WHAT MOVES TOGETHER: COMPOSITIONAL MOTION MODES FOR PHYSICAL DYNAMICS PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Physical dynamics prediction aims to forecast the trajectories of interacting components across diverse physical systems, from atoms in molecules to joints in human motion. The dynamics of such systems combine distinct but interacting kinds of motion: system-wide translation and rotation, system-wide deformation, and neighborhood-level motion. However, most existing approaches predict the resulting trajectory without explicitly separating these kinds of motion, leaving their individual contributions and interactions entangled. We introduce *Compositional Mode Dynamics (CMD)*, a framework that represents physical motion with a compositional mode dictionary and models how its coefficients evolve and interact. The dictionary collects the spatial patterns of modes, grouped into three complementary families: *rigid*, *collective*, and *local*. Each mode follows its own linear dynamics, while structured coupling allows interactions within and across mode families without changing the underlying modal dynamics. This formulation yields an explicit and interpretable decomposition of predicted trajectories into physically meaningful motion components. Experiments on molecular dynamics and human motion datasets show that CMD achieves competitive prediction performance, while ablations and analyses validate the roles of the mode families and their coupled coefficient dynamics.

1 INTRODUCTION

Physical dynamics prediction seeks to forecast how a system of interacting components, such as atoms in a molecule or joints in a human body, evolves in three-dimensional space from an observed state. Accurate prediction of such dynamics is important across a broad range of scientific applications, including drug discovery (Durrant & McCammon, 2011; Hollingsworth & Dror, 2018), protein folding (Shaw et al., 2010; Lindorff-Larsen et al., 2011), materials design (Behler & Parrinello, 2007; Deng et al., 2023), robotics (Sanchez-Gonzalez et al., 2018; Li et al., 2018), and human motion forecasting (Martinez et al., 2017; Mao et al., 2019). Learning-based approaches focused on modeling spatial interactions among components with equivariant graph networks (Satorras et al., 2021; Thomas et al., 2018; Fuchs et al., 2020; Köhler et al., 2019), whereas recent work increasingly models the temporal evolution of entire trajectories (Huang et al., 2020; Diez et al., 2025; Wu et al., 2023; Xu et al., 2024; Sun et al., 2024; Kim et al., 2026).

Despite these advances, long-horizon prediction remains challenging because a physical trajectory is generated by several kinds of motion at once: the system may move as a whole, deform across its extent, and move within local neighborhoods, as when a molecule rotates while its bonds vibrate or a human body translates while individual joints swing. These kinds of motion interact as the trajectory unfolds. Yet most learned dynamics models predict the trajectory as a single entangled state, without explicitly representing the contribution of each kind of motion or how these motions interact.

Recent approaches partially address this by decomposing dynamics in the frequency domain, along time (Xu et al., 2024), across the graph (Sun et al., 2024), or jointly in space and time (Kim et al., 2026). Such decompositions improve long-horizon prediction, but frequency and physical motion are different things: each kind of motion spans multiple frequencies, and each frequency mixes several of them. Frequency decomposition therefore does not separate the physically distinct kinds of motion in a trajectory, nor does it offer a direct way to model their interactions.

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

This motivates a different perspective: rather than decomposing a trajectory by how fast it changes, can we decompose it by *what moves together*, and then model how these components evolve and interact? Classical modal decomposition (Brooks & Karplus, 1983; Amadei et al., 1993; Schmid, 2010) is a natural starting point, representing motion as a combination of spatial modes and time-varying coefficients, where each mode is a displacement pattern in physical space rather than a latent feature. Conventional modal methods, however, treat the modes as a single undifferentiated set, neither organizing them by the kind of motion they represent nor modeling interactions among such groups.

We introduce *Compositional Mode Dynamics (CMD)*, which represents physical dynamics as a composition of complementary mode families and models their coupled evolution in coefficient space. CMD organizes modes by two criteria, whether the motion preserves the shape of the system and the spatial scope over which it acts, yielding three families: *rigid* for system-wide translation and rotation, *collective* for system-wide deformation, and *local* for neighborhood-level motion. Their spatial patterns form a compositional mode dictionary, so that every predicted trajectory decomposes into the contribution of each family. Each mode is assigned its own linear dynamics in coefficient space, parameterized by its oscillatory or non-oscillatory temporal behavior. Then, an orthogonal mixing of these dynamics lets the coefficients interact within and across families while preserving the modal spectrum. CMD thus comprises three parts with distinct roles: the mode dictionary specifies *what moves together*, the initial coefficients specify *how strongly each mode is expressed initially*, and the evolution operator specifies *how the modes evolve and interact over time*.

On molecular dynamics and human motion prediction, CMD achieves competitive accuracy while providing a per-family decomposition of the predicted motion, and experiments confirm that each element of the design contributes as intended. Our main contributions are summarized as follows:

- We introduce *Compositional Mode Dynamics (CMD)*, which represents physical trajectories through complementary rigid, collective, and local mode families defined directly in physical space.
- We develop *structured coupled coefficient dynamics* that preserve individual modal behavior while allowing interactions within and across mode families, yielding an explicit decomposition of predicted trajectories.
- We demonstrate CMD on molecular dynamics and human motion prediction, showing competitive accuracy together with an interpretable per-family decomposition, and validate the coupling and the auxiliary rigid branch through ablations and the learned representation through coefficient-space analyses.

2 Related works

Physical dynamics prediction. Learned physical dynamics models typically build on geometric architectures that encode interactions among system components. Graph networks model pairwise and neighborhood interactions through message passing (Battaglia et al., 2016; Kipf et al., 2018; Sanchez-Gonzalez et al., 2020), while equivariant architectures such as TFN, SE(3)-Transformer, and EGNN further incorporate geometric symmetries (Thomas et al., 2018; Fuchs et al., 2020; Satorras et al., 2021). Recent approaches extend these architectures to long-horizon prediction: EGNO predicts trajectories at multiple future times using an equivariant neural operator (Xu et al., 2024), GF-NODE evolves graph-frequency components in continuous time (Sun et al., 2024), and SGDD distills spatio-temporal characteristics of future trajectories into the initial-state representation (Kim et al., 2026). These methods still model the trajectory as a whole, without decomposing it into physically distinct kinds of motion or modeling their interactions.

Structured decomposition of motion. Modal decomposition represents motion as a combination of spatial modes with time-varying coefficients, with modes derived from a harmonic approximation of the energy (Brooks & Karplus, 1983; Tirion, 1996; Atilgan et al., 2001), from the covariance of simulated trajectories (Amadei et al., 1993), or jointly with linear temporal dynamics (Schmid, 2010). Because these modes are defined directly in physical space, their spatial patterns are interpretable. However, the methods serve mainly as analysis tools and treat the modes as a single set rather than organizing them by kind of motion. Learned trajectory predictors instead introduce structure through temporal-frequency decomposition (Mao et al., 2019) or spatial hierarchies of joints or

nodes (Han et al., 2022; Dang et al., 2021), but their components are not physical-space modes and their interactions are not modeled. CMD keeps the physical-space modes of classical decomposition while organizing them into rigid, collective, and local families and coupling their dynamics.

Koopman-based linear dynamics. Koopman theory describes nonlinear dynamics through linear evolution in an infinite-dimensional space of observable functions (Koopman, 1931; Mezić, 2005), which practical methods approximate on a finite set: dynamic mode decomposition uses a fixed dictionary (Schmid, 2010; Williams et al., 2015), whereas learned Koopman models jointly train an encoder, a linear operator $\mathbf{K}$, and a decoder (Takeishi et al., 2017; Lusch et al., 2018; Azencot et al., 2020), sometimes parameterizing the dynamics through their spectral properties (Lusch et al., 2018). CMD also evolves its state linearly, but its state consists of coefficients of explicit physical-space modes rather than unconstrained latent embeddings, and $\mathbf{K}$ is an orthogonal mixing of mode-wise dynamics that lets modes interact within and across families while preserving their spectrum.

3 METHOD

Problem setup. We represent a physical system as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes. A trajectory window consists of $T + 1$ time steps indexed by $t \in \{0, \dots, T\}$, where $\mathbf{X}_t \in \mathbb{R}^{3\times N}$ denotes the node coordinates at time step t and $\mathbf{V}_0 \in \mathbb{R}^{3\times N}$ denotes the initial velocity computed from consecutive coordinates. Given the initial state $(\mathbf{X}_0, \mathbf{V}_0, \mathcal{G})$, our goal is to predict the future coordinates $\mathbf{X}_{1:T}$. Notation is summarized in Appendix A.1.

Canonicalization and Displacement. Trajectory windows may differ in their global position and orientation even when they exhibit similar underlying motion. We refer to the original global coordinate system as the *lab frame*. To represent motion in a common coordinate system, we canonicalize each trajectory window into a *body frame* using its initial center of mass $\mathbf{c}_0$ and a Kabsch rotation $\mathbf{R}_c$. Specifically, $\mathbf{R}_c^\top$ aligns the centered coordinates at $t = 0$ with a fixed reference structure $\mathbf{X}^{\text{ref}}$, whose orientation defines the canonical pose. The same transformation is applied to all subsequent frames in the window. The resulting body-frame coordinates and displacement are

$$\mathbf{b}_t = \mathbf{R}_c^\top(\mathbf{X}_t - \mathbf{c}_0), \qquad \Delta\mathbf{x}_t = \mathbf{b}_t - \mathbf{b}_0. \tag{1}$$

We use the body-frame displacement $\Delta\mathbf{x}_t$ at each future time step as the prediction target.

3.1 OVERVIEW

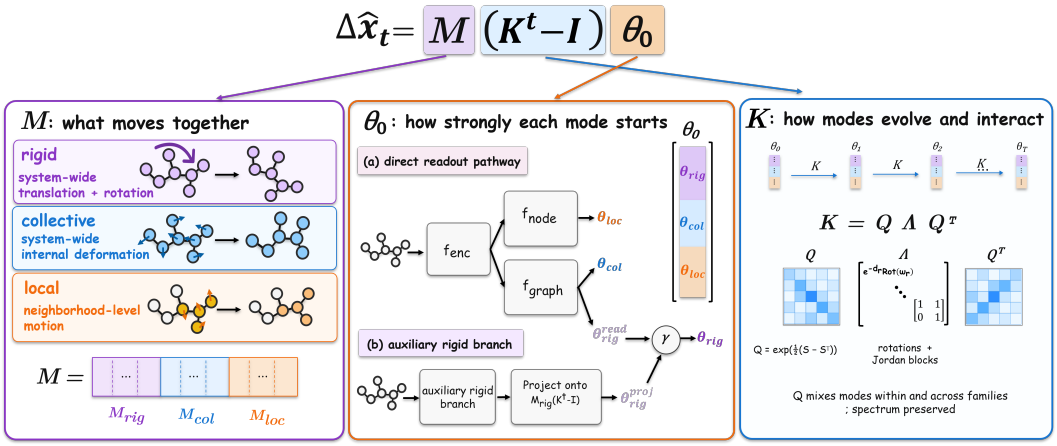
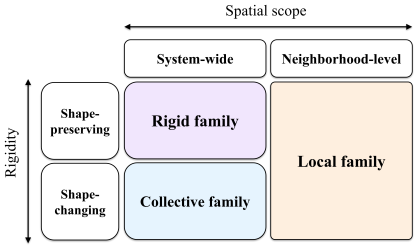


Figure 1: **Overview of CMD.** $\mathbf{M}$ collects the spatial patterns of the rigid, collective, and local families; $\boldsymbol{\theta}_0$ is estimated from the initial state; $\mathbf{K} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$ evolves the coefficients, with $\mathbf{Q}$ coupling modes within and across families.

We propose Compositional Mode Dynamics (CMD), which represents physical motion as a composition of complementary *rigid*, *collective*, and *local* mode families and evolves the coefficients of their modes through structured linear dynamics. Within CMD, body-frame displacement is represented through a compositional mode dictionary $\mathbf{M}$. The dictionary consists of three complementary

Table 1: **Mode families of CMD.** (Left) The three families are distinguished by rigidity and spatial scope; the local family covers neighborhood-level motion regardless of rigidity. (Right) Specification of each family's motion types and temporal forms.



Mode family	Motion type	Temporal form	Pattern	ω_r	d_r
Rigid	rotation	oscillatory	analytic	learned	0
Rigid	translation	non-oscillatory	analytic	–	–
Collective	deformation	oscillatory	learned	learned	learned
Local	deformation	oscillatory	learned	learned	learned
Local	drift	non-oscillatory	analytic	–	–

mode families, distinguished by rigidity and spatial scope: the rigid family captures system-wide translation and rotation, the collective family captures system-wide internal deformation, and the local family captures neighborhood-level motion. CMD estimates the initial mode coefficients $\boldsymbol{\theta}_0$ from the initial state and evolves them using a linear operator $\mathbf{K}$. Since $\boldsymbol{\theta}_t = \mathbf{K}^t\boldsymbol{\theta}_0$, the predicted body-frame displacement at time step t is

$$\boxed{\mathrm{vec}(\Delta\widehat{\mathbf{x}}_t) = \mathbf{M}\left(\mathbf{K}^t - \mathbf{I}\right)\boldsymbol{\theta}_0} \tag{2}$$

As illustrated in Figure 1, $\mathbf{M}$ specifies *what moves together*, $\boldsymbol{\theta}_0$ specifies *how strongly each mode is expressed initially*, and $\mathbf{K}$ specifies *how the modes evolve and interact over time*.

3.2 Compositional Mode Dictionary $\mathbf{M}$

We organize physical motion along two axes: rigidity (whether the motion preserves the shape of the system) and spatial scope (whether it spans the whole system or a local neighborhood). This yields three complementary mode families: *rigid* for system-wide translation and rotation, *collective* for system-wide internal deformation, and *local* for neighborhood-level motion. We collect their spatial patterns into a compositional mode dictionary,

$$\mathbf{M} = [\mathbf{M}_{\mathrm{rig}} \mid \mathbf{M}_{\mathrm{col}} \mid \mathbf{M}_{\mathrm{loc}}] \in \mathbb{R}^{3N\times P}, \tag{3}$$

where P is the dimension of the mode coefficient vector $\boldsymbol{\theta}_t$.

Each mode is associated with one or more spatial patterns and corresponding coefficients, with temporal evolution taking one of two forms: oscillatory or non-oscillatory. An oscillatory mode r uses a pair of spatial patterns $\mathbf{m}_r = [\mathbf{m}_r^c\ \mathbf{m}_r^s]$, with coefficients $(\alpha_{r,t}, \beta_{r,t})$, angular frequency ω_r, and damping rate d_r. A non-oscillatory mode j, such as translation or drift, uses a single spatial pattern $\mathbf{m}_j$ weighted by $p_{j,t}$ and does not require ω_r or d_r.

Let $\mathcal{O}$ and $\mathcal{N}$ denote the oscillatory and non-oscillatory modes. Their composition at time t is

$$\mathbf{M}\boldsymbol{\theta}_t = \sum_{r\in\mathcal{O}} (\alpha_{r,t}\mathbf{m}_r^c + \beta_{r,t}\mathbf{m}_r^s) + \sum_{j\in\mathcal{N}} p_{j,t}\mathbf{m}_j. \tag{4}$$

The vectorized body-frame displacement is obtained from the change in this composition relative to $t = 0$. Depending on the mode family and motion type, the spatial patterns and temporal parameters are either defined analytically or learned from data (see Table 1).

Rigid family. The rigid family captures system-wide translation and rotation, moving the system as a whole without changing its shape. Its spatial patterns are defined analytically from the reference structure $\mathbf{X}^{\mathrm{ref}}$ and remain fixed during training. Translation is represented along the Cartesian directions $\mathbf{e}_a$, $a \in \{x, y, z\}$. For rotation, we use first- and second-order Rodrigues patterns $\mathbf{r}_a$ and $\mathbf{r}_{ab}$, analytically derived from $\mathbf{X}^{\mathrm{ref}}$ and the Cartesian axes $\mathbf{e}_a, \mathbf{e}_b$, which satisfy

$$\mathrm{vec}\big(\mathbf{R}(\phi)\mathbf{X}^{\mathrm{ref}} - \mathbf{X}^{\mathrm{ref}}\big) = \sin\phi\,\mathbf{r}_a + (1-\cos\phi)\,\mathbf{r}_{ab}, \tag{5}$$

so that finite rotations are represented exactly. Since these spatial patterns are shared across trajectory windows while their rotation rates may vary, we instantiate each pattern at multiple learnable angular frequencies. The corresponding coefficients then compose these frequency-specific modes to represent the rigid motion of each window.

Collective family. The collective family captures system-wide internal deformation, which changes the overall shape of the system. We first construct a fixed set of candidate patterns $\{\mathbf{p}_p\}_{p=1}^{n_{\mathrm{pca}}}$. To construct these candidates, each frame is further Kabsch-aligned to $\mathbf{b}_0$, and principal components of the remaining deformation are used to define an empirical deformation subspace. We then learn the two spatial patterns of each collective mode as linear combinations within this subspace,

$$\mathbf{m}_r = \mathbf{\Phi}\,\mathbf{C}_r, \qquad \mathbf{\Phi} = [\mathbf{p}_1 \cdots \mathbf{p}_{n_{\mathrm{pca}}}] \in \mathbb{R}^{3N \times n_{\mathrm{pca}}}, \qquad \mathbf{C}_r \in \mathbb{R}^{n_{\mathrm{pca}} \times 2}, \tag{6}$$

where $\mathbf{C}_r$ contains the learnable combination coefficients of mode r. The candidate subspace is computed from the training data and kept fixed, while the modes within it are learned. This construction restricts collective modes to empirically observed internal-deformation patterns rather than arbitrary displacement fields.

Local family. The local family captures neighborhood-level motion around each node, including both rigid and deformable motion of the neighborhood. Each node has its own set of local modes. For node i, we construct fixed analytic candidates $\{\mathbf{u}_{i,q}\}_{q=1}^{n_i}$ that describe local translation and rotation of the neighborhood, bond stretching, and angle bending. We remove the system-wide rigid-motion component from each candidate so that the resulting local modes capture motion distinct from global rigid motion. The learnable coefficients $\mathbf{D}_i$ combine these candidates into local mode patterns, analogously to the collective modes (Eq. 6). Each node additionally has an analytically defined non-oscillatory drift mode. Details of the mode construction are provided in the Appendix B.1.

3.3 Initial Mode Coefficient Estimation

Given the compositional mode dictionary $\mathbf{M}$, we estimate the initial mode coefficients $\boldsymbol{\theta}_0$ from the initial state using two pathways: a direct readout and an auxiliary rigid branch. The direct readout estimates the collective and local coefficients, $\boldsymbol{\theta}_{\mathrm{col}}$ and $\boldsymbol{\theta}_{\mathrm{loc}}$, together with a direct estimate $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{read}}$ of the rigid coefficients. The auxiliary rigid branch provides a second estimate $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}}$. The two rigid estimates are fused into $\boldsymbol{\theta}_{\mathrm{rig}}$, yielding the initial coefficient vector

$$\boldsymbol{\theta}_0 = [\boldsymbol{\theta}_{\mathrm{rig}}; \boldsymbol{\theta}_{\mathrm{col}}; \boldsymbol{\theta}_{\mathrm{loc}}]\,. \tag{7}$$

Direct readout pathway. We first use a graph encoder f_{enc} to map the graph-structured initial state to node features,

$$\{\mathbf{h}_{i,0}\}_{i=1}^{N} = f_{\mathrm{enc}}(\mathbf{X}_0, \mathbf{V}_0, \mathcal{G}). \tag{8}$$

From these features, a graph-level readout f_{graph} estimates the collective coefficients $\boldsymbol{\theta}_{\mathrm{col}}$ together with a direct estimate of the rigid coefficients, $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{read}}$, while a node-level readout f_{node} estimates the local coefficients $\boldsymbol{\theta}_{\mathrm{loc},i}$. These readouts provide the initial coefficients of the modes in all three families directly from the observed initial state.

Auxiliary rigid branch. Although the rigid patterns are defined analytically, the rigid component of the trajectory is determined by coefficients estimated only once at $t = 0$ and subsequently evolved through the linear dynamics. Global rotation is therefore particularly sensitive to coefficient errors: it affects all nodes simultaneously and, being undamped, any error can persist throughout the prediction window. To improve the estimation of the initial rigid coefficients, we introduce an auxiliary branch that predicts a coarse coordinate trajectory. The branch encodes node-wise latent states $\mathbf{z}_i$, evolves them with its own linear operator $\mathbf{K}_{\mathrm{rig}}$, and decodes coordinates at each time step. Kabsch alignment of the decoded trajectory then extracts the predicted rotation and translation $(\widehat{\mathbf{R}}_t, \widehat{\mathbf{c}}_t)$. The resulting rigid displacement trajectory is projected by ridge regression onto the rigid columns of $\mathbf{M}(\mathbf{K}^t - \mathbf{I})$, stacked over $t = 1, \ldots, T$. This yields $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}}$, the rigid coefficients that best explain the auxiliary trajectory under the shared mode dictionary $\mathbf{M}$ and dynamics $\mathbf{K}$. The auxiliary branch contributes to CMD only through $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}}$; its decoded coordinates are never used as the final prediction.

Rigid coefficient fusion. The two pathways provide independent but complementary estimates of the initial rigid coefficients. The direct estimate $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{read}}$ is inferred from the observed initial state, whereas the projected estimate $\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}}$ is derived from the rigid trajectory predicted by the auxiliary branch. We adaptively fuse these estimates using a per-window gate $\gamma \in (0, 1)$ predicted from the initial-state features:

$$\boldsymbol{\theta}_{\mathrm{rig}} = \gamma\,\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}} + (1 - \gamma)\,\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{read}}. \tag{9}$$

3.4 MODE COEFFICIENT DYNAMICS $\mathbf{K}$

Given the initial coefficients $\boldsymbol{\theta}_0 \in \mathbb{R}^P$, we model their temporal evolution with a linear operator $\mathbf{K} \in \mathbb{R}^{P\times P}$,

$$\boldsymbol{\theta}_{t+1} = \mathbf{K}\,\boldsymbol{\theta}_t. \tag{10}$$

We first construct independent dynamics for individual modes and then introduce structured coupling, allowing the rigid, collective, and local families to interact while preserving their underlying modal dynamics.

Independent dynamics. For each oscillatory mode, the coefficient pair $(\alpha_{r,t}, \beta_{r,t})$ encodes its amplitude and phase,

$$(\alpha_{r,t}, \beta_{r,t}) = a_{r,t}(\cos\phi_{r,t}, \sin\phi_{r,t}). \tag{11}$$

Over one time step, the phase advances by ω_r while the amplitude decays by a factor of e^{-d_r},

$$\begin{bmatrix}\alpha_{r,t+1}\\ \beta_{r,t+1}\end{bmatrix} = e^{-d_r}\mathbf{Rot}(\omega_r)\begin{bmatrix}\alpha_{r,t}\\ \beta_{r,t}\end{bmatrix}. \tag{12}$$

Non-oscillatory translation and drift instead follow simple linear updates. Collecting these mode-wise dynamics yields a block-diagonal operator $\boldsymbol{\Lambda} \in \mathbb{R}^{P\times P}$, which defines the independent evolution of all modes. Thus, in the absence of mode coupling, $\mathbf{K} = \boldsymbol{\Lambda}$. The exact non-oscillatory blocks are provided in the Appendix B.2

Coupled dynamics. To model interactions among mode coefficients, we introduce an orthogonal mixing matrix $\mathbf{Q}$,

$$\mathbf{K} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top, \qquad \mathbf{Q} = \exp\!\left(\tfrac{1}{2}(\mathbf{S} - \mathbf{S}^\top)\right), \tag{13}$$

where $\mathbf{S} \in \mathbb{R}^{P\times P}$ is learnable. Since $\mathbf{Q}$ is orthogonal, $\mathbf{K}$ preserves the spectrum of $\boldsymbol{\Lambda}$ while allowing the corresponding mode coefficients to interact. This construction separates temporal behavior from interaction: $\boldsymbol{\Lambda}$ specifies the intrinsic frequency and damping of individual modes, while $\mathbf{Q}$ determines how their coefficients interact without introducing additional eigenvalues. Initializing $\mathbf{S} = \mathbf{0}$ gives $\mathbf{Q} = \mathbf{I}$, so the model starts from independent mode dynamics and learns coupling only as needed.

We consider three levels of coupling: *no coupling* ($\mathbf{Q} = \mathbf{I}$), where all modes evolve independently; *intra-family coupling*, where modes mix only within each family; and *full coupling*, where modes mix both within and across families (Appendix B.3). Importantly, the coupling acts on the coefficient dynamics rather than on the spatial dictionary. The columns of $\mathbf{M}$ remain fixed to their rigid, collective, and local families; $\mathbf{Q}$ only determines how their coefficients influence one another during evolution. Therefore, even under full coupling, the predicted displacement can still be decomposed according to the spatial support encoded by each block of $\mathbf{M}$.

3.5 TRAINING

During training, we additionally estimate mode coefficients $\tilde{\boldsymbol{\theta}}_t$ from intermediate observed frames to directly supervise both trajectory prediction and coefficient dynamics. The overall training objective combines trajectory supervision, coefficient-dynamics supervision, regularization, and auxiliary supervision.

Trajectory supervision. For a k-step prediction starting from time step t, we define $\mathbf{A}_k = \mathbf{M}(\mathbf{K}^k - \mathbf{I})$, such that $\mathbf{A}_k\tilde{\boldsymbol{\theta}}_t$ predicts the displacement from t to $t+k$. To account for uncertainty in the estimated mode coefficients, we model $\boldsymbol{\theta}_t \sim \mathcal{N}(\tilde{\boldsymbol{\theta}}_t, \boldsymbol{\Sigma}_\theta)$, which induces the predictive distribution

$$\mathbf{y}_{t,k} \sim \mathcal{N}\Big(\mathbf{A}_k\tilde{\boldsymbol{\theta}}_t,\ \sigma_y^2\mathbf{I} + \mathbf{A}_k\boldsymbol{\Sigma}_\theta\mathbf{A}_k^\top\Big), \qquad \mathcal{L}_{\mathrm{nll}} = -\log p(\mathbf{y}_{t,k}), \tag{14}$$

where $\mathbf{y}_{t,k} = \mathrm{vec}(\mathbf{b}_{t+k} - \mathbf{b}_t)$. Here, $\boldsymbol{\Sigma}_\theta$ captures uncertainty in the estimated mode coefficients, while σ_y accounts for residual observation noise; both are learned. We average $\mathcal{L}_{\mathrm{nll}}$ over the sampled (t,k) pairs. In addition, we directly supervise the reconstructed trajectory with the coordinate loss

$$\mathcal{L}_{\mathrm{pos}} = \frac{1}{T}\sum_{t=1}^{T}\|\widehat{\mathbf{X}}_t - \mathbf{X}_t\|_2^2. \tag{15}$$

Coefficient-dynamics supervision. We directly supervise the learned coefficient dynamics using two objectives. First, we enforce multi-step consistency between coefficients evolved from the initial state and those independently estimated from the corresponding observed frames,

$$\mathcal{L}_{\text{cons}} = \sum_k \left\| \mathbf{K}^k \tilde{\boldsymbol{\theta}}_0 - \tilde{\boldsymbol{\theta}}_k \right\|_2^2, \qquad \mathcal{L}_{\text{rec}} = \sum_k \left\| \mathbf{M}(\mathbf{K} - \mathbf{I}) \tilde{\boldsymbol{\theta}}_k - \text{vec}(\mathbf{b}_{k+1} - \mathbf{b}_k) \right\|_2^2. \tag{16}$$

Together, these objectives constrain the learned dynamics in coefficient and displacement spaces.

Regularization and auxiliary supervision. We apply a Gram-matrix regularizer $\mathcal{L}_{\text{gram}}$ to discourage redundancy among the learned collective and local mode patterns. The auxiliary rigid branch is trained with reconstruction, linear-dynamics, and rigid-motion objectives, collectively denoted by $\mathcal{L}_{\text{rig}}$. Additional details are provided in the Appendix B.6. The overall training objective is

$$\mathcal{L} = \lambda_{\text{nll}}\mathcal{L}_{\text{nll}} + \lambda_{\text{pos}}\mathcal{L}_{\text{pos}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{gram}}\mathcal{L}_{\text{gram}} + \lambda_{\text{rig}}\mathcal{L}_{\text{rig}}. \tag{17}$$

4 EXPERIMENTS

We evaluated CMD on molecular dynamics (MD17) and human motion (CMU Motion Capture). On both, CMD attained accuracy comparable to the strongest baselines.

Evaluation. Given the initial state, the model predicts the trajectory over the following T future steps. For a window with ground truth $\mathbf{X}_{1:T}$ and prediction $\widehat{\mathbf{X}}_{1:T}$, we computed the mean squared error over nodes and time steps, $\text{MSE} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N} \|\widehat{\mathbf{x}}_{i,t} - \mathbf{x}_{i,t}\|_2^2$, and reported the average over all test windows.

Baselines. We compared against EGNN (Satorras et al., 2021), EGNO (Xu et al., 2024), GF-NODE (Sun et al., 2024), SGDD-EGNO and SGDD-GF-NODE (Kim et al., 2026), NDCN (Zang & Wang, 2020), ITO (Diez et al., 2025), and LG-ODE (Huang et al., 2020).

Implementation details. Our framework was implemented in PyTorch, and all experiments were run on a single NVIDIA A6000 GPU with CUDA 12.1. The encoder f_{enc} that extracts node features from the initial state was based on GemNet (Gasteiger et al., 2021), as were the encoder and decoder of the auxiliary rigid branch. All models were trained with AdamW at a fixed learning rate of 3×10^{-4} without scheduling, with gradient clipping at 0.5, for up to 5000 epochs; the checkpoint with the best validation error, evaluated every 5 epochs, was used for testing. Unless stated otherwise, we used full coupling. Dataset-specific settings (e.g., batch size, per-parameter-group weight decay, and model configurations) and source code are included in Appendix C.2

4.1 MOLECULAR DYNAMICS

Dataset. MD17 (Seyler & Beckstein, 2017) contains DFT-based molecular dynamics trajectories of eight small organic molecules, whose atoms serve as nodes and whose number differs across molecules. Following Xu et al. (2024), we sample 500/2000/2000 sub-trajectories per molecule for training, validation, and testing. Each sub-trajectory consists of an initial state followed by $T = 8$ uniformly spaced future steps, the last of which lies 3000 frames ahead of the initial state. Per-molecule statistics are given in Appendix C.1.

Results. Table 2 compares CMD with the baselines on MD17. CMD attains the lowest error on four of the eight molecules, improving over the second-best method by 21% on aspirin, 17% on salicylic acid, 15% on uracil, and 4% on naphthalene. In these molecules, internal vibration dominates over rotation, which is the regime the local family is built for: its modes are constructed per node from bond stretching, angle bending, and local rotation, so motion confined to a few atoms is carried by modes that act only there. Conversely, CMD performs worse than the best baselines on benzene and toluene, where rigid rotation dominates. We attribute this to the initial coefficient estimator rather than to the dictionary or the operator: the rigid coefficients are estimated once from $\mathbf{V}_0$, a one-frame difference, whereas each prediction step spans 375 frames, and since rotation is undamped, the resulting error persists over the whole horizon. As a diagnostic, Appendix D.1 feeds the same model an initial velocity computed over the prediction step; with the dictionary and operator unchanged, the toluene error falls to 1.33 and the benzene error to 1.03, respectively. This is not a fair comparison with the baselines, which use the one-frame velocity, but it shows that the gap comes from what the estimator is given, not from what CMD can represent.

Table 2: **MSE** $(\times 10^{-2})$ **on the MD17 dataset (mean $\pm$ standard deviation over 3 runs)**. The best performance is highlighted in **bold**, and the second best is underlined.

	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil
NDCN	31.73±0.40	56.21±0.30	10.74±0.02	46.55±0.28	2.25±0.01	3.58±0.11	13.92±0.02	2.38±0.00
ITO	20.56±0.03	57.25±0.58	8.60±0.27	28.44±0.73	1.82±0.17	2.48±0.34	12.47±0.30	1.33±0.12
LG-ODE	19.36±0.12	53.92±1.32	7.08±0.01	24.41±0.03	1.73±0.02	3.82±0.04	11.18±0.04	2.11±0.02
EGNN	9.24±0.07	57.85±2.70	4.63±0.00	12.81±0.01	0.38±0.01	0.85±0.00	10.41±0.04	0.56±0.02
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	7.37±0.07	22.41±0.31	3.28±0.02	10.67±0.01	0.32±0.01	0.77±0.01	4.58±0.03	0.47±0.01
GFNODE	6.07±0.09	1.51±0.07	2.74±0.01	**9.43**±0.02	0.24±0.02	0.63±0.05	**1.80**±0.03	0.41±0.02
SGDD-EGNO	6.20±0.01	7.79±0.18	2.88±0.01	11.01±0.05	0.33±0.00	0.69±0.00	4.23±0.09	0.50±0.00
SGDD-GFNODE	5.63±0.01	**1.36**±0.01	**2.67**±0.01	10.95±0.04	0.26±0.00	0.60±0.00	2.39±0.03	0.44±0.00
CMD (Ours)	**4.42**±0.18	1.70±0.07	2.71±0.02	10.75±0.03	**0.23**±0.00	**0.50**±0.01	3.40±0.83	**0.35**±0.00

4.2 MOTION CAPTURE

Dataset. The CMU Motion Capture dataset (Carnegie Mellon University, 2003) provides 3D skeleton sequences in which each of the 31 joints is treated as a node. We use the walking and running motions following the protocol of Xu et al. (2024), splitting them into 198/600/600 and 200/240/240 sub-trajectories for training, validation, and testing, respectively. Each sub-trajectory consists of an initial state followed by $T = 5$ uniformly spaced future steps, the last of which lies 30 frames ahead of the initial state.

Results. Table 3 reports results on the motion capture dataset. On Walk, CMD attains the lowest error, improving over the second-best method by 10%. Walking is periodic: each limb swings back and forth, and the limbs swing in a consistent timing relative to one another. One oscillatory mode already produces a back-and-forth swing, while reproducing the timing between limbs requires the modes to evolve jointly; consistent with this, removing the coupling raises the error by 16% (Section 4.3). On Run, CMD is comparable to SGDD-GFNODE but behind EGNO and SGDD-EGNO. Running changes the rotation rate of the body at every footfall, whereas CMD infers the coefficients once and carries them through the horizon with a fixed linear operator, which is less flexible than the nonlinear networks of the baselines. This is the cost of an explicit composition of modes; an operator that can follow such changes, by updating the coefficients within the horizon or by letting $\mathbf{K}$ vary with time, is left to future work.

Table 3: **MSE** $(\times 10^{-2})$ **on the motion capture dataset**.

Model	Walk	Run
EGNN-R	32.0±1.6	277.3±1.8
EGNN-S	14.3±1.2	28.5±1.3
EGNO	3.5±0.5	14.9±0.9
GFNODE	4.5±0.1	23.2±2.2
SGDD-EGNO	3.2±0.0	**13.5**±0.9
SGDD-GFNODE	3.4±0.0	17.7±1.0
CMD (Ours)	**2.89**±0.06	17.73±0.68

4.3 ABLATION STUDIES

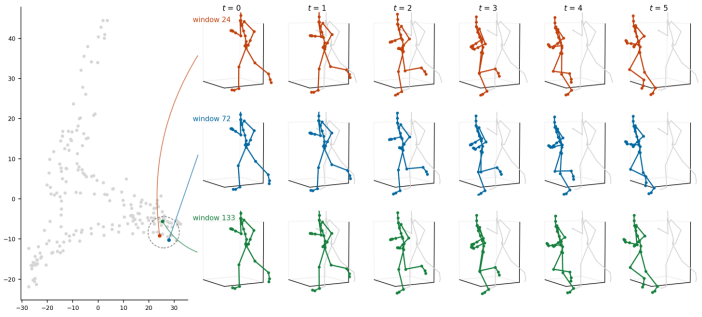
We validate the design of CMD by varying the two components that govern its dynamics: the level of coupling among mode families and the auxiliary rigid branch. Table 4 reports results on six MD17 molecules and Mocap-Walk; the full table is shown in Table 15.

Coupling. The first three rows of Table 4 compare the three coupling levels with the rigid branch enabled. Full coupling attains the lowest error on every dataset in the table, with the largest gain on Mocap-Walk ($3.34 \rightarrow 2.89$). Intra-family coupling alone does not consistently help: it improves benzene and Mocap-Walk over no coupling but is worse on aspirin and ethanol. These results suggest that restricting interactions to within individual families is insufficient to obtain the full benefit of coefficient coupling; allowing cross-family interactions provides the most consistent improvement across the evaluated systems.

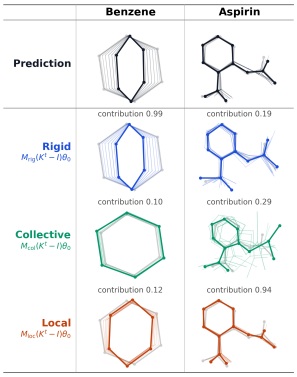
Auxiliary rigid branch. The last two rows compare the full model with and without the branch. Removing the branch and estimating the rigid coefficients from the direct readout alone degrades accuracy on most datasets. The two largest relative increases in error occur on benzene (about $5\times$) and ethanol (25%), the two molecules whose trajectories are dominated by global rotation. This agrees with the motivation of the branch: rotation is undamped, so an error in its initial coefficient persists over the whole horizon and cannot be compensated by the other families.

Table 4: **Ablation on coupling (rows 1–3) and rigid branch (rows 3–4) on MD17 and motion capture (Mocap) datasets**. MSE ($\times 10^{-2}$). The best performance is highlighted in bold.

Coupling	Rigid branch	Aspirin	Benzene	Ethanol	Naphthalene	Salicylic	Uracil	Mocap-Walk
No	✓	4.77	1.96	2.70	0.26	0.52	0.40	3.34
Intra	✓	5.04	1.80	2.74	0.24	0.51	0.40	3.24
Full	✓	**4.61**	**1.76**	**2.69**	**0.23**	**0.50**	**0.35**	**2.89**
Full	–	4.65	8.77	3.35	0.24	**0.50**	**0.35**	3.10



(a) θ_0 coefficient space – the three nearest windows and how they move.



(b) Per-family decomposition of predicted motion.

Figure 2: **Visualization of the learned coefficient space and motion decomposition**.

4.4 Analysis & Visualization

Coefficient space. Figure 2a projects the initial coefficients $\boldsymbol{\theta}_0$ of all test windows running motion (Mocap-Run) and marks one window with its three nearest neighbors in this space. Since $\boldsymbol{\theta}_0$ is estimated from the initial state alone, the neighbors are chosen without seeing any future frame. The right panels show the ground-truth trajectories of the three windows, each drawn over its own initial pose in gray: they start from nearly the same pose and follow nearly the same motion over the whole window. This shows that the coefficients carry information about how the system will move.

Per-family decomposition of predicted motion. Since the prediction is a sum over families, each family's displacement, $\mathbf{M}_{\text{rig}}(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0$, $\mathbf{M}_{\text{col}}(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0$, and $\mathbf{M}_{\text{loc}}(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0$, can be rendered separately. Figure 2b shows this for benzene and aspirin; each column should be read by comparing the full prediction at the top with the three family panels below it. For benzene, the predicted motion is a rotation of the ring that preserves its shape, and the rigid panel reproduces it, carrying 0.99 of the total displacement (Appendix D.3). For aspirin, the ring stays in place and only the two side groups move, and the local panel reproduces it, carrying 0.94 of the displacement. The ablation in Section 4.3 is consistent with this: removing the rigid branch degrades benzene about 400% but on aspirin by less than 1%. The individual modes behind this decomposition are visualized in Appendix D.4 (Figures 3$\sim$12), which show the learned collective and local mode patterns with their frequencies.

5 Conclusion

We introduced Compositional Mode Dynamics (CMD), which represents physical motion as a composition of rigid, collective, and local mode families defined in physical space and evolves their coefficients with structured linear dynamics that let modes interact within and across families. Because the prediction is a sum over families, every predicted trajectory decomposes into physically distinct components. On molecular dynamics and human motion, CMD attains accuracy comparable to strong baselines, and ablations and analyses show that the mode families and their coefficient dynamics behave as intended, supporting the design of the model.

One limitation is that the initial coefficients are estimated only once, from the initial state, so any estimation error propagates through the fixed linear operator and persists over the entire horizon. Future work could address this limitation through a more expressive estimator of $\boldsymbol{\theta}_0$ and adaptive coefficient dynamics during rollout while retaining the explicit compositional structure.

AI USE STATEMENT

In this work, we used generative AI tools to help develop the conceptual framework for organizing motion modes into families, to provide feedback on the design of experiments, to assist in implementing the method, to translate between Korean and English during manuscript preparation, and to assist in interpreting experimental results. We have not used generative AI tools to formulate mathematical claims or to propose or refine hypotheses, and generating synthetic data, proving mathematical claims, writing proofs, cleaning or reformatting datasets, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to suggest experimental parameters, to draft parts of the paper, to summarize and identify relevant literature, for brainstorming, to search for information, to edit the paper for readability, and to suggest the structure of the paper. We have reviewed all AI-assisted work: AI-assisted code was verified and tested by the authors, and all reported results were produced from the authors' own runs; the mathematical formulation and all equations were written by the authors; AI-suggested references were checked against the original publications; and all AI-drafted or AI-edited text was reviewed by the authors for accuracy and consistency with the method and results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Anonymized source code, configuration files, and dataset-specific hyperparameters are provided in Appendix C.2. The mode dictionary, coefficient dynamics, and training objectives are detailed in Appendices B.1, B.2, and B.6, and dataset processing and statistics are detailed in Appendix C.1. The evaluation protocol and baselines are stated in Section 4.

REFERENCES

Andrea Amadei, Antonius BM Linssen, and Herman JC Berendsen. Essential dynamics of proteins. *Proteins: Structure, Function, and Bioinformatics*, 17(4):412–425, 1993.

Ali Rana Atilgan, Stewart R Durell, Robert L Jernigan, Melik C Demirel, Ozlem Keskin, and Ivet Bahar. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophysical journal*, 80(1):505–515, 2001.

Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *ICML*, 2020.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 2016.

Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

Bernard Brooks and Martin Karplus. Harmonic dynamics of proteins: normal modes and fluctuations in bovine pancreatic trypsin inhibitor. *Proceedings of the National Academy of Sciences*, 80(21): 6571–6575, 1983.

Carnegie Mellon University. Cmu motion capture database. `http://mocap.cs.cmu.edu`, 2003. Accessed: 7, 13.

Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *ICCV*, 2021.

Bowen Deng, Peichen Zhong, KyuJung Jun, Janosh Riebesell, Kevin Han, Christopher J Bartel, and Gerbrand Ceder. Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 5(9):1031–1041, 2023.

Juan Viguera Diez, Mathias Schreiner, Ola Engkvist, and Simon Olsson. Boltzmann priors for implicit transfer operators. *ICLR*, 2025.

Jacob D Durrant and J Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC biology*, 9(1):71, 2011.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. In *NeurIPS*, 2020.

Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in neural information processing systems*, 2021.

Jiaqi Han, Wenbing Huang, Tingyang Xu, and Yu Rong. Equivariant graph hierarchy-based neural networks. *Advances in Neural Information Processing Systems*, 2022.

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

Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *NeurIPS*, 2020.

Youjin Kim, Dagyeong Na, JaeYong Lee, and Junseok Kwon. Spectral-guided physical dynamics distillation. In *ICLR*, 2026.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.

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

Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *ICCV*, 2019.

Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017.

Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *ICML*, 2020.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *ICML*, 2021.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Sean L. Seyler and Oliver Beckstein. Molecular dynamics trajectory for benchmarking mdanalysis, 2017. URL `https://figshare.com/articles/Molecular_dynamics_trajectory_for_benchmarking_MDAnalysis/5108170`.

David E Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, Michael P Eastwood, Joseph A Bank, John M Jumper, John K Salmon, Yibing Shan, et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

Fang Sun, Zijie Huang, Haixin Wang, Huacong Tang, Xiao Luo, Wei Wang, and Yizhou Sun. Graph fourier neural odes: Modeling spatial-temporal multi-scales in molecular dynamics. *arXiv preprint arXiv:2411.01600*, 2024.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in neural information processing systems*, 2017.

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.

Monique M Tirion. Large amplitude elastic motions in proteins from a single-parameter, atomic analysis. *Physical review letters*, 77(9):1905, 1996.

Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

Liming Wu, Zhichao Hou, Jirui Yuan, Yu Rong, and Wenbing Huang. Equivariant spatio-temporal attentive graph networks to simulate physical dynamics. In *NeurIPS*, 2023.

Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. In *ICML*, 2024.

Chengxi Zang and Fei Wang. Neural dynamics on complex networks. In *ACM SIGKDD*, 2020.

A PRELIMINARIES

A.1 NOTATIONS

Table 5: **Data and coordinate frames**.

Notation	Dim.	Description
N	–	Number of nodes (atoms or joints).
i	–	Node index, $i \in \{1, \dots, N\}$.
T	–	Prediction horizon; $t \in \{0, \dots, T\}$ with $t = 0$ observed.
Δt	–	Interval between consecutive sampled time steps.
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	–	Interaction graph over the nodes.
$\mathbf{X}_t$	$3 \times N$	Lab-frame coordinates at step t.
$\mathbf{x}_{i,t}$	3	Coordinate of node i at step t.
$\mathbf{V}_t$	$3 \times N$	Velocity, $(\mathbf{X}_t - \mathbf{X}_{t-1})/\Delta t$; only $\mathbf{V}_0$ is used as input.
$\mathbf{X}^{\mathrm{ref}}$	$3 \times N$	Fixed COM-centered reference structure; its orientation defines the canonical pose.
$\mathbf{c}_0$	3	Center of mass at $t = 0$.
$\mathrm{Kabsch}(\cdot, \cdot)$	–	SVD-based rigid alignment returning the optimal rotation between two configurations.
$\mathbf{R}_c$	3×3	Canonical rotation, $\mathrm{Kabsch}(\mathbf{X}_0 - \mathbf{c}_0, \mathbf{X}^{\mathrm{ref}})$.
$\mathbf{b}_t$	$3 \times N$	Body-frame coordinates, $\mathbf{R}_c^\top (\mathbf{X}_t - \mathbf{c}_0)$; the same transform is applied to every step of a window.
$\mathbf{b}_0$	$3 \times N$	Initial body-frame configuration; COM-centered by construction.
$\Delta \mathbf{x}_t$	$3 \times N$	Prediction target, $\mathbf{b}_t - \mathbf{b}_0$.
$\mathbf{R}_t^\star,\ \mathbf{c}_t^\star$	$3 \times 3,\ 3$	Ground-truth rigid rotation and COM displacement from step 0 to t in the body frame; $\mathbf{R}_0^\star = \mathbf{I}$, $\mathbf{c}_0^\star = \mathbf{0}$.
$\widehat{\mathbf{X}}_t$	$3 \times N$	Predicted lab-frame coordinates, $\mathbf{R}_c(\mathbf{b}_0 + \Delta \widehat{\mathbf{x}}_t) + \mathbf{c}_0$.
d_h	–	Node-feature dimension of the encoder.
$\mathbf{h}_{i,0}$	d_h	Feature of node i from the encoder at $t = 0$.

Table 6: **Mode dictionary**.

Notation	Dim.	Description
$\mathbf{M} = [\mathbf{M}_{\mathrm{rig}} \mid \mathbf{M}_{\mathrm{col}} \mid \mathbf{M}_{\mathrm{loc}}]$	$3N \times P$	Compositional mode dictionary; each column is a displacement pattern over all nodes.
P	–	Number of columns, $P = P_{\mathrm{rig}} + P_{\mathrm{col}} + NP_{\mathrm{loc}}$.
$P_{\mathrm{rig}}, P_{\mathrm{col}}, P_{\mathrm{loc}}$	–	Columns of the rigid block, of the collective block, and of each node's local block.
R	–	Number of oscillatory modes, $R = R_{\mathrm{rig}} + R_{\mathrm{col}} + NR_{\mathrm{loc}}$; each occupies two columns.
$r,\ j$	–	Index of an oscillatory and of a non-oscillatory mode.
$\mathcal{O},\ \mathcal{N}$	–	Index sets of oscillatory and non-oscillatory modes.
$\mathbf{m}_r = [\mathbf{m}_r^c\ \mathbf{m}_r^s]$	$3N \times 2$	Pattern pair of oscillatory mode r, contributing $\alpha_{r,t}\mathbf{m}_r^c + \beta_{r,t}\mathbf{m}_r^s$.
$\mathbf{m}_j$	$3N$	Single pattern of non-oscillatory mode j, weighted by $p_{j,t}$.
ω_r	–	Angular frequency of mode r; learned for all oscillatory modes.
d_r	–	Damping rate; fixed to 0 for rigid rotation, learned otherwise.

Notation	Dim.	Description
$\mathbf{e}_a$	3	Cartesian axis, $a \in \{x, y, z\}$; translation moves every node by $\mathbf{e}_a$.
$\mathbf{r}_a,\ \mathbf{r}_{ab}$	$3N$	First- and second-order Rodrigues patterns, $\mathbf{e}_a \times \mathbf{X}^{\mathrm{ref}}$ and $\mathbf{e}_a \times (\mathbf{e}_b \times \mathbf{X}^{\mathrm{ref}})$, so finite rotations are exact.
n_{grp}	–	Number of rotation-rate groups; the nine Rodrigues patterns are instantiated once per group, and ω is shared within a group.
$\mathbf{\Phi} = [\mathbf{p}_1 \cdots \mathbf{p}_{n_{\mathrm{pca}}}]$	$3N \times n_{\mathrm{pca}}$	Fixed collective candidates: principal components of the Kabsch-aligned training deformations, with the six rigid directions removed.
n_{pca}	–	Size of the collective candidate subspace, $\lceil (3N-6)/3 \rceil$.
$\mathbf{C}_r$	$n_{\mathrm{pca}} \times 2$	Learned coefficients forming collective mode r, $\mathbf{m}_r = \mathbf{\Phi}\mathbf{C}_r$.
h	–	Hop radius of a local neighborhood ($h = 2$ for molecules, 1 for motion).
$\mathbf{u}_{i,q}$	$3N$	Fixed local candidate q of node i (local translation and rotation, bond stretching, angle bending) supported on its h-hop neighborhood.
n_i	–	Number of local candidates of node i, capped at $n_{\max}$.
$\mathbf{Q}_{\mathrm{rig}}$	$3N \times 6$	Orthonormal basis of the global rigid directions, projected out of the local drift candidates.
$\mathbf{D}_i$	$R_{\mathrm{loc}} \times n_i \times 2$	Learned coefficients forming the local modes of node i.
$R_{\mathrm{col}},\ R_{\mathrm{loc}}$	–	Collective modes per system, $R_{\mathrm{col}} = 2n_{\mathrm{pca}}$, and local modes per node, $R_{\mathrm{loc}} = 16$.

Table 7: **Mode coefficients and the evolution operator**.

Notation	Dim.	Description
$\boldsymbol{\theta}_t$	P	Mode coefficients at step t; $\boldsymbol{\theta}_t = \mathbf{K}^t \boldsymbol{\theta}_0$.
$[\boldsymbol{\theta}_{\mathrm{rig}}; \boldsymbol{\theta}_{\mathrm{col}}; \boldsymbol{\theta}_{\mathrm{loc}}]$	P	Block structure of $\boldsymbol{\theta}_0$, matching the column blocks of $\mathbf{M}$.
$(\alpha_{r,t}, \beta_{r,t})$	2	Coefficient pair of oscillatory mode r, encoding amplitude $a_{r,t}$ and phase $\phi_{r,t}$.
(p, v)	2	Coefficient pair of a non-oscillatory slot, evolving as $p \leftarrow p + v$.
o	–	Order of the translation Jordan block ($o = 2$ for molecules, 3 for motion).
$\mathbf{J}$	$o \times o$	Jordan block acting on a non-oscillatory slot; $\mathbf{J}_{kk} = 1$, $\mathbf{J}_{k,k+1} = 1$.
$\mathbf{Rot}(\omega)$	2×2	Planar rotation by ω.
$\mathbf{\Lambda}$	$P \times P$	Block-diagonal independent dynamics: $e^{-d_r}\mathbf{Rot}(\omega_r)$ for oscillatory modes and $\mathbf{J}$ for non-oscillatory ones.
$\mathbf{S}$	$P \times P$	Learnable matrix generating the mixing, initialized to $\mathbf{0}$.
$\mathbf{Q} = \exp\big(\frac{1}{2}(\mathbf{S} - \mathbf{S}^\top)\big)$	$P \times P$	Orthogonal mixing matrix; $\mathbf{Q} = \mathbf{I}$ recovers independent modes.
$\mathbf{K} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$	$P \times P$	Evolution operator; preserves the spectrum of $\mathbf{\Lambda}$ while letting modes interact.
$\mathbf{W}$	$P \times P$	Binary mask applied to $\mathbf{S}$; all-ones for full coupling, family block-diagonal for intra-family coupling, and $\mathbf{Q} = \mathbf{I}$ for no coupling.
$\mathbf{A}_k = \mathbf{M}(\mathbf{K}^k - \mathbf{I})$	$3N \times P$	Map from coefficients to the k-step displacement.
$\mathbf{A}$	$3NT \times P$	Stack of $\mathbf{A}_1, \ldots, \mathbf{A}_T$ over the window.
$\Delta\widehat{\mathbf{x}}_t$	$3 \times N$	Predicted body-frame displacement, $\mathbf{A}_t\boldsymbol{\theta}_0$ reshaped.

Table 8: **Estimation of the initial coefficients**.

Notation	Dim.	Description
f_{enc}	–	Graph encoder mapping $(\mathbf{X}_0, \mathbf{V}_0, \mathcal{G})$ to node features.
f_{graph}	–	Graph-level readout producing $\boldsymbol{\theta}_{\text{col}}$ and $\boldsymbol{\theta}_{\text{rig}}^{\text{read}}$.
f_{node}	–	Node-level readout producing $\boldsymbol{\theta}_{\text{loc},i}$.
f_γ	–	Head predicting the fusion gate from pooled features at $t = 0$.
d_z	–	Latent dimension of the auxiliary rigid branch.
$\mathbf{z}_i$	d_z	Latent state of node i in the rigid branch.
$\mathbf{K}_{\text{rig}}$	$d_z \times d_z$	Linear operator of the rigid branch, $\mathbf{z}_i(t) = \mathbf{K}_{\text{rig}}^t \mathbf{z}_i$.
$\widehat{\mathbf{X}}_t^{\text{aux}}$	$3 \times N$	Coarse trajectory decoded by the rigid branch; never used as the final prediction.
$(\widehat{\mathbf{R}}_t, \widehat{\mathbf{c}}_t)$	$3 \times 3,\ 3$	Rotation and COM displacement extracted from $\widehat{\mathbf{X}}_t^{\text{aux}}$ by Kabsch alignment.
$\Delta\mathbf{x}_t^{\text{rig}}$	$3 \times N$	Implied rigid field, $\widehat{\mathbf{R}}_t \mathbf{b}_0 - \mathbf{b}_0 + \widehat{\mathbf{c}}_t$.
$\mathbf{A}_{\text{rig}}$	$3NT \times P_{\text{rig}}$	Rigid columns of $\mathbf{A}$.
ϵ	–	Ridge coefficient of the projection, 10^{-3}.
$\boldsymbol{\theta}_{\text{rig}}^{\text{proj}}$	P_{rig}	Ridge projection of $\Delta\mathbf{x}^{\text{rig}}$ onto $\mathbf{A}_{\text{rig}}$.
$\boldsymbol{\theta}_{\text{rig}}^{\text{read}}$	P_{rig}	Direct readout estimate of the rigid coefficients.
γ	1	Per-window gate, $\boldsymbol{\theta}_{\text{rig}} = \gamma\boldsymbol{\theta}_{\text{rig}}^{\text{proj}} + (1-\gamma)\boldsymbol{\theta}_{\text{rig}}^{\text{read}}$.

Table 9: **Training objectives**.

Notation	Dim.	Description
$\tilde{\boldsymbol{\theta}}_t$	P	Coefficients read out from an observed frame at step t; used only during training.
(t, k)	–	Start step and horizon of a supervised prediction, sampled per window.
$\mathbf{y}_{t,k}$	$3N$	Observed displacement $\text{vec}(\mathbf{b}_{t+k} - \mathbf{b}_t)$.
$\boldsymbol{\Sigma}_\theta$	$P \times P$	Learned diagonal uncertainty of the estimated coefficients.
σ_y	–	Learned observation noise.
$\mathcal{L}_{\text{nll}}$	–	Gaussian negative log-likelihood of $\mathbf{y}_{t,k}$ under $\mathcal{N}(\mathbf{A}_k\tilde{\boldsymbol{\theta}}_t,\ \sigma_y^2\mathbf{I} + \mathbf{A}_k\boldsymbol{\Sigma}_\theta\mathbf{A}_k^\top)$.
$\mathcal{L}_{\text{pos}}$	–	Coordinate loss on the reconstructed trajectory.
$\mathcal{L}_{\text{cons}}$	–	Multi-step consistency between evolved and independently read-out coefficients.
$\mathcal{L}_{\text{rec}}$	–	One-step reconstruction of the observed displacement from $\tilde{\boldsymbol{\theta}}_k$.
$\mathcal{L}_{\text{gram}}$	–	Gram-matrix regularizer discouraging redundancy among learned collective and local patterns.
$\mathcal{L}_{\text{rig}}$	–	Auxiliary rigid-branch objective, combining reconstruction, linear-dynamics, and rigid-motion terms.
$\lambda_{(\cdot)}$	–	Weight of each loss term in the total objective.

Table 10: **Evaluation and analysis**.

Notation	Dim.	Description
MSE	–	Mean squared error over nodes and steps of a window, averaged over the test set.

Notation	Dim.	Description
$f \in \{\mathrm{rig}, \mathrm{col}, \mathrm{loc}\}$	–	Family index.
$\Delta\widehat{\mathbf{x}}_t^f = \mathbf{M}_f(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0$	$3 \times N$	Displacement produced by family f alone.
ρ_f	–	Contribution of family f, $\|\Delta\widehat{\mathbf{x}}^f\| / \big(\sum_{f'} \|\Delta\widehat{\mathbf{x}}^{f'}\|^2\big)^{1/2}$; the three need not sum to one, since the families overlap.
$\mathbf{Z}$	$n_{\mathrm{win}} \times 2$	Two-dimensional PCA projection of the standardized $\boldsymbol{\theta}_0$ of all evaluated windows.
$\delta(w, w')$	–	Coefficient-space distance between windows, the Euclidean distance between their $\boldsymbol{\theta}_0$ after per-dimension standardization.

B Compositional Mode Dynamics (CMD)

B.1 Mode Dictionary Construction

The dictionary is built once, before training, from the reference structure $\mathbf{X}^{\mathrm{ref}}$ and the training split. Its columns are grouped into modes. An oscillatory mode owns two patterns, one driven by a cosine and one by a sine of the same frequency. A non-oscillatory mode owns a short chain of columns driven by a single Jordan block, of which only the first carries a pattern; the others act on the coefficients alone (Appendix B.2).

B.1.1 Rigid family

Translation. Three patterns, one per Cartesian axis: every node moves together along that axis. Each is the head of an order-o chain, with $o = 2$ for MD17 and $o = 3$ for Mocap.

Rotation. This family has to represent a rotation of the whole system as it is, not only a small one: over a single prediction step an MD17 molecule can turn by tens of degrees. Rodrigues' formula splits the displacement produced by a rotation into two parts, one scaling with the sine of the angle and one with one minus its cosine. Whatever the axis, the first part is a combination of three fields and the second of six,

$$(\mathbf{r}_c)_n = \mathbf{e}_c \times \mathbf{x}_n, \qquad (\mathbf{r}_{cd})_n = \mathbf{e}_c(\mathbf{x}_n \cdot \mathbf{e}_d) + \mathbf{e}_d(\mathbf{x}_n \cdot \mathbf{e}_c), \qquad 1 \le c \le d \le 3, \tag{18}$$

where $\mathbf{x}_n$ is the n-th column of $\mathbf{X}^{\mathrm{ref}}$. The first three are the familiar infinitesimal rotations and the six are their second-order counterparts. Together the nine span the displacement of a finite rotation about any axis, which is what makes the representation exact rather than a small-angle approximation. The patterns are fixed and never trained.

Temporal assignment. Leaving one of the two patterns of an oscillatory mode empty makes that mode act through a single time factor, and we use this to reproduce the two parts above. At a constant rotation rate the first-order patterns are needed with a sine and the second-order patterns with one minus a cosine, so we give each first-order pattern the sine slot and each second-order pattern the cosine slot, leaving the other slot empty in both cases. The "one minus" comes for free: the prediction is a displacement measured from the initial frame, so the constant part of every coefficient cancels and a cosine that starts at one becomes a cosine minus one.

A rate group is nine such modes, three of the first kind and six of the second, sharing one rotation rate. Their nine amplitudes are exactly what is needed to name a rotation axis and its size, and they are supplied by the initial coefficients. A group therefore fixes only how fast the system turns, while how it turns is decided per window; we use three groups, so a window may mix a slow, a medium and a fast rate. Rigid modes are left undamped, since a rotation at a constant rate should neither fade nor grow. In total, the family has 27 oscillatory modes and 54 coefficients, of which 27 carry a pattern.

Frequency initialization. For each training window, we measure how far the system turns between its first and last frame and divide by the number of steps. The three group rates start at the 20th, 50th

and 80th percentile of these per-step rates over the training set, with a floor of one degree per step, and are trained afterwards.

B.1.2 COLLECTIVE FAMILY

This family should contain the ways the system deforms and nothing else, so we read those ways off the training data.

Deformation snapshots. From the body-frame coordinates of a training window we remove the rigid motion that is still left in each frame: we align every frame to the initial one with a Kabsch rotation $\mathbf{R}_t$ and keep the difference,

$$\mathbf{d}_t = \mathbf{R}_t^\top \bar{\mathbf{b}}_t - \bar{\mathbf{b}}_0, \tag{19}$$

where $\bar{\mathbf{b}}_t$ are the centered body-frame coordinates. These snapshots are collected over all training windows and all $t \geq 1$ and centered.

Removing what is not deformation. The alignment removes the rigid motion of each snapshot individually, but a little of the six rigid directions, three translations and three infinitesimal rotations, survives in the ensemble, so we subtract those six components from every snapshot.

Candidate subspace. We take the SVD of what remains, keep the directions whose singular value exceeds 10^{-4} of the largest, and retain the leading $n_{\mathrm{pca}} = \max\big(2, \lceil (3N-6)/3 \rceil\big)$ of them as the candidate subspace $\mathbf{\Phi}$. It is computed once on the training split and held fixed.

Learned modes. The family has $R_{\mathrm{col}} = 2n_{\mathrm{pca}}$ modes, and the two patterns of each are a learned combination of the columns of $\mathbf{\Phi}$, normalized together to unit norm. The patterns are therefore learned but cannot leave the deformations the system was actually observed to undergo.

Initialization. Frequencies come from a fixed bank: one near-zero slot, then 1°, 3° and 6°, then values equally spaced between 8° and 178.5°, assigned in slot order. Since the rigid slots are overwritten by the percentiles above, the collective slots inherit the upper part of the bank. For each frequency, we read the real FFT of the deformation time series at the nearest bin and initialize the combination weights by projecting that component onto $\mathbf{\Phi}$ and normalizing. Damping is learnable and starts at 0.05. The Gram penalty of Appendix B.6 discourages duplicate patterns within this family.

B.1.3 LOCAL FAMILY

Support. The support of node i is the set of nodes within h bonds of it, including itself, with $h = 2$ for MD17 and $h = 1$ for Mocap.

Candidates. For each node we build the following displacement fields, in this order: three local translations, in which the support moves together along a Cartesian axis; three local rotations, in which the support turns about its own centroid; one stretch per incident bond, in which that bond lengthens and shortens; and one bend per bond angle with the node at its apex, in which that angle opens and closes. The last two are the Wilson B-vectors,

$$\mathbf{u}_j = \tfrac{1}{2}\hat{\mathbf{u}}_{ij}, \quad \mathbf{u}_i = -\tfrac{1}{2}\hat{\mathbf{u}}_{ij}; \qquad \mathbf{u}_j = \frac{\cos\theta\,\hat{\mathbf{u}}_{ij} - \hat{\mathbf{u}}_{ik}}{\|\mathbf{x}_j - \mathbf{x}_i\|\sin\theta}, \quad \mathbf{u}_i = -(\mathbf{u}_j + \mathbf{u}_k), \tag{20}$$

with $\hat{\mathbf{u}}_{ij}$ the unit vector from node i to node j and $\mathbf{u}_k$ given by exchanging j and k. The list is truncated to 16 candidates per node in the order above.

Removing global motion. Left as they are, sums of these per-node candidates could reproduce a translation or a rotation of the whole system, so the same displacement would be representable in two families. We therefore subtract the six rigid components from every candidate before normalizing it. The operation takes away the part of a candidate that lies in the rigid directions rather than placing it there: after it, "node i moves one way" becomes "node i moves one way while the rest of the system compensates, so that the system as a whole neither shifts nor spins".

Learned modes. Each node carries 16 oscillatory modes whose two patterns are a learned combination of that node's candidates, normalized per mode. They are initialized, node by node, by the least-squares fit of the same FFT components used for the collective family within that node's candidate span. Damping is learnable and starts at 0.05.

Drift. Each node additionally carries three order-two non-oscillatory chains, one per axis, whose leading column is the unit displacement of that node with the rigid components removed and whose second column has no pattern.

Dimensions. A node contributes $6 + 2 \cdot 16 = 38$ columns and the system-wide part contributes $3o + 2(27 + R_{\mathrm{col}})$, so $P = d_G + 38N$. Of the $3o$ translation columns three are non-zero, of the 54 rotation coefficients 27 carry a pattern, and of the $6N$ drift columns $3N$ are non-zero; the rest are higher Jordan components, which carry coefficients but no pattern. For benzene ($N = 6$, $o = 2$, $n_{\mathrm{pca}} = 4$) this gives $d_G = 76$ and $P = 304$.

B.2 Non-oscillatory Blocks of $\mathbf{\Lambda}$

Since $\mathbf{K} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, the behaviour of the individual modes lives entirely in $\mathbf{\Lambda}$. The oscillatory blocks are given in Section 3.4; here we give the remaining blocks and the layout of $\mathbf{\Lambda}$ as a whole.

Jordan chain. A non-oscillatory mode occupies one chain of order o. Its coefficients read as a position, a velocity and, if $o > 2$, an acceleration, and one time step adds each coefficient to the one above it. The block is the $o \times o$ matrix with ones on the diagonal and on the first superdiagonal,

$$\mathbf{J}_o = \begin{bmatrix} 1 & 1 & & \\ & 1 & \ddots & \\ & & \ddots & 1 \\ & & & 1 \end{bmatrix}, \qquad (\mathbf{J}_o^t)_{ij} = \binom{t}{j-i} \ (j \geq i), \qquad (\mathbf{J}_o^t)_{ij} = 0 \ \ (j < i), \tag{21}$$

where t is an integer step index and the binomial vanishes for $j - i > t$. For the two orders we use,

$$\mathbf{J}_2^t = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{J}_3^t = \begin{bmatrix} 1 & t & \frac{t(t-1)}{2} \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}. \tag{22}$$

An order-o chain therefore produces motion that is polynomial of degree $o - 1$ in t. Only its leading coefficient owns a pattern; the others act through this coupling alone (Appendix B.1).

Translation. One chain per Cartesian axis, three in total. MD17 uses $o = 2$, a translation at constant speed, and Mocap uses $o = 3$, which lets the system-wide translation curve within a window.

Drift. Every node carries three order-two chains, one per axis, so that a local displacement of a single node can continue at a constant rate through the window.

Assembly. The coefficient vector holds the system-wide part first and then one block per node. The system-wide part holds the three translation chains, axis by axis and in chain order within each, followed by the R_g oscillatory pairs with the rotation slots before the collective slots. Each node block holds its three drift chains followed by its R_{loc} oscillatory pairs. Hence

$$\mathbf{\Lambda} = \mathrm{blkdiag}\Big(\underbrace{\mathbf{J}_o, \mathbf{J}_o, \mathbf{J}_o}_{\text{translation}}, \ \big\{e^{-d_r}\mathbf{Rot}(\omega_r)\big\}_{r=1}^{R_g}, \ \underbrace{\big[\mathbf{J}_2, \mathbf{J}_2, \mathbf{J}_2, \ \big\{e^{-d_l}\mathbf{Rot}(\omega_l)\big\}_{l=1}^{R_{\mathrm{loc}}}\big]}_{\text{once for each of the } N \text{ nodes}}\Big). \tag{23}$$

Powers. We never form $\mathbf{\Lambda}^t$ by repeated multiplication: each block has a closed form, an oscillatory block becoming $e^{-d_r t}\mathbf{Rot}(\omega_r t)$ and a chain block following the binomial expression above. Every horizon therefore costs the same, and the blocks $\mathbf{A}_k = \mathbf{M}(\mathbf{K}^k - \mathbf{I})$ can be built for all prediction steps at once.

Damping. Damping rates are parameterized as $d = \exp(\cdot)$ so that they stay positive, and are learned, except that $d = 0$ is imposed on every rigid rotation slot and on the first slot of each bank. The chain blocks carry no damping.

B.3 Coupling Levels

The three coupling levels are realized by masking $\mathbf{S}$ before the exponential. With a binary $\mathbf{W} \in \{0, 1\}^{P \times P}$ we form

$$\mathbf{Q} = \exp\Big(\tfrac{1}{2}\big(\widetilde{\mathbf{S}} - \widetilde{\mathbf{S}}^\top\big)\Big), \qquad \widetilde{\mathbf{S}} = \mathbf{W} \odot \mathbf{S}. \tag{24}$$

Full coupling uses $\mathbf{W} = \mathbf{1}$. Intra-family coupling sets $\mathbf{W}$ to one on the three diagonal blocks corresponding to the rigid, collective and local columns and zero elsewhere; the masked generator is then block diagonal, so $\mathbf{Q}$ is block diagonal as well and modes mix only within a family. No coupling fixes $\mathbf{Q} = \mathbf{I}$, in which case $\mathbf{S}$ is not used at all.

B.4 Initial Coefficient Estimation

This section describes how $\boldsymbol{\theta}_0$ is read off the initial state. The auxiliary rigid branch is described in Appendix B.5.

Encoder. The encoder f_{enc} follows GemNet. Starting from an embedding of the atom or joint type, it forms a message for every bond, bond angle and torsion and passes them through three interaction blocks to produce node features. Each message is built from the features of its endpoints, a geometric basis (a 16-component Gaussian radial basis with a $5\,\text{Å}$ cutoff for bond lengths, and four-frequency Fourier bases for angles and torsions), a 16-dimensional type embedding, and the one-step change of that geometric quantity. The last term is how velocity enters: bond lengths, angles and torsions are evaluated both at the current frame and at the preceding one, and their differences are appended. Node features are 64-dimensional.

Graph-level readout. The readout f_{graph} produces a single vector for the system, containing the collective coefficients together with the direct estimate of the rigid coefficients. Its input has three parts: the mean and the maximum of the node features; fifteen rigid observables, namely the total linear momentum, the total angular momentum, the six upper-triangular entries of the inertia tensor and the node-wise $\mathbf{b} \cdot \mathbf{v}$ summed over axes, normalized by dataset statistics and scaled by a learned weight; and the output of a vector attention, in which four learned queries attend over the nodes through keys computed from the node features and take a weighted sum of five vectors prepared at each node, $\mathbf{b}$, $\mathbf{v}$, $\mathbf{b} \times \mathbf{v}$, $\mathbf{v}_\perp$ and $\mathbf{b} \times \mathbf{v}_\perp$, where $\mathbf{v}_\perp$ is the velocity with its component along that node's bond directions removed. The three parts are concatenated and passed through a two-layer MLP of width 256. All inputs are body-frame quantities, so the output does not depend on the orientation of the lab frame.

Node-level readout. The readout f_{node} produces the drift and local oscillatory coefficients of each node. It first forms a node representation by adding to the node feature a projection of fifteen local geometric quantities ($\mathbf{b}$, $\mathbf{v}$, $\mathbf{b} \times \mathbf{v}$ and the upper triangle of $\mathbf{b}\mathbf{b}^\top$) together with a 32-dimensional summary of their graph-wide average. A drift head, an MLP of width 128, emits one value per axis, to which the observed velocity of that node is added through a learned per-system gate. These three values are placed in the *velocity* entry of the corresponding drift chains, the position entry being left at zero, since a displacement measured from $t = 0$ has no constant part. An oscillatory head with a FiLM structure lifts the node representation to width 128, modulates it by a scale and a shift produced from an eight-band Fourier embedding of the mode frequency, and reads out two coefficients per mode. The same module serves every mode; only the frequency distinguishes them.

Regularization. Dropout with probability 0.3 is applied in the hidden layers of both readouts. During training the same readouts are applied to several observed frames to obtain $\tilde{\boldsymbol{\theta}}_t$ (Appendix B.6); at test time they are applied at $t = 0$ only.

B.5 Auxiliary Rigid Branch

Motivation. The rigid coefficients are estimated once, at $t = 0$, and are then carried through the horizon by an undamped linear operator, so an error in that single estimate survives to the end of the window, and because a rotation acts on every node at once the cost of such an error is large. The auxiliary branch predicts the rotation a second time, in coordinate space, and supplies a second estimate.

Branch. The branch takes the same encoder output as CMD and forms a latent state per node. It adds to the node feature the position and velocity of that node together with a 32-dimensional summary, averaged over the system, of position, velocity, $\mathbf{x} \times \mathbf{v}$ and the upper triangle of $\mathbf{x}\mathbf{x}^\top$, and then reads out the coefficient pairs of 128 latent oscillators with the same FiLM structure as the oscillatory head of Appendix B.4. The latent state is advanced by its own linear operator $\mathbf{K}_{\mathrm{rig}}$, a block-diagonal matrix of damped planar rotations whose first four blocks are undamped; its frequencies are initialized from a bank built as in Appendix B.1.2 and are learned. A two-layer GemNet-based decoder

maps the latent state at each step back to coordinates, with a Fourier embedding of the frequencies added to its input. During training the latent state is perturbed by Gaussian noise of ten percent of its magnitude and dropout with probability 0.3 is applied in the readout.

Branch losses. The branch is trained by its own objectives only: the reconstruction error of the coordinates decoded from the encoding of each observed frame; the mismatch between the latent state rolled out from the initial frame and the latent state encoded directly at that frame, whose target is detached; the error of the rolled-out coordinates against the ground truth; the error of their center of mass; and the squared Frobenius error between the rotation extracted from the rollout by Kabsch alignment and the ground-truth rotation. The first two carry weight 1 and the last three weight 0.1.

Extraction. Only the rigid part of the rolled-out trajectory is used. Centering the coordinates at each step and aligning them to the initial frame with a Kabsch rotation gives $\widehat{\mathbf{R}}_t$, and their mean gives $\widehat{\mathbf{c}}_t$. Mapped into the body frame these define the rigid displacement field

$$\mathbf{y}_t^{\mathrm{rig}} = \widehat{\mathbf{R}}_t \bar{\mathbf{b}}_0 - \bar{\mathbf{b}}_0 + \widehat{\mathbf{c}}_t. \tag{25}$$

The coordinates produced by the decoder are never used as a prediction.

Projection. We then ask which coefficients explain this displacement field under CMD's own dictionary and operator. Let $\mathbf{A}_{\mathrm{r}}$ be the rigid columns of $\mathbf{A}_k = \mathbf{M}(\mathbf{K}^k - \mathbf{I})$ stacked over $k = 1, \dots, T$. The projected estimate is the ridge-regularized least-squares solution

$$\boldsymbol{\theta}_{\mathrm{rig}}^{\mathrm{proj}} = \left(\mathbf{A}_{\mathrm{r}}^\top \mathbf{A}_{\mathrm{r}} + 10^{-3}\mathbf{I}\right)^{-1} \mathbf{A}_{\mathrm{r}}^\top \mathbf{y}^{\mathrm{rig}}, \tag{26}$$

taken over the rotation slots and the non-position entries of the translation chains. Because the dictionary and the operator are shared with CMD, the result can be placed directly into the corresponding entries of $\boldsymbol{\theta}_0$.

Fusion. The two estimates are combined by a gate predicted per window: a scalar obtained by applying a sigmoid to the output of a width-128 MLP fed with the mean and the maximum of the node features at the initial frame together with the fifteen rigid observables of Appendix B.4. The gate is applied only to the entries listed above. By default $\widehat{\mathbf{R}}_t$ and $\widehat{\mathbf{c}}_t$ are detached, so the loss of CMD does not train the branch and the branch is trained by its own objectives alone.

B.6 Training Objective

Coefficients at intermediate frames. During training, the readouts of Appendix B.4 are applied to every observed frame of the window, giving $\tilde{\boldsymbol{\theta}}_t$; the rigid fusion of Appendix B.5 is applied at $t = 0$ only. By default, all frames of the window are used.

Horizon-balanced sampling. For every horizon length $j = 1, \dots, T$ we draw a start frame t uniformly among the observed frames with $t + j \le T$. Every horizon length therefore appears exactly once per batch, and long horizons are not trained less often than short ones; at test time the start is always $t = 0$. Horizons that share a start are handled together as one joint Gaussian, so the correlation between them induced by the shared $\tilde{\boldsymbol{\theta}}_t$ is part of the model.

Likelihood. For such a group, we stack the corresponding $\mathbf{A}_k$ and evaluate the negative log-likelihood under

$$\boldsymbol{\Sigma} = \sigma_y^2 \mathbf{I} + \mathbf{A}\boldsymbol{\Sigma}_\theta \mathbf{A}^\top. \tag{27}$$

The value is divided by $3N$ and averaged over the (t, j) pairs. The matrix $\boldsymbol{\Sigma}_\theta$ is diagonal and learned per system, initialized at half the coefficient scales of Appendix B.4; σ_y is learned and starts at $0.5\,p$. The Cholesky factorization is carried out in float64, retrying with jitter of 10^{-8}, 10^{-6} and 10^{-4} times the mean diagonal if it fails.

Coordinate loss. $\mathcal{L}_{\mathrm{pos}}$ is the mean squared error between the prediction mapped back to the lab frame and the ground-truth coordinates, divided by w, the mean squared within-window displacement of that system, so that systems of different amplitude contribute comparably. This term uses the actual prediction, i.e. the one built from $\boldsymbol{\theta}_0$ after the rigid fusion.

Coefficient-dynamics losses. $\mathcal{L}_{\mathrm{cons}}$ compares the initial coefficients advanced by k steps with the coefficients estimated directly at frame k. The target is detached, the difference is divided by p, and it is measured on the collective and local columns only, the rigid columns being the responsibility

of the auxiliary branch. $\mathcal{L}_{\mathrm{rec}}$ checks that the one-step transition $\mathbf{M}(\mathbf{K}-\mathbf{I})$ reproduces the observed one-step displacement; it uses all columns and is divided by w.

Gram penalty. We measure how far the Gram matrix of the collective patterns departs from the identity, and likewise, node by node, for the local patterns, averaging over nodes. Both terms are divided by the number of modes involved and added. The penalty discourages duplicate patterns within a family.

Weights. $\mathcal{L}_{\mathrm{rig}}$ is already the weighted sum of the five branch terms of Appendix B.5, with weight 1 on reconstruction and latent consistency and 0.1 on the coordinate, center-of-mass and rotation terms.

Table 11: **Weights**.

λ_{nll}	λ_{pos}	λ_{cons}	λ_{rec}	λ_{gram}	λ_{rig}
1	1	1	1	0.01	1

C Experiment detail

C.1 Dataset Details

MD17. MD17 contains DFT molecular dynamics trajectories of eight small organic molecules. Following the usual convention, hydrogen atoms are excluded and only the heavy atoms are kept as nodes, giving the node counts in Table 12. Bonds are determined from the interatomic distances at the first frame with a covalent-radius threshold of 1.6, and bond angles and torsions are derived from that bond graph. We do not extend the edges to two-hop neighbours as is common in the EGNN family, since the local family already builds its support from multi-hop neighbourhoods. Each trajectory is randomly split into $500/2000/2000$ state-trajectory pairs for training, validation and testing. A window consists of an initial state and $T = 8$ uniformly spaced future steps, the last of which lies 3000 frames ahead of the initial state.

Table 12: **Statistics of the MD17 dataset**.

Name	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil
# nodes	13	6	3	5	10	10	7	8

Motion Capture. We use the Walk and Run motions of the CMU Motion Capture dataset following the protocol of EGNO. Each skeleton is a graph over its 31 joints, with edges given by the joints that are directly connected in the body. The numbers of windows are $198/600/600$ for Walk and $200/240/240$ for Run. A window consists of an initial state and $T = 5$ uniformly spaced future steps, the last of which lies 30 frames ahead.

C.2 Implementation Details

Setup. Our framework is implemented in PyTorch with PyTorch Lightning, and every experiment is run on a single NVIDIA RTX A6000 with CUDA 12.1. We optimize with AdamW at a fixed learning rate of 3×10^{-4} without scheduling, with gradient clipping at 0.5, for up to 5000 epochs. Validation is performed every five epochs and the checkpoint with the lowest validation error is used for testing. Weight decay is applied in three groups: 10^{-4} on the mode patterns and the operator parameters $\mathbf{S}$, $\mathbf{C}_r$ and $\mathbf{D}_i$, zero on the frequencies, damping rates, gates and attention queries, and 10^{-12} on everything else.

Reproducibility. We provide a link to the source code for reproducibility. Please refer to the following repository[1].

[1] https://anonymous.4open.science/r/CMD-62EA/

Table 13: **Hyperparameters of CMD**.

	MD17	**Mocap**
Horizon T / step Δt (frames)	8 / 375	5 / 6
Batch size	100	50
Rate groups G	3	3
Collective modes R_{col}	$2\lceil(3N-6)/3\rceil$	58
Local modes per node R_{loc} / candidate cap	16 / 16	16 / 16
Support hop h	2	1
Translation order o	2	3
Coupling	full	full
Damping initialization	0.05	0.05
Encoder d_{atom} / blocks / cutoff	64 / 3 / 5 Å	same
Encoder RBF / angle, torsion frequencies / type embedding	16 / 4, 4 / 16	same
Graph readout hidden / node readout hidden	256 / 128	same
FiLM bands / band scale / pooling dimension	8 / 50 / 32	same
Vector attention queries / key dimension	4 / 32	same
Dropout	0.3	0.3
Rigid branch oscillators / hidden / undamped / decoder layers	128 / 128 / 4 / 2	same
Rigid branch frequency MLP hidden / latent noise / ridge	64 / 0.1 / 10^{-3}	same
Scale factors for σ_θ, σ_y, local σ	0.5, 0.5, 0.5	same

D Additional Experimental Results

D.1 Estimating the initial coefficients

Motivation. CMD estimates $\boldsymbol{\theta}_0$ once, from the initial state alone, and the whole trajectory follows from $\mathbf{M}(\mathbf{K}^t-\mathbf{I})\boldsymbol{\theta}_0$; there is no expressive decoder afterwards, so information absent from the initial state is never recovered. This exposes the rigid family in particular. Its rotation modes are undamped ($d_r = 0$), act on every node at once, and are slow—below 1° per step on most molecules—so an error in $\boldsymbol{\theta}_{\mathrm{rig}}$ neither decays nor stays local: it accumulates over the whole window. Yet it must be inferred from $\mathbf{V}_0$, a one-step finite difference, while the prediction step is Δt (375 frames). The slowest component of the motion is estimated from the shortest available baseline.

The auxiliary rigid branch is already one response to this. It recovers $\boldsymbol{\theta}_{\mathrm{rig}}$ from a decoded coarse trajectory rather than from a single linear readout of $\mathbf{V}_0$, and therefore constructs the Δt-scale evidence the direct readout lacks. Removing it collapses exactly the rigid-dominated molecules (benzene $1.76 \to 8.77$; ethanol's error rises by 25%) while leaving the local-dominated ones essentially unchanged (aspirin and uracil within 2%). It does not remove the bottleneck. Interventions on it are of two kinds: *architectural* ones change how $\boldsymbol{\theta}_{\mathrm{rig}}$ is extracted from the same input, as the rigid branch does, while *input-side* ones change what the estimator is given. The two below are of the second kind and leave the architecture untouched.

Interventions. **(v)** computes $\mathbf{V}_0$ from the frame Δt earlier instead of the preceding one, raising the signal-to-noise ratio of the slow component; the dictionary, the operator and everything else are unchanged. **(h)** includes hydrogen atoms in the input graph while evaluating on heavy atoms exactly as before, adding information about molecular orientation and torsional state without altering the prediction target.

Table 14: **Diagnosis of the $\boldsymbol{\theta}_0$ bottleneck on MD17, MSE**$(\times 10^{-2})$.

	Aspirin	Benzene	Ethanol	Malon.	Naphth.	Salicylic	Toluene	Uracil
CMD	4.61	1.76	2.69	10.76	0.23	0.50	3.91	0.35
+ (v)	5.18 (−12%)	**1.03** (+41%)	2.54 (+6%)	10.79 (−0%)	0.25 (−10%)	0.59 (−18%)	**1.33** (+66%)	0.45 (−28%)
+ (h)	4.93 (−7%)	1.59 (+10%)	2.71 (−1%)	11.21 (−4%)	0.23 (−3%)	0.53 (−5%)	**1.30** (+67%)	0.39 (−10%)
Rigid contribution	0.18	0.99	0.97	0.50	0.16	0.14	0.79	0.16

Table 15: **Ablation on coupling and rigid branch on MD17 and motion capture (Mocap)**. MSE ($\times 10^{-2}$). The best performance is highlighted in bold.

Coupling	Rigid branch	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil	Mocap-Walk	Mocap-Run
No	✓	4.77	1.96	2.70	10.87	0.26	0.52	**3.45**	0.40	3.34	19.40
Intra	✓	5.04	1.80	2.74	10.78	0.24	0.51	3.73	0.40	3.24	**14.76**
Full	✓	**4.61**	**1.76**	**2.69**	10.76	**0.23**	**0.50**	3.91	**0.35**	**2.89**	17.70
Full	–	4.65	8.77	3.35	**10.72**	0.24	**0.50**	4.32	**0.35**	3.10	20.52

Results. Both interventions improve exactly the molecules where the rigid family dominates the predicted displacement—benzene (0.99), ethanol (0.97) and toluene (0.79)—and degrade the local-dominated ones. Toluene falls from 3.91 to 1.33 and 1.30; benzene reaches 1.03 with (v). These are precisely the two molecules on which CMD trails the strongest baselines.

Decomposing the error of both models with the same orthogonal operator—a Kabsch fit for the rigid part, projection of the residual onto the collective PCA subspace $\mathbf{\Phi}$ for the collective part, and what remains for the local part—localises the gain. On toluene the rigid component falls by 73% (3.72 $\rightarrow$ 1.01) and accounts for the entire 65% reduction in total error, while the collective component worsens by 8% and the local one changes by 10%. The model's own attribution shifts accordingly: toluene's local contribution drops from 0.58 to 0.40 while its rigid contribution rises from 0.79 to 0.83, indicating that the local family had been absorbing motion belonging to the rigid one.

Why it is not adopted. The Δt velocity averages over the interval and therefore suppresses precisely the fast, spatially confined motion that the local family represents; on local-dominated molecules this removes information rather than adding it (gains of -28%, -18% and -12% on uracil, salicylic and aspirin). Both variants trade accuracy on the slow rigid component for accuracy on the fast local one, which is why we report them as a diagnosis of how $\boldsymbol{\theta}_0$ is estimated rather than as a better configuration. The limiting factor in CMD is thus not the mode dictionary or the operator, but the fact that $\boldsymbol{\theta}_0$ is read from a single frame pair: the same $\mathbf{M}$ and the same $\mathbf{K}$ produce a two-thirds lower error once the input is presented on a time scale matched to the component being estimated. An estimator that conditions on multiple time scales is the natural next step.

D.2 Ablation studies

Ablation on coupling and rigid branch. We conducted ablation studies on the MD17 and Motion Capture datasets to evaluate the effects of mode coupling and the auxiliary rigid branch. The results are summarized in Table 15.

D.3 Analysis Details

Family contribution. Since the predicted displacement is a sum of per-family terms, each term can be evaluated on its own. We define the contribution of family f as

$$\rho_f = \frac{\left\|\mathbf{M}_f(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0\right\|}{\left(\sum_g \left\|\mathbf{M}_g(\mathbf{K}^t - \mathbf{I})\boldsymbol{\theta}_0\right\|^2\right)^{1/2}}, \tag{28}$$

with Frobenius norms taken over nodes, time steps and windows. The denominator is not the norm of the total displacement but the root of the sum of the squared family norms, so that $\sum_f \rho_f^2 = 1$. This normalization is used because the column spaces of the families are not orthogonal; ρ_f is therefore the relative size of the displacement a family produces, not its share of an orthogonal decomposition. Table 16 reports ρ_f over the whole test split.

Family-wise error. ρ_f is the attribution the model makes for itself, so when comparing errors we use a decomposition that does not depend on the model. A displacement field is split into the rigid component obtained by a Kabsch fit to the initial structure, the projection of the remainder onto the collective candidate subspace $\mathbf{\Phi}$, and what is left. The three components are mutually orthogonal, so the split is unique. We apply the same operator to the predicted and to the ground-truth displacement and compare the squared error component by component, which is what makes the errors of two models comparable on the same terms.

Table 16: **Contribution ρ_f of each mode family to the predicted displacement, over the test split**.

	Aspirin	Benzene	Ethanol	Malon.	Naphth.	Salicylic	Toluene	Uracil	Mocap-Walk	Mocap-Run
Rigid	0.18	0.99	0.97	0.50	0.16	0.14	0.79	0.16	0.03	0.01
Collective	0.27	0.10	0.08	0.21	0.29	0.34	0.19	0.34	0.01	0.01
Local	0.95	0.12	0.24	0.84	0.94	0.93	0.58	0.93	1.00	1.00

Coefficient-space neighbors. Figure 2a shows $\boldsymbol{\theta}_0$ for every window of the test split. The coefficients are standardized coordinate-wise, and neighbors are chosen by Euclidean distance in the full standardized coefficient space; the two-dimensional plane in the figure is a PCA projection used for display only and plays no part in the selection. Since $\boldsymbol{\theta}_0$ is estimated from the initial state alone, no future frame is used in choosing the neighbours.

D.4 VISUALIZATION

We show the learned mode patterns of each system. The top two rows are collective modes, the bottom two the local modes of one node, marked by a red circle. Every mode has a cosine and a sine pattern, drawn in the first and the second row of each pair. Arrows are node displacements, scaled within each panel, and ω is the angular frequency per prediction step. Modes are ordered by the mean amplitude of their initial coefficients over the test split, and the node shown is the one with the largest local amplitude.

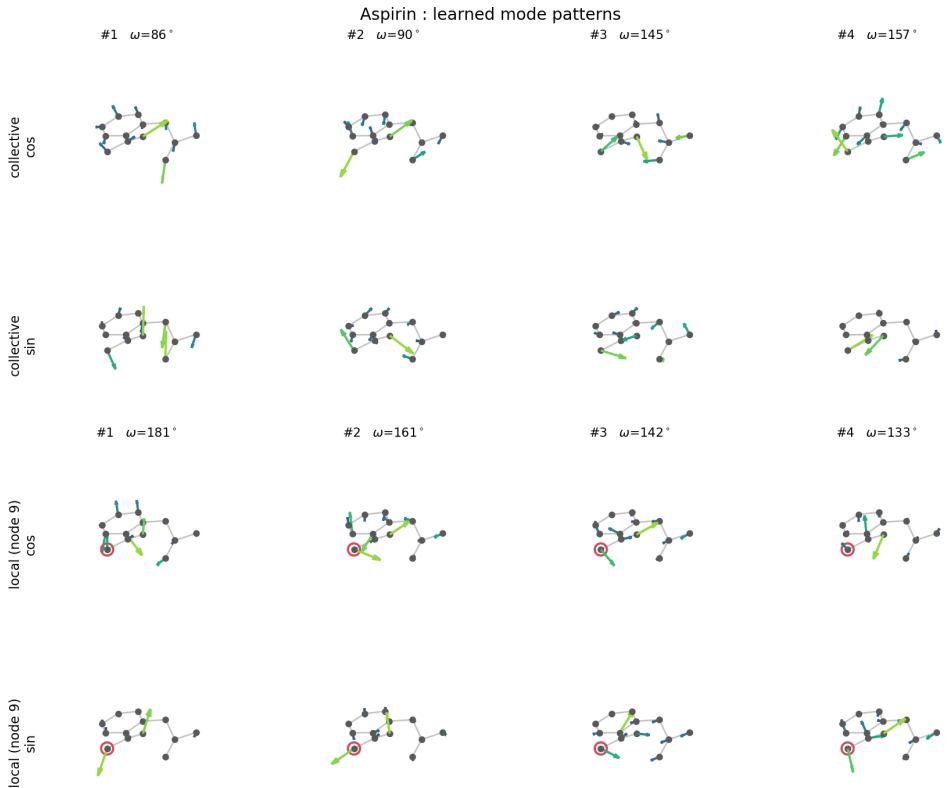


Figure 3: **Learned mode patterns, aspirin**.

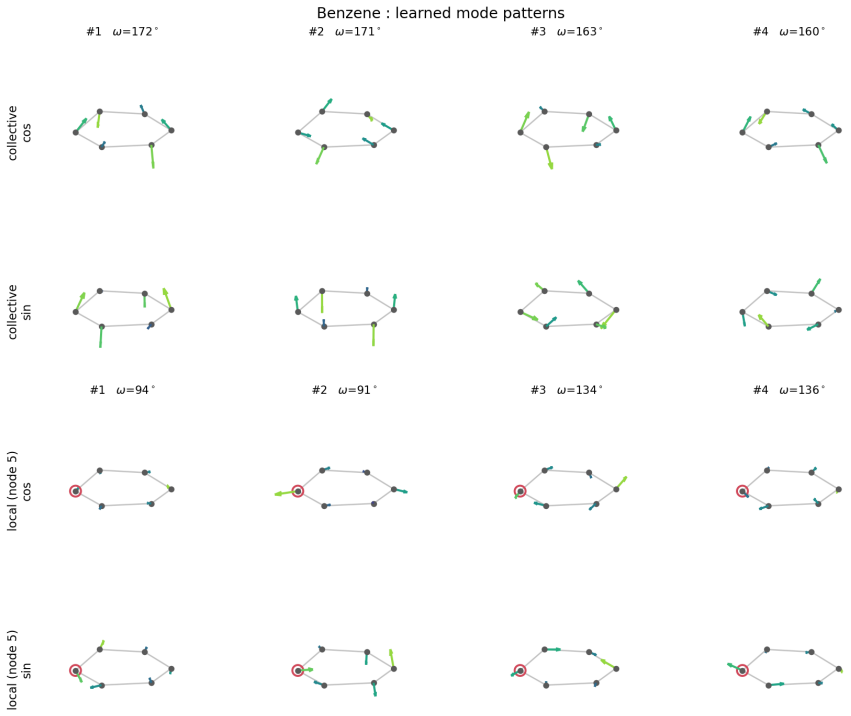


Figure 4: **Learned mode patterns, benzene**.

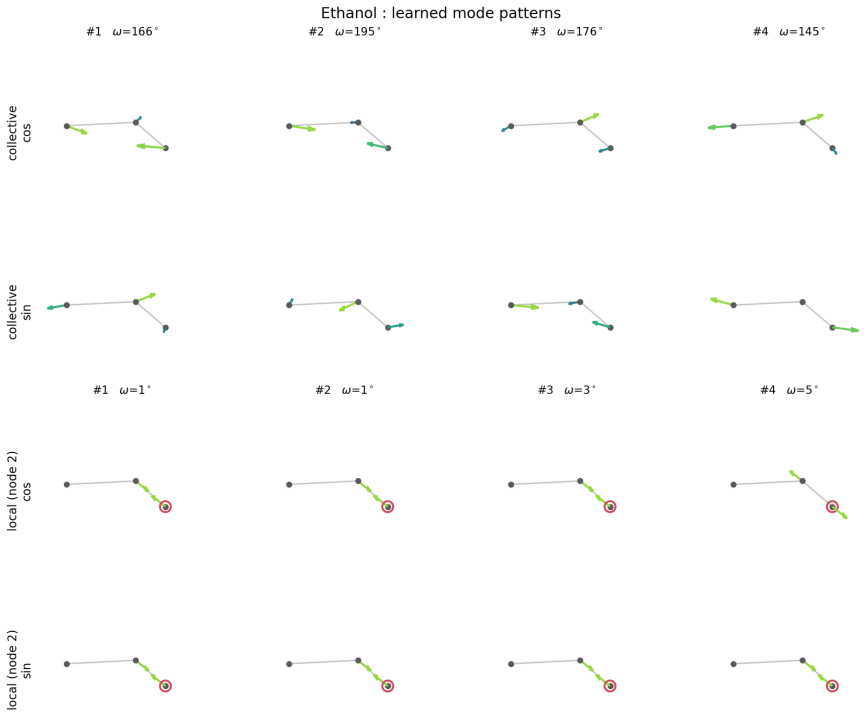


Figure 5: **Learned mode patterns, ethanol**.

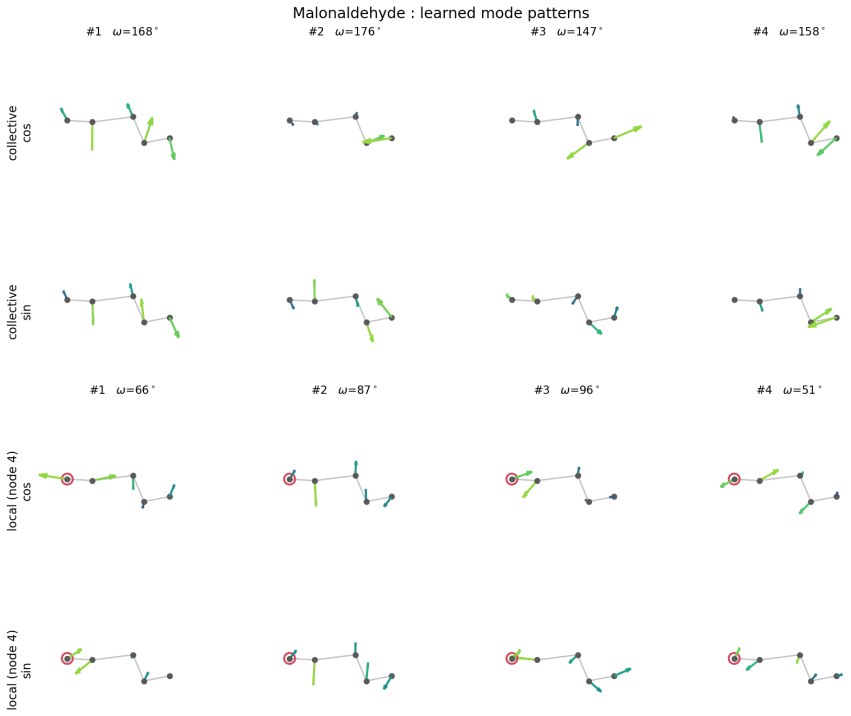


Figure 6: **Learned mode patterns, malonaldehyde**.

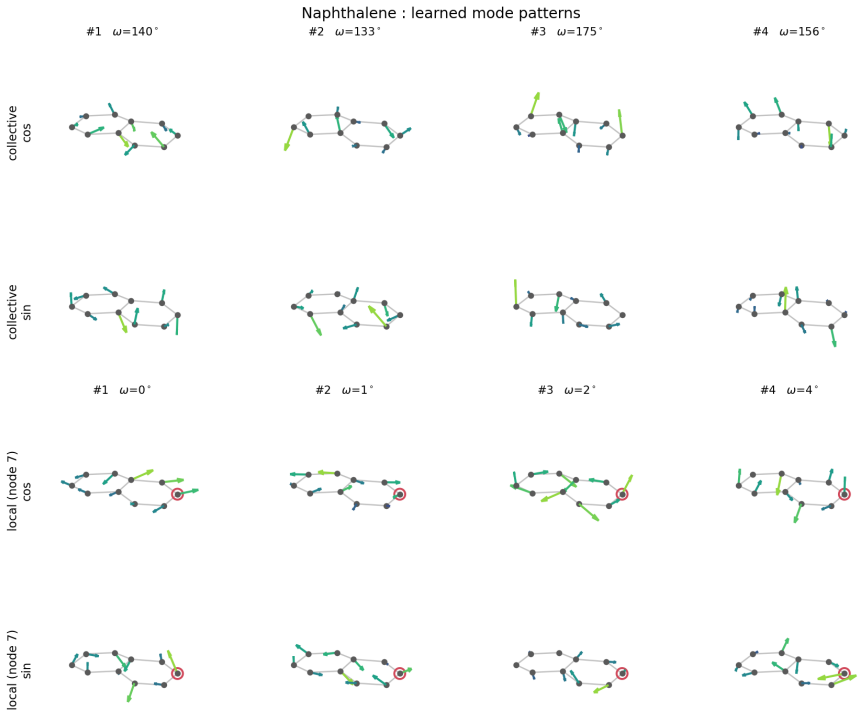


Figure 7: **Learned mode patterns, naphthalene**.

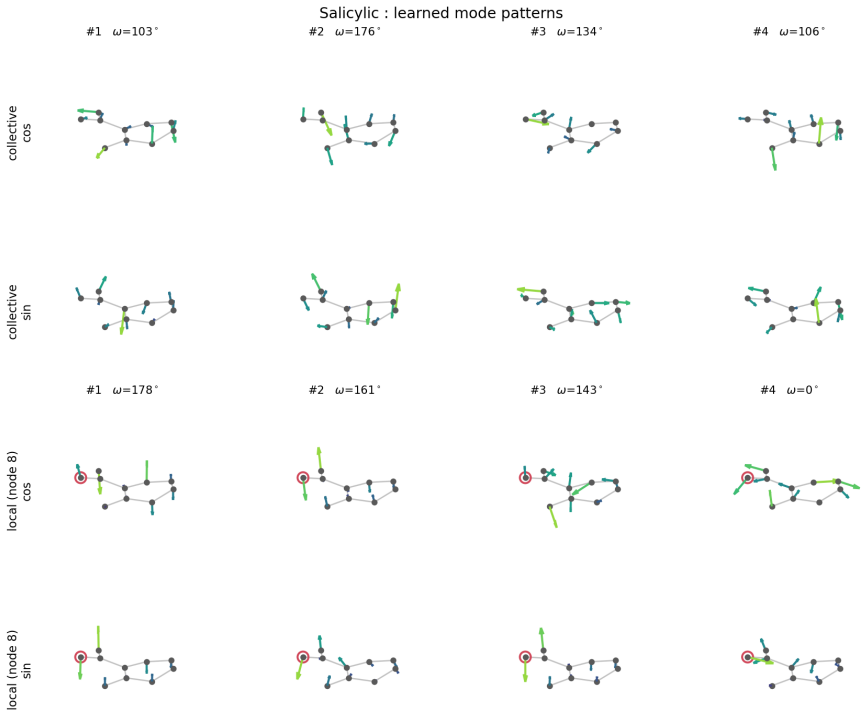


Figure 8: **Learned mode patterns, salicylic acid**.

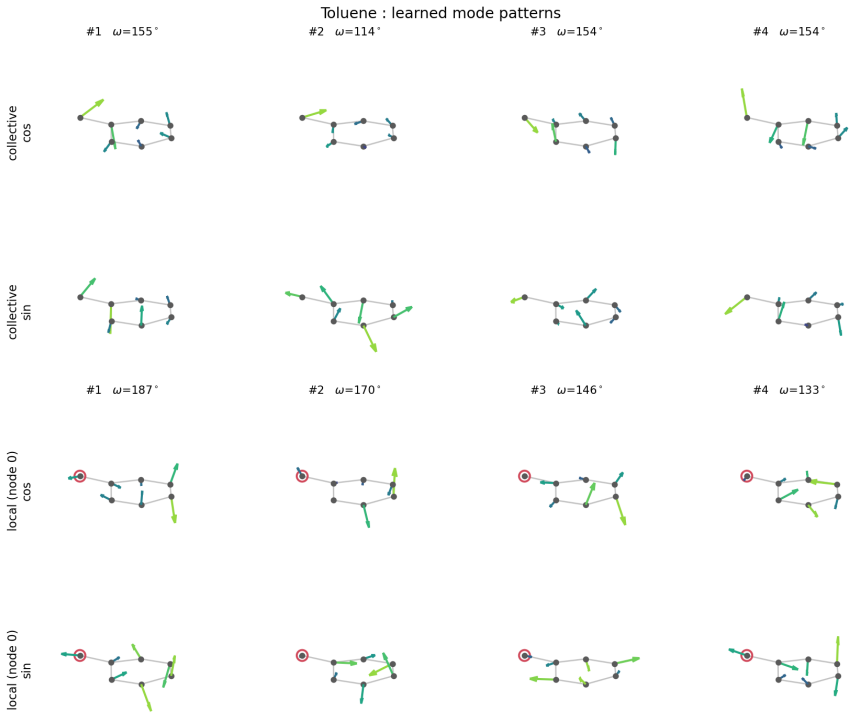


Figure 9: **Learned mode patterns, toluene**.

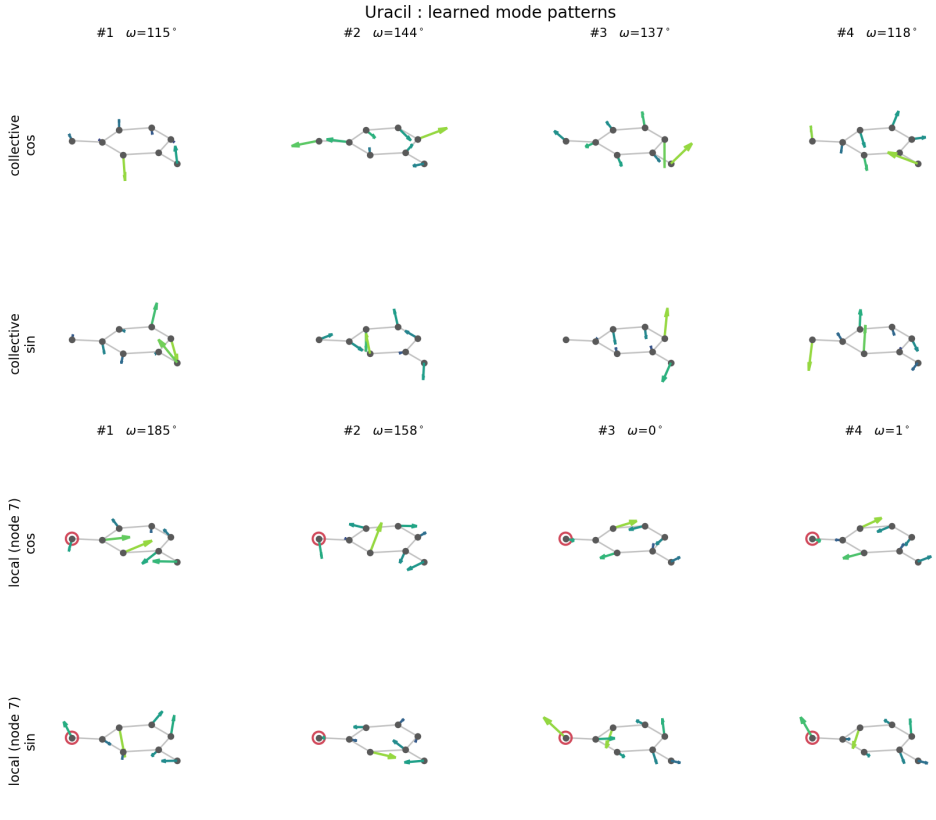


Figure 10: **Learned mode patterns, uracil**.

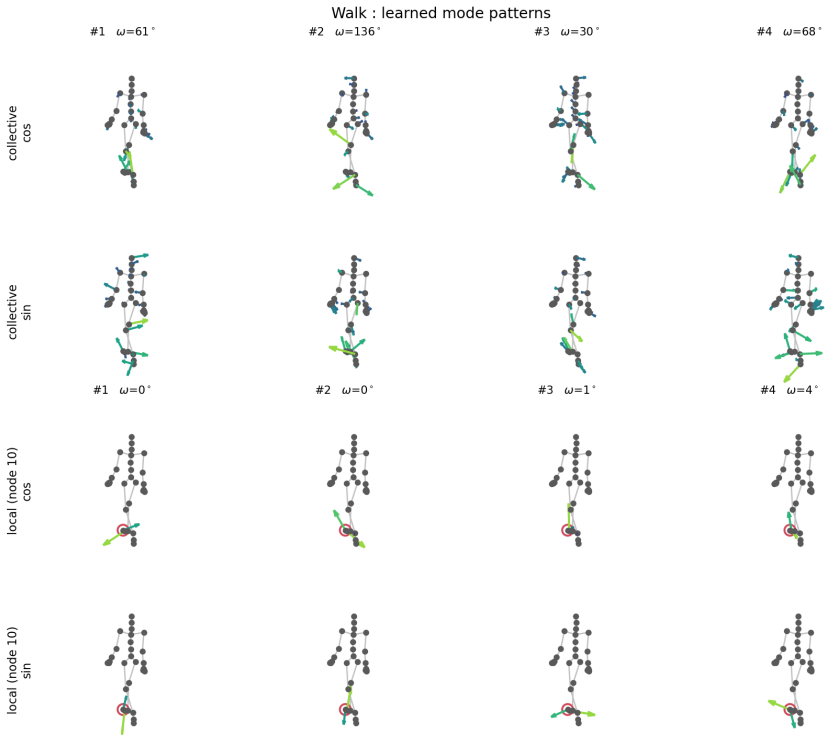


Figure 11: **Learned mode patterns, Mocap-Walk**.

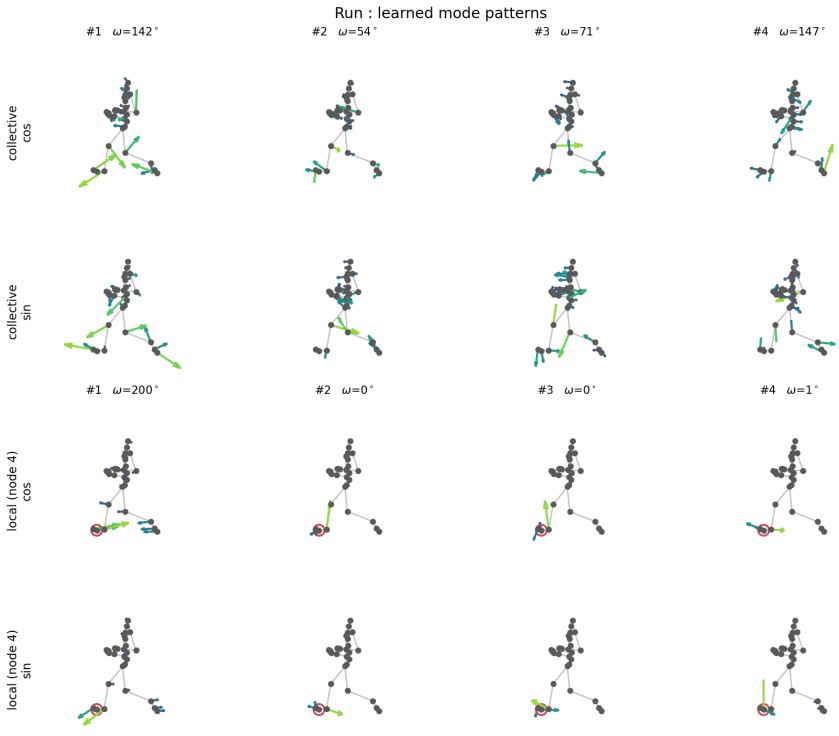


Figure 12: **Learned mode patterns, Mocap-Run**.