GRAPH MAMBA OPERATOR FOR LEARNING THE DYNAMICS OF PARTICLE-BASED SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modelling complex 3D dynamics—from molecular conformations to particle interactions and human motion—requires capturing dependencies spanning long temporal horizons and non-local spatial interactions. Graph neural networks (GNNs) have shown promise in spatio-temporal settings but often suffer from instability and degraded accuracy in long-range forecasting. We propose the *Graph Mamba Operator* (GRAMO), a neural operator that integrates state-space models (SSMs) with geometric learning to capture spatio-temporal correlations jointly. To jointly model complex dynamics, GRAMO integrates a stable, SSM-based temporal backbone with an SSM-parameterized graph update to capture long-range spatial dependencies. Unlike stepwise predictors that accumulate errors over time, GRAMO learns entire trajectories in a single forward pass. Across diverse benchmarks ranging from molecular dynamics to human motion capture, GRAMO shows notable improvements in trajectory fidelity, stability, and robustness over strong baselines with relative improvements of over 26.3% on motion capture benchmarks and 25.2% on MD17 final-state prediction. Ablation studies reveal that temporal SSM components consistently improve performance, while spatial SSM updates show task-dependent benefits—helping with long-range interactions in large molecules but potentially hindering performance on systems with primarily local dependencies. Altogether, these results suggest that selective integration of SSM components with graph neural networks can improve performance on particle-based systems, with applications in molecular simulations, articulated rigid body dynamics, and particle systems.

1 INTRODUCTION

Particle-based systems are fundamental to understanding natural phenomena across diverse domains ranging from physics to biology to engineering. Molecular dynamics (MD) simulations capture how atoms interact to drive chemical reactions and protein folding Frenkel & Smit (2023); Hollingsworth & Dror (2018). Gravitational N-body problems describe celestial mechanics that shape galactic structures. These systems, while operating at vastly different scales, share a common mathematical foundation: collections of interacting entities whose collective behavior emerges from local interaction rules governed by ordinary differential equations. Accurately modeling and predicting the evolution of such systems is essential for scientific discovery and technological advancement. Traditional numerical methods solve these equations using symplectic integrators such as Verlet or velocity Verlet schemes (Donnelly & Rogers, 2005; Bou-Rabee, 2013). However, these approaches become computationally prohibitive for complex systems with many particles or long temporal horizons (Hollingsworth & Dror, 2018; Bou-Rabee, 2013), motivating alternative data-driven methods.

Graph neural networks have emerged as a promising approach for learning particle dynamics by naturally representing entities as nodes and their interactions as edges (Kipf et al., 2018; Martinkus et al., 2021; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Bihani et al., 2024; Bishnoi et al., 2023). This representation enables physically informed models that can generalize across different system configurations (Xu et al., 2024; Bishnoi et al., 2022). Neural operators provide a robust framework for modeling dynamical systems by learning mappings between infinite-dimensional functional spaces and serving as universal approximators for complex dynamics. Among these, the Fourier Neural Operator (Li et al., 2020; 2023; Bonev et al., 2023) has achieved state-of-the-art results on partial differential equations by leveraging Fourier-domain representations. However, current graph-based

approaches face fundamental limitations when modeling long-term dynamics. Their reliance on local message passing restricts information flow to immediate neighbors at each step, leading to over-squashing of long-range dependencies and error accumulation over time (Dwivedi et al., 2022; Di Giovanni et al., 2023; Arroyo et al., 2025; Topping et al., 2021). These methods also suffer from vanishing gradients and degraded performance on long horizons (Liu et al., 2019).

State-space models (SSMs) offer a complementary perspective by treating discrete observations as samples from underlying continuous-time processes (Gu & Dao, 2023). SSMs maintain compact latent representations and update them through linear dynamics, enabling efficient modeling of long-range temporal dependencies while avoiding the vanishing gradient problems that plague recurrent approaches (Mehta et al., 2022). Recent work has demonstrated that SSMs can be effective in modelling partial differential equations and have shown competitive performance compared to traditional neural operators on various benchmarks (Tiwari et al., 2025). Additionally, selective SSMs have achieved remarkable success in sequence modeling tasks across multiple domains (Dao & Gu, 2024). However, extending SSMs to graph-structured data remains challenging because it requires reconciling sequential temporal updates with complex spatial topologies (Wang et al., 2024).

We propose the Graph Mamba Operator (GRAMO), which integrates state-space modeling with graph neural networks to capture both long-range spatial and temporal dependencies in particle dynamics. GRAMO employs bidirectional SSMs for temporal modeling and introduces a SSM-parameterized message passing scheme that enables non-local spatial interactions while maintaining computational efficiency. The key insight is that spatial message passing can itself be formulated as a state-space process, where node states evolve according to graph-aware dynamics. Unlike stepwise predictors that accumulate errors over time, GRAMO learns to predict entire trajectories in a single forward pass. Our main contributions are:

1. **GRAMO.** We present GRAMO, unifying SSM-based temporal modeling with graph message passing to jointly capture spatial and temporal dependencies in particle systems.
2. **Theoretical Analysis.** We analyze the stability properties of our SSM-based spatial updates and show how they extend local graph interactions to capture long-range dependencies.
3. **Empirical Validation.** Across benchmarks spanning molecular dynamics, human motion, and N-body systems, GRAMO achieves consistent improvements over strong baselines, with mean relative gains of 26.3% on motion capture and 25.2% on MD17.

2 RELATED WORK

Neural Operators. Neural operators learn mappings between infinite-dimensional function spaces, providing data-driven alternatives to classical numerical solvers (Kovachki et al., 2023). Specifically, Fourier Neural Operator (FNO) (Li et al., 2020; 2023), a seminal work in the field, parameterizes integral operators in Fourier space, achieving state-of-the-art performance on fluid dynamics and atmospheric modeling (Li et al., 2020; 2023; Bonev et al., 2023). Recent operator frameworks leverage state-space models to capture long-range temporal dependencies (Gu & Dao, 2023; Tiwari et al., 2025), showing competitive performance on PDE benchmarks. However, these approaches treat spatial and temporal dynamics independently, limiting effectiveness for coupled particle systems where spatial interactions evolve over time.

Graph Neural Networks. Message-passing networks (Kipf & Welling, 2017a; Hamilton et al., 2017; Veličković et al., 2018) naturally represent particle systems by treating entities as nodes and interactions as edges (Kipf & Welling, 2017b). Despite conceptual appeal, they face fundamental limitations: over-squashing prevents long-range information flow (Arroyo et al., 2025), while iterative prediction leads to error accumulation. Graph Transformers address some issues through global attention but incur quadratic computational costs. Continuous-time formulations like Graph ODEs (Fang et al., 2021; Luo et al., 2023) avoid discretization errors but suffer from expensive solvers and numerical instability. Most relevant to our work, EGNOs (Xu et al., 2024) predict entire trajectories in single passes, reducing error accumulation, but struggle with long-range temporal dependencies and non-stationary dynamics due to their reliance on fixed Fourier parameterizations.

State Space Models on Graphs. Structured SSMs have demonstrated remarkable success in sequence modeling (Gu & Dao, 2023; Dao & Gu, 2024) and recently shown promise for PDE solving (Tiwari et al., 2025). Extensions to graphs have explored various approaches: survey work (Qu et al., 2024)

categorizes SSM applications across domains, including vision (Zhu et al., 2024) and graphs (Wang et al., 2024). However, current graph SSM methods impose artificial sequential orderings through degree-based sorting (Wang et al., 2024) or random walks (Behrouz & Hashemi, 2024), breaking permutation equivariance or losing structural information. Recent spatio-temporal approaches (Sahili & Awad, 2023; Cini et al., 2023) address temporal modeling but do not fully exploit SSMs' long-range capabilities for spatial interactions. To the best of our knowledge, no prior work successfully integrates SSM temporal operators with principled graph message passing for particle dynamics.

3 Background and Problem Setting

State-Space Models (SSMs). A continuous-time linear state-space model (also known as a Linear Time-Invariant, LTI system) characterizes the evolution of a hidden state under the influence of external inputs. Let the input be $\mathbf{x}(t) \in \mathbb{R}^{d_x}$, the hidden state be $\mathbf{h}(t) \in \mathbb{R}^{d_h}$, and the output be $\mathbf{y}(t) \in \mathbb{R}^{d_y}$. The system dynamics are described by the continuous-time state-space as follows:

$$\dot{\mathbf{h}}(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t), \qquad \mathbf{y}(t) = \mathbf{C}\mathbf{h}(t), \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{d_h \times d_h}$ is the state transition matrix, $\mathbf{B} \in \mathbb{R}^{d_h \times d_x}$ is the input projection, and $\mathbf{C} \in \mathbb{R}^{d_y \times d_h}$ is the output mapping and the model learnable parameters.

Discretization. For integration of continuous-time SSMs into neural network architectures, it is necessary to derive a discrete-time formulation. Let $\Delta \in \mathbb{R}_+$ denote the sampling interval, and assume a zero-order hold (ZOH) scheme for discretization. The continuous dynamics in equation 2 takes the following discrete representation form:

$$\mathbf{h}[k] = \bar{\mathbf{A}}\,\mathbf{h}[k-1] + \bar{\mathbf{B}}\,\mathbf{x}[k], \quad \mathbf{y}[k] = \mathbf{C}\,\mathbf{h}[k]. \tag{2}$$

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are the discretized system matrices (derivation in Appendix B), defined as follows:

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}), \quad \bar{\mathbf{B}} = \mathbf{A}^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I})\,\mathbf{B}. \tag{3}$$

with $\mathbf{I}$ denoting the identity matrix. For sufficiently small Δ, one may approximate $\bar{\mathbf{B}} \approx \Delta\mathbf{B}$.

An equivalent view of the discrete-time SSM is obtained by unfolding the recurrence into a convolutional operator (Gu & Dao, 2023). Specifically, the output sequence $\mathbf{y}$ can be expressed as:

$$\mathbf{y} = \mathbf{x} * \mathbf{K}, \qquad \mathbf{K} = \big[\,\mathbf{CB},\, \mathbf{CAB},\, \ldots,\, \mathbf{CA}^{L-1}\mathbf{B}\,\big], \tag{4}$$

where $\mathbf{K} \in \mathbb{R}^L$ is the convolutional kernel the system's impulse response.

Selective State-Space Models (S6). The Selective SSM framework, introduced in Mamba (Gu & Dao, 2023), extends classical SSMs by allowing certain parameters to depend on the input signal. In particular, the input-dependent parameterization of $\mathbf{B}$, $\mathbf{C}$, and the discretization step Δ enhances both expressivity and efficiency, bringing the performance of SSMs closer to transformer-based sequence models. Under this formulation, the convolutional kernel in Equation 4 is generalized to following:

$$\mathbf{K} = \big\{\,\mathbf{C}_L\mathbf{B}_L,\ \mathbf{C}_L\mathbf{A}_{L-1}\mathbf{B}_{L-1},\ \ldots,\ \mathbf{C}_L\Big(\prod_{i=1}^{L-1}\mathbf{A}_i\Big)\mathbf{B}_1\,\big\}, \tag{5}$$

where the matrices $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i\}$ are selectively modulated by the input at each step. This adaptive construction yields a richer class of convolutional operators.

Newtonian–SSM Connection. Consider Newtonian dynamics for a particle system governed by $\dot{\mathbf{x}} = \mathbf{v}$ and $m\dot{\mathbf{v}} = -\mathbf{F}(\mathbf{x}, \mathbf{v}, t) - \gamma\mathbf{v}$, where $\mathbf{F}(\mathbf{x}, \mathbf{v})$ represents forces, m represent the masses, and γ is a damping coefficient. By defining the state vector $\mathbf{h} = [\mathbf{x}; \mathbf{v}]$, the second-order system can be written in first-order state-space form:

$$\dot{\mathbf{h}} = \mathbf{A}\mathbf{h} + \mathbf{B}\mathbf{u}(\mathbf{h}), \tag{6}$$

where $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\gamma\mathbf{I} \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{I} \end{bmatrix}$, and $\mathbf{u}(\mathbf{h}) = \mathbf{F}(\mathbf{x}, \mathbf{v})$.

When forces are linear in the state variables, this reduces to a linear time-invariant (LTI) system. For nonlinear forces, the system becomes a nonlinear state-space model. The explicit form is:

$$\dot{\mathbf{Z}} = \begin{bmatrix} \dot{\mathbf{x}} \\ m\dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\mathbf{F}(\mathbf{x}, \mathbf{v}) - \gamma\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\gamma\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{I} \end{bmatrix} \mathbf{F}(\mathbf{x}, \mathbf{v}) \tag{7}$$

This formulation provides a natural connection between particle dynamics and state-space representations, motivating the use of SSM-based approaches for trajectory prediction. Note that this equation is analogous to Hamiltonian dynamics with $\mathbf{x}$ and $m\mathbf{v}$ corresponding to the canonical coordinates $\mathbf{q}$ and $\mathbf{p}$ representing positions and momenta of the particles, respectively.

Problem Setting. We study the temporal evolution of interacting particle systems represented as a sequence of graphs $\{\mathcal{G}^t\}_{t=1}^T$. Each graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}, \mathbf{H}_t, \mathbf{Z}_t)$ shares a fixed node set with $|\mathcal{V}| = N$ nodes, defining a mapping $f : t \mapsto \mathcal{G}^t$.

The graph structure captures both static and dynamic particle properties. Node features $\mathbf{H}^t \in \mathbb{R}^{N\times d}$ encode time-invariant attributes such as atom type or charge. Time-varying geometric descriptors $\mathbf{Z}^t \in \mathbb{R}^{N\times m\times 3}$ represent evolving particle states, commonly instantiated as $\mathbf{Z}^t = [\mathbf{x}^t, \mathbf{v}^t]$ where $\mathbf{x}^t \in \mathbb{R}^{N\times 3}$ denotes positions and $\mathbf{v}^t \in \mathbb{R}^{N\times 3}$ denotes velocities, giving $m = 2$.

In our formulation, structural features remain constant over time ($\mathbf{H}_t = \mathbf{H}$) while system dynamics evolve through the geometric tensors $\mathbf{Z}_t$. The graph connectivity is defined by an adjacency matrix $\mathbb{A} \in \mathbb{R}^{N\times N}$, constructed from either domain-specific interactions (e.g., chemical bonds in molecular systems) or distance-based neighborhoods.

Problem Statement (Trajectory Prediction). Let $\mathbb{T}$ be a set of spatio-temporal trajectories of interacting particle systems sampled from data distribution p_{data}. Each trajectory consists of a sequence of graphs $\{\mathcal{G}^{(t+\Delta t)} : \Delta t \in [0, \Delta T]\}$ representing particle states over time. Let $\mathcal{U} : \mathcal{D} \to \mathbb{R}^{N\times m\times 3}$ denote the target trajectory space. Our goal is to learn a neural operator that predicts complete trajectories from initial conditions. Towards that end, we want to learn:

1. A neural operator $\mathcal{F}_\theta : \mathcal{G}^{(t)} \mapsto \{\mathcal{G}^{(t+\Delta t)} : \Delta t \in [0, \Delta T]\}$ that approximates the true solution operator $\mathcal{F}^\dagger$ and maps initial graph states to complete future trajectories in a single forward pass, unlike prior approaches (Thomas et al., 2018; Wang & Chodera, 2023).
2. The operator must capture both long-range spatial dependencies through graph structure and long-range temporal dependencies across the prediction horizon.
3. The learned operator should maintain permutation equivariance with respect to particle ordering and exhibit stable predictions over extended time horizons.

The training objective minimizes the expected trajectory discrepancy:

$$\min_\theta \ \mathbb{E}_{\mathcal{G}^{(t)}\sim p_{\text{data}}} \left[\mathcal{L}\left(\mathcal{F}_\theta(\mathcal{G}^{(t)}), \mathcal{F}^\dagger(\mathcal{G}^{(t)}) \right) \right], \tag{8}$$

where $\mathcal{L}$ measures trajectory discrepancy using the L_2 norm.

Once learned, this neural operator can predict trajectories of unseen particle systems across domains, including molecular dynamics, human motion, and N-body systems.

In practice, we approximate the continuous objective through empirical risk minimization (ERM) over P uniformly sampled time points $\{\tau_p\}_{p=1}^P \subset [0, \Delta T]$:

$$\min_\theta \ \mathbb{E}_{\mathcal{G}^{(t)}\sim p_{\text{data}}} \frac{1}{P} \sum_{p=1}^{P} \left\| \mathcal{F}_\theta(\mathcal{G}^{(t)})(\tau_p) - \mathcal{F}^\dagger(\mathcal{G}^{(t)})(\tau_p) \right\|_2. \tag{9}$$

We focus on modeling geometric evolution: the time-varying descriptors $\mathbf{Z}_t$ capture system dynamics while structural node features $\mathbf{H}$ remain time-invariant.

4 Graph Mamba Operator

We propose GraMO, a neural operator that predicts entire future trajectories in a single step. Inspired by Mamba (Gu & Dao, 2023), which employs dynamic weights (input dependent) for SSM, we employ a temporal SSM that learns frequency-dependent temporal dynamics, coupled with an SSM-based spatial update that preserves graph topology while capturing long-range interactions.

4.1 Proposed Method

Overview. The proposed method, illustrated in Figure 1, can be summarized as follows:

$$\mathcal{F}_\theta = (\mathcal{S} \circ \mathcal{T})^l \circ (\mathcal{S} \circ \mathcal{T})^{l-1} \circ \cdots \circ (\mathcal{S} \circ \mathcal{T})^2 \circ (\mathcal{S} \circ \mathcal{T})^1 \circ \mathcal{E}, \tag{10}$$

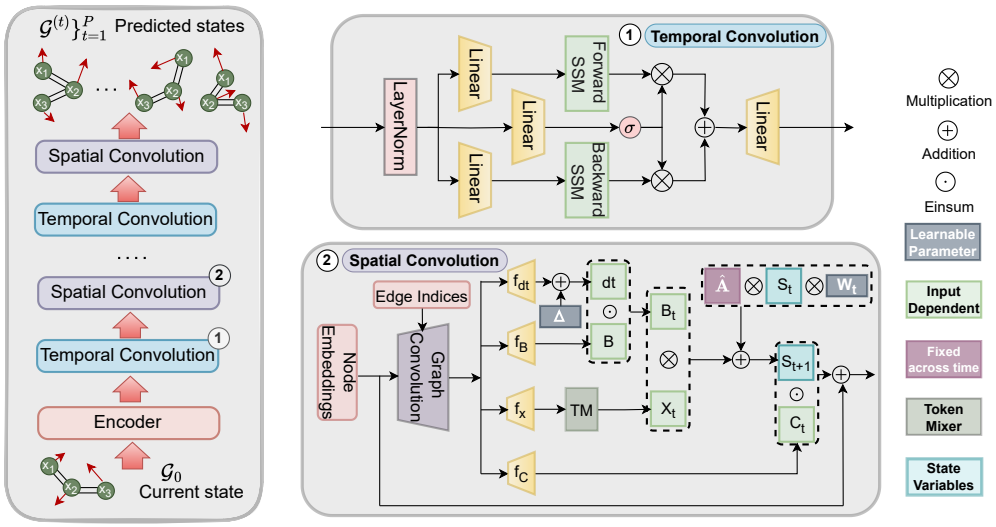


Figure 1: **Overview.** The framework encodes the input graph $\mathcal{G}^{(t)}$, processes it through L stacked GRAMO blocks, and decodes future trajectories. (**Left**.) Overall architecture. (**Right Top**.) Temporal block: bidirectional Mamba capturing long-range temporal dependencies. (**Right Bottom**.) Spatial block: SSM-inspired message passing enabling non-local interactions (in Appendix Algorithm 1-2). The horizon ΔT is discretized into P steps $\{\Delta t_p\}_{p=1}^{P}$, and the decoder outputs trajectories $\{\mathcal{G}^{(t+\Delta t_p)}\}_{p=1}^{P}$ in parallel as predicted dynamics $f_{\mathcal{G}}(t)$.

where $\circ$ denotes composition. Specifically, the encoder $\mathcal{E}$ maps the initial graph state $\mathcal{G}^{(0)}$ to the trajectory sequence $\{\mathcal{G}^{(t)}\}$. Each $\mathcal{S}$ represents a *spatial convolution block*, while $\mathcal{T}$ corresponds to a *temporal convolution block*. The parameter l denotes the total number of stacked GRAMO layers. Finally, it generates the complete trajectory, producing the final output representation.

Encoder ($\mathcal{E}$). The encoder $\mathcal{E}$ maps the initial graph $\mathcal{G}^{(0)}$ into a predicted trajectory $\{\mathcal{G}^{(t)}\}_{t=1}^{P}$. Following (Xu et al., 2024), we replicate $\mathcal{G}^{(0)}$ into P copies $\{\mathcal{G}^{(0)}\}_{p=1}^{P}$ and augment each with a learnable time embedding $e(\Delta t_p)$, enabling the model to incorporate timestep information for trajectory prediction (Ho et al., 2020). The sequence is then processed by interleaved temporal blocks ($\mathcal{T}$) and spatial blocks ($\mathcal{S}$), allowing stacked GRAMO layers to evolve the full trajectory.

GRAMO Blocks. GRAMO is structured as a hierarchical stack of temporal convolution blocks ($\mathcal{T}$) and spatial convolution blocks ($\mathcal{S}$). The temporal blocks capture long-range dynamics across nodes, while the spatial blocks operate on the graph topology to model inter-node dependencies. By interleaving and stacking these components, GRAMO jointly encodes temporal and spatial correlations, yielding a rich representation for learning the dynamics of graph-structured systems.

Temporal Convolution ($\mathcal{T}$). Let $f : D \to \mathcal{G}$ denote the input trajectory function with $f(t) = [f_h;\, f_Z(t)]^\top$, where f_h is time-invariant and $f_Z(t)$ is time-varying. [1] The temporal convolution layer $\mathcal{T}$ applies a residual update which can be described as follows:

$$(\mathcal{T}f)(t) \;=\; f(t) \;+\; \sigma\big((\mathcal{K}f)(t)\big), \tag{11}$$

where σ denotes a pointwise nonlinearity, and $\mathcal{K}$ represents the integral operator (Li et al., 2020) defined as $\mathcal{K}f(t) = \int_T \mathcal{K}(t,\tau)\, f(\tau)\, \mathrm{d}\tau$. We parameterize $\mathcal{K}$ using a bidirectional state-space model (SSM) applied across nodes to capture temporal dynamics. As shown in (Tiwari et al., 2025), SSMs approximate integral kernels, which allows $\mathcal{K}$ to represent both causal and anti-causal responses. As shown in Equation 2 we can represent the $\mathcal{K}f$ in discretized form as follows:

$$\mathcal{K}f = \mathrm{SSM}_{\text{forward}}(\mathbf{A}, \bar{\mathbf{B}}, \bar{\mathbf{C}})f + \mathrm{SSM}_{\text{backward}}(\mathbf{A}, \bar{\mathbf{B}}, \bar{\mathbf{C}})f \tag{12}$$

where $\mathrm{SSM}_{\text{forward}}$ and $\mathrm{SSM}_{\text{backward}}$ denote the forward and backward state-space models applied along the temporal dimension of the nodes. The forward model is parameterized as $\mathrm{SSM}(\mathbf{A}, \bar{\mathbf{B}}, \bar{\mathbf{C}})$, and the backward model as $\mathrm{SSM}(\mathbf{A}, \bar{\mathbf{B}}, \bar{\mathbf{C}})$. Unlike EGNO, where the kernel parameters remain fixed, our formulation employs dynamic parameters that adapt to the temporal evolution of the system.

Spatial Convolution ($\mathcal{S}$). We design a Mamba-inspired message passing layer that extends state-space updates to graph-structured data. Let $\{\mathcal{G}^{(t)}\}$ denote a temporal sequence of graphs with fixed

[1] For simplicity, In this paper f denotes $f_{\mathcal{G}}$.

topology $(\mathcal{V}, \mathcal{E})$ and node features $\{\mathbf{X}^{(t)}\}$, obtained by concatenating the intrinsic node features $\mathbf{H}^{(t)}$ with the geometric features $\mathbf{Z}^{(t)}$, where $\mathbf{Z}^{(t)}$ encodes the positions and velocities of the nodes. Each graph is associated with a hidden state $\mathbf{S}^{(t)}$, which evolves through the following structured update:

$$\mathbf{S}^{(t+1)} = \hat{\mathbf{A}}\,\mathbf{S}^{(t)}\,\mathbf{W} + \mathbf{X}^{(t)}\,\mathbf{B}, \qquad \mathbf{Y}^{(t+1)} = \mathbf{S}^{(t+1)}\,\mathbf{C}. \tag{13}$$

Here, $\mathbf{S}^{(t)} \in \mathbb{R}^{N\times n}$ denotes the hidden state at step t, $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times d_x}$ the input node features, and $\mathbf{Y}^{(t)} \in \mathbb{R}^{N\times d_y}$ the output node features. The operators are given by $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}, \mathbf{W} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{d_x\times n}, \mathbf{C} \in \mathbb{R}^{n\times d_y}$. The matrix $\hat{\mathbf{A}}$ denotes the normalized adjacency $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbb{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$, where $\mathbb{A}$ is the adjacency matrix, $\mathbf{I}$ adds self-loops, and $\mathbf{D}$ is the corresponding degree matrix. The matrices $\mathbf{W}, \mathbf{B}, \mathbf{C}$ are model learnable parameters.

Remark 4.1. If we set $\mathbf{B} = \mathbf{C} = \mathbf{0}$ and initialize $\mathbf{S}^{(t)} = \mathbf{X}^{(t)}$, then after applying a nonlinearity $\sigma(\cdot)$, the update reduces to $\mathbf{Y} = \sigma\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}\right)$, which recovers the standard GCN message-passing rule.

Selectivity and Interpretability. Spatial convolution design is inspired by the notion of *selectivity* in sequence models, where dynamics are adaptively weighted by input content. To achieve this, the parameters $\mathbf{B}$ and $\mathbf{C}$ are made input-dependent, dynamically generated from $\mathbf{X}^{(t)}$ via graph convolution (e.g., GCN, GraphSAGE) at each step. This provides finer control over information flow: $\hat{\mathbf{A}}$ governs state propagation through normalized adjacency (as in message passing neural networks), while $\mathbf{B}$ and $\mathbf{C}$ regulate how new inputs update the hidden state $\mathbf{S}^{(t)}$ and how the state contributes to the output $\mathbf{Y}^{(t)}$. Such selectivity enables the model to filter irrelevant signals, compress long contexts into compact states, and balance content-driven input modulation with context-driven state dynamics.

4.2 Theoretical Insights

We now provide a theoretical analysis of the proposed components, with complete proofs in Appendix B. Lemma 4.2 establishes the stability of Equation 13 by showing that the normalized adjacency $\hat{\mathbf{A}}$ has spectral radius equal to unity. Moreover, unrolling the recurrence introduces powers of $\hat{\mathbf{A}}$, and Appendix Proposition B.7 demonstrates that $\hat{\mathbf{A}}^t$ encodes all walks of length t. Consequently, after t steps, each node aggregates information from all nodes within t hops, thereby enabling the model to effectively capture long-range information across nodes in the graph.

Lemma 4.2 (Spectrum and Neumann Series). *Let $\mathcal{G}$ be an undirected graph and $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbb{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$. Then $\sigma(\hat{\mathbf{A}}) \subseteq [-1,1]$ and $\|\hat{\mathbf{A}}\|_2 = \rho(\hat{\mathbf{A}}) = 1$, with max eigenvalue 1 and eigenvector proportional to $\mathbf{D}^{1/2}\mathbf{1}$. Consequently, the Neumann series $\sum_{t=0}^{\infty}\hat{\mathbf{A}}^t$ diverges. However, for any $\alpha \in (0,1)$ the damped series $\sum_{t=0}^{\infty}(\alpha\hat{\mathbf{A}})^t$ converges to $(\mathbf{I}-\alpha\hat{\mathbf{A}})^{-1}$.*

Next, in Proposition 4.3, we establish the permutation equivariance of Equation 13: permuting the node ordering results in the output being permuted in the same way, a desired property for GNNs.

Proposition 4.3 (Permutation Equivariance). *Let $\Pi \in \{0,1\}^{N\times N}$ be any permutation matrix and define $\hat{\mathbf{A}}' = \Pi\hat{\mathbf{A}}\Pi^\top$, $\mathbf{S}'^{(\mathbf{t})} = \Pi\mathbf{S}^{(\mathbf{t})}$, $\mathbf{X}'^{(\mathbf{t})} = \Pi\mathbf{X}^{(\mathbf{t})}$. Then the update:*

$$\mathbf{S}^{(\mathbf{t+1})} = \hat{\mathbf{A}}\mathbf{S}^{(\mathbf{t})}\mathbf{W} + \mathbf{X}^{(\mathbf{t})}\mathbf{B}, \qquad \mathbf{Y}^{(\mathbf{t+1})} = \mathbf{S}^{(\mathbf{t+1})}\mathbf{C} \tag{14}$$

is permutation equivariant:

$$\mathbf{S}'^{(\mathbf{t+1})} = \hat{\mathbf{A}}'\mathbf{S}'^{(\mathbf{t})}\mathbf{W} + \mathbf{X}'^{(\mathbf{t})}\mathbf{B} = \Pi\mathbf{S}^{(\mathbf{t+1})}, \qquad \mathbf{Y}'^{(\mathbf{t+1})} = \mathbf{S}'^{(\mathbf{t+1})}\mathbf{C} = \Pi\mathbf{Y}^{(\mathbf{t+1})}. \tag{15}$$

Finally, Proposition 4.4 establishes that repeated updates can either amplify or attenuate signals depending on the spectral properties of $\hat{\mathbf{A}}$ and $\mathbf{W}$. The dynamics remain stable provided the spectral radii are bounded: as shown in Lemma 4.2, $\hat{\mathbf{A}}$ has spectral radius 1, and by ensuring that $\rho(\mathbf{W})$ is bounded, the update equation maintains stable dynamics throughout rollouts.

Proposition 4.4 (Multi–Step Jacobian). *For the spatial SSM update*

$$\mathbf{S}^{(t)} = \hat{\mathbf{A}}\,\mathbf{S}^{(t-1)}\mathbf{W} + \mathbf{X}^{(t-1)}\mathbf{B}, \tag{16}$$

with $\mathbf{S}^{(t)} \in \mathbb{R}^{N\times n}$, $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$, and $\mathbf{W} \in \mathbb{R}^{n\times n}$, the Jacobian of $\mathbf{S}^{(t)}$ with respect to $\mathbf{S}^{(s)}$ for $t > s$ is

$$\frac{\partial\,\mathbf{S}^{(t)}}{\partial\,\mathbf{S}^{(s)}} = \left[\,(\hat{\mathbf{A}}^{\,t-s})_{ij}\,\mathbf{W}^{\,t-s}\,\right]_{i,j=1}^{N} \quad \textit{and} \quad \frac{\partial\,\mathrm{vec}(\mathbf{S}^{(t)})}{\partial\,\mathrm{vec}(\mathbf{S}^{(s)})} = \left(\mathbf{W}^\top \otimes \hat{\mathbf{A}}\right)^{t-s}. \tag{17}$$

Table 1: **Evaluation of the N-Body and Mocap Capture**. F-MSE ($\times 10^{-2}$) on the N-Body simulation and Motion Capture datasets (Subject #35 Walk, Subject #9 Run). The "Gain %" quantifies the relative reduction achieved by our model compared to the second-best performer on each benchmark. **First** and **Second** denotes best results. Lower value denotes better performance. Baseline results reported from Xu et al. (2024).

Dataset	SE(3)-Tr.	TFN	MPNN	RF	ClofNet	EGNN	EGNO	GRAMO	Gain %
N-body	2.44 $\pm$0.03	1.55 $\pm$0.02	1.07 $\pm$0.03	1.04 $\pm$0.03	**0.65 $\pm$0.02**	0.71 $\pm$0.02	**0.65 $\pm$0.01**	**0.62 $\pm$0.02**	**4.61%**
#35 (Walk)	31.5 $\pm$2.1	32.0 $\pm$1.8	36.1 $\pm$1.5	188.0 $\pm$1.9	–	28.7 $\pm$1.6	**9.61 $\pm$1.6**	**6.9 $\pm$0.21**	**28.19%**
#9 (Run)	61.2 $\pm$2.3	56.6 $\pm$1.7	66.4 $\pm$2.2	521.3 $\pm$2.3	–	50.9 $\pm$0.9	**37.18 $\pm$0.7**	**28.12 $\pm$0.43**	**24.37%**

5 NUMERICAL EXPERIMENTS

We systematically evaluate GRAMO across diverse trajectory prediction tasks, comparing against established baselines and performing detailed component analyses to validate our design choices.

5.1 IMPLEMENTATION DETAILS

Benchmark Details. We evaluate GRAMO on diverse trajectory-prediction tasks spanning physical, biological, and human-motion domains. Benchmarks include the 3D N-body suite (Satorras et al., 2021a), human motion from CMU Mocap (Walk #35 and Run #9) (CMU, 2003; Huang et al., 2022; Han et al., 2022), protein dynamics from the AdK equilibrium trajectory (Seyler & Beckstein, 2017; Richard J. Gowers et al., 2016), and the MD17 molecular dataset (Chmiela et al., 2017). Together, these tasks evaluate the GRAMO 's ability to capture both short- and long-range interactions.

Baselines. We compare GRAMO against a broad range of graph networks and operators, including Linear (Satorras et al., 2021a), SE(3)-Transformer (Fuchs et al., 2020), TFN (Thomas et al., 2018), MPNN (Gilmer et al., 2017), RF (Köhler et al., 2019), ClofNet (Du et al., 2022), EGNN (Satorras et al., 2021a), EGNO (Xu et al., 2024), and ITO (Schreiner et al., 2023). Following Xu et al. (2024), we also evaluate two EGNN variants: (i) *EGNN-Roll*, trained on short horizons and evaluated by iterative rollout (Sanchez-Gonzalez et al., 2020); and (ii) *EGNN-Sequential*, which generates trajectories frame by frame through successive EGNN layers.

Implementation Details. All models are trained using the Adam optimizer (Kingma & Ba, 2014) along with the StepLR scheduler. Experiments are carried out on a Linux system running Ubuntu 20.04.3LTS, equipped with an Intel(R) Core(TM) i9-10900X CPU and a single NVIDIA RTX A6000 GPU with 48 GB memory. Further details are provided in the Appendix Section D.

Evaluation Metrics. We evaluate performance under two settings: state-to-state (`S2S`) and state-to-trajectory (`S2T`). The state-to-state setting measures accuracy at the final timestep using the Final Mean Squared Error (F-MSE), defined as F-MSE $= \|\mathbf{x}(t_P) - \mathbf{x}^\dagger(t_P)\|^2$, where $\mathbf{x}^\dagger$ is the reference state. In contrast, the state-to-trajectory setting evaluates the entire rollout using the Average MSE (A-MSE), given by A-MSE $= \frac{1}{P}\sum_{p=1}^{P} \|\mathbf{x}(t_p) - \mathbf{x}^\dagger(t_p)\|^2$, which serves as the evaluation metric.

Results. Table 1 shows that GRAMO performs slightly out perform existing approaches on the relatively simple N-body system (4.61% gain), while achieving substantial gains on the more complex Motion Capture tasks (Subject #35 Walk, Subject #9 Run), with a relative improvement of 26.28% over the second-best baseline. Furthermore, Table 2 demonstrates that EGNN and EGNO suffer from pronounced error accumulation over long trajectories, whereas GRAMO maintains accurate and stable rollouts, preserving both fidelity and stability.

Table 2: A-MSE ($\times 10^{-2}$) comparison across models on N-Body simulation and MoCap datasets: Subject #35 (Walk) and Subject #9 (Run). **First** and **Second** denotes best results.

Model	N-Body	#35 (Walk)	#9 (Run)
EGNN-R	2.15 $\pm$0.02	32.0 $\pm$1.6	277.3 $\pm$1.8
EGNN-S	0.45 $\pm$0.01	14.3 $\pm$1.2	**28.5 $\pm$1.3**
EGNO	**0.27 $\pm$0.03**	**9.3 $\pm$1.5**	37.2 $\pm$0.7
GRAMO	**0.25 $\pm$0.02**	**6.5 $\pm$0.2**	**26.28 $\pm$0.4**
Gain %	**7.4%**	**30.1%**	**7.8%**

Now, we analyze the performance on the MD17 dataset (see Table 3). Here, GRAMO consistently outperforms all baselines in the final-state prediction, achieving a relative gain of 25.2% over the second-best method. These improvements are especially pronounced for molecules such as benzene, aspirin, and toluene, where GRAMO yields substantially lower MSE than prior approaches. Notably,

Table 3: **Evaluation on the MD17 dataset.** The upper half of the table reports F-MSE ($\times 10^{-2}$), while the lower half presents A-MSE ($\times 10^{-2}$). **First** and **Second** denotes best results. Lower value denotes better performance. Baseline results reported from Xu et al. (2024).

Model	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil
RF	10.94±0.01	103.72±1.29	4.64±0.01	13.93±0.03	0.50±0.01	1.23±0.01	10.93±0.04	0.64±0.01
TFN	12.37±0.18	58.48±1.98	4.81±0.04	13.62±0.08	0.49±0.01	1.03±0.02	10.89±0.01	0.84±0.02
SE(3)-Tr.	11.12±0.06	68.11±0.67	4.74±0.13	13.89±0.02	0.52±0.01	1.13±0.02	10.88±0.06	0.79±0.02
EGNN	14.41±0.15	62.40±0.53	4.64±0.01	13.64±0.01	0.47±0.02	1.02±0.02	11.78±0.07	0.64±0.01
EGNN-R	14.51±0.19	62.61±0.75	4.94±0.21	17.25±0.05	0.82±0.02	1.35±0.02	11.59±0.04	1.11±0.02
EGNN-S	9.50±0.10	66.45±0.89	4.63±0.01	**12.88±0.01**	0.45±0.01	1.00±0.02	10.78±0.05	0.60±0.01
ITO	20.56±0.03	57.85±0.58	8.60±0.27	28.44±0.73	1.82±0.17	2.48±0.34	12.47±0.30	1.33±0.02
EGNO	**9.42±0.03**	**55.70±0.32**	**4.6±0.05**	13.12±0.01	**0.39±0.01**	**0.86±0.02**	**10.31±0.12**	**0.56±0.01**
GRAMO	**7.55±0.04**	**2.06±0.12**	**3.65±0.03**	**12.85±0.04**	**0.37±0.02**	**0.84±0.02**	**4.75±0.03**	**0.54±0.02**
Gain %	**19.85%**	**96.30%**	**20.65%**	**0.233%**	**5.13%**	**2.32%**	**53.93%**	**3.57%**
EGNN-R	12.07±0.11	23.73±0.30	3.44±0.17	13.38±0.03	0.63±0.01	**1.15±0.02**	5.04±0.02	0.89±0.01
EGNN-S	9.49±0.12	29.99±0.65	**3.29±0.01**	**11.21±0.01**	0.43±0.01	1.36±0.02	4.85±0.04	0.68±0.01
EGNO	**7.01±0.01**	**22.06 ±0.02**	3.30±0.02	**10.73**±0.01	**0.33**±0.01	1.20±0.02	**4.67±0.02**	**0.51±0.01**
GRAMO	**6.21±0.03**	**1.08±0.02**	**2.88±0.03**	**11.21±0.05**	**0.36±0.02**	**0.72±0.02**	**2.31±0.03**	**0.55±0.01**
Gain %	**11.41%**	**95.10%**	**12.46%**	**-4.47%**	**-9.09%**	**37.39%**	**50.53%**	**-7.84%**

GRAMO performs strongly in both S2S and S2T evaluations, highlighting the benefit of combining temporal and spatial convolution. In contrast to EGNO, which employs a fixed kernel, GRAMO leverages dynamic weights to parameterize the temporal kernel, enabling it to effectively capture non-stationary signals.

Furthermore, we evaluate the performance of GRAMO on the AdK equilibrium benchmark and compare it against the state-of-the-art EGHN (Han et al., 2022) baseline. As shown in Table 4, our method outperforms the baseline by a relative improvement of 4.45%, demonstrating its ability to effectively capture long-range interactions in modelling the equilibrium trajectory.

Table 4: **Evaluation on the Protein dataset.** F-MSE ($\times 10^{-2}$) on the AdK equilibrium trajectory dataset. **First** and **Second** denotes best results. Lower value denotes better performance.

Dataset	MPNN	RF	EGNN	EGHN	EGNO	EGHNO	GRAMO	Gain%
AdK	2.322	2.846	2.735	2.034	2.231	**1.80**	**1.72**	**4.44%**

Ablations. To assess the contributions of the temporal and spatial convolution modules in GRAMO, we perform an ablation on the Aspirin and MoCap-Run datasets. Incorporating temporal modeling consistently improves performance (see Table 5), whether combined with an EGNN or our SSM-based spatial update. When comparing the spatial components, our Mamba-based convolution outperforms EGNN on molecules, reflecting the importance of capturing long-range interactions. Conversely, the MoCap task, which is characterized by shorter-range dynamics, benefits more from the EGNN spatial update (Bishnoi et al., 2022).

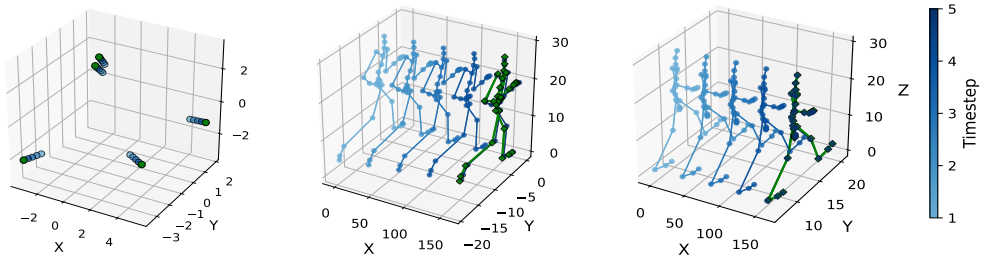
Table 5: Ablation on Aspirin and MoCap-Run. F-MSE ($\times 10^{-2}$). **First** and **Second** denotes best results. w denotes with and w/o denotes without.

Model Component	Aspirin	MoCap-Run
EGNN	9.55±0.02	50.9±0.02
GRAMO w/o Temporal	**8.10±0.05**	33.5±0.03
GRAMO w/ EGNN	8.82±0.03	**28.22±0.21**
GRAMO	**7.55±0.04**	**34.12±0.43**

We further validate GRAMO through additional experiments, including temporal loss evolution, extrapolation beyond the training horizon, data efficiency in low-data regimes, scaling with depth and embedding dimension, quantitative visualization, and physics-based metrics. Comprehensive results are provided in Appendix E.

Temporal Loss. Figure 3 demonstrates that GRAMO consistently attains lower temporal MSE across timesteps compared to EGNO on both MD17 (Aspirin) and MoCap tasks. Moreover, even when extrapolated to twice the training horizon (Appendix Figure 6), GRAMO remains stable and continues to generalize effectively, achieving robust rollouts. In contrast, EGNO exhibits instability, with error drift compounding severely on molecules such as Benzene, leading to error explosion, while GRAMO maintains reliable long-horizon predictions.

Model Efficiency. To assess data-efficiency, we evaluate GRAMO in low-data regimes. As shown in Figure 3, our model maintains strong performance and consistently surpasses EGNO on MD17 and MoCap, even when using as little as 25% of the training data. Further, we analyze the impact of



(a) (b) (c)

Figure 2: Visualization of trajectories generated by GraMO with uniform discretization on (a) N-Body Simulation, (b) Mocap (Run), and (c) Mocap (Walk). Predicted trajectories are shown with timestep progression indicated by a **Blue** color gradient, while the ground truth final snapshot is marked in **Green**.

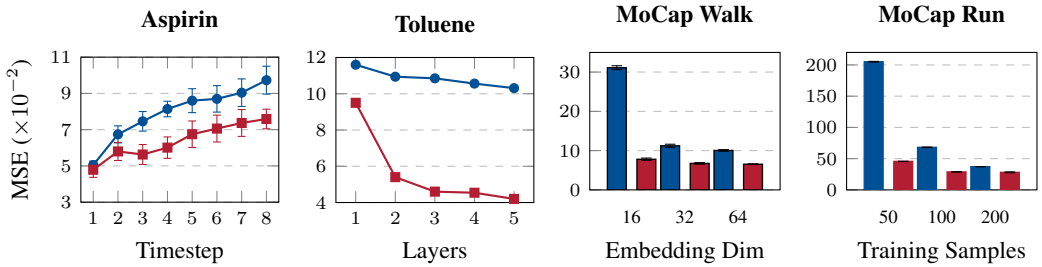


Figure 3: Comparison of GRAMO and EGNO under different experimental settings. (**Left.**) Temporal MSE across timesteps. (**Left-Middle.**) MSE as a function of the number of layers. (**Right-Middle.**) MSE with varying embedding dimensions. (**Right.**) MSE across different training sample sizes.

model depth and embedding size (shown in Figure 3). Deeper architectures improve performance while remaining stable; even a single layer surpasses EGNO on Toluene, and the model reliably captures dynamics in low-dimensional embeddings.

Visual Demonstrations. Figure 2 provides qualitative trajectory visualizations on the N-Body and MoCap datasets. The predictions closely match the ground-truth dynamics, highlighting GRAMO 's ability to model complex trajectories with both accuracy and fidelity.

Physics Metrics. To evaluate the realistic nature of the trajectory, we introduce velocity-based losses, as optimizing for position alone often leads to poor conservation of momentum and kinetic energy. This approach allows GRAMO to outperform EGNO in both positional prediction and physics-aware metrics (Appendix Tables 9–16). Specifically, our model achieves significantly lower errors in momentum and kinetic energy across all timesteps, resulting in more stable trajectory predictions. We further assess structural fidelity using system-averaged radial distribution functions (RDFs) (Bihani et al., 2024), which capture atomic density as a function of distance. Appendix Figures 11–18 show that GRAMO accurately reproduces RDFs across molecules.

6 Conclusion and Future Work

We presented GRAMO, a neural operator that integrates state-space models with graph neural networks for particle dynamics. The method combines bidirectional SSMs for temporal modeling with SSM-parameterized spatial message passing, learning entire trajectories in single forward passes rather than iterative rollouts. Experiments across molecular dynamics (MD17), human motion capture, N-body systems, and protein dynamics show consistent improvements over baselines, with relative gains of 26.3% on motion capture and 25.2% on MD17 final-state prediction. The approach maintains stability during extrapolation beyond training horizons and provides theoretical guarantees.

Future work. Key directions include incorporating SE(3) equivariance for broader physical applicability, systematic evaluation on larger systems to establish computational limits, and adaptive mechanisms for SSM state dimensions. Integration of physical conservation laws and uncertainty quantification would enhance the method's utility for scientific applications.

7 Reproducibility Statement

We ensure reproducibility by providing the full anonymous implementation at https://anonymous.4open.science/r/GraMO/, including code and training scripts.

References

Álvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in gnns: Bridging recurrent and graph learning. *arXiv preprint arXiv:2502.10818*, 2025. 2

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 119–130, 2024. 3

Vaibhav Bihani, Sajid Mannan, Utkarsh Pratiush, Tao Du, Zhimin Chen, Santiago Miret, Matthieu Micoulaut, Morten M Smedskjaer, Sayan Ranu, and NM Anoop Krishnan. Egraffbench: evaluation of equivariant graph neural network force fields for atomistic simulations. *Digital Discovery*, 3(4): 759–768, 2024. 1, 9

Suresh Bishnoi, Ravinder Bhattoo, Sayan Ranu, and NM Krishnan. Enhancing the inductive biases of graph neural ode for modeling dynamical systems. *arXiv preprint arXiv:2209.10740*, 2022. 1, 8

Suresh Bishnoi, Ravinder Bhattoo, Jayadeva Jayadeva, Sayan Ranu, and NM Anoop Krishnan. Learning the Dynamics of Physical Systems with Hamiltonian Graph Neural Networks. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023. 1

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. In *International conference on machine learning*, pp. 2806–2823. PMLR, 2023. 1, 2

Nawaf Bou-Rabee. Time integrators for molecular dynamics. *Entropy*, 16(1):138–162, 2013. 1

Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017. 7, 21

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. 3

CMU. Carnegie-mellon motion capture database. 2003. URL `http://mocap.cs.cmu.edu`. 7, 21

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024. 2

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International conference on machine learning*, pp. 7865–7885. PMLR, 2023. 2

Denis Donnelly and Edwin Rogers. Symplectic integrators: An introduction. *American Journal of Physics*, 73(10):938–945, 2005. 1

Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie-Yan Liu. SE(3) equivariant graph neural networks with complete local frames. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5583–5608. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/du22e.html`. 7

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022. 2

Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 364–373, 2021. 2

Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023. 1

Fabian B Fuchs, Daniel E Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020. 7

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*, pp. 1263–1272. PMLR, 2017. 7

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2, 3, 4, 23

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

Jiaqi Han, Wenbing Huang, Tingyang Xu, and Yu Rong. Equivariant graph hierarchy-based neural networks. *Advances in Neural Information Processing Systems*, 35:9176–9187, 2022. 7, 8, 21, 22

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 5, 22

Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6): 1129–1143, 2018. 1

Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SHbhHHfePhP. 7, 21

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7, 22

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pp. 2688–2697. Pmlr, 2018. 1, 21

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017a. 2

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017b. URL https://openreview.net/forum?id=SJU4ayYgl. 2

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019. 7

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. 2

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020. 1, 2, 5

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023. 1, 2

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019. 2

Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. Hope: High-order graph ode for modeling interacting dynamics. In *International conference on machine learning*, pp. 23124–23139. PMLR, 2023. 2

Alexander D MacKerell Jr, Nilesh Banavali, and Nicolas Foloppe. Development and current status of the charmm force field for nucleic acids. *Biopolymers: original Research on biomolecules*, 56(4): 257–265, 2000. 22

Karolis Martinkus, Aurelien Lucchi, and Nathanaël Perraudin. Scalable graph networks for particle simulations. *AAAI*, 2021. 1

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022. 2

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 1

Haohao Qu, Liangbo Ning, Rui An, Wenqi Fan, Tyler Derr, Hui Liu, Xin Xu, and Qing Li. A survey of mamba. *arXiv preprint arXiv:2408.01129*, 2024. 2

Richard J. Gowers, Max Linke, Jonathan Barnoud, Tyler J. E. Reddy, Manuel N. Melo, Sean L. Seyler, Jan Domański, David L. Dotson, Sébastien Buchoux, Ian M. Kenney, and Oliver Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In Sebastian Benthall and Scott Rostrup (eds.), *Proceedings of the 15th Python in Science Conference*, pp. 98 – 105, 2016. doi: 10.25080/Majora-629e541a-00e. 7, 22

Zahraa Al Sahili and Mariette Awad. Spatio-temporal graph neural networks: A survey. *arXiv preprint arXiv:2301.10569*, 2023. 3

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020. 1, 7

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021a. 7

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021b. 21

Mathias Schreiner, Ole Winther, and Simon Olsson. Implicit transfer operator learning: Multiple time-resolution surrogates for molecular dynamics. *arXiv preprint arXiv:2305.18046*, 2023. 7

Sean Seyler and Oliver Beckstein. Molecular dynamics trajectory for benchmarking mdanalysis. *URL: https://figshare. com/articles/Molecular_dynamics_ trajectory_for_benchmarking_MDAnalysis/5108170, doi*, 10:m9, 2017. 7, 21

Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. *arXiv preprint arXiv:2105.03902*, 2021. 21

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 4, 7

Karn Tiwari, Niladri Dutta, NM Krishnan, et al. Latent mamba operator for partial differential equations. *arXiv preprint arXiv:2505.19105*, 2025. 2, 5, 16, 23

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. 2

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*, 30, 2017. 22

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 2

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024. 2, 3

Yuanqing Wang and John Chodera. Spatial attention kinetic networks with e(n)-equivariance. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3DIpIf3wQMC. 4

Robert L Williams, Douglas A Lawrence, et al. *Linear state-space control systems*. John Wiley & Sons, 2007. 16, 17

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PzcvxEMzvQC. 21

Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. *arXiv preprint arXiv:2401.11037*, 2024. 1, 2, 5, 7, 8, 22, 23

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 3

APPENDIX TABLE OF CONTENTS

CONTENTS

A Notation and Conventions

Table 6 summarizes the mathematical notations used throughout the paper for clarity.

Table 6: **Summary of Notations.** This table compiles the notations used throughout the paper, grouped into general setup, state-space models, and model-specific parameters.

Notation	Description
General Setup and Objectives	
$\mathcal{G}$	A graph
$\{\mathcal{G}^t\}_{t=1}^T$	Sequence of graphs from $t=1$ to T
$\mathcal{G}_t = (\mathcal{V}, \mathcal{E}, \mathbf{H}_t, \mathbf{Z}_t)$	Node set $\mathcal{V}$, edge set $\mathcal{E}$, node features $\mathbf{H}_t$, and geometry $\mathbf{Z}_t$
$\lvert\mathcal{V}\rvert = N$	Number of nodes
$\mathbf{Z}^t = [\mathbf{x}^t, \mathbf{v}^t]$	Node positions $\mathbf{x}^t \in \mathbb{R}^{N\times 3}$ and velocities $\mathbf{v}^t \in \mathbb{R}^{N\times 3}$
$\mathbb{A} \in \mathbb{R}^{N\times N}$	Adjacency matrix of the graph
$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbb{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$	Normalized adjacency matrix, $\mathbf{D}$ is the degree matrix
$\mathcal{F}^\dagger$	Ground-truth solution operator
$\mathcal{F}_\theta$	Neural operator with parameters θ
p_{DATA}	Empirical data distribution
$\lVert\cdot\rVert$	Euclidean L_2 norm
P	Uniformly sampled timesteps
ΔT	Time interval
State-Space Models (SSMs)	
$x(t) \in \mathbb{R}^H$	Continuous-time input sequence
$h(t) \in \mathbb{R}^N$	Continuous-time hidden state
$y(t) \in \mathbb{R}^M$	Continuous-time output sequence
$x[k] \in \mathbb{R}^H$	Discrete-time input sequence
$h[k] \in \mathbb{R}^N$	Discrete-time hidden state
$y[k] \in \mathbb{R}^M$	Discrete-time output sequence
$\mathbf{A} \in \mathbb{R}^{N\times N}$	State matrix (continuous SSM)
$\mathbf{B} \in \mathbb{R}^{N\times H}$	Input matrix (continuous SSM)
$\mathbf{C} \in \mathbb{R}^{M\times N}$	Output matrix (continuous SSM)
$\mathbf{D} \in \mathbb{R}^{M\times H}$	Feedthrough matrix (continuous SSM)
$\bar{\mathbf{A}} \in \mathbb{R}^{N\times N}$	State matrix (discrete SSM)
$\bar{\mathbf{B}} \in \mathbb{R}^{N\times H}$	Input matrix (discrete SSM)
$\bar{\mathbf{C}} \in \mathbb{R}^{M\times N}$	Output matrix (discrete SSM)
$\bar{\mathbf{D}} \in \mathbb{R}^{M\times H}$	Feedthrough matrix (discrete SSM)
$\Delta \in \mathbb{R}^+$	Discretization step size
$\mathbf{K}$	State kernel (convolutional SSM)
$\mathcal{K}$	Kernel integral operator
Model Components	
$\mathcal{S}, \mathcal{T}, \mathcal{E}$	Spatial and temporal convolution modules, encoder
σ	Non-linear activation function
$f_B, f_C, f_{\Delta t}$	Input-dependent, learnable SSM parameters
W	Generic learnable weight matrix

B Theory and Formal Proofs

B.1 Theoretical Insights

Newtonian Dynamics: Further dynamics introduced problem statement are governed by Newtonian motion, for which the incremental form reads as follows:

$$\mathrm{d}\mathbf{Z} = \begin{bmatrix}\mathrm{d}\mathbf{x}\\ \mathrm{d}\mathbf{v}\end{bmatrix} = \begin{bmatrix}\mathbf{v}\,\mathrm{d}t\\ -\mathbf{r}\,\mathrm{d}t - \gamma\,\mathbf{v}\,\mathrm{d}t\end{bmatrix}, \tag{18}$$

where $\mathbf{r}$ encodes inter-particle forces and γ scales a friction term. Consequently, an exact trajectory $\mathbf{Z}^{(t+\Delta t)}$ exists via the solver $\mathcal{F}^\dagger$, i.e., a mapping that takes $\mathcal{G}^{(t)}$ to the function $f_{\mathcal{G}}(\Delta t)$ describing future states for $\Delta t \in [0, \Delta T]$.

Equivalence to an SSMs: Defining the first-order state $\mathbf{h} = [\mathbf{x}; \mathbf{v}]$, the Newtonian second-order dynamics can be written as follows:

$$\dot{\mathbf{h}} = \begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{v}}\end{bmatrix} = \begin{bmatrix}\mathbf{v}\\ -\mathbf{r}(\mathbf{x}, \mathbf{v}) - \gamma\mathbf{v}\end{bmatrix} \tag{19}$$

This has the generic state-space structure $\dot{\mathbf{h}} = \mathbf{A}\mathbf{h} + \mathbf{B}\,\mathbf{u}(\mathbf{h})$, e.g. with $\mathbf{A} = \begin{bmatrix}0 & I\\ 0 & -\gamma I\end{bmatrix}$ and $B = \begin{bmatrix}0\\ -I\end{bmatrix}$; if $\mathbf{r}$ is linear in $\mathbf{h}$, the system reduces to an LTI model, otherwise it is a nonlinear SSM. Hence, the Newtonian evolution admits a (possibly nonlinear) state-space representation, and the exact solver $\mathcal{F}^\dagger$ corresponds to the solution operator of this SSM mapping $\mathcal{G}^{(t)}$ to future trajectories.

B.2 Mathematical Proofs

In this section, we summarize several standard results on state-space models (SSMs) in an informal manner from the existing literature. Complete proofs can be found in the Williams et al. (2007); Tiwari et al. (2025); here, we provide a concise overview. We begin with the closed-form solution of linear continuous-time SSMs, followed by their discretization via the Zero-Order Hold (ZOH) method and its connection to the forward Euler scheme. Finally, we discuss the approximation capability of linear SSMs in capturing nonlinear dynamical systems.

We begin by analyzing the closed-form solution of the state-space model in Lemma B.1, which shows that the solution consists of the contribution from the initial condition together with a convolution against an exponential kernel.

Lemma B.1 (**Continuous SSM Solution** Williams et al. (2007))**.** *Consider a linear time-invariant system of the form*

$$\frac{d}{dt}h(t) = Ah(t) + Bx(t). \tag{20}$$

Then the state $h(t)$ admits the closed-form representation

$$h(t) = e^{A(t-t_0)}h(t_0) + \int_{t_0}^{t} e^{A(t-s)}B\,x(s)\,ds, \tag{21}$$

where $h(t_0)$ is the initial condition at time t_0.

Next, we present the discretization of the SSM parameters in Proposition B.2, and establish its equivalence to the Euler method. A complete proof is provided in Tiwari et al. (2025).

Proposition B.2 (**Discretization of SSM**)**.** *Using a Zero-Order Hold (ZOH) with sampling interval Δ, the discrete-time system matrices corresponding to the continuous-time model are given by*

$$\begin{aligned}\overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}),\\ \overline{\mathbf{B}} &= \mathbf{A}^{-1}\big(\exp(\Delta\mathbf{A}) - \mathbf{I}\big)\,\mathbf{B},\end{aligned} \tag{22}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{A}, \mathbf{B}$ denote the continuous-time system parameters.

Corollary B.3 (Euler Equivalence of SSM). *The discretization rule introduced above reduces to the forward Euler scheme when the matrix exponential is approximated by retaining only the first-order term of its Taylor expansion.*

Next, we establish an important result in Lemma B.4, which states that any nonlinear continuous dynamics can be locally approximated by a linear SSM.

Lemma B.4 (Linear Approximation of Non Linear Dynamics Williams et al. (2007)). *Let a continuous, nonlinear, and differentiable dynamical system be described by*

$$\dot{h}(t) = f(h(t), x(t), t). \tag{23}$$

Then, its dynamics can be locally approximated by a linear state-space model (SSM) of the form

$$\dot{h}(t) \approx Ah(t) + Bx(t) + O(h, x), \tag{24}$$

where the system matrices A and B are obtained from the Jacobians

$$A = \frac{\partial f}{\partial h}\big(\tilde{h}(t), \tilde{x}(t), t\big), \qquad B = \frac{\partial f}{\partial x}\big(\tilde{h}(t), \tilde{x}(t), t\big), \tag{25}$$

and $O(h, x)$ denotes higher-order infinitesimal terms.

Next, we show in the following lemma that the normalized adjacency $\hat{\mathbf{A}}$ always has spectral radius 1, which prevents the plain Neumann series from converging. Introducing a damping factor $\alpha \in (0, 1)$ ensures convergence and yields the standard resolvent form.

Lemma B.5 (Spectrum and Neumann series). *Let $\mathcal{G}$ be an undirected graph and*

$$\hat{\mathbf{A}} \;=\; \mathbf{D}^{-1/2}(\mathbb{A} + \mathbf{I})\mathbf{D}^{-1/2}, \qquad \mathbf{D} = \mathrm{Diag}(\mathbb{A}\mathbf{1} + \mathbf{1}). \tag{26}$$

Then $\sigma(\hat{\mathbf{A}}) \subseteq [-1, 1]$ and $\|\hat{\mathbf{A}}\|_2 = \rho(\hat{\mathbf{A}}) = 1$, with eigenvalue 1 having eigenvector $\mathbf{D}^{1/2}\mathbf{1}$. Hence $\sum_{t=0}^{\infty} \hat{\mathbf{A}}^t$ diverges, while for any $\alpha \in (0, 1)$,

$$\sum_{t=0}^{\infty} (\alpha\hat{\mathbf{A}})^t = (\mathbf{I} - \alpha\hat{\mathbf{A}})^{-1} \quad \textit{(converges in operator norm).}$$

Proof. $\hat{\mathbf{A}}$ is symmetric, so its spectrum is real and $\|\hat{\mathbf{A}}\|_2 = \rho(\hat{\mathbf{A}})$.

Define the normalized Laplacian as

$$\mathcal{L} := \mathbf{I} - \hat{\mathbf{A}} = \mathbf{D}^{-1/2}\big(\mathbf{D} - (\mathbb{A} + \mathbf{I})\big)\mathbf{D}^{-1/2}, \tag{27}$$

which is symmetric positive semi-definite. The quadratic form identity gives

$$z^\top \mathcal{L} z = \tfrac{1}{2}\sum_{i,j} (\mathbb{A} + \mathbf{I})_{ij}\big(y_i - y_j\big)^2, \quad y = \mathbf{D}^{-1/2} z, \tag{28}$$

hence $0 \le \mu \le 2$ for every eigenvalue μ of $\mathcal{L}$ (using $(a-b)^2 \le 2(a^2 + b^2)$). Thus every eigenvalue λ of $\hat{\mathbf{A}} = \mathbf{I} - \mathcal{L}$ satisfies $\lambda \in [-1, 1]$. Moreover,

$$\hat{\mathbf{A}}\,\mathbf{D}^{1/2}\mathbf{1} = \mathbf{D}^{-1/2}(\mathbb{A} + \mathbf{I})\mathbf{1} = \mathbf{D}^{-1/2}\mathbf{D}\mathbf{1} = \mathbf{D}^{1/2}\mathbf{1}, \tag{29}$$

so $1 \in \sigma(\hat{\mathbf{A}})$ with eigenvector $\mathbf{D}^{1/2}\mathbf{1}$. This implies $\rho(\hat{\mathbf{A}}) = 1$ and $\|\hat{\mathbf{A}}\|_2 = 1$. Hence $\hat{\mathbf{A}}^t(\mathbf{D}^{1/2}\mathbf{1}) = \mathbf{D}^{1/2}\mathbf{1}$ for all t, so $\sum_{t=0}^{T} \hat{\mathbf{A}}^t$ diverges (unbounded on that vector).

For $\alpha \in (0, 1)$, $\|\alpha\hat{\mathbf{A}}\|_2 = \alpha < 1$, so the Neumann series converges and sums to

$$\sum_{t=0}^{\infty} (\alpha\hat{\mathbf{A}})^t = (\mathbf{I} - \alpha\hat{\mathbf{A}})^{-1}.$$

□

Then we show the following proposition, which establishes that the proposed state-space update rule is permutation equivariant, meaning that relabeling the nodes of the input graph with any permutation matrix Π leads to correspondingly permuted outputs. The proof relies on the fact that the graph operator $\hat{\mathbf{A}}$ conjugates naturally under permutations, while the learnable parameters $\mathbf{W}, \mathbf{B}, \mathbf{C}$ act only on feature dimensions and thus commute with node permutations. This ensures that the model's predictions are independent of node ordering, a fundamental property for GNNs.

Proposition B.6 (Permutation Equivariance). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with $|\mathcal{V}| = N$. Consider the update*

$$\mathbf{S}^{(t+1)} = \hat{\mathbf{A}}\,\mathbf{S}^{(t)}\mathbf{W} \;+\; \mathbf{X}^{(t)}\,\mathbf{B}, \tag{30}$$

$$\mathbf{Y}^{(t+1)} = \mathbf{S}^{(t+1)}\,\mathbf{C}, \tag{31}$$

where $\mathbf{S}^{(t)} \in \mathbb{R}^{N\times n}$, $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times d_x}$, $\mathbf{Y}^{(t)} \in \mathbb{R}^{N\times d_y}$, $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$, and $\mathbf{W} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{d_x\times n}$, $\mathbf{C} \in \mathbb{R}^{n\times d_y}$. Let $\Pi \in \mathbb{R}^{N\times N}$ be any permutation matrix (node relabeling), and define

$$\hat{\mathbf{A}}_\Pi := \Pi\hat{\mathbf{A}}\Pi^\top, \qquad \mathbf{S}_\Pi^{(t)} := \Pi\mathbf{S}^{(t)}, \qquad \mathbf{X}_\Pi^{(t)} := \Pi\mathbf{X}^{(t)}.$$

Run equation 13 on the permuted triplet $(\hat{\mathbf{A}}_\Pi, \mathbf{S}_\Pi^{(t)}, \mathbf{X}_\Pi^{(t)})$ to obtain $(\mathbf{S}_\Pi^{(t+1)}, \mathbf{Y}_\Pi^{(t+1)})$. If the initialization satisfies $\mathbf{S}_\Pi^{(0)} = \Pi\mathbf{S}^{(0)}$, then for all $t \geq 0$,

$$\mathbf{S}_\Pi^{(t)} = \Pi\mathbf{S}^{(t)} \quad \textit{and} \quad \mathbf{Y}_\Pi^{(t)} = \Pi\mathbf{Y}^{(t)}.$$

Hence, the layer equation 13 is permutation equivariant.

Proof. We proceed by induction on t.

Base case. By assumption, $\mathbf{S}_\Pi^{(0)} = \Pi\mathbf{S}^{(0)}$.

Induction step. Assume $\mathbf{S}_\Pi^{(t)} = \Pi\mathbf{S}^{(t)}$. Then using equation 13 and the definitions above,

$$\mathbf{S}_\Pi^{(t+1)} = \hat{\mathbf{A}}_\Pi\mathbf{S}_\Pi^{(t)}\mathbf{W} \;+\; \mathbf{X}_\Pi^{(t)}\mathbf{B} \tag{32}$$

$$= (\Pi\hat{\mathbf{A}}\Pi^\top)(\Pi\mathbf{S}^{(t)})\mathbf{W} \;+\; (\Pi\mathbf{X}^{(t)})\mathbf{B} \tag{33}$$

$$= \Pi\big(\hat{\mathbf{A}}\mathbf{S}^{(t)}\mathbf{W} + \mathbf{X}^{(t)}\mathbf{B}\big) = \Pi\mathbf{S}^{(t+1)}. \tag{34}$$

The third equality uses $\Pi^\top\Pi = \mathbf{I}_N$ and the fact that right-multiplications by $\mathbf{W}, \mathbf{B}$ commute with left-multiplication by Π on the node dimension.

For the outputs,

$$\mathbf{Y}_\Pi^{(t+1)} = \mathbf{S}_\Pi^{(t+1)}\mathbf{C} = (\Pi\mathbf{S}^{(t+1)})\mathbf{C} = \Pi(\mathbf{S}^{(t+1)}\mathbf{C}) = \Pi\mathbf{Y}^{(t+1)}. \tag{35}$$

Thus, the claim holds for $t+1$, completing the induction. □

Proposition B.7 (Information Flow in Graph Mamba). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with $|\mathcal{V}| = N$ and normalized adjacency $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$. Consider the Graph Mamba recurrence*

$$\mathbf{S}^{(t+1)} \;=\; \hat{\mathbf{A}}\,\mathbf{S}^{(t)}\mathbf{W} \;+\; \mathbf{X}^{(t)}\mathbf{B}. \tag{36}$$

Ignoring input injection for clarity, unrolling for k steps gives

$$\mathbf{S}^{(t+k)} \;=\; \hat{\mathbf{A}}^k\,\mathbf{S}^{(t)}\,\mathbf{W}^k. \tag{37}$$

For any nodes $i, j \in V$, the (i,j)-entry of $\hat{\mathbf{A}}^k$ equals

$$\big(\hat{\mathbf{A}}^k\big)_{ij} \;=\; \sum_{(p_1,\ldots,p_{k-1})\in V^{\,k-1}} \hat{\mathbf{A}}_{ip_1}\,\hat{\mathbf{A}}_{p_1p_2}\cdots\hat{\mathbf{A}}_{p_{k-1}j}, \tag{38}$$

i.e., the total weight of all walks of length k from i to j, where each walk's weight is the product of normalized edge weights.

Proof. By definition of matrix multiplication, $(\hat{\mathbf{A}}^{k+1})_{ij} = \sum_{r\in V}(\hat{\mathbf{A}}^{k})_{ir}\hat{\mathbf{A}}_{rj}$. Expanding recursively expresses $(\hat{\mathbf{A}}^{k})_{ij}$ as a sum over all ordered k-step walks from i to j, with weights given by products of edge entries. Substituting into the recurrence yields $\mathbf{S}^{(t+k)} = \hat{\mathbf{A}}^{k}\mathbf{S}^{(t)}\mathbf{W}^{k}$, proves the claim. □

Proposition B.7 shows that repeated applications of the Graph Mamba update propagate information along walks of increasing length. Thus, each node's hidden state at time $t + k$ aggregates influences from nodes up to k hops away, enabling effective modeling of long-range dependencies.

Next Proposition B.8 characterizes how input perturbations propagate through space and time. The Jacobian decomposes into a scalar walk term $(\hat{\mathbf{A}}^{\,t-s-1})_{ij}$ capturing node-to-node influence via all walks, and a channel term $(\mathbf{B}\mathbf{W}^{\,t-s-1}\mathbf{C})$ capturing feature transformations. This separation highlights how structural diffusion and feature dynamics jointly govern spatiotemporal sensitivity.

Proposition B.8 (**Spatiotemporal Jacobian**). *Consider the update*

$$\mathbf{S}^{(t+1)} = \hat{\mathbf{A}}\,\mathbf{S}^{(t)}\mathbf{W} \;+\; \mathbf{X}^{(t)}\mathbf{B}, \tag{39}$$

$$\mathbf{Y}^{(t+1)} = \mathbf{S}^{(t+1)}\mathbf{C}. \tag{40}$$

with time–invariant parameters $\hat{\mathbf{A}}, \mathbf{W}, \mathbf{B}, \mathbf{C}$. Fix nodes i, j and times $t > s$. Then the Jacobians w.r.t. the input at time s factor into a scalar walk term over nodes and a matrix channel term:

$$\frac{\partial\,\mathbf{S}^{(t)}(i,:)}{\partial\,\mathbf{X}^{(s)}(j,:)} = \big(\hat{\mathbf{A}}^{\,t-s-1}\big)_{ij}\,\mathbf{B}\,\mathbf{W}^{\,t-s-1}, \tag{41}$$

$$\frac{\partial\,\mathbf{Y}^{(t)}(i,:)}{\partial\,\mathbf{X}^{(s)}(j,:)} = \big(\hat{\mathbf{A}}^{\,t-s-1}\big)_{ij}\,\mathbf{B}\,\mathbf{W}^{\,t-s-1}\mathbf{C}. \tag{42}$$

Here $(\hat{\mathbf{A}}^{\,m})_{ij}$ is the total weight of all length-m walks from j to i and acts as a scalar gain on the node axis, while $\mathbf{B}\mathbf{W}^{\,m}$ (or $\mathbf{B}\mathbf{W}^{\,m}\mathbf{C}$) acts on the feature axis.

Proof. Unrolling the recurrence for $m = t - s$ gives

$$\mathbf{S}^{(t)} \;=\; \hat{\mathbf{A}}^{\,m}\,\mathbf{S}^{(s)}\,\mathbf{W}^{\,m} \;+\; \sum_{k=0}^{m-1}\hat{\mathbf{A}}^{\,k}\,\mathbf{X}^{(t-1-k)}\,\mathbf{B}\,\mathbf{W}^{\,k}. \tag{43}$$

Among the terms in the sum, only the one with $t - 1 - k = s$ (i.e., $k = m - 1$) depends on $\mathbf{X}^{(s)}$, yielding

$$\frac{\partial\,\mathbf{S}^{(t)}}{\partial\,\mathbf{X}^{(s)}} \;=\; \hat{\mathbf{A}}^{\,m-1}\,(\cdot)\,\mathbf{B}\,\mathbf{W}^{\,m-1}. \tag{44}$$

Extracting the (i, j) block selects the scalar factor $(\hat{\mathbf{A}}^{\,m-1})_{ij}$ on the left, while the right-hand factor is the feature map $\mathbf{B}\,\mathbf{W}^{\,m-1}$. Finally, post-multiplication by $\mathbf{C}$ gives the above Equation. □

Proposition B.9 (**One–step Jacobian of the state update**). *Let the spatial SSM update be*

$$\mathbf{S}^{(t)} \;=\; \hat{\mathbf{A}}\,\mathbf{S}^{(t-1)}\mathbf{W} \;+\; \mathbf{X}^{(t-1)}\mathbf{B}, \tag{45}$$

with $\mathbf{S}^{(t)} \in \mathbb{R}^{N\times n}$, $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$, and $\mathbf{W} \in \mathbb{R}^{n\times n}$. Then the Jacobian of $\mathbf{S}^{(t)}$ w.r.t. $\mathbf{S}^{(t-1)}$ is the block matrix

$$\frac{\partial\,\mathbf{S}^{(t)}}{\partial\,\mathbf{S}^{(t-1)}} \;=\; \big[\,J_{ij}\,\big]_{i,j=1}^{N}, \qquad J_{ij} \;=\; \hat{\mathbf{A}}_{ij}\,\mathbf{W} \;\in \mathbb{R}^{n\times n}. \tag{46}$$

Equivalently, in vectorized form

$$\frac{\partial\,\mathrm{vec}\big(\mathbf{S}^{(t)}\big)}{\partial\,\mathrm{vec}\big(\mathbf{S}^{(t-1)}\big)} \;=\; \mathbf{W}^{\top}\otimes\hat{\mathbf{A}}. \tag{47}$$

Proof. For each node pair (i, j), we know,

$$\mathbf{S}^{(t)}(i,:) = \sum_{j=1}^{N} \hat{\mathbf{A}}_{ij}\, \mathbf{S}^{(t-1)}(j,:)\mathbf{W} + \mathbf{X}^{(t-1)}(i,:)\mathbf{B}. \tag{48}$$

The term involving $\mathbf{X}^{(t-1)}$ does not depend on $\mathbf{S}^{(t-1)}$. Differentiating the first term gives $\partial \mathbf{S}^{(t)}(i,:)/\partial \mathbf{S}^{(t-1)}(j,:) = \hat{\mathbf{A}}_{ij}\, \mathbf{W}$, yielding the stated block structure. Vectorization uses the identity $\mathrm{vec}(ASW) = (W^\top \otimes A)\,\mathrm{vec}(S)$. □

Corollary B.10 (Operator norm/spectral bound). *For any sub-multiplicative matrix norm $\|\cdot\|$,*

$$\left\| \frac{\partial \,\mathrm{vec}\big(\mathbf{S}^{(t)}\big)}{\partial \,\mathrm{vec}\big(\mathbf{S}^{(t-1)}\big)} \right\| \;\le\; \|\hat{\mathbf{A}}\|\,\|\mathbf{W}\|, \quad \textit{and} \quad \rho\left(\frac{\partial \,\mathrm{vec}\big(\mathbf{S}^{(t)}\big)}{\partial \,\mathrm{vec}\big(\mathbf{S}^{(t-1)}\big)} \right) = \rho(\hat{\mathbf{A}})\,\rho(\mathbf{W}), \tag{49}$$

where $\rho(\cdot)$ denotes spectral radius. Hence, a sufficient condition for one–step contractivity is $\|\hat{\mathbf{A}}\|\,\|\mathbf{W}\| < 1$ (or $\rho(\hat{\mathbf{A}})\rho(\mathbf{W}) < 1$ for spectral criteria).

The above corollary provides a spectral bound on the Jacobian, showing that the growth of perturbations across a single update is controlled by the product of the operator norms (or spectral radii) of $\hat{\mathbf{A}}$ and $\mathbf{W}$. This result establishes a simple contractivity condition: if $\|\hat{\mathbf{A}}\|\|\mathbf{W}\| < 1$, then the update is stable in operator norm. Building on this, Proposition B.11 extends the analysis to multiple steps, where the Jacobian naturally factorizes into powers of $\hat{\mathbf{A}}$ and $\mathbf{W}$, both in block form and in the compact vectorized form $(\mathbf{W}^\top \otimes \hat{\mathbf{A}})^{t-s}$. This highlights how repeated updates amplify or dampen signals depending on the spectral properties of $\hat{\mathbf{A}}$ and $\mathbf{W}$ and remains stable if bounded.

Proposition B.11 (Multi–step Jacobian of the state update). *Let the spatial SSM update be*

$$\mathbf{S}^{(t)} \;=\; \hat{\mathbf{A}}\,\mathbf{S}^{(t-1)}\mathbf{W} \;+\; \mathbf{X}^{(t-1)}\mathbf{B}, \tag{50}$$

with $\mathbf{S}^{(u)} \in \mathbb{R}^{N\times n}$, $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$, and $\mathbf{W} \in \mathbb{R}^{n\times n}$. For any $t > s$, the Jacobian of $\mathbf{S}^{(t)}$ w.r.t. $\mathbf{S}^{(s)}$ is

$$\frac{\partial\, \mathbf{S}^{(t)}}{\partial\, \mathbf{S}^{(s)}} \;=\; \big[\, J_{ij}^{(t-s)} \,\big]_{i,j=1}^{N}, \qquad J_{ij}^{(t-s)} \;=\; \big(\hat{\mathbf{A}}^{\,t-s}\big)_{ij}\, \mathbf{W}^{\,t-s} \in \mathbb{R}^{n\times n}. \tag{51}$$

Equivalently, in vectorized form

$$\frac{\partial \,\mathrm{vec}\big(\mathbf{S}^{(t)}\big)}{\partial \,\mathrm{vec}\big(\mathbf{S}^{(s)}\big)} \;=\; \big(\mathbf{W}^\top \otimes \hat{\mathbf{A}}\big)^{t-s}. \tag{52}$$

Proof. From the previous Proposition B.9 we know

$$\mathbf{S}^{(u)}(i,:) = \sum_{j=1}^{N} \hat{\mathbf{A}}_{ij}\, \mathbf{S}^{(u-1)}(j,:)\mathbf{W} + \mathbf{X}^{(u-1)}(i,:)\mathbf{B}. \tag{53}$$

where u denotes the timestep, and by applying the previous result we obtain:

$$\frac{\partial \,\mathrm{vec}(\mathbf{S}^{(u)})}{\partial \,\mathrm{vec}(\mathbf{S}^{(u-1)})} = \mathbf{W}^\top \otimes \hat{\mathbf{A}}. \tag{54}$$

Next, for $t > s$ using chain rule,

$$\frac{\partial \,\mathrm{vec}(\mathbf{S}^{(t)})}{\partial \,\mathrm{vec}(\mathbf{S}^{(s)})} = \prod_{u=s+1}^{t} \frac{\partial \,\mathrm{vec}(\mathbf{S}^{(u)})}{\partial \,\mathrm{vec}(\mathbf{S}^{(u-1)})} = \prod_{u=s+1}^{t} \big(\mathbf{W}^\top \otimes \hat{\mathbf{A}}\big) = \big(\mathbf{W}^\top \otimes \hat{\mathbf{A}}\big)^{t-s}, \tag{55}$$

since the factor is the same at each step. It proves the vectorized formula.

To get matrix block diagonal form, unrolling the state (ignoring inputs, which do not affect the Jacobian w.r.t. $\mathbf{S}^{(s)}$), $\mathbf{S}^{(t)} = \hat{\mathbf{A}}^{\,t-s}\,\mathbf{S}^{(s)}\,\mathbf{W}^{\,t-s}$. Differentiating entrywise gives, for each node pair (i, j), $\frac{\partial \mathbf{S}^{(t)}(i,:)}{\partial \mathbf{S}^{(s)}(j,:)} = \big(\hat{\mathbf{A}}^{\,t-s}\big)_{ij}\, \mathbf{W}^{\,t-s}$, which yields the stated block matrix $\big[\, J_{ij}^{(t-s)} \,\big]_{i,j=1}^{N}$ with $J_{ij}^{(t-s)} = \big(\hat{\mathbf{A}}^{\,t-s}\big)_{ij} \mathbf{W}^{\,t-s}$. □

Table 7: Summary statistics for molecular structures in the MD17 dataset. Shown are the number of atoms, the number of modes used to obtain optimal results, position extrema ($X_{\min}$, $X_{\max}$) and mean (X_{mean}) in Å, and velocity extrema ($V_{\min}$, $V_{\max}$) and mean (V_{mean}) in Å ps^{-1}.

Molecule	Atoms	$X_{\min}$	$X_{\max}$	X_{mean}	$V_{\min}$	$V_{\max}$	V_{mean}
Benzene	6	-178.112	197.981	−27.737	−0.004	0.003	0.000
Aspirin	13	-3.720	3.105	0.026	−0.011	0.012	0.000
Ethanol	3	-1.398	1.417	−0.004	−0.011	0.010	0.000
Malonaldehyde	5	-2.397	2.370	0.000	−0.010	0.009	0.000
Naphthalene	10	-2.597	2.593	0.000	−0.012	0.011	0.000
Salicylic	10	-2.734	2.581	−0.051	−0.013	0.012	0.000
Toluene	7	-1.990	2.630	−0.015	−0.010	0.012	0.000
Uracil	8	-2.338	2.558	0.012	−0.012	0.011	0.000

C Benchmark Details

This section provides comprehensive details for the benchmarks used in our evaluation, including dataset characteristics, training, validation, and test splits, and the graph construction process.

C.1 N-body Simulation

The N-body dataset, first introduced by Kipf et al. (2018) and subsequently adapted to 3D in Satorras et al. (2021b), contains trajectories of N charged particles evolving under pairwise Coulomb forces. Each example supplies particle charges together with initial positions and velocities; the task is to forecast future states of the system (for instance, positions at a target time). We adopt the experimental setup of Satorras et al. (2021b): $N = 5$, time horizon $\Delta T = 10$, and splits of 3000 / 2000 / 2000 trajectories for training/validation/test. Unless noted otherwise, we set $P = 5$. Node features use the speed $\|\mathbf{v}\|_2$, edge features are given by $c_i c_j$ for charges c_i, c_j, and the interaction graph is complete.

C.2 MD17

The MD17 dataset (Chmiela et al., 2017) contains molecular-dynamics trajectories for eight small molecules. For each molecule, we partition the data at random into train/val/test sets of 500/2000/2000 state–trajectory pairs. We set the prediction horizon to ΔT, which is chosen to be 3000, we use $P = 8$ and uniform discretization by default. and obtain the input velocity by finite differencing the states across this interval. Node features comprise the atom type concatenated with the speed $\|\mathbf{v}\|_2$. Following common practice, hydrogen atoms are removed, and only heavy-atom dynamics are modeled. For the graph topology, we augment the native molecular graph by adding 2-hop edges as in prior work (Shi et al., 2021; Xu et al., 2022); edge attributes are formed by concatenating the hop type, the atomic types of the two endpoints, and the chemical bond type. Furthermore, Table 7 provides the complete summary statistics for each molecular structure.

C.3 CMU Motion Capture

The CMU Motion Capture dataset (CMU, 2003) provides 3D trajectories of human actions. We evaluate on two subsets—Subject #35 (`Walk`) and Subject #9 (`Run`)—adopting the data splits and preprocessing of Huang et al. (2022); Han et al. (2022). Subject #35 uses 200 / 600 / 600 trajectories for train / val / test, and Subject #9 uses 200 / 240 / 240. Each sample is represented as a skeletal graph with 31 nodes (joint locations) and edges encoding the connecting bones. As with the N-body experiments, inputs comprise the nodes' initial positions and velocities; we set the prediction horizon $\Delta T = 30$, use $P = 5$ unless stated otherwise, and employ uniform temporal discretization by default.

C.4 Protein

We utilize the preprocessed AdK equilibrium trajectories provided by Han et al. (2022), which originate from the AdK dataset of Seyler & Beckstein (2017) and are made available via the MD-

Analysis toolkit (Richard J. Gowers et al., 2016). The simulations employ the CHARMM27 force field (MacKerell Jr et al., 2000) with explicit solvent and ions under NPT conditions at 300 K and 1 bar. Trajectory frames were recorded every 240 ps, yielding a total simulated time of 1.004 μs. Following Han et al. (2022), we adopt their partitioning into 2481 training, 827 validation, and 878 test sub-trajectories. and use $P = 4$. For model construction, we retain the protein backbone and form graphs by connecting atoms within a 10 Å cutoff.

D Implementation Details

This section outlines implementation details of the proposed method to ensure clarity.

D.1 Training Details

Encoder. From a theoretical standpoint, GRAMO directly addresses the state-to-trajectory task, whereas our formulation employs neural operators to learn mappings between function spaces. This requires defining an input function $f(t)$ (i.e., a trajectory) that evolves over time; a simple implementation is to repeat $\mathcal{G}(t)$, yielding a constant trajectory similar to Xu et al. (2024). From a practical standpoint, GRAMO layers are equipped with residual connections, which demand input and output tensors of identical shape. Consequently, $\mathcal{G}(t)$ is repeated to match the length of the predicted trajectory.

Time Embedding. To incorporate temporal information, GRAMO augments the input features with explicit encoding of the time index for each structure in the trajectory. Specifically, we construct a set of sinusoidal functions at varying frequencies to generate fixed-time embeddings. For a timestep Δt_i, the embedding is given as follows:

$$\begin{aligned} \text{emb}_{2j} &= \sin\left(\frac{i}{10000^{2j/d_{\text{emb}}}}\right), \\ \text{emb}_{2j+1} &= \cos\left(\frac{i}{10000^{2j/d_{\text{emb}}}}\right), \end{aligned} \tag{56}$$

where d_{emb} is the dimensionality of the time embedding space.

To ensure the model is aware of the timestep information required for future trajectory prediction, we incorporate *temporal embeddings*. This practice is widely adopted across domains—for example, positional encodings in large language models (Vaswani et al., 2017) and timestep embeddings in diffusion models (Ho et al., 2020).

D.2 Hyperparameter Details

We provide detailed hyperparameter settings for all datasets in Table D.2. Here, `batch` denotes batch size, `lr` is the learning rate, `wd` is the weight decay, `layer` is the number of network layers, `hidden` is the hidden dimension, `timestep` is the number of time steps, and `time_emb` is the dimension of the time embedding. All models are trained using the Adam optimizer (Kingma & Ba, 2014) with a StepLR scheduler. Models are trained until convergence, with early stopping triggered by validation loss with patience of 50 epochs, and a maximum of 2000 training epochs. For fair comparison with baselines, we use the same number of layers as EGNO.

Table 8: Summary of hyperparameter for GRAMO on all datasets.

	Batch	LR	Weight Decay	Layer	Embedding Dim	Timestep	Time_emb
N-body	100	1e-4	1e-8	4	64	5	32
Walk/Run	12	5e-4	1e-10	6	128	5	32
MD17	100	1e-4	1e-15	6	64	8	32
Protein	4	5e-5	1e-4	4	128	4	32

D.3 BASELINES DETAILS

The baseline results reported in the main table are taken from the EGNO paper (Xu et al., 2024). Since EGNO represents the strongest existing operator, we further re-ran its official codebase to ensure fairness and consistency. All results are averaged over three independent runs using the official implementation to provide reliable and reproducible benchmark comparisons.

D.4 ALGORITHM

In this subsection, we present the complete block-level algorithm for GRAMO. Algorithm 1 describes the temporal convolution module, which employs bidirectional Mamba (Gu & Dao, 2023; Tiwari et al., 2025) by discretizing the SSM and applying both forward and backward updates as defined in Equation 2. This operation is applied across the temporal dimension of nodes, producing updated node features. Subsequently, Algorithm 2 defines the Graph Mamba update, which processes temporal graph snapshots using the update rule from Equation 13.

Algorithm 1 GRAMO Temporal Convolution

Require: Input Node Features of Snapshots $X \in \mathbb{R}^{BN\times P\times D}$,
Ensure: Output Node Features $Y \in \mathbb{R}^{BN\times P\times D}$

Project input: $[z, xBC, dt] \leftarrow \text{split}(\text{input\_projection}(X))$
Compute $A \leftarrow -\exp(A_{\log})$
Initialize states if learnable
Adjust Δt: concatenate flipped halves, apply softplus
Convolve: $xBC \leftarrow \text{act}(\text{conv1d}(xBC))$
Split: $(x, BC) \leftarrow \text{split}(xBC)$
Duplicate & flip x, split BC into (B, C)
Apply chunk scan: $y \leftarrow \text{mamba\_chunk\_scan}(x, dt, A, B, C)$
Rearrange y, roll by 1, reset first timestep
Split into forward/backward: $y \leftarrow y_{\text{fw}} + y_{\text{bw}} + x_{\text{og}} \cdot \text{linear}(x_{\text{og}})$
Apply gated norm: $y \leftarrow \text{norm}(y, z)$
Output: $Y \leftarrow \text{output\_projection}(y)$

Algorithm 2 GRAMO Spatial Convolution

Require: Input snapshots $\{X_j\}_1^P$, adjacency $\hat{\mathbf{A}}$
Ensure: Updated representations $\{X_j\}$

Compute residuals $X_j^{\text{res}} \leftarrow \text{ResConn}_i(X_j)$
Initialize lists $dts, B, C \leftarrow \emptyset$
for each snapshot j **do**
    Apply graph convolution $[X_j, dt, B_j, C_j] \leftarrow \text{Conv}_i(X_j, \text{edge\_index})$
    Append dt, B_j, C_j to dts, B, C
end for
Perform token mixing $X'_j \leftarrow \text{TokenMixer}_i(\{X_j\})$
Initialize hidden state $S \leftarrow 0$
for each snapshot j **do**
    Compute step size $\Delta t \leftarrow \text{softplus}(dts[j] + \delta_i)$
    Compute input transform $B_X \leftarrow \text{einsum}(B_j, X'_j)$
    Propagate state $S \leftarrow \hat{\mathbf{A}}_{\text{norm}} S W_i + B_X$
    Update representation $X_j \leftarrow \text{Expand}_i(\sigma(\text{einsum}(S, C_j)) + X_j^{\text{res}})$
end for

E ADDITIONAL RESULTS

In this section, we present a set of complementary experiments—including temporal error analysis, scaling studies, data efficiency evaluations, physics-informed losses, and radial distribution function (RDF) analyses. These results provide a more comprehensive understanding of the proposed method in comparison with baseline models, highlighting its effectiveness for particle-based simulations.

E.1 TEMPORAL LOSS ANALYSIS

In this experiment, we evaluate the temporal loss across timesteps to assess how prediction errors accumulate across timesteps. Figure 4 compares our proposed GRAMO with EGNO across the MD17 dataset. The results show that GRAMO consistently outperforms EGNO for all molecules, demonstrating better stability over timesteps. In particular, GRAMO achieves significant performance on Aspirin, Ethanol, Toluene, and Benzene, which span diverse challenges in MD17—from large multifunctional flexibility (Aspirin), to torsional motions (Ethanol), mixed rigid–flexible structures (Toluene), and highly symmetric aromatic systems (Benzene). These results highlight the ability of GRAMO to better capture long-range molecular dynamics interactions.

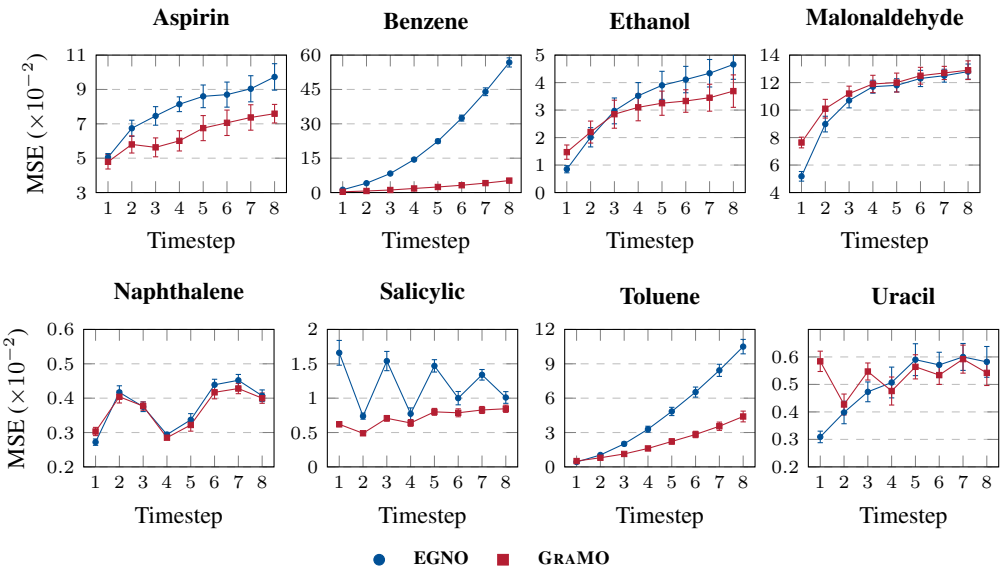


Figure 4: **Temporal Loss on MD17.** Mean squared error (MSE) ($\times 10^{-2}$) across timesteps for all MD17 molecules, comparing EGNO and GRAMO.

Further, we evaluate the temporal loss across timesteps on the motion capture datasets. As shown in Figure 5, our proposed GRAMO demonstrates consistently lower timestep compared to EGNO. This indicates that GRAMO is able to maintain stable and accurate predictions over horizons, capturing both the temporal dependencies and articulated joint dynamics more effectively. The improved performance across timesteps highlights the robustness of GRAMO in modelling long-term human motion trajectories.

E.2 EXTRAPOLATION ANALYSIS

In this experiment, we evaluate model performance under temporal extrapolation, extending beyond the training horizon to assess generalization on unseen timesteps. As shown in Figure 6, GRAMO not only achieves strong accuracy within the training horizon but also maintains stability when extrapolated to longer rollouts. Furthermore, Table 7 highlights that benzene exhibits substantial drift during simulation. While EGNO struggles to capture the global dynamics—leading to exploding temporal losses under extrapolation—GRAMO remains stable and continues to deliver accurate predictions. These results demonstrate that GRAMO more effectively captures global dynamics and long-range interactions, yielding superior performance even beyond the training horizon.

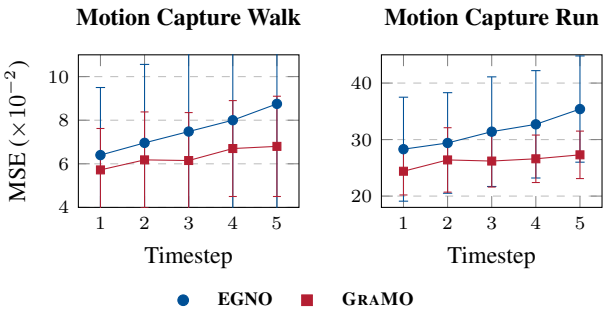


Figure 5: **Temporal Loss on Motion Capture.** MSE ($\times 10^{-2}$) across timesteps for motion capture tasks, comparing EGNO and GRAMO. Lower MSE indicates better performance.

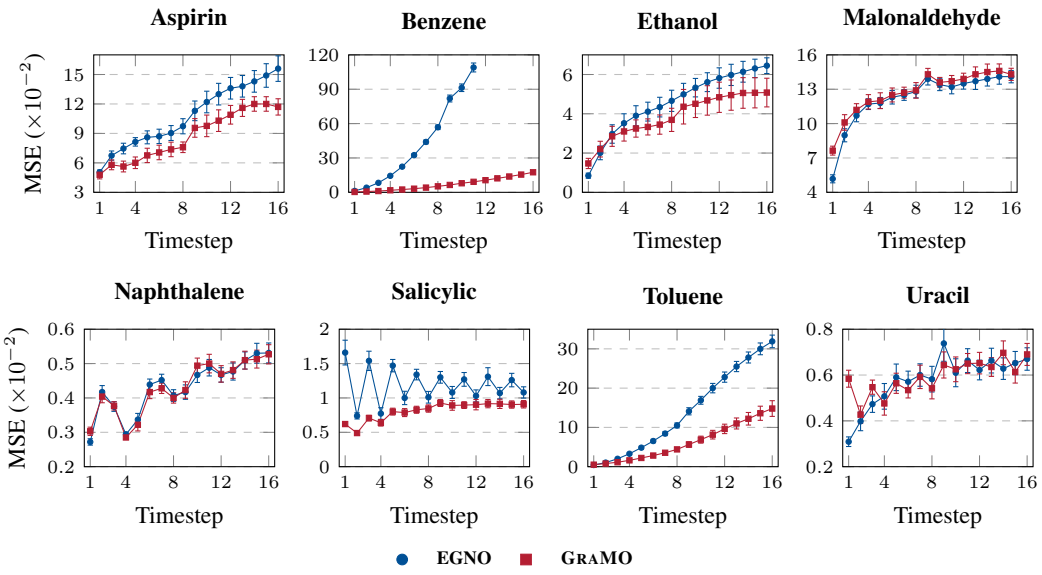


Figure 6: **Temporal Loss on MD17 Extrapolation.** MSE ($\times 10^{-2}$) across timesteps for the MD17 dataset, comparing EGNO and GRAMO. Models are trained on 8 timesteps and extrapolated to predict the subsequent 8 timesteps. Lower MSE indicates better performance.

E.3 SCALABILITY AND DATA EFFICIENCY ANALYSIS

In this experiment, we evaluate the performance of EGNO and GRAMO as a function of network depth. As shown in Figure 7, GRAMO exhibits more favorable scaling with the number of layers, consistently improving as depth increases. Notably, it achieves performance comparable to or better than EGNO even with fewer layers, highlighting its superior scalability and efficiency.

Next, we evaluate the data efficiency of our approach on the MD17 dataset. As shown in Table 8, our method outperforms EGNO even with substantially fewer training samples. This highlights the robustness of the proposed model in data-scarce regimes, where only limited training data are available. We further extend the data efficiency analysis to the MoCap dataset, as shown in Figure 9. Our method achieves consistent and significant improvements over EGNO, demonstrating stronger temporal modelling capabilities. Notably, even when trained with only 25% of the dataset, the proposed approach outperforms the baseline, underscoring its effectiveness in low-data regimes.

Finally, we assess the effect of embedding dimension on the motion capture datasets. As shown in Figure 10, our method significantly outperforms EGNO even at the lowest dimension (16), and maintains better performance as the embedding dimension increases.

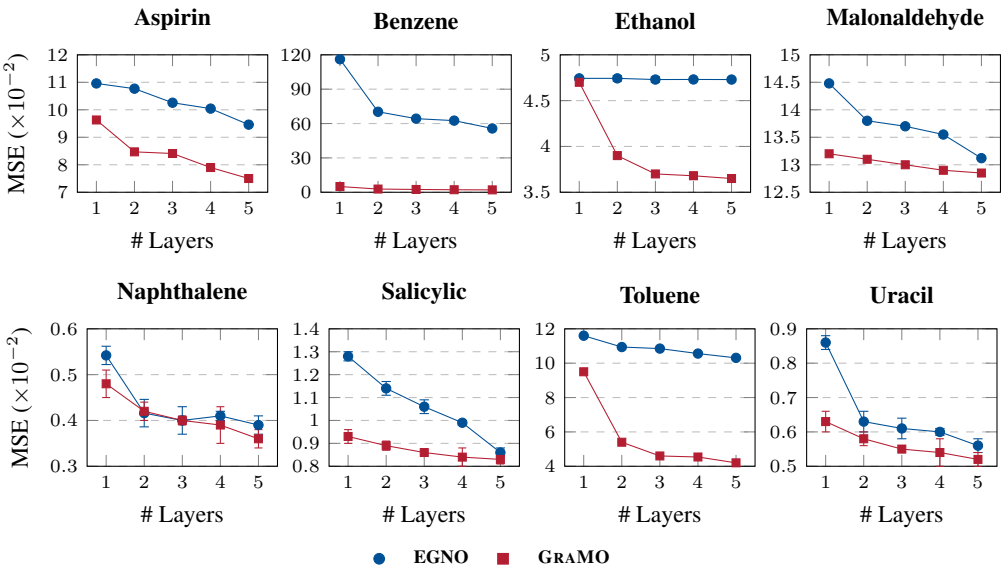


Figure 7: **Scaling with Layers on MD17.** MSE ($\times 10^{-2}$) for EGNO and GRAMO across different molecular systems as a function of the number of layers. Lower MSE indicates better performance.

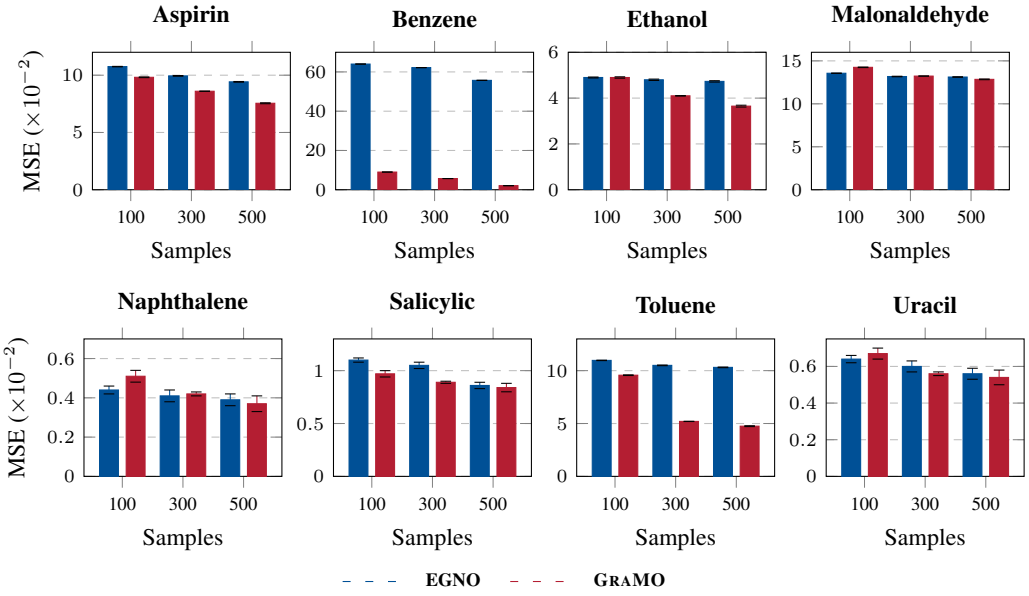


Figure 8: **Data Efficiency on MD17.** Comparison of EGNO and GRAMO across MD17 molecular systems under varying training sample sizes. Results are reported as MSE ($\times 10^{-2}$). Lower MSE indicates better performance.

E.4 PHYSICS-BASED METRICS ANALYSIS

In this experiment, we evaluate the *physics loss* of the operator, focusing on momentum and kinetic energy. Prior work primarily optimized over position only, which led to poor preservation of physical quantities across molecules for all operators. To address this, we introduce an additional velocity loss alongside the position loss, ensuring better alignment with desired physical constraints. As shown in Tables 9–16, GRAMO consistently outperforms EGNO not only in position loss but also in physics-aware metrics such as momentum and kinetic energy, achieving significantly lower errors.

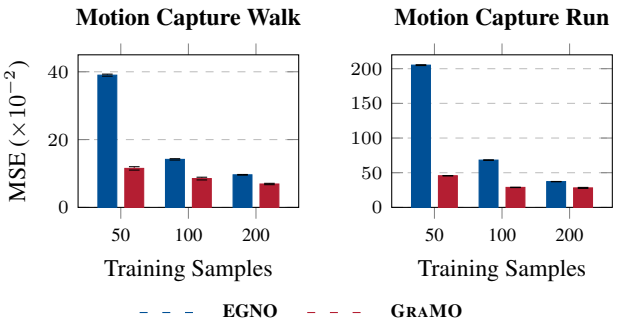


Figure 9: **Data Efficiency on Motion Capture.** Comparison of EGNO and GRAMO under varying training sample sizes. Results are reported as MSE ($\times 10^{-2}$). Lower MSE indicates better performance.

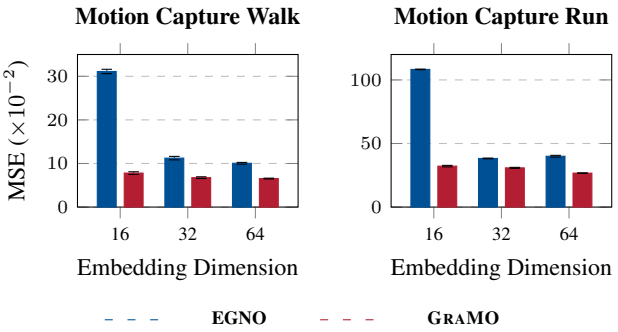


Figure 10: **Embedding Dimension on Motion Capture.** Comparison of EGNO and GRAMO under varying training sample sizes. Results are reported as MSE ($\times 10^{-2}$). Lower MSE indicates better performance.

E.5 RADIAL DISTRIBUTION FUNCTION ANALYSIS

The radial distribution function (RDF), $g(r)$, measures the likelihood of finding a particle at distance r from a reference particle and serves as a key descriptor of molecular structure. It reflects local ordering, coordination shells, and medium-range correlations. Comparing predicted and ground-truth RDFs across rollouts thus provides a stringent test of a model's ability to preserve geometry and capture physically meaningful interactions. Figures 11–18 show that GRAMO closely matches the ground-truth distributions, indicating robust structural fidelity.

F QUALITATIVE VISUALIZATIONS

In this section, we present visualizations of the dynamics predicted by GRAMO. Results for particle simulations, MOCAP-WALK, and MOCAP-RUN are shown in Figures 19–21. As illustrated, GRAMO not only produces accurate final-state predictions but also effectively captures the underlying dynamics by explicitly modelling both spatial and temporal correlations. Furthermore, Figure 22 provides a qualitative visualization of the trajectories predicted by GRAMO. The figure illustrates how the model accurately reproduces the temporal evolution of the system, closely aligning with the ground-truth dynamics and capturing both local and global structural changes over time.

G IMPACT STATEMENT

This work presents a new graph neural framework for particle-based simulations, leveraging state-space model (SSM) inspired message passing to capture long-range interactions. Our primary aim is to advance machine learning methods for scientific simulation by addressing fundamental modelling challenges in particle dynamics. The contributions are technical and scientific in nature, and we do not foresee any direct ethical or societal risks arising from this work.

Table 9: **Physics Loss for Aspirin**. Lower value shows better metrics.

	Position ($\times 10^{-2}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-2}$)		Kinetic ($\times 10^{-6}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	4.41 ± 0.23	4.43 ± 0.23	63.4 ± 0.74	1.85 ± 0.045	11.6 ± 0.13	0.33 ± 0.0077	248 ± 5.7	0.050 ± 0.0036
2	6.18 ± 0.46	5.67 ± 0.45	54.4 ± 0.58	1.84 ± 0.028	9.89 ± 0.11	0.33 ± 0.0050	174 ± 4.6	0.051 ± 0.0037
3	6.68 ± 0.56	6.16 ± 0.50	48.7 ± 0.45	1.84 ± 0.031	8.77 ± 0.088	0.33 ± 0.0057	134 ± 3.9	0.051 ± 0.0036
4	7.33 ± 0.47	6.64 ± 0.44	47.7 ± 0.50	1.85 ± 0.033	8.46 ± 0.10	0.33 ± 0.0065	123 ± 3.8	0.051 ± 0.0038
5	8.16 ± 0.68	7.41 ± 0.55	48.0 ± 0.56	1.84 ± 0.043	8.42 ± 0.11	0.33 ± 0.0077	127 ± 4.2	0.051 ± 0.0036
6	8.41 ± 0.76	7.92 ± 0.66	53.5 ± 0.58	1.83 ± 0.037	9.42 ± 0.11	0.33 ± 0.0068	161 ± 5.2	0.051 ± 0.0041
7	8.77 ± 0.80	8.33 ± 0.73	63.7 ± 0.73	1.83 ± 0.046	11.4 ± 0.13	0.33 ± 0.0089	235 ± 6.4	0.051 ± 0.0034
8	9.51 ± 0.82	8.65 ± 0.66	69.0 ± 0.62	1.84 ± 0.041	12.5 ± 0.11	0.33 ± 0.0070	287 ± 6.6	0.052 ± 0.0044

Table 10: **Physics Loss for Benzene**. Lower value shows better metrics.

	Position ($\times 10^{-2}$)		Velocity ($\times 10^{-4}$)		Momentum ($\times 10^{-2}$)		Kinetic ($\times 10^{-6}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	2.21 ± 0.050	0.35 ± 0.022	20.0 ± 0.54	0.54 ± 0.017	28.9 ± 0.78	0.78 ± 0.024	1710 ± 97	1.61 ± 0.20
2	6.06 ± 0.17	0.43 ± 0.033	20.2 ± 0.55	0.56 ± 0.030	29.2 ± 0.80	0.81 ± 0.043	1730 ± 97	1.79 ± 0.24
3	11.5 ± 0.36	0.55 ± 0.051	14.9 ± 0.46	0.42 ± 0.019	21.5 ± 0.67	0.60 ± 0.027	989 ± 62	0.96 ± 0.11
4	19.0 ± 0.52	0.72 ± 0.061	9.12 ± 0.32	0.62 ± 0.021	13.2 ± 0.47	0.89 ± 0.031	409 ± 30	2.60 ± 0.20
5	28.4 ± 0.80	0.86 ± 0.080	6.41 ± 0.26	0.71 ± 0.048	9.25 ± 0.37	1.03 ± 0.069	216 ± 17	3.66 ± 0.48
6	39.4 ± 1.1	1.06 ± 0.10	6.38 ± 0.25	0.91 ± 0.044	9.21 ± 0.37	1.31 ± 0.064	213 ± 17	5.93 ± 0.77
7	51.6 ± 1.5	1.31 ± 0.12	9.03 ± 0.33	1.12 ± 0.051	13.0 ± 0.47	1.62 ± 0.074	395 ± 29	8.11 ± 0.92
8	64.0 ± 1.9	1.69 ± 0.15	14.7 ± 0.45	1.37 ± 0.071	21.2 ± 0.65	1.97 ± 0.10	956 ± 61	12.4 ± 1.8

H USE OF LARGE LANGUAGE MODELS (LLMS)

We employed a Large Language Model (LLM) as a supporting tool to refine the manuscript text and to automatically generate LaTeX code for figures and plots. This integration streamlined the creation of consistent, high-quality visualizations and improved the overall readability of the presentation.

Table 11: **Physics Loss for Ethanol**. Lower value shows better metrics.

	Position ($\times 10^{-2}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-8}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	0.88 $\pm$0.13	1.29 $\pm$0.15	1.17 $\pm$0.069	1.14 $\pm$0.065	1.92 $\pm$0.12	1.88 $\pm$0.095	2.19 $\pm$0.28	2.09 $\pm$0.34
2	2.05 $\pm$0.35	2.25 $\pm$0.33	1.28 $\pm$0.072	1.12 $\pm$0.055	2.13 $\pm$0.12	1.85 $\pm$0.086	2.25 $\pm$0.26	2.00 $\pm$0.36
3	3.00 $\pm$0.46	2.90 $\pm$0.49	1.23 $\pm$0.066	1.16 $\pm$0.060	2.05 $\pm$0.11	1.92 $\pm$0.096	2.22 $\pm$0.32	2.01 $\pm$0.36
4	3.54 $\pm$0.48	3.26 $\pm$0.50	1.19 $\pm$0.073	1.15 $\pm$0.039	1.98 $\pm$0.11	1.90 $\pm$0.055	2.07 $\pm$0.33	2.00 $\pm$0.34
5	3.91 $\pm$0.51	3.54 $\pm$0.49	1.18 $\pm$0.052	1.16 $\pm$0.043	1.97 $\pm$0.083	1.91 $\pm$0.068	2.04 $\pm$0.22	2.00 $\pm$0.30
6	4.12 $\pm$0.49	3.77 $\pm$0.52	1.18 $\pm$0.057	1.15 $\pm$0.058	1.96 $\pm$0.099	1.90 $\pm$0.090	1.96 $\pm$0.29	2.12 $\pm$0.37
7	4.34 $\pm$0.50	3.92 $\pm$0.57	1.18 $\pm$0.072	1.15 $\pm$0.072	1.96 $\pm$0.12	1.91 $\pm$0.11	1.99 $\pm$0.28	2.15 $\pm$0.40
8	4.66 $\pm$0.54	4.18 $\pm$0.63	1.20 $\pm$0.078	1.18 $\pm$0.052	1.99 $\pm$0.12	1.95 $\pm$0.083	2.15 $\pm$0.34	2.25 $\pm$0.49

Table 12: **Physics Loss for Malonaldehyde**. Lower value shows better metrics.

	Position ($\times 10^{-1}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-8}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	0.52 $\pm$0.032	0.70 $\pm$0.038	9.09 $\pm$0.31	1.78 $\pm$0.067	18.1 $\pm$0.60	3.16 $\pm$0.12	53.3 $\pm$4.7	4.11 $\pm$0.48
2	0.90 $\pm$0.056	1.04 $\pm$0.061	11.4 $\pm$0.29	1.84 $\pm$0.069	24.3 $\pm$0.56	3.29 $\pm$0.12	90.3 $\pm$5.6	4.39 $\pm$0.58
3	1.07 $\pm$0.055	1.15 $\pm$0.061	14.6 $\pm$0.59	1.80 $\pm$0.064	33.0 $\pm$1.4	3.19 $\pm$0.11	226 $\pm$18	3.95 $\pm$0.54
4	1.16 $\pm$0.045	1.22 $\pm$0.056	16.8 $\pm$0.67	1.80 $\pm$0.072	38.4 $\pm$1.7	3.19 $\pm$0.12	346 $\pm$29	4.01 $\pm$0.51
5	1.18 $\pm$0.047	1.22 $\pm$0.066	13.7 $\pm$0.55	1.79 $\pm$0.064	31.2 $\pm$1.3	3.17 $\pm$0.11	226 $\pm$20	3.98 $\pm$0.51
6	1.23 $\pm$0.058	1.27 $\pm$0.060	7.78 $\pm$0.31	1.80 $\pm$0.048	16.9 $\pm$0.70	3.20 $\pm$0.086	59.0 $\pm$6.0	4.01 $\pm$0.51
7	1.25 $\pm$0.043	1.29 $\pm$0.043	5.39 $\pm$0.18	1.80 $\pm$0.061	10.4 $\pm$0.35	3.18 $\pm$0.10	13.9 $\pm$1.5	3.90 $\pm$0.53
8	1.28 $\pm$0.050	1.32 $\pm$0.057	7.18 $\pm$0.25	1.82 $\pm$0.059	13.7 $\pm$0.53	3.21 $\pm$0.11	28.3 $\pm$2.8	4.05 $\pm$0.57

Table 13: **Physics Loss for Naphthalene**. Lower value shows better metrics.

	Position ($\times 10^{-3}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-7}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	2.91 $\pm$0.11	3.10 $\pm$0.14	7.92 $\pm$0.10	3.26 $\pm$0.076	11.4 $\pm$0.15	4.70 $\pm$0.11	17.0 $\pm$0.45	2.30 $\pm$0.18
2	3.64 $\pm$0.13	4.23 $\pm$0.19	6.96 $\pm$0.13	3.00 $\pm$0.093	10.0 $\pm$0.19	4.33 $\pm$0.13	11.9 $\pm$0.36	2.14 $\pm$0.14
3	3.78 $\pm$0.13	3.81 $\pm$0.11	5.75 $\pm$0.12	2.13 $\pm$0.055	8.29 $\pm$0.17	3.07 $\pm$0.079	7.24 $\pm$0.25	0.75 $\pm$0.056
4	3.04 $\pm$0.082	2.86 $\pm$0.062	5.10 $\pm$0.085	2.55 $\pm$0.077	7.36 $\pm$0.12	3.68 $\pm$0.11	5.13 $\pm$0.18	1.25 $\pm$0.14
5	3.40 $\pm$0.16	3.24 $\pm$0.19	5.09 $\pm$0.074	2.64 $\pm$0.076	7.34 $\pm$0.11	3.80 $\pm$0.11	5.19 $\pm$0.17	1.34 $\pm$0.18
6	4.42 $\pm$0.14	4.29 $\pm$0.18	6.01 $\pm$0.070	2.35 $\pm$0.057	8.67 $\pm$0.10	3.38 $\pm$0.082	7.86 $\pm$0.20	0.74 $\pm$0.036
7	4.65 $\pm$0.17	4.39 $\pm$0.16	7.32 $\pm$0.097	2.34 $\pm$0.068	10.6 $\pm$0.14	3.38 $\pm$0.097	13.5 $\pm$0.35	0.73 $\pm$0.052
8	4.24 $\pm$0.16	4.07 $\pm$0.16	8.15 $\pm$0.12	2.71 $\pm$0.076	11.8 $\pm$0.18	3.91 $\pm$0.11	18.0 $\pm$0.43	1.26 $\pm$0.10

Table 14: **Physics Loss for Salicylic**. Lower value shows better metrics.

	Position ($\times 10^{-3}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-6}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	17.7 ±2.0	6.32 ±0.36	9.51 ±0.11	4.68 ±0.26	20.0 ±0.23	8.78 ±0.57	12.2 ±0.13	1.93 ±0.44
2	6.68 ±0.44	4.91 ±0.30	8.74 ±0.11	4.45 ±0.13	18.0 ±0.23	7.89 ±0.28	9.00 ±0.098	1.11 ±0.20
3	16.2 ±1.6	7.24 ±0.47	7.69 ±0.088	3.51 ±0.21	15.3 ±0.18	6.83 ±0.47	5.75 ±0.065	1.20 ±1.6
4	7.53 ±0.85	6.44 ±0.51	7.69 ±0.10	3.58 ±0.16	14.6 ±0.22	6.66 ±0.36	4.27 ±0.053	1.11 ±0.80
5	15.5 ±0.95	7.93 ±0.48	8.16 ±0.088	3.19 ±0.10	15.4 ±0.17	6.11 ±0.24	4.43 ±0.051	0.71 ±0.26
6	10.4 ±1.0	7.77 ±0.46	8.43 ±0.090	3.88 ±0.14	16.3 ±0.18	7.41 ±0.35	6.07 ±0.065	1.35 ±1.1
7	14.2 ±0.79	8.27 ±0.34	8.92 ±0.090	3.28 ±0.10	18.0 ±0.19	6.23 ±0.24	9.28 ±0.11	0.62 ±0.24
8	10.7 ±0.93	8.49 ±0.54	9.62 ±0.091	3.82 ±0.16	20.0 ±0.19	7.04 ±0.40	12.3 ±0.12	1.43 ±1.1

Table 15: **Physics Loss for Toluene**. Lower value shows better metrics.

	Position ($\times 10^{-2}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-8}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	0.46 ±0.027	0.72 ±0.034	6.97 ±0.20	1.98 ±0.049	10.1 ±0.29	2.86 ±0.071	159 ±7.6	4.62 ±0.35
2	1.15 ±0.094	1.13 ±0.092	7.11 ±0.18	2.01 ±0.045	10.3 ±0.26	2.90 ±0.064	164 ±7.7	4.91 ±0.35
3	2.16 ±0.21	1.78 ±0.17	6.21 ±0.12	1.98 ±0.052	8.96 ±0.18	2.86 ±0.074	110 ±5.4	4.76 ±0.32
4	3.47 ±0.33	2.60 ±0.26	4.85 ±0.10	1.98 ±0.049	7.00 ±0.15	2.85 ±0.071	60.0 ±3.2	4.68 ±0.39
5	5.08 ±0.46	3.61 ±0.36	4.12 ±0.087	1.97 ±0.049	5.95 ±0.13	2.85 ±0.071	40.7 ±2.3	4.67 ±0.35
6	6.84 ±0.55	4.74 ±0.44	4.02 ±0.092	1.98 ±0.046	5.80 ±0.13	2.85 ±0.067	41.7 ±2.7	4.51 ±0.31
7	8.80 ±0.65	6.05 ±0.57	4.62 ±0.10	1.95 ±0.060	6.67 ±0.15	2.81 ±0.086	59.1 ±3.9	4.49 ±0.32
8	10.9 ±0.76	7.51 ±0.73	5.77 ±0.13	1.94 ±0.044	8.33 ±0.18	2.79 ±0.064	103 ±5.5	4.44 ±0.32

Table 16: **Physics Loss for Uracil**. Lower value shows better metrics.

	Position ($\times 10^{-3}$)		Velocity ($\times 10^{-5}$)		Momentum ($\times 10^{-3}$)		Kinetic ($\times 10^{-7}$)	
P	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO	EGNO	GRAMO
1	4.41 ±0.32	5.52 ±0.41	8.44 ±0.54	2.95 ±0.13	16.7 ±1.1	5.67 ±0.28	140 ±30	4.50 ±0.93
2	4.28 ±0.41	4.20 ±0.40	7.44 ±0.34	2.50 ±0.10	14.2 ±0.66	4.82 ±0.20	97.1 ±17	2.25 ±0.29
3	5.08 ±0.24	5.47 ±0.26	6.88 ±0.35	2.20 ±0.080	12.6 ±0.64	4.13 ±0.17	55.7 ±7.3	1.69 ±0.40
4	5.25 ±0.56	4.66 ±0.48	7.46 ±0.35	2.48 ±0.070	13.5 ±0.64	4.81 ±0.14	43.1 ±5.7	2.23 ±0.31
5	6.07 ±0.48	5.82 ±0.38	7.81 ±0.28	2.34 ±0.065	14.3 ±0.50	4.47 ±0.14	43.0 ±5.9	1.78 ±0.29
6	5.70 ±0.41	5.28 ±0.33	7.61 ±0.25	2.42 ±0.090	14.3 ±0.53	4.45 ±0.17	55.9 ±14	1.36 ±0.12
7	6.30 ±0.44	6.10 ±0.49	7.26 ±0.48	2.72 ±0.047	14.2 ±1.0	4.79 ±0.098	94.0 ±27	1.70 ±0.18
8	6.18 ±0.61	5.66 ±0.51	7.42 ±0.48	2.23 ±0.057	14.7 ±1.0	4.10 ±0.12	137 ±36	1.08 ±0.13

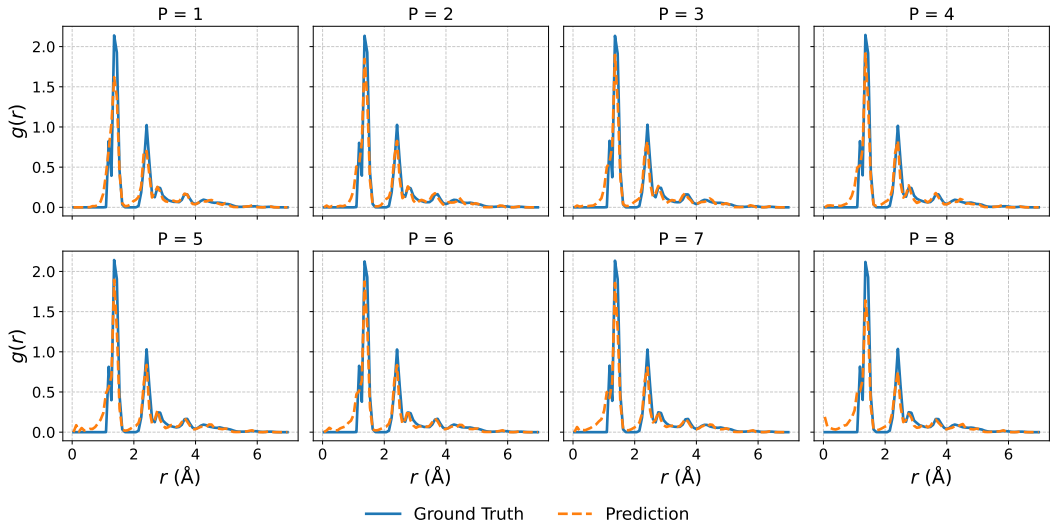


Figure 11: **Radial distribution function (RDF) for Aspirin.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

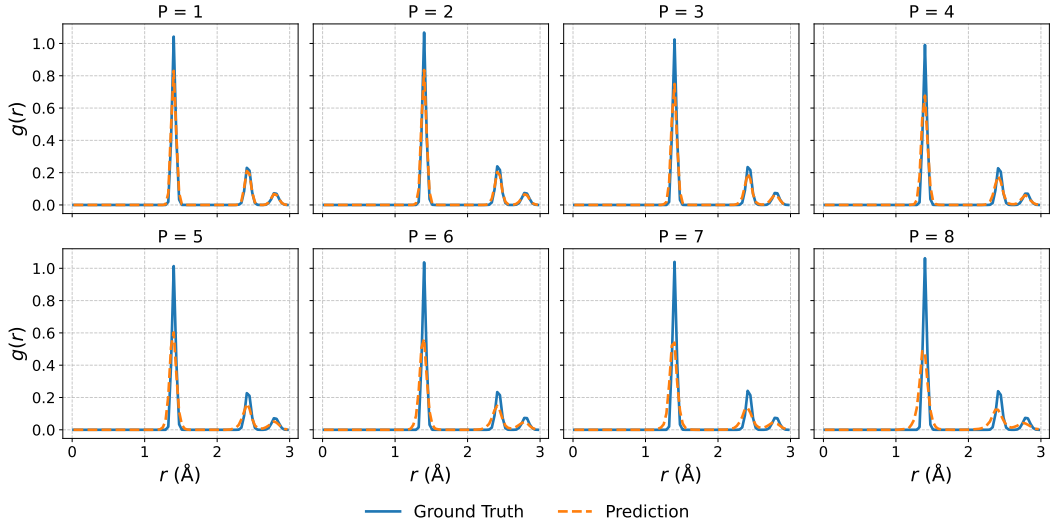


Figure 12: **Radial distribution function (RDF) for Benzene.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

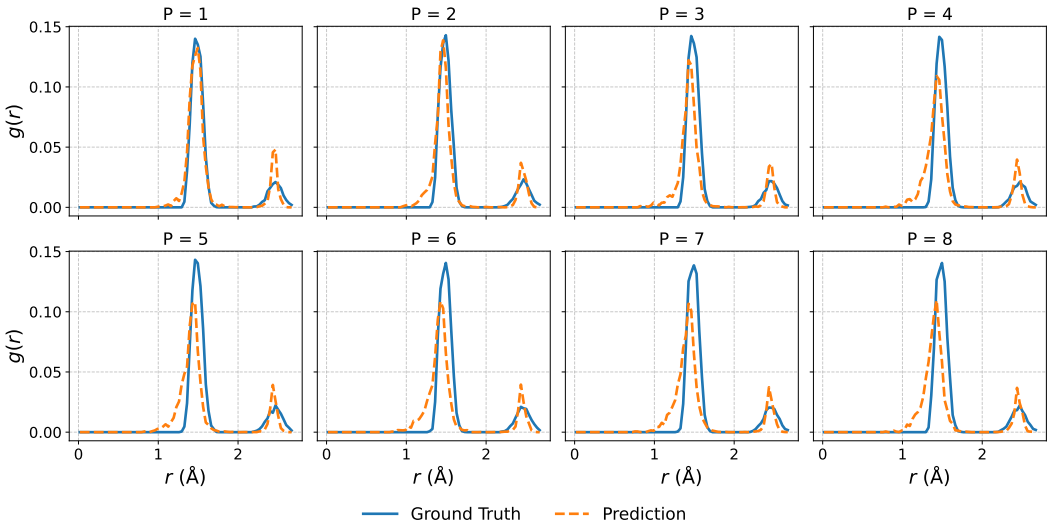


Figure 13: **Radial distribution function (RDF) for Ethanol.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

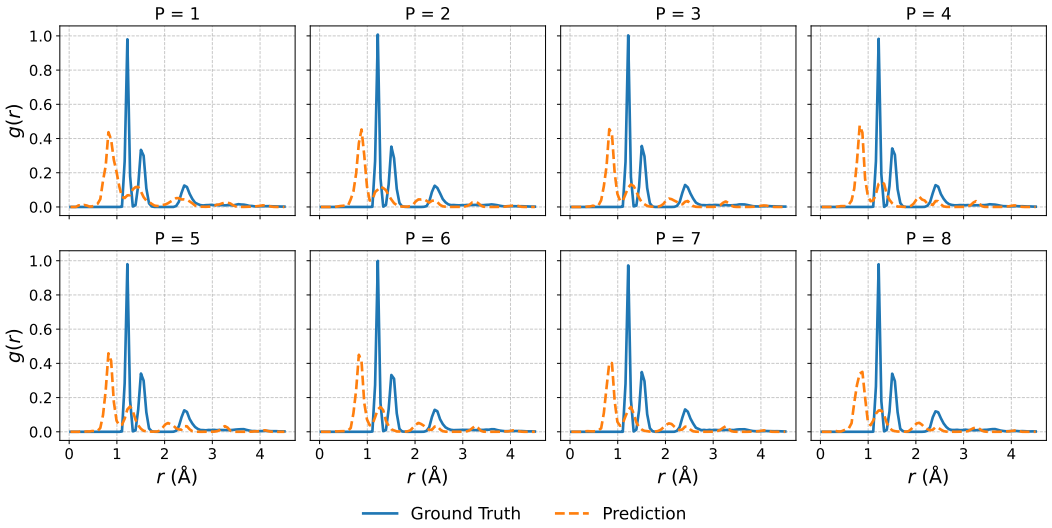


Figure 14: **Radial distribution function (RDF) for Malonaldehyde.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

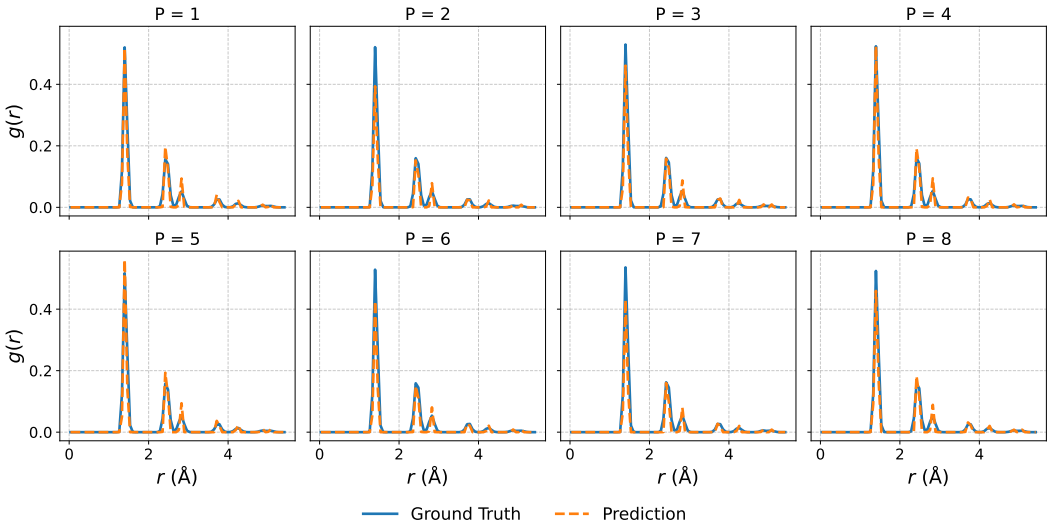


Figure 15: **Radial distribution function (RDF) for Naphthalene.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

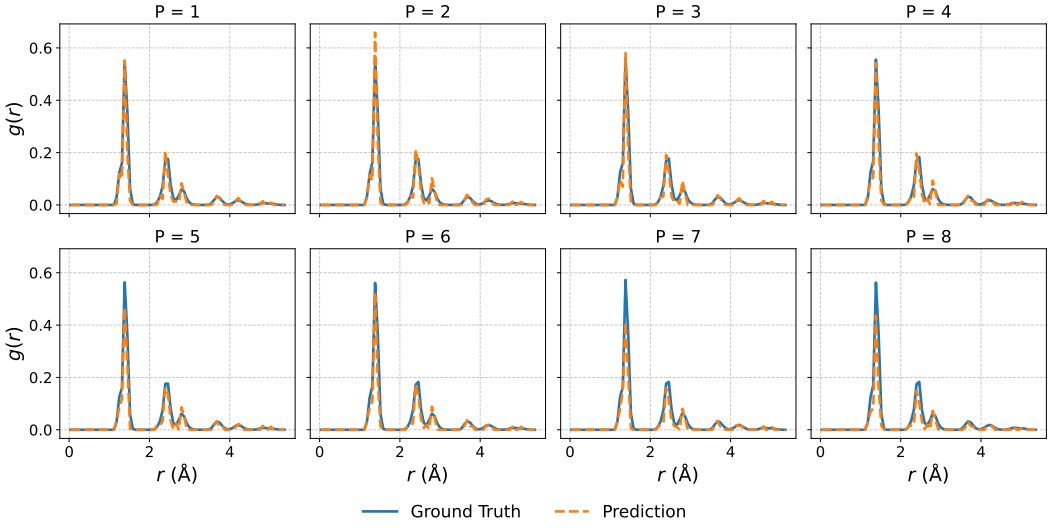


Figure 16: **Radial distribution function (RDF) for Salicylic.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

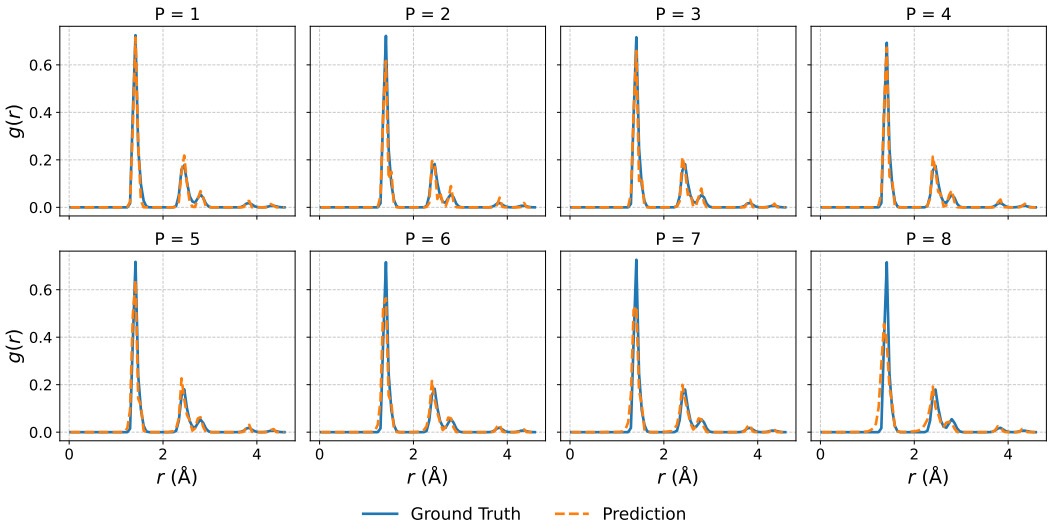


Figure 17: **Radial distribution function (RDF) for Toluene.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

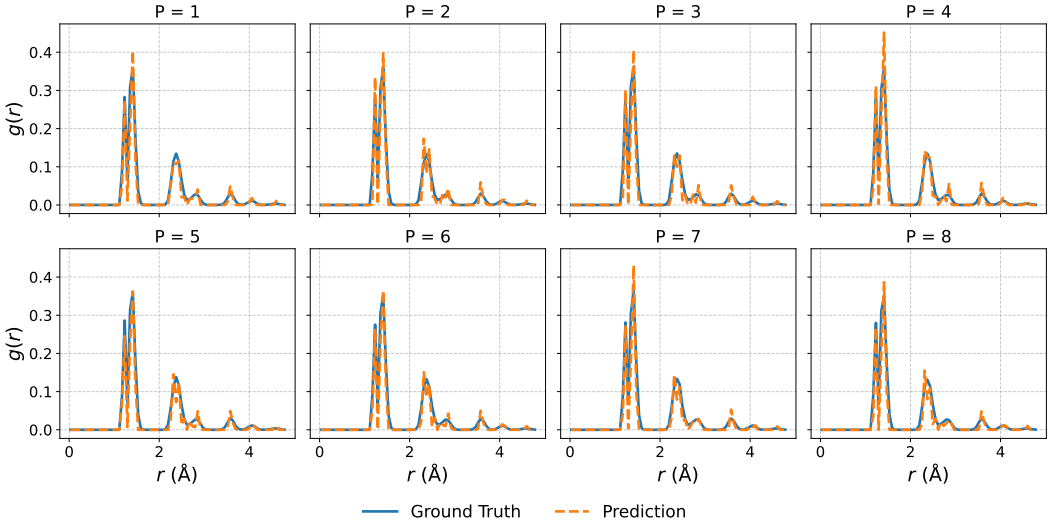


Figure 18: **Radial distribution function (RDF) for Uracil.** We compare the ground-truth (solid) and model-predicted (dashed) RDFs $g(r)$ across $P = 1 \dots 8$ timesteps.

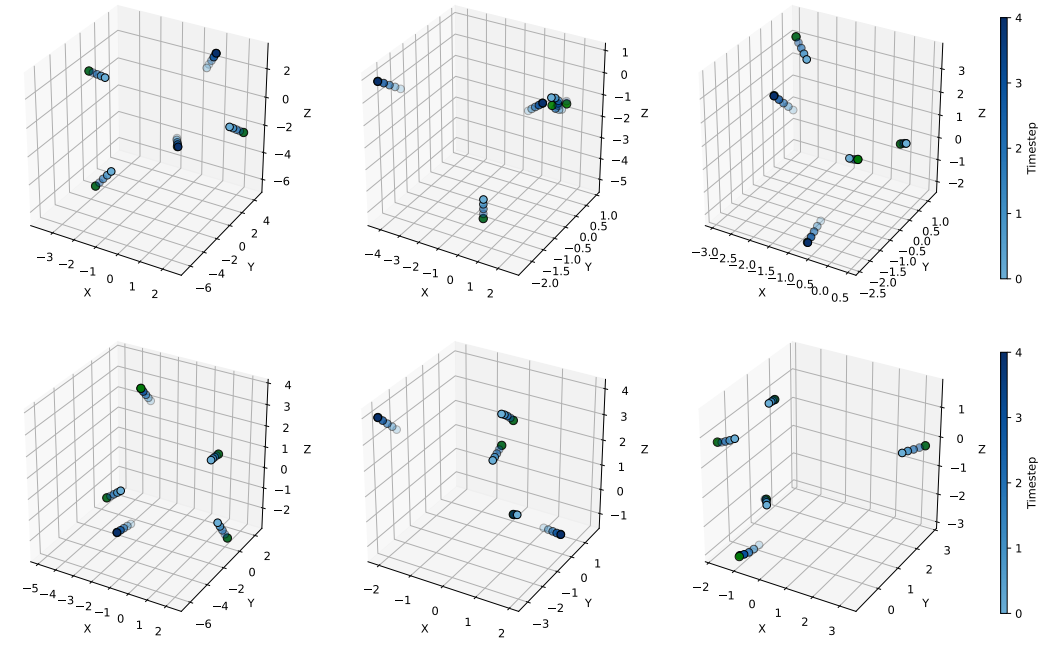


Figure 19: Visualization of trajectories generated by GraMO with uniform discretization on the N-Body dataset. Predicted trajectories are shown with timestep progression indicated by a **Blue** color gradient, while the ground truth final snapshot is marked in **Green**.

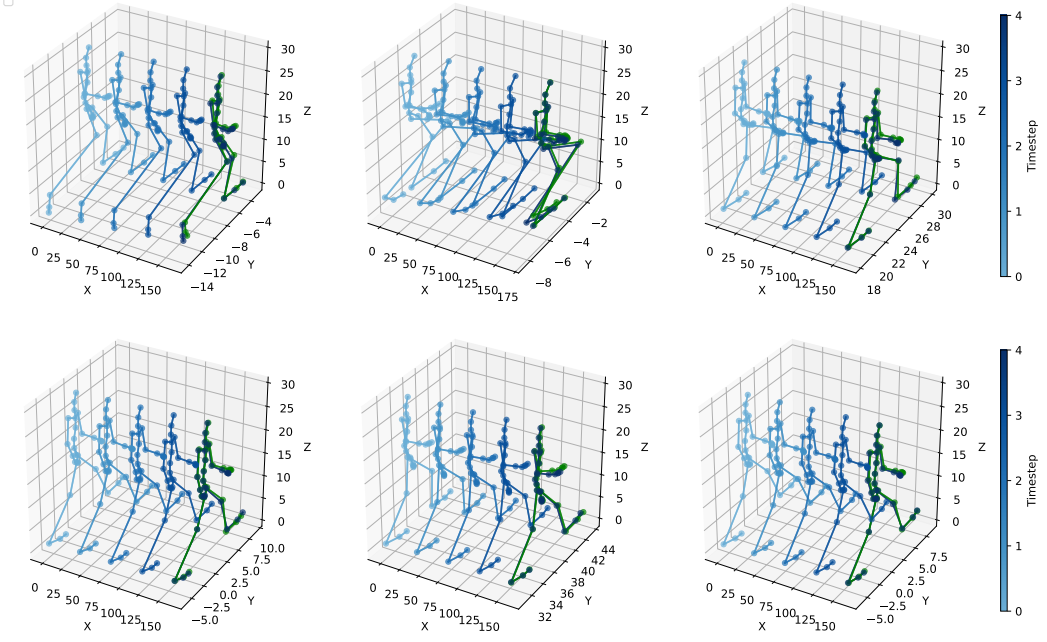


Figure 20: Visualization of trajectories generated by GraMO with uniform discretization on Mocap (Walk) dataset. Predicted trajectories are shown with timestep progression indicated by a **Blue** color gradient, while the ground truth final snapshot is marked in **Green**.

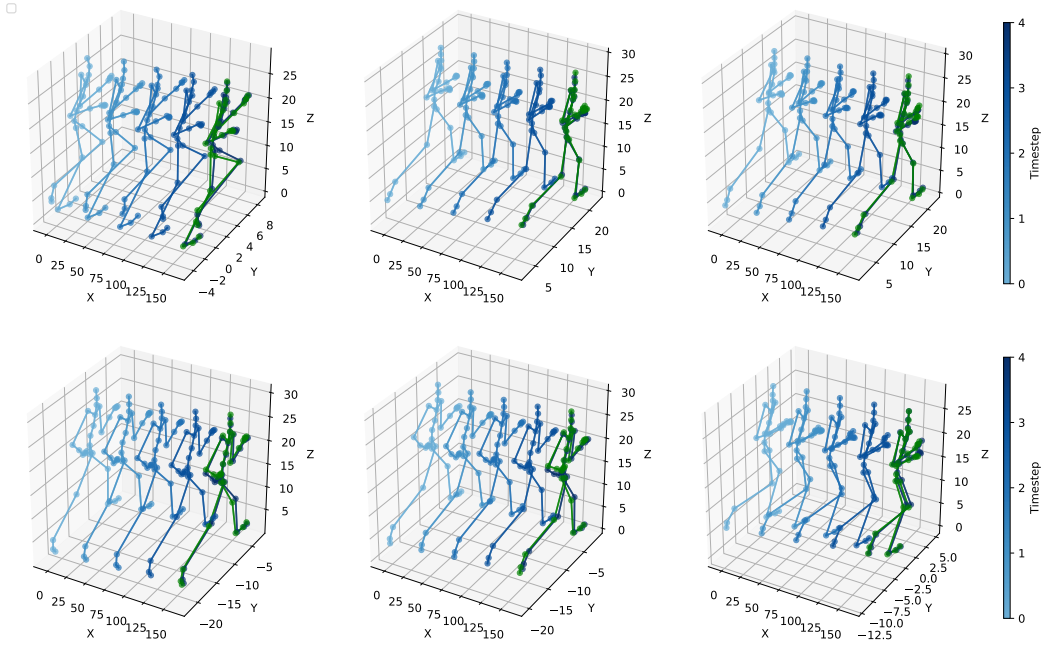


Figure 21: Visualization of trajectories generated by GraMO with uniform discretization on Mocap (Run) dataset. Predicted trajectories are shown with timestep progression indicated by a **Blue** color gradient, while the ground truth final snapshot is marked in **Green**.

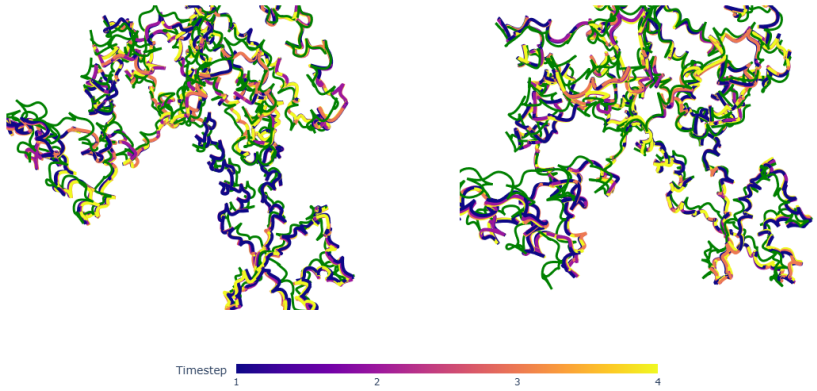


Figure 22: Visualization of trajectories generated by GraMO with uniform discretization on AdK equilibrium trajectory dataset. Predicted trajectories are shown with timestep progression indicated by the colorbar shown above, while the ground truth final snapshot is marked in **Green**.