bound is finite even when an edge remains inactive throughout all intermediate micro-steps. □

H.4 Terminal Synchronization

Proposition H.6 (Terminal synchronization)**.** *Under Eq. S26,* $\boldsymbol{b}_{\square}^{L} = \mathbf{0}$ *for every* e *and* $\square \in \{\mathrm{T}, \mathrm{D}\}$.

Proof. At $\ell = L - 1$, $a_{\square}^{L-1} = 1$. Substitution into Eq. S25 multiplies the parenthesized buffer update by zero.

At the same step, Eq. S25 includes both the current candidate increment and the complete retained buffer. Hence every retained increment is delivered and every final buffer is exactly zero. □

This property guarantees that asynchronous propagation only changes the intermediate communication schedule, rather than removing any spatial interaction from the final prediction.

H.5 Pairwise Conservation of Edge Increments

Proposition H.7 (Zero-sum edge scattering)**.** *For any delivered edge-increment matrix* $\boldsymbol{D}_{\square}$ *scattered by an oriented incidence matrix,*

$$\mathbf{1}^{\top}\boldsymbol{B}\boldsymbol{D}_{\square} = \mathbf{0}. \tag{S34}$$

Proof. Each column of $\boldsymbol{B}$ has one -1 and one $+1$, so $\mathbf{1}^{\top}\boldsymbol{B} = \mathbf{0}^{\top}$. Multiplying by $\boldsymbol{D}_{\square}$ proves the claim. □

Remark H.8. Proposition H.7 concerns the edge contribution only. Reaction, forcing, projection, boundary replacement, and the decoder may change the global sum. It also does not imply conservation of a physical quantity unless that quantity is represented linearly in the scattered latent channels.

Table S3: Dataset-specific post-integration operators.

Dataset	$\mathcal{G}_\theta$	Learnable	$\Pi_{\mathcal{C}}$	Learnable
2D Burgers	$\mathcal{G}_{\text{att}}$	Yes	Π_{per}	No
3D Gray–Scott	$\mathcal{G}_{\text{att}}$	Yes	Π_{per}	No
2D Black Sea	$\mathcal{G}_{\text{att}}$	Yes	Π_{bc}	No
2D Cylinder Flow	$\mathcal{G}_{\text{att}}$	Yes	Π_{bc}	No

H.6 Autoregressive Error Propagation

Corollary H.9 (Long-horizon rollout bound)**.** *Let Ψ_A be one learned forecasting step and F the true one-step map. Suppose*

$$\|\Psi_A(x) - \Psi_A(y)\| \le K_A \|x - y\|, \tag{S35}$$
$$\|\Psi_A(x) - F(x)\| \le \varepsilon_A \tag{S36}$$

on the rollout domain. If $\widehat{u}_{t_{k+1}} = \Psi_A(\widehat{u}_{t_k})$, $u_{t_{k+1}} = F(u_{t_k})$, and $e_n = \|\widehat{u}_{t_k} - u_{t_k}\|$, then

$$e_{t_{k+H}} \le K_A^H e_{t_k} + \varepsilon_A \sum_{q=0}^{H-1} K_A^q. \tag{S37}$$

For $K_A < 1$, the accumulated term is bounded by $\varepsilon_A/(1-K_A)$. When $K_A = 1$, the same expression reduces to $e_{t_{k+H}} \le e_{t_k} + H\varepsilon_A$.

Proof. Using the rollout definitions, adding and subtracting $\Psi_A(u_{t_k})$, and applying Eq. S35– Eq. S36 gives

$$\begin{aligned} e_{t_{k+1}} &= \|\Psi_A(\widehat{u}_{t_k}) - F(u_{t_k})\| \\ &\le \|\Psi_A(\widehat{u}_{t_k}) - \Psi_A(u_{t_k})\| + \|\Psi_A(u_{t_k}) - F(u_{t_k})\| \\ &\le K_A e_{t_k} + \varepsilon_A. \end{aligned} \tag{S38}$$

Induction over H steps yields Eq. S37. For $K_A \neq 1$, the geometric sum equals $(K_A^H - 1)/(K_A - 1)$; the stated contractive and $K_A = 1$ cases follow immediately. □

Remark H.10. The bound is conditional. It does not establish $K_A < K_S$ for a synchronous model, nor does it compare one-step errors. Any stability advantage must be supported by analysis or measurement of the trained models.

I Projection and Choices

We factor the post-integration transformation into a learnable global refinement $\mathcal{G}_\theta$ and an explicit constraint projection $Pi_{\mathcal{C}}$):

$$\boldsymbol{H}^{\ell+1} = \Pi_{\mathcal{C}} \circ \mathcal{G}_\theta(\dot{\boldsymbol{H}}^{\ell}), \tag{S39}$$

such that $\mathcal{P} = \Pi_{\mathcal{C}} \circ \mathcal{G}_\theta$. The refinement module is chosen from

$$\begin{aligned} \mathcal{G}_{\text{id}}(\boldsymbol{H}) &= \boldsymbol{H}, \\ \mathcal{G}_{\text{att}}(\boldsymbol{H}) &= \phi_{\text{att}}(\boldsymbol{H}), \end{aligned} \tag{S40}$$

whereas explicit constraint operators include

$$\begin{aligned} \Pi_{\text{id}}(\boldsymbol{H}) &= \boldsymbol{H}, \\ \Pi_{\text{box}}(\boldsymbol{H}) &= \text{clip}(\boldsymbol{H}, \boldsymbol{h}_{\min}, \boldsymbol{h}_{\max}), \\ \Pi_{\text{bc}}(\boldsymbol{H}) &= \boldsymbol{M}_{\text{int}} \odot \boldsymbol{H} + \boldsymbol{M}_{\text{bc}} \odot \boldsymbol{H}_{\text{bc}}, \\ \Pi_{\text{per}}(\boldsymbol{H}) &= \boldsymbol{M}_{\text{int}} \odot \boldsymbol{H} + \boldsymbol{M}_{\text{per}} \odot \boldsymbol{H}_{\text{per}}, \\ \Pi_{\text{linear}}(\boldsymbol{H}) &= \boldsymbol{H} - \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{A}^\top)^\dagger \cdot (\boldsymbol{A}\boldsymbol{H} - \boldsymbol{c}). \end{aligned} \tag{S41}$$

Table S4: Dataset information. The split is reported as train/validation/test trajectories.

Dataset	Domain	Δx	Δt	State	Nodes per trajectory	Trajectory length	Sample Split	Boundary condition	Force term
2D Burgers	$[0,1]^2$	0.02	0.001 s	u, v	2,500 (50^2)	1,000	30/5/5	Periodic	–
3D GS	$[0,48]^3$	2	0.25 s	u, v	13,824 (24^3)	3,000	2/2/2	Periodic	–
2D BS	Irregular	–	1 day	u, v, T	1,000	365	20/2/2	–	–
2D CF	Irregular	–	0.01 s	u, v, p	3,000–5000	1000	100/10/10	–	–

These choices correspond to the identity map, component-wise box clipping, boundary replacement, global nonlinear map via self-attention mechanism, and projection onto a linear affine set, respectively.

For boundary replacement, $\boldsymbol{M}_{\mathrm{int}}, \boldsymbol{M}_{\mathrm{bc}} \in \{0,1\}^{|\mathcal{V}|\times 1}$ are complementary masks satisfying $\boldsymbol{M}_{\mathrm{int}} + \boldsymbol{M}_{\mathrm{bc}} = \mathbf{1}$, with broadcasting over latent channels.

For periodic boundaries, $\boldsymbol{M}_{\mathrm{per}}$ denotes the periodic boundary mask, and $\boldsymbol{H}_{\mathrm{per}}$ is obtained by averaging paired states on opposite boundaries, such that $\boldsymbol{H}(0,y) = \boldsymbol{H}(L_x,y)$ and $\boldsymbol{H}(x,0) = \boldsymbol{H}(x,L_y)$.

For the linear affine case, let $\boldsymbol{A} \in \mathbb{R}^{m\times|\mathcal{V}|}$ and $\boldsymbol{c} \in \mathbb{R}^{m\times d_h}$. Provided that the set

$$\mathcal{C} = \{\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}|\times d_h} : \boldsymbol{A}\boldsymbol{H} = \boldsymbol{c}\} \tag{S42}$$

is nonempty, the last line of Eq. S41 is the Euclidean projection onto $\mathcal{C}$. The pseudo-inverse permits redundant or linearly dependent projections; when $\boldsymbol{A}$ has full row rank, $(\boldsymbol{A}\boldsymbol{A}^\top)^\dagger$ reduces to $(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}$.

For the nonlinear refinement case, the map learned by self-attention mechanism (without index-dependent positional embeddings) is not assumed to be orthogonal or idempotent and does not guarantee exact conservation, range projections, or boundary satisfaction.

J Datasets

The experimental suite spans complementary regimes, including directional transport, dissipative smoothing, nonlinear local effects, and irregular physical boundaries. Complete trajectories are partitioned into disjoint training, validation, and test sets. The split is performed at trajectory level to prevent future states of a test trajectory from leaking into training. More details are summarized in Table S4.

J.1 Two-Dimensional Burgers System

We generate the Burgers trajectories by solving the viscous two-dimensional system

$$\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \nu\nabla^2\boldsymbol{u}, \quad \boldsymbol{u} = (u,v)^\top, \tag{S43}$$

on the unit square $\Omega = (0,1)^2$. Periodic boundary conditions are applied in both spatial directions. Each trajectory covers $t \in [0,1]$, with viscosity $\nu = 0.01$.

To obtain diverse smooth initial states, each velocity component is sampled independently from a truncated random Fourier series. For $s \in \{u,v\}$, we define

$$\theta_{pq}(x,y) = 2\pi\left[\left(p - \frac{N}{2}\right)x + \left(q - \frac{N}{2}\right)y\right], \tag{S44a}$$

$$\widetilde{s}_0(x,y) = \sum_{p=0}^{N}\sum_{q=0}^{N}\left[\lambda_{pq}^s \cos\theta_{pq}(x,y) + \gamma_{pq}^s \sin\theta_{pq}(x,y)\right], \tag{S44b}$$

$$s_0(x,y) = \frac{1}{3}\left[2\,\frac{\widetilde{s}_0(x,y)}{\max_{(x,y)\in\Omega}\widetilde{s}_0(x,y)} + c_s\right]. \tag{S44c}$$

We use $N = 10$, draw $\lambda_{pq}^s, \gamma_{pq}^s \sim \mathcal{N}(0,1)$ independently, and sample $c_s \sim \mathcal{N}(-1,1)$. Reference trajectories are computed using a fourth-order Runge–Kutta scheme, following the numerical setup of (Ren et al., 2025).

J.2 THREE-DIMENSIONAL GRAY–SCOTT SYSTEM

The Gray–Scott (GS) dataset describes the interaction of two chemical species through diffusion, nonlinear conversion, feed, and removal. Their concentrations $u(\boldsymbol{x}, t)$ and $v(\boldsymbol{x}, t)$ satisfy

$$\partial_t u = D_u \nabla^2 u - uv^2 + \alpha(1-u), \tag{S45a}$$

$$\partial_t v = D_v \nabla^2 v + uv^2 - (\alpha+\beta)v, \tag{S45b}$$

where $\boldsymbol{x} = (x, y, z) \in \Omega$. We solve this system with $D_u = 0.2, D_v = 0.1, \alpha = 0.025, \beta = 0.055$.

Initial concentrations are obtained by perturbing the homogeneous state $(u, v) = (1, 0)$:

$$u_0(\boldsymbol{x}) = (1 - r_t) + r_t\, \xi_u(\boldsymbol{x}), \quad v_0(\boldsymbol{x}) = r_t\, \xi_v(\boldsymbol{x}), \tag{S46}$$

where $r_t = 0.1$ and the entries of ξ_u and ξ_v are sampled independently from $\mathcal{N}(0, 1)$. A fourth-order Runge–Kutta method is used for temporal integration (Ren et al., 2025).

J.3 TWO-DIMENSIONAL CYLINDER FLOW

The cylinder-flow (CF) dataset contains unsteady incompressible flows past a circular obstacle. The simulations are performed in COMSOL Multiphysics using the flow-past-a-cylinder configuration [1]. The rectangular domain is

$$\Omega = [0, 1.2] \times [0, 0.41]\ \mathrm{m}^2, \quad t \in [0, 10]\ \mathrm{s}. \tag{S47}$$

A no-slip condition is imposed on the solid boundary. Across simulation instances, the mean inlet velocity is fixed at $U_{\mathrm{mean}} = 1\,\mathrm{m\,s^{-1}}$, and the reported dynamic viscosity is $\mu = 10^{-3}\,\mathrm{Pa\,s}$.

Geometric diversity is introduced by the cylinder radius r and center $\boldsymbol{c} = (c_x, c_y)$. They are sampled uniformly as

$$r \sim \mathcal{U}(0.04, 0.081)\ \mathrm{m}, \tag{S48}$$

$$\boldsymbol{c} \sim \mathcal{U}([0.1, 0.31] \times [0.1, 0.31])\ \mathrm{m}. \tag{S49}$$

The resulting cases cover the reported Reynolds-number range $\mathrm{Re} \in [800, 1600]$, producing distinct wake evolutions and periodic vortex-shedding patterns.

J.4 BLACK SEA OCEAN DATASET

The Black Sea (BS) dataset [2] is derived from simulations produced by the Nucleus for European Modelling of the Ocean (NEMO) circulation model [3]. Atmospheric surface fluxes are computed from ECMWF ERA5 fields with a spatial resolution of 0.25° and an hourly temporal resolution from June 1, 1993 to June 30, 2021 (Hersbach et al., 2020). NEMO is coupled online with the OceanVar assimilation system (Farina et al., 2015), which assimilates along-track sea-level-anomaly observations distributed by the Copernicus Marine Service [4]. For graph-based learning, we uniformly select $1{,}000$ ocean mesh locations from the more than $40{,}000$ locations in the original computational domain. Model outputs are resampled to one snapshot per day, and the resulting time series are organized into annual sequences.

K BASELINES

K.1 DEEPONET

DeepONet (Lu et al., 2021) is a popular neural operator-based PDE solver. It comprises two distinct sub-networks: the branch network and the trunk network. They are responsible for encoding the input function and the value of the output function at a specific position, respectively. DeepONet significantly minimizes generalization errors and efficiently learns operators from the relatively small dataset.

[1] https://cn.comsol.com/model/flow-past-a-cylinder-97

[2] https://data.marine.copernicus.eu/product/BLKSEA_MULTIYEAR_PHY_007_004/description

[3] https://doi.org/10.5281/zenodo.1464816

[4] https://marine.copernicus.eu

K.2 MeshGraphNet (MGN)

MGN (Pfaff et al., 2021) serves as a representative graph-based simulator for mesh-discretized physical systems. Its encoder–processor–decoder pipeline first embeds node and edge attributes into a latent space, repeatedly exchanges information along mesh edges, and finally maps the processed node features to physical state updates. Since the computational graph follows the original mesh connectivity, MGN is a natural reference model for evaluating dynamics prediction on irregular spatial discretizations.

K.3 Message-Passing Neural PDE Solver (MP-PDE)

MP-PDE (Brandstetter et al., 2022) combines graph message passing with training strategies tailored to autoregressive PDE forecasting. In particular, temporal bundling allows several future states to be predicted jointly, while push-forward training constructs training inputs from intermediate model predictions. These mechanisms expose the solver to rollout-like inputs during optimization and alleviate the distribution mismatch between teacher-forced training and recursive inference. We include MP-PDE as a graph-based baseline specifically designed for stable long-horizon simulation.

K.4 Fourier Neural Operator (FNO)

FNO (Li et al., 2021) learns a resolution-compatible mapping between input and output fields by parameterizing nonlocal interactions in the spectral domain. For an input field $v : \Omega \to \mathbb{R}^{d_v}$, a general integral operator takes the form

$$(\mathcal{K}v)(\mathbf{x}) = \int_{\Omega} \kappa(\mathbf{x}, \mathbf{y}) v(\mathbf{y}) \, \mathrm{d}\mathbf{y}, \tag{S50}$$

where κ is a learnable kernel. FNO approximates this nonlocal operation using a finite number of Fourier modes. A typical spectral layer is written as

$$v^{\ell+1}(\mathbf{x}) = \sigma\left(\mathbf{W}^{\ell} v^{\ell}(\mathbf{x}) + \mathcal{F}^{-1}\left[\mathbf{R}^{\ell} \odot \mathcal{F}\big(v^{\ell}\big)\right](\mathbf{x})\right), \tag{S51}$$

where $\mathbf{W}^{\ell}$ represents a local linear transformation, $\mathbf{R}^{\ell}$ contains the learnable spectral coefficients of the retained modes, and σ denotes a nonlinear activation. We use the two- or three-dimensional FNO implementation according to the spatial dimension of each regular-grid dataset.

K.5 Factorized Fourier Neural Operator (FFNO)

FFNO (Tran et al., 2023) replaces dense multidimensional spectral mixing with a sequence of axis-wise Fourier operations. The resulting factorization lowers the parameter and computational requirements of the spectral layers while maintaining a domain-wide receptive field. It is included to compare our method against a more economical Fourier-based operator on regular discretizations.

K.6 Vanilla Transformer

We also consider a standard Transformer (Vaswani et al., 2017) as a global-interaction baseline. Given query, key, and value matrices, its scaled dot-product attention is

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{S52}$$

where d denotes the query–key dimension. Unlike graph processors whose receptive fields expand through repeated local propagation, self-attention directly couples every pair of input tokens within a layer. This baseline evaluates the benefit of unrestricted global interaction for capturing long-range spatial dependencies.

K.7 Transolver

Transolver (Wu et al., 2024) introduces physics-attention to model physical fields on both structured and unstructured discretizations. It first aggregates spatial points into a compact collection of learned

physical slices, carries out attention among the slice-level representations, and then distributes the updated information back to the original points. By performing global interaction in the reduced slice space, Transolver avoids the full cost of pointwise attention and can accommodate geometries with different spatial organizations. We adopt it as a recent attention-based neural solver for comparison with our graph propagation framework.

K.8 Conservation-informed Graph Neural Network (CiGNN)

CiGNN (Mi et al., 2025) proposes a graph-based framework that incorporates general conservation principles directly into neural message passing for spatiotemporal dynamics prediction. By imposing symmetry-inspired structural constraints, the model captures both conservative and non-conservative interactions, while multi-scale spatial propagation and latent temporal marching improve long-term forecasting over complex geometries. The approach achieves strong predictive accuracy, physical consistency, and data efficiency, particularly under limited training data.

L Training setup

L.1 Computing Infrastructure

For the hardware, we trained all models on the server with the NVIDIA A100 GPU (80GB) and the Intel(R) Xeon(R) Platinum 8380 CPU (2.30GHz, 64 cores). For the software, we use Pytorch and Pytorch-Geometric for our implementations.

L.2 Preprocessing Protocol

Preprocessing follows three rules. First, complete trajectories or simulation instances are split before temporal windows are created. Second, when normalization is used, all per-channel statistics are computed exclusively from the training trajectories and are reused unchanged for validation and testing. Finally, each sample contains one input frame and the next T_{out} targets without crossing a trajectory or split boundary.

L.3 Training Objective and Other Metrics

The model is trained using the mean squared prediction error

$$\mathcal{L}_{\text{train}} = \frac{1}{T_{\text{out}}|\mathcal{V}|d_x} \sum_{q=1}^{T_{\text{out}}} \sum_{i\in\mathcal{V}} \left\| \widehat{\boldsymbol{u}}_{i,t_{k+q}} - \boldsymbol{u}_{i,t_{k+q}} \right\|_2^2, \tag{S53}$$

where d_x is the number of predicted state channels. In the reported experiments, $T_{\text{out}} = 1$ during training, and predictions are recursively fed back during evaluation.

The reported rollout RMSE is

$$\text{RMSE} = \frac{1}{T_{\text{eval}}} \sum_{q=1}^{T_{\text{eval}}} \sqrt{\frac{1}{|\mathcal{V}|d_x} \sum_{i\in\mathcal{V}} \left\| \widehat{\boldsymbol{u}}_{i,t_{k+q}} - \boldsymbol{u}_{i,t_{k+q}} \right\|_2^2}. \tag{S54}$$

Efficiency reporting must distinguish edge computation from aggregation:

$$R_{\text{comp}} = \frac{\sum_{\ell,\square} \#\{\text{edges evaluated by head } \square\}}{2L|\mathcal{E}_{\text{u}}|},$$
$$R_{\text{agg}} = \frac{\sum_{\ell,\square} \sum_e a_{\square}^{\ell}}{2L|\mathcal{E}_{\text{u}}|}. \tag{S55}$$

These rates include terminal refresh unless stated otherwise. Training time is seconds per epoch, inference latency is seconds per physical forecasting step, and memory is peak allocated accelerator memory.

L.4 HYPERPARAMETER

All models are trained over 1,000 epochs with the ReduceLROnPlateau Scheduler, Adam Optimizer, and MSE loss function; evaluated by RMSE metric. Notably, all hyperparameters are selected by validation-set search.

Generally, the default setting is that the number of processor layers is set to 4, the standard deviation of noise is set to 0.0001, the number of attention head is set to 8, MLPs are the 2-layer architecture, the learning rate and latent feature dimension are uniformly fixed at 0.0001 and 128 respectively, across all datasets. Moreover, we encode the cyclical timestamp features for handling the periodic patterns in the 2D BS dataset. Trajectories are temporally subsampled using dataset-specific sampling intervals. For example, 1 for BS and 5 for others. Namely, for a 1000-step trajectory, there is a 200-step data for training. In addition, T_{eval} is set as 200 for Burgers, 200 for CF, 600 for GS, and 20 for BS. Task-specific adjustments for all models are described as follows.

DeepONet. For the basic settings of DeepONet, the number of branch layer is set to 3, the number of trunk layer is set to 1, learning rate is 0.0005, weight decay is 0.00001, latent feature dimension is 512, and batch size is 10. In addition, the number of branch layer is set to 1 for CF. The number of trunk layer is set to 3 for CF and 2 for BS. Batch size is set to 5 for 3D GS and 1 for CF.

MGN. The noise standard deviation is configured as 0.01 for CF. Batch sizes are set to 100 for the Burgers, 5 for 3D GS, 20 for BS, and 10 for CF.

MP-PDE. The standard deviation of noise is set to 0.01 for CF. The batch size is configured as 100 for the Burgers, 5 for 3D GS, 20 for BS, and 10 for CF. The training rollout step is uniformly set to 2 across all benchmarks.

FNO. Batch sizes are set to 100 for the Burgers and 10 for 3D GS. The number of Fourier modes is uniformly set to 8 for all benchmarks.

FFNO. Batch sizes are configured as 100 for the Burgers and 10 for 3D GS. The number of Fourier modes is uniformly set to 8 across all benchmarks.

Transformer. Transformer uses 6 processor layers for 3D GS, and 8 for others. The noise standard deviation is set to 0.01 for CF. Batch sizes are 30 for Burgers, 2 for 3D GS, 20 for BS, and 1 for CF.

Transolver. Transolver uses 8 processor layers for all benchmarks. The noise standard deviation is set to 0.01 for CF. Batch sizes are 100 for the Burgers, 10 for 3D GS, 20 for BS, and 1 for CF. The slice size is set to 32 across all benchmarks. The grid size for mapping is 8×8 for the Burgers, BS, and CF systems, and $8\times8\times8$ for the 3D GS system.

CiGNN. The noise standard deviation is configured as 0.01 for CF. Batch sizes are set to 10 for the Burgers, 5 for 3D GS, 20 for BS, and 10 for CF. The mask ratio is 0.9 for all benchmarks.

Ours. The standard deviation of noise is set to 0.01 for CF. The batch size is set to 30 for the Burgers, 2 for 3D GS, 20 for BS, and 1 for CF.

L.5 ADDITIONAL VISUALIZED RESULTS

Additional visualized results are provided in Figure S1.

M ADDITIONAL EXPERIMENTAL DETAILS

M.1 MASKED DEFERRED ACCUMULATION

For a deferred directed edge $j \to i$, let $\mathbf{b}_{j\to i}^{\ell}$ denote its persistent buffer, $\mathbf{m}_{j\to i}^{\ell+1}$ the new message, and $\mathbf{z}_{j\to i}^{\ell+1} \in [0,1]^{d_m}$ a componentwise retention gate. For an inactive edge, the update is

$$\mathbf{b}_{j\to i}^{(\ell+1)} = \mathbf{b}_{j\to i}^{(\ell)} + \mathbf{z}_{j\to i}^{(\ell+1)} \odot \mathbf{m}_{j\to i}^{(\ell+1)}. \tag{S56}$$

The hard-mask implementation is recovered by restricting the gate to $\{0,1\}^{d_m}$. The ablations in Table 5 separate delayed delivery from selective retention. Random and shuffled variants retain the same gate architecture and parameter count but replace or permute its outputs during evaluation.

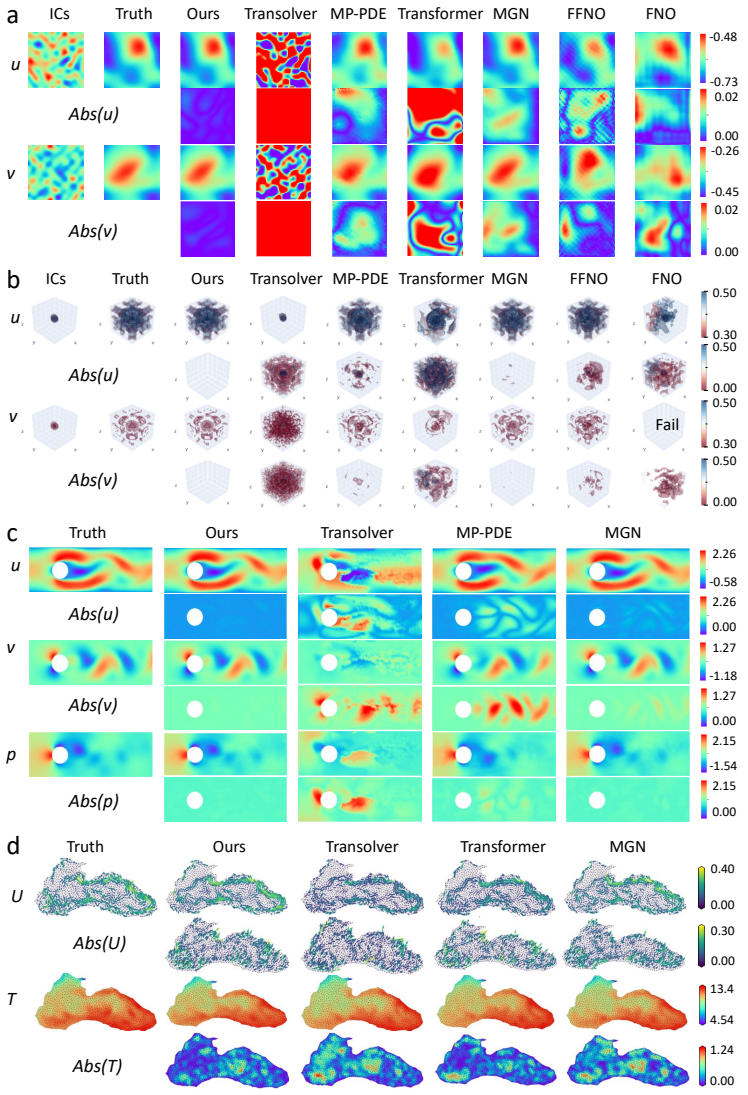

Figure S1: Full visualized results. The "Abs(*)" denotes $|\hat{*} - *|$.

Masked accumulation gives the lowest RMSE on all four datasets. The smaller gap between full and masked accumulation indicates that retaining deferred information is the primary benefit, while the learned gate provides an additional selective-filtering gain.

M.2 Comparison of Scheduling Policies

To determine whether the gain can be explained by generic sparsification, we compare asynchronous policies using the same intermediate activation rate. Random scheduling activates the same number of edges as the proposed policy, whereas the static schedule is fixed before rollout. Representative ascending priority scores are

$$\tau_{j\to i}^{\text{distance}} = d_{ji}, \quad \tau_{j\to i}^{\text{magnitude}} = \frac{1}{\|\boldsymbol{m}_{j\to i}\|_2 + \varepsilon}, \quad \tau_{j\to i}^{\text{proposed}} = \frac{d_{ji}}{\|\boldsymbol{m}_{j\to i}\|_2 + \varepsilon}. \tag{S57}$$

Because smaller scores are activated first, the magnitude-only policy prioritizes larger message norms. The synchronous row is a full-update reference. The no-terminal-refresh row is a mechanism ablation and has a smaller total aggregation budget unless its final-layer rate is compensated. The proposed priority achieves the lowest error on every dataset in Table 6, showing that the gain is not

reproduced by random sparsification or a single-factor score. The larger Burgers gain suggests that state-dependent delivery is particularly useful for strongly directional dynamics.

M.3 Message-Retention and Staleness Diagnostics

The retention ratio is evaluated only over inactive edge–operator pairs because the accumulation gate does not affect an active edge:

$$R_{\mathrm{ret}} = \frac{\sum_{\ell,\square}(1-a_{\square}^{\ell})\|\boldsymbol{z}_{\square}^{\ell}\|_1/d_h}{\sum_{\ell,\square}(1-a_{\square}^{\ell})+\varepsilon_{\square}}. \tag{S58}$$

For operator $\square$, pre-terminal edge coverage is

$$C_{\mathrm{edge}}^{\mathrm{pre}}(\square) = \frac{\left|\bigcup_{\ell=0}^{L-2}\{e : a_{\square}^{\ell}=1\}\right|}{|\mathcal{E}_{\square}|}. \tag{S59}$$

Table 7 reports its average over physical steps and edge-based operators. The normalized buffer magnitude immediately before terminal refresh is

$$R_{\mathrm{buffer}} = \frac{\sum_{\square}\|\boldsymbol{b}_{\square}^{L-1}\|_2}{\sum_{\square}\|\square\boldsymbol{m}_{\square}^{L-1}\|_2+\varepsilon_{\square}}. \tag{S60}$$

Waiting depth measures the number of consecutive inactive micro-steps for an edge–operator pair. We initialize $w_{\square}^{0}=0$ and update

$$w_{\square}^{\ell+1} = \left(1-a_{\square}^{\ell}\right)\left(w_{\square}^{\ell}+1\right), \qquad \ell=0,\dots,L-2. \tag{S61}$$

Thus, an active edge resets its waiting depth to zero, whereas an inactive edge increments its consecutive waiting depth by one.

Let

$$\mathcal{W} = \left\{(\ell,\square) : \ell \le L-2,\ a_{\square}^{\ell}=0\right\} \tag{S62}$$

denote the set of inactive pre-terminal edge–operator events. The mean waiting depth is

$$\bar{w} = \frac{1}{|\mathcal{W}|}\sum_{(\ell,\square)\in\mathcal{W}} w_{\square}^{\ell+1}. \tag{S63}$$

The maximum waiting depth is defined as

$$w_{\max} = \max_{(\ell,\square)\in\mathcal{W}} w_{\square}^{\ell+1}, \tag{S64}$$

which satisfies

$$w_{\max} \le L-1. \tag{S65}$$