JOINT CROSS-MODE REPRESENTATION LEARNING FOR TENSOR WITH A HAMILTONIAN SINGULAR FIELD

Anonymous authors
Paper under double-blind review

ABSTRACT

Tensor factorization provides a compact framework for modeling high-order data by learning low-dimensional mode-wise representations and composing them for individual tensor elements. Neural tensor factorization further enriches this paradigm through nonlinear cross-mode interaction, yet such flexible fusion does not explicitly organize how changes in one mode affect the refinement of others. To address this limitation, we propose the Hamiltonian-Structured Neural Tensor Factorization (HSNTF), which introduces a Hamiltonian Singular Field (HSF) for context-specific latent representation refinement. For each tensor element, HSNTF assembles the corresponding mode-wise embeddings into a joint latent representation. A shared scalar-valued Hamiltonian induces a joint vector field that couples the context-dependent evolution of mode-specific representations. The field Jacobian maps second-order curvature, including mixed derivatives, into local cross-mode influence operators that characterize how perturbations to one mode affect the evolution of others. The behavior of these operators is structurally coupled during representation refinement. Short-horizon, multi-step integration repeatedly evaluates the field at the evolving latent state, producing context-adapted representations. Experiments on five real-world tensor datasets demonstrate competitive and stable predictive performance. Analysis results further establish Hamiltonian structure as a unified inductive bias for coupled cross-mode representation dynamics rather than a physical model of tensor data.

1 INTRODUCTION

Many real-world datasets naturally exhibit multidimensional, multi-entity, and multi-relational structures, such as user–service–time interaction in cloud computing and space–variable–time interactions in weather or traffic system (Kolda and Bader, 2009; Wang and Zhe, 2022; Wang et al., 2023). Tensor factorization provides an effective low-rank framework for representing such high-order data by learning low-dimensional embeddings for entities in each mode and modeling structured interactions across modes (Tucker, 1966; Kolda and Bader, 2009). CP and Tucker decomposition-based factorization approaches generally combine mode-specific factors through predefined multilinear operations, which heavily restrict the forms of cross-mode interaction they can express (Kolda and Bader, 2009; Tang and Luo, 2025). Neural Tucker Factorization enhances the expressiveness of cross-mode interactions by taking the outer product of mode-specific embeddings to construct a higher-dimensional interaction representation. A neural network then applies a nonlinear mapping to this representation (Tang and Luo, 2025). Subsequent methods further enrich this interaction stage. ANTucF employs auto-encoding latent interaction learning to model nonlinear spatiotemporal dependencies (Tang et al., 2025b), whereas 3DA-NTC introduces three-dimensional channel attention to enhance the learned interaction features (Kang et al., 2024a). These methods improve how latent representations are combined, transformed, and reconstructed. However, each entity within a mode still receives a fixed, context-independent embedding, irrespective of the entities from other modes (Wang and Zhe, 2022).

This limitation has motivated a shift from static embeddings toward dynamic representation learning (Bengio et al., 2013; Rubanova et al., 2019; Dupont et al., 2019). Neural ODEs parameterize the evolution of latent states through a learnable continuous-time vector field (Chen et al., 2018). Extending this perspective to tensor factorization, NONFAT replaces fixed entity factors with continuous trajectories, allowing representations to evolve along the temporal mode (Wang and Zhe, 2022).

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.

DEMOTE further makes relational dependence explicit by constructing a multipartite graph from observed tensor entries and propagating information through graph diffusion–reaction dynamics (Wang et al., 2023).

Together, these approaches move beyond context-independent embeddings by allowing latent representations to evolve and exchange relational information. Yet a central question remains: **how can cross-mode influences be organized within a common latent geometry and systematically exploited during joint representation evolution?**

To address this question, we propose the Hamiltonian-Structured Neural Tensor Factorization (HSNTF), a structured latent dynamics model with context-conditioned representation refinement. Element-wisely, HSNTF assembles the corresponding mode-specific representations into joint latent coordinates q and augments them with auxiliary momentum coordinates p. Together, they form the joint state $z = (q, p)$. Rather than learning mode-wise updates as independent responses, a **Hamiltonian Singular Field (HSF)** is designed to derive them from a shared scalar-valued Hamiltonian function $H_\theta(z)$, which induces the joint vector field $f_\theta(z) = J\nabla_z H_\theta(z)$ (Greydanus et al., 2019; Toth et al., 2020; Zhong et al., 2020; Sanchez-Gonzalez et al., 2019). This construction places all mode-specific representations within a common latent field, allowing their dynamics to be coupled through a single geometric structure.

The shared Hamiltonian serves as a common generator of all mode-wise updates. Specifically, the field Jacobian satisfies $Df_\theta(z) = J\nabla^2 H_\theta(z)$, whose cross-mode blocks quantify how perturbations to one mode alter the evolution of another. Consequently, the mixed curvature of H_θ encodes local cross-mode sensitivities and couples directional responses through the geometry of a single latent field. We therefore use Hamiltonian structure as a geometric inductive bias for organizing representation interactions, rather than as a literal physical assumption (Battaglia et al., 2018; Bronstein et al., 2021).

Our main contributions are as follows:

- **Context-conditioned dynamic representation refinement.** We formulate the element-wise mode-specific representations as a joint dynamical state. This formulation allows each entity embedding to adapt to its current cross-mode context instead of being retrieved unchanged from a global embedding table.
- **Geometric organization of cross-mode dynamics.** We introduce the Hamiltonian Singular Field (HSF), which derives all mode-wise updates from a single scalar-valued function. Its mixed curvature captures local cross-mode sensitivities, coupling the updates through a common latent geometry rather than independent response functions.
- **Inductive Bias analysis and validation of the Hamiltonian.** We analyze how the geometric structure supports cross-mode information transfer and identify the conditions under which it is beneficial. Controlled experiments examine how structural mismatch, the amount of supervision, and noise affect its gains and limitations.

2 Related Work

2.1 Tensor Representation Learning and Dynamic Tensor Models

Classical tensor factorization models multiway observations through structured low-rank factors across modes. At the formulation level, CP reconstructs the tensor by summing rank-one outer products, whereas Tucker uses a core tensor to coordinate interactions among mode-wise factors (Tucker, 1966; Kolda and Bader, 2009). (Tucker, 1966; Kolda and Bader, 2009). Recent neural tensor models retain the basic factorization framework but introduce learned nonlinear components into the interaction mechanism. For example, ANTucF learns latent interactions through an auto-encoding formulation, while NeAT and 3DA-NTC introduce neural modules to model richer dependencies among the participating factors (Tang and Luo, 2025; Tang et al., 2025b; Ahn et al., 2024; Kang et al., 2024a). These studies form a line of work centered on learning richer interactions among mode-wise factors.

Another line of research considers how latent factors evolve (Rubanova et al., 2019; Wang and Zhe, 2022; Wang et al., 2023). NONFAT introduces dynamics along the temporal dimension by modeling entity representations as continuous functions of time, rather than explicitly modeling how different modes influence one another during representation evolution (Wang and Zhe, 2022).

DEMOTE instead incorporates explicit structural relations through a multipartite graph and injects the graph information into the ODE-based evolution, allowing relational information to propagate across entities (Wang et al., 2023). *Our proposed HSF adopts a unified evolution architecture in which the selected factors of tensor element are organized as a single jointly evolving state. Their updates are derived from a shared scalar generator, so cross-mode influence and representation evolution are determined by the same latent field.*

2.2 Hamiltonian Neural Dynamics

Hamiltonian dynamics provide a fundamental formulation of motion in classical physics (Arnold, 1989a; Hairer et al., 2006). Neural Hamiltonian models incorporate physical structure into learned dynamics, providing an inductive prior for systems governed by conservation laws such as energy conservation (Greydanus et al., 2019; Finzi et al., 2020; Chen et al., 2020; Jin et al., 2020). Hamiltonian-based models have also exploited this structured dynamical organization to construct hierarchical and multi-scale evolution, enabling information to be transferred across different spatial or temporal levels while maintaining coherent dynamics (Miller et al., 2020; Mou et al., 2026). *Our proposed HSF uses the Hamiltonian architecture not as a prior for energy conservation or hierarchical structure, but as a mechanism for organizing interactions throughout representation evolution.*

3 Hamiltonian-Structured Neural Tensor Factorization

3.1 Overview

For each element of a third-order tensor indexed by (i, j, τ), the embedding layer first retrieves the mode-wise embeddings e_i^u, e_j^s, and e_τ^t. Standard factorization models combine these representations through an interaction function to approximate the element value (Tucker, 1966; Kolda and Bader, 2009). In HSNTF, we refine the embeddings jointly before prediction so that each mode representation can depend on the other modes in the queried element. Figure 1 depicts its overall architecture which consists of three stages. First, the *Hamiltonian Singular Field (HSF)* places the selected embeddings in a joint latent state and generates their updates through a shared scalar function. Second, *multi-step dynamical refinement* repeatedly evaluates this field at updated latent states, allowing the representations to undergo successive context-dependent adjustment. Finally, a *representation projection* maps the evolved state back to mode-specific factors for downstream prediction.

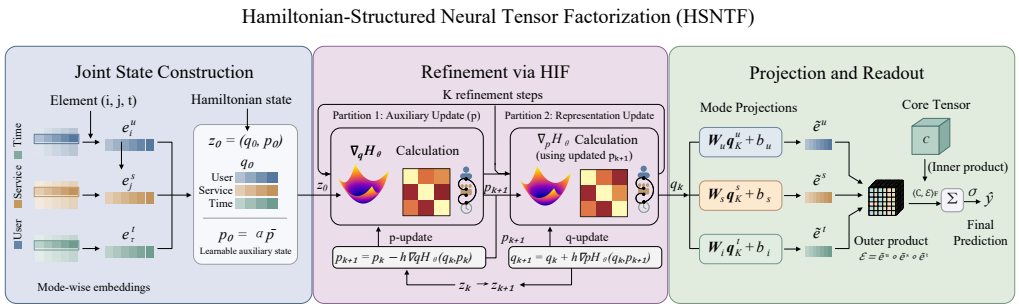


Figure 1: **Overall architecture of HSNTF.**

3.2 Joint Position–Auxiliary State Construction

For each tensor mode, we maintain a learnable embedding table $E^u \in \mathbb{R}^{I_u \times d_u}$, $E^s \in \mathbb{R}^{I_s \times d_s}$, $E^t \in \mathbb{R}^{I_t \times d_t}$, where I_u, I_s, and I_t denote the numbers of entities in the three modes. Given a queried tensor entry (i, j, τ), the corresponding indices are first mapped to mode-specific embeddings,

$$e_i^u = E^u[i, :], \qquad e_j^s = E^s[j, :], \qquad e_\tau^t = E^t[\tau, :]. \tag{1}$$

These retrieved representations are concatenated to form the initial position coordinate

$$q_0 = [e_i^u; e_j^s; e_\tau^t] \in \mathbb{R}^d, \qquad d = d_u + d_s + d_t. \tag{2}$$

Accordingly, we define $q_0 = [q_0^u; q_0^s; q_0^t]$ for the ease of presentation, where each block corresponds to one tensor mode.

We augment the position representation with an auxiliary coordinate of the same dimension,

$$p_0 = \alpha \bar{p}, \qquad \bar{p} = [\bar{p}^u; \bar{p}^s; \bar{p}^t] \in \mathbb{R}^d, \tag{3}$$

where $\bar{p}$ is a global learnable auxiliary vector and α is a learnable scalar controlling its initial scale. The queried tuple is therefore represented by the joint state

$$z_0 = (q_0, p_0) \in \mathbb{R}^{2d}. \tag{4}$$

This construction places the retrieved mode embeddings and their auxiliary coordinates in a common joint state before any cross-mode refinement is performed.

3.3 Refinement via the Hamiltonian Singular Field

Starting from the joint state $z = (q, p)$, we introduce a shared scalar generator $H_\theta : \mathbb{R}^{2d} \to \mathbb{R}$. Following the Hamiltonian neural parameterization (Greydanus et al., 2019), the generator induces the joint representation field

$$f_\theta(z) = J \nabla H_\theta(z), \qquad J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}. \tag{5}$$

Equivalently, its continuous-time dynamics are

$$\dot{q} = \nabla_p H_\theta(q, p), \qquad \dot{p} = -\nabla_q H_\theta(q, p). \tag{6}$$

Importantly, the different mode coordinates are not assigned independent update functions. Instead, all directional responses are generated through the same scalar field H_θ, so the evolution of one mode can depend on the current joint configuration of the others.

Local cross-mode interaction structure. Assuming that H_θ is twice continuously differentiable, the local sensitivity of the field is characterized by

$$A_\theta(z) = D f_\theta(z) = J \nabla^2 H_\theta(z). \tag{7}$$

Let $z^a = (q^a, p^a)$ denote the state block associated with mode a, and let J_a be the canonical symplectic matrix restricted to that block. For source mode b and target mode a, the corresponding cross-mode Jacobian block is

$$A_{a \leftarrow b}(z) = J_a \nabla^2_{z^a z^b} H_\theta(z). \tag{8}$$

This operator maps a local perturbation of mode b to the first-order change in the instantaneous update of mode a. Hence, the mixed Hessian blocks of the shared generator characterize local directional cross-mode sensitivities.

Because opposite directional responses originate from the same scalar generator, Hessian symmetry further yields

$$A_{b \leftarrow a}(z) = J_b A_{a \leftarrow b}(z)^\top J_a. \tag{9}$$

Thus, the responses $a \leftarrow b$ and $b \leftarrow a$ need not be identical, but they remain mutually compatible with the same generating structure.

Multi-step representation refinement. The field above specifies the instantaneous evolution of the joint representation. Motivated by partitioned integration of Hamiltonian systems (Hairer et al., 2006), we realize this evolution through a short sequence of sequential updates,

$$p_{k+1} = p_k - h \nabla_q H_\theta(q_k, p_k), \tag{10}$$

$$q_{k+1} = q_k + h \nabla_p H_\theta(q_k, p_{k+1}), \tag{11}$$

where h denotes the refinement step size.

For a separable Hamiltonian $H(q, p) = V(q) + T(p)$, this update reduces to symplectic Euler (Hairer et al., 2006). For the general coupled H_θ used here, we employ it as a first-order partitioned refinement map without assuming that the discrete update is exactly symplectic or energy preserving.

Each refinement step first updates the auxiliary coordinates using the current joint state (q_k, p_k). The same generator is then re-evaluated at the intermediate state (q_k, p_{k+1}) to determine the update of the representation coordinates. Consequently, the update of each mode is conditioned on the intermediate joint configuration rather than on a fixed interaction pattern.

The sequential structure can also mediate interaction through an intermediate mode within one complete refinement step. For example, a perturbation in mode a may first affect the auxiliary coordinate of mode b through Eq. (10), after which the updated p^b_{k+1} can influence the representation coordinate of mode c through Eq. (11), as in the process of $z^a_k \longrightarrow p^b_{k+1} \longrightarrow q^c_{k+1}$. This refinement therefore supports both direct cross-mode response and intermediate-state-mediated interaction pathways.

Repeating the update for K steps gives

$$z_K = \Phi_h^{(K)}(z_0), \tag{12}$$

allowing the queried mode representations to undergo successive state-dependent interaction before downstream reconstruction. Detailed derivations of the interaction operator and the corresponding structural properties are provided in the Appendix.

3.4 Mode-Specific Factor Mapping and Tucker Readout

After K refinement steps, the position component $q_K \in \mathbb{R}^d$ represents the jointly refined state of the queried tuple. Rather than reading each mode from an isolated coordinate block, we use three mode-specific affine projections of the same joint representation:

$$\tilde{e}^a = W_a q_K + b_a, \qquad a \in \{u, s, t\}, \tag{13}$$

where $W_a \in \mathbb{R}^{r_a \times d}$ and $b_a \in \mathbb{R}^{r_a}$. Each projection maps the jointly refined state into the corresponding mode-specific representation space.

For tensor completion, the mapped factors are combined through a learnable Tucker core (Tucker, 1966; Kolda and Bader, 2009). The predicted tensor entry is

$$\hat{y}_{ij\tau} = \sigma\big(\mathcal{C} \times_1 \tilde{e}^u \times_2 \tilde{e}^s \times_3 \tilde{e}^t\big), \tag{14}$$

where $\mathcal{C} \in \mathbb{R}^{r_u \times r_s \times r_t}$ is the learnable Tucker core and $\times_a$ denotes contraction along mode a.

In this way, HSF determines how the queried mode representations are jointly refined, while the mode-specific mappings and Tucker readout translate the resulting representations into the final completion prediction.

4 Experiments

Our experiments evaluate HSNTF from three complementary perspectives: predictive effectiveness, interaction mechanism, and applicability boundary. In the following experiments, we examine whether joint dynamic refinement improves tensor completion, whether the learned Hamiltonian structure governs cross-mode influence and its propagation, and under what data conditions this structured interaction is beneficial.

4.1 Tensor Completion Evaluation

Setup. We evaluate HSNTF on five real-world tensor datasets spanning dense spatiotemporal, sparse relational, and heterogeneous spatial settings, and compare against CP, 3DA-NTC, NeAT, and DEMOTE (Kolda and Bader, 2009; Kang et al., 2024b; Ahn et al., 2024; Wang et al., 2023). We report MAE, MRE, and RMSE using identical data partitions and evaluation protocols for all methods. Dataset statistics, preprocessing, model configurations, optimization, and repeated-run settings are provided in Appendix I–J.2.

Results. Table 1 shows that HSNTF ranks first in 20 of the 21 dataset–metric comparisons, including all seven MAE and RMSE cases and six of seven MRE cases. Relative to the strongest competing baseline in each evaluation setting, HSNTF reduces MAE, MRE, and RMSE by an average of 9.57%,

Table 1: Performance comparison on five datasets and seven splits. PCTemp-1/2 and Taxi-1/2 denote two evaluation splits. Values are mean ± standard deviation; lower is better. The best and second-best means in each row are shown in **bold** and underlined, respectively.

Dataset	Metric	CP	3D	NeAT	DEMOTE	HSNTF
PCTemp-1	MAE	1.021±0.005	0.580±0.121	0.521±0.010	0.411±0.008	**0.305±0.010**
	MRE	0.0413±0.001	0.0231±0.005	0.0207±0.001	0.0164±0.001	**0.0121±0.001**
	RMSE	1.327±0.002	0.764±0.163	0.679±0.014	0.536±0.010	**0.397±0.012**
PCTemp-2	MAE	1.015±0.005	0.569±0.086	0.453±0.028	0.387±0.009	**0.300±0.008**
	MRE	0.0412±0.001	0.0227±0.003	0.0171±0.001	0.0154±0.001	**0.0119±0.001**
	RMSE	1.322±0.003	0.743±0.115	0.588±0.037	0.502±0.012	**0.390±0.009**
Taxi-1	MAE	4.255±0.048	3.279±0.123	3.317±0.034	3.534±0.068	**2.990±0.083**
	MRE	0.505±0.004	0.435±0.026	0.417±0.006	0.448±0.007	**0.396±0.033**
	RMSE	8.667±0.270	5.986±0.340	6.138±0.074	6.719±0.149	**5.415±0.162**
Taxi-2	MAE	4.255±0.044	3.149±0.124	3.176±0.050	2.965±0.011	**2.871±0.073**
	MRE	0.498±0.003	0.416±0.026	0.413±0.005	0.405±0.002	**0.386±0.034**
	RMSE	8.871±0.196	5.734±0.240	5.785±0.158	5.397±0.029	**5.086±0.262**
Network	MAE	10.615±0.452	4.285±0.105	4.476±0.061	4.430±0.026	**4.226±0.026**
	MRE	4.492±0.102	0.909±0.195	0.995±0.093	0.834±0.049	**0.691±0.114**
	RMSE	83.618±15.667	30.440±0.205	30.433±0.078	30.430±0.142	**30.428±0.063**
QoS	MAE	0.671±0.006	0.657±0.002	0.665±0.002	0.646±0.005	**0.639±0.008**
	MRE	1.328±0.011	1.184±0.015	1.167±0.015	**1.160±0.026**	1.229±0.010
	RMSE	1.864±0.016	1.856±0.023	1.847±0.020	1.827±0.016	**1.804±0.026**
Soil	MAE	1.004±0.093	0.626±0.048	0.608±0.029	1.016±0.021	**0.582±0.030**
	MRE	0.651±0.169	0.326±0.031	0.289±0.008	0.489±0.018	**0.285±0.015**
	RMSE	1.748±0.136	1.144±0.077	1.110±0.045	1.679±0.029	**1.060±0.039**
MAE wins / 7 ↑		0/7	0/7	0/7	0/7	**7/7**
MRE wins / 7 ↑		0/7	0/7	0/7	**1/7**	**6/7**
RMSE wins / 7 ↑		0/7	0/7	0/7	0/7	**7/7**
Average rank ↓		4.95	3.29	3.00	2.62	**1.14**

10.18%, and 9.90%, respectively. The only exception is QoS MRE. These results support the value of HSNTF representation refinement before Tucker reconstruction. The only weaker result on QoS MRE further motivates us to examine the HSF mechanism underlying these gains and the conditions under which its structured interaction is beneficial.

4.2 Structural Ablation of Joint Cross-Mode Evolution

We first examine whether the predictive gain of HSNTF comes from joint cross-mode evolution. Using the same evaluation protocol as in Sec. 4.1, We compare three nested variants on five datasets: Static Tucker removes latent evolution, Separable HSF retains mode-wise dynamics without cross-mode coupling, and Full HSF jointly evolves the queried mode representations. All variants share the same reconstruction and evaluation protocol; detailed definitions are given in Appendix I.3.

As shown in Table 2, Full HSF achieves the lowest RMSE on all five datasets. Introducing dynamics alone, however, does not yield a consistent improvement. Separable HSF substantially improves Network and Soil, provides only marginal gains on PCTemp-1 and QoS, and even underperforms Static Tucker on Taxi-1.This contrast indicates that the benefit of HSF does not simply come from allowing each mode representation to evolve independently under the reconstruction objective.

Table 2: **Structural ablation in RMSE.** I.1/I.2 denote the relative improvement of Full HSF over Static Tucker/Separable HSF.

Dataset	Static	Sep.	Full	I.1	I.2
PCTemp-1	1.138	1.135	**0.397**	**65.11%**	**65.08%**
Taxi-1	8.106	8.271	**5.415**	**33.20%**	**34.53%**
Network	72.324	30.570	**30.329**	**58.07%**	**0.79%**
QoS	1.870	1.845	**1.804**	**3.53%**	**2.22%**
Soil	1.596	1.174	**1.060**	**33.58%**	**9.71%**

Instead, the gain arises when the representations associated with a queried tuple are evolved as a joint state, so that the update of each mode can depend on and be influenced by the states of the others. Full HSF therefore turns representation evolution into a mechanism for modeling cross-mode interaction rather than independent mode-wise adaptation.

From Eqs. (7) and (8), the mixed curvature of the shared generator predicts local cross-mode responses. We test this relation using a fixed Full HSF checkpoint on Soil: for each of the six directed mode pairs, we perturb the source representation and compare the measured target response with that predicted by the corresponding mixed-curvature operator. The first-order derivation is given in Appendix E.

As shown in Figure 2, all six directed mode pairs, the measured-to-predicted response ratios remain close to one(Figure 2). A ratio of one indicates near-perfect agreement between the response predicted by the mixed-Hessian operator and the response measured from the actual perturbation, indicating that the mixed-curvature operator accurately captures the local cross-mode sensitivities realized by the trained field. As established by Eqs. (6) and (8), the first-order derivatives of H_θ generate the joint representation evolution, while its mixed second-order derivatives characterize the local cross-mode influence embedded in that evolution.

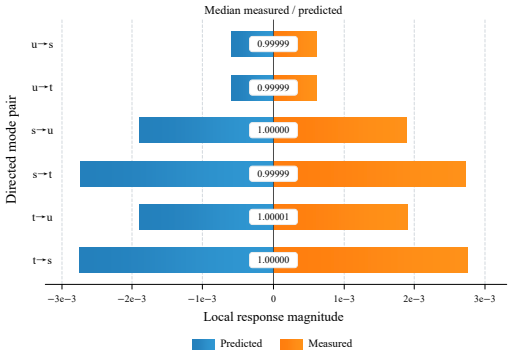


Figure 2: Mixed-curvature local response prediction.

4.3 How Does Cross-Mode Influence Propagate across Refinement Steps?

The sequential update in Eqs. (10)–(11) allows a source-mode change to modify the auxiliary state before the representation update is computed. We therefore ask whether this dynamic intermediate state enables cross-mode influence to propagate more efficiently within a refinement step and whether the resulting effect persists across multiple steps.

Intervention design. Following Sec. 4.2, we use the same fixed Full HSF checkpoint on Soil, perturb one source-mode representation, and measure the resulting target response over $K = 1, 2, 3$ refinement steps. We evaluate all six directed source–mediator–target paths shown in Figure 3. Full HSF retains the complete p–q evolution, whereas Q-only removes the evolution of the auxiliary state and updates only q. Both conditions reuse the same trained parameters, queried tuples, perturbations, initial states, and evaluation protocol. Appendix F explains how this intervention probes the selected mediated route and why its earliest possible appearance differs between Full HSF and Q-only.

For a directed path $b \to m \to a$, let $\Delta_a^B(K) = q_a^{B,\mathrm{pert}}(K) - q_a^{B,\mathrm{ref}}(K)$, $B \in \{\mathrm{Full}, \mathrm{Q\text{-}only}\}$. We measure the contribution of the selected mediated path by

$$\Gamma^B_{b\to m\to a}(K) = \frac{\left\| \Delta_a^B(K) - \Delta_a^{B,\mathrm{block}(m)}(K) \right\|_2}{\varepsilon}, \tag{15}$$

where $\Delta_a^{B,\mathrm{block}(m)}$ is obtained from a matched counterfactual rollout in which the selected mediator path is blocked, and ε is the source perturbation magnitude. Thus, Γ measures the target response attributable specifically to the selected source–mediator–target route.

Dynamic auxiliary state enables earlier and stronger propagation. Figure 3 shows a consistent distinction between the two rollout conditions. At $K = 1$, Full HSF already produces a nonzero mediated response on all six directed paths, whereas the corresponding Q-only response is zero. This follows directly from the partitioned update: a source perturbation can first alter the intermediate p state and then influence the target q update within the same refinement step. Without auxiliary-state evolution, the corresponding two-stage route cannot be completed until information carried by q reaches a subsequent step.

At $K = 2$ and $K = 3$, Q-only develops nonzero path responses, confirming that repeated q updates can also propagate information across states. Full HSF nevertheless remains stronger on every directed path. The difference therefore reflects more than an additional latent coordinate: the evolving p state provides an intermediate channel that makes mediated influence available within one step and strengthens its composition over successive refinements. These results identify the dynamic auxiliary state as an active component of cross-mode representation propagation rather than a passive extension of the latent state.

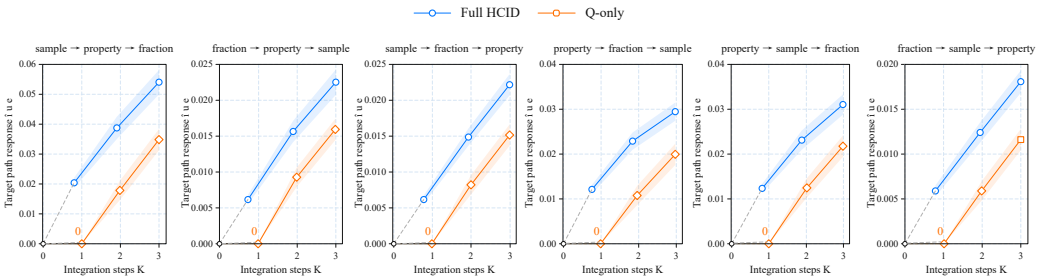


Figure 3: Cross-mode interaction propagation across six directed paths.

4.4 When Does the Shared Hamiltonian Field Help?

The preceding experiments show that modeling cross-mode influence is important, but such coupling can also be learned by an unconstrained vector field. We therefore ask when the *shared Hamiltonian organization* itself is beneficial. As established in Eqs. (8)–(9), reciprocal directional responses are coupled through the same mixed curvature structure, suggesting that supervision from one direction may inform its reciprocal counterpart.

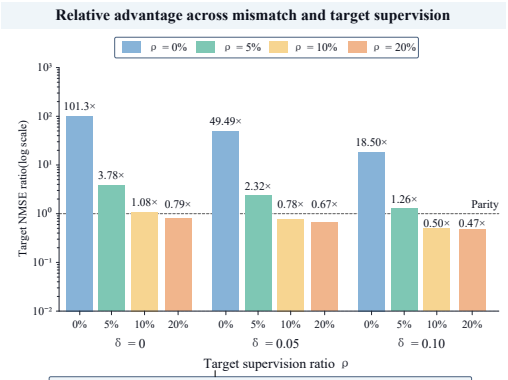
Controlled reciprocal-transfer benchmark. To isolate this effect from embedding and reconstruction, we construct an analytic nonlinear teacher with controllable Hamiltonian mismatch:

$$H_0(z) = \tfrac{1}{2} z^\top Q z + \sum_r c_r \cos(w_r^\top z), \qquad f_\delta(z) = \sqrt{1-\delta^2}\, \bar{f}_{\mathrm{H}}(z) + \delta\, \bar{f}_\perp(z). \tag{16}$$

where $f_{\mathrm{H}} = J\nabla H_0$, $Q = Q^\top$, and $f_\perp(z) = JKz$ with $K = -K^\top$. The compatible component preserves the reciprocal relation $A_{a\leftarrow b} = J_a H_{ab}$ and $A_{b\leftarrow a} = J_b H_{ab}^\top$, whereas increasing δ progressively violates this shared-block structure.

We compare HSF with an unconstrained MLP under identical directional observations. We first use reciprocal-direction supervision only to test recovery of an unseen target direction, and then progressively add direct target observations to test whether HSF can adapt under structural mismatch. Full construction and training details are given in Appendix J.4.

Figure 4 shows that HSF can recover an unobserved target from reciprocal-direction supervision. At $\rho = 0$, the Vector-field MLP has $101.3\times$, $49.49\times$, and $18.50\times$ higher target error than HSF for $\delta = 0$, 0.05, and 0.10, respectively. HSF also improves when direct target observations are added; at $\delta = 0.10$, its NMSE decreases from 0.06052 to 0.01831 as ρ increases from 0 to 20%.



Target NMSE under increasing direct target supervision

δ	Model	0%	5%	10%	20%
δ = 0	HSIF	**0.01085**	**0.01092**	**0.01090**	0.01081
	Vector-field MLP	1.09893	0.04130	0.01173	**0.00855**
δ = 0.05	HSIF	**0.02266**	**0.01820**	0.01533	0.01294
	Vector-field MLP	1.12143	0.04222	**0.01196**	**0.00861**
δ = 0.10	HSIF	**0.06052**	**0.03264**	0.02354	0.01831
	Vector-field MLP	1.11984	0.04104	**0.01174**	**0.00869**

Figure 4: Reciprocal transfer and target adaptation.

The advantage is therefore strongest when target supervision is scarce and decreases as direct target evidence becomes sufficient for the unrestricted MLP. These results show that the shared Hamiltonian structure supports reciprocal information transfer while still allowing the learned field to adapt to target-specific observations.

Statistical boundary under sparsity and noise. The reciprocal-transfer experiment shows that the shared Hamiltonian field enables information transfer across directions. It also reduces effective estimation freedom by generating all vector components from one scalar function rather than fitting

them independently. We therefore ask when this reduction is beneficial under limited and noisy observations. To characterize when the shared Hamiltonian constraint is statistically beneficial, we

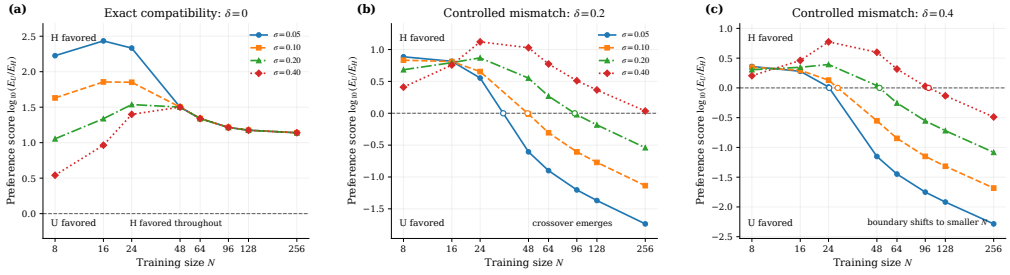


Figure 5: **Statistical boundary of the shared Hamiltonian field.**

generate a nonlinear teacher field $g_\delta = g_{\mathrm{H}} + \delta g_\perp$, where δ controls structural mismatch, and vary the number of observations N and the Gaussian noise level σ. We compare Hamiltonian-constrained and unrestricted field estimators on clean test fields using $S = -\log_{10} \overline{E_{\mathrm{H}}/E_{\mathrm{U}}}$, where $S > 0$ favors the Hamiltonian estimator; the complete teacher construction, normalization, sampling, and estimation protocol are given in Appendix J.4.

Figure 5 reveals a clear transition. When the teacher is Hamiltonian-compatible, the structured estimator retains an advantage across the tested sample sizes; under increasing mismatch, this advantage persists primarily when supervision is sparse or noisy, while sufficient observations eventually favor the unrestricted field. Higher noise shifts this crossover toward larger N, whereas stronger mismatch shifts it toward smaller N. Together with the reciprocal-transfer experiment, these results show that the shared Hamiltonian field is most useful when directional interactions remain sufficiently compatible and supervision is limited, while strong, well-observed incompatible structure eventually makes the constraint overly restrictive.

5 Conclusion and Future Work

Conclusion. This work asks how cross-mode influences should be organized during dynamic representation evolution. We propose the **Hamiltonian-Structured Neural Tensor Factorization (HSNTF)**, whose core refinement module is the **Hamiltonian Singular Field (HSF)**. HSF generates mode-wise updates from a shared scalar function. Its first-order derivatives define joint dynamics, while mixed curvature characterizes local cross-mode influence and links reciprocal sensitivities through a common generating structure. Multi-step refinement then propagates these interactions to produce query-specific representations. Experiments on five real-world tensor datasets show the predictive value of HSNTF. Structural ablations, local perturbations, and rollout interventions show that effective refinement requires joint cross-mode evolution, whose local influence is captured by mixed curvature and further propagated through the dynamic auxiliary state. Controlled experiments further show that shared Hamiltonian organization supports information transfer under sparse or noisy supervision when the underlying interactions remain sufficiently compatible. Together, these results establish HSNTF as a tensor-factorization architecture in which HSF jointly organizes representation dynamics and mutual cross-mode influence.

Limitations and future work. The main limitation of HSF is that generating directional interactions through a common scalar field restricts the class of vector fields it can represent. Under strong structural mismatch and sufficient observations, this constraint can introduce approximation error relative to an unrestricted field. Future work can therefore develop adaptive structured interaction fields that combine a Hamiltonian-compatible component with a learnable residual interaction field, allowing the strength of shared organization to vary with the data. Another promising direction is to incorporate contrastive objectives into the representation refinement process, encouraging related cross-mode contexts to induce consistent latent responses while separating incompatible interactions. Such objectives could exploit reciprocal or perturbation-induced responses as additional supervision, providing a complementary signal when direct observations of a target direction are limited. Direction-aware data augmentation may further expose the model to diverse cross-mode interaction patterns under sparse supervision.

AI USE STATEMENT

Generative AI tools were used to assist with English-language polishing. They were not used to invent scientific claims, alter experimental results, or introduce unverified technical content. In addition, They were used to assist in generating the synthetic data for the controlled reciprocal-transfer and statistical-boundary experiments. The corresponding generation code and resulting data were manually inspected and validated against the intended experimental assumptions and design requirements; the complete data-generation mechanisms are documented in Appendix J.4. All mathematical statements, experimental procedures, reported results, and conclusions were reviewed and verified by the authors, who take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

The information required to reproduce the reported experiments is specified in the main paper and Appendix. Dataset sources and tensor construction are documented in Appendix I.1, while data partitions, training-only preprocessing, inverse transformation, and evaluation procedures are given in Appendix J.1. More details on the HSNTF architecture and HSF refinement procedure are further specified in Appendix I.3. Model hyperparameters and optimization details are reported in Appendix C; random seeds, repeated-run settings, checkpoint selection, and shared comparison rules are documented in Appendix J.2. The reported MAE, MRE, and RMSE metrics are described in Sec. 4.1 and Appendix J.1. Baseline definitions and the protocols for the mechanism and controlled synthetic experiments are provided in Appendices I.2, J.3, and J.4, respectively.

REFERENCES

Dawon Ahn, Uday Singh Saini, Evangelos E. Papalexakis, and Ali Payani. Neural additive tensor decomposition for sparse tensors. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 14–23, 2024.

V. I. Arnold. *Mathematical Methods of Classical Mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer New York, 2 edition, 1989a.

Vladimir I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer, 2 edition, 1989b.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible research is reproducible. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 725–734. PMLR, 2019.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Ana Cannas da Silva. *Lectures on Symplectic Geometry*, volume 1764 of *Lecture Notes in Mathematics*. Springer, 2001.

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations*, 2020.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2185–2194. Association for Computational Linguistics, 2019.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Syeda Ferdous. Soil POM & MAOM data, 2026. Version 1.

Marc Finzi, Ke Alexander Wang, and Andrew Gordon Wilson. Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2 edition, 2008.

Nate Gruver, Marc Finzi, Samuel Stanton, and Andrew Gordon Wilson. Deconstructing the inductive biases of hamiltonian neural networks. In *International Conference on Learning Representations*, 2022.

Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2 edition, 2006.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002.

Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

Xu Kang, Zhiyang Jia, Jia Jia, and Jiadong Ren. 3DA-NTC: 3D channel attention aided neural tensor completion for crowdsensing data inference. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024a.

Xu Kang, Zhiyang Jia, Jia Jia, and Jiadong Ren. 3da-ntc: 3d channel attention aided neural tensor completion for crowdsensing data inference. In *2024 International Joint Conference on Neural Networks*, pages 1–8, 2024b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, 2009.

John M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, 2 edition, 2013.

Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2004.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? a large-scale study. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Xin Luo, Hao Wu, Zhi Wang, Jianjun Wang, and Deyu Meng. A novel approach to large-scale dynamically weighted directed network representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9756–9773, 2022.

Jerrold E. Marsden and Tudor S. Ratiu. *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer, 2 edition, 1999.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

Michael I. Miller, Daniel J. Tward, and Alain Trouvé. Coarse-to-fine hamiltonian dynamics of hierarchical flows in computational anatomy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3760–3765, 2020.

Max D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

Changhong Mou, Dinghua Xu, Xiyue Zuo, Keji Liu, and Yeyu Zhang. ATLAS-NN: Adaptive transfer learnable symplectic-aware neural network for long-time hamiltonian dynamics. *arXiv preprint arXiv:2606.04447*, 2026.

New York City Taxi and Limousine Commission. TLC trip record data. Accessed: September 20, 2026.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994.

Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics, 2017.

Richard W. Reynolds, Thomas M. Smith, Chunying Liu, Dudley B. Chelton, Kenneth S. Casey, and Michael G. Schlax. Daily high-resolution-blended analyses for sea surface temperature. *Journal of Climate*, 20(22):5473–5496, 2007.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 3 edition, 1976.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian Graph Networks with ODE Integrators. *arXiv preprint arXiv:1909.12790*, 2019.

J. M. Sanz-Serna. Symplectic integrators for hamiltonian problems: An overview. *Acta Numerica*, 1: 243–286, 1992.

Peng Tang and Xin Luo. Neural Tucker Factorization. *IEEE/CAA Journal of Automatica Sinica*, 12 (2):475–477, 2025.

Peng Tang, Xin Luo, and J. C. P. Woodcock. Auto-encoding neural tucker factorization. *IEEE Transactions on Knowledge and Data Engineering*, 37(10):5795–5807, 2025a.

Peng Tang, Xin Luo, and Jim Woodcock. Auto-encoding neural tucker factorization. *IEEE Transactions on Knowledge and Data Engineering*, 37(10):5795–5807, 2025b.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian Generative Networks. In *International Conference on Learning Representations*, 2020.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(91), 2006.

Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge University Press, 2019.

Zheng Wang and Shandian Zhe. Nonparametric Factor Trajectory Learning for Dynamic Tensor Decomposition. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23459–23469. PMLR, 2022.

Zheng Wang, Shikai Fang, Shibo Li, and Shandian Zhe. Dynamic tensor decomposition via neural diffusion-reaction processes. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yilei Zhang, Zibin Zheng, and Michael R. Lyu. WSPred: A time-aware personalized QoS prediction framework for web services. In *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering*, pages 210–219, 2011.

Zibin Zheng, Yilei Zhang, and Michael R. Lyu. Investigating QoS of real-world web services. *IEEE Transactions on Services Computing*, 7(1):32–39, 2014.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 1218–1229. PMLR, 2021.

A UNIFIED SETUP AND NOTATION

Table 3 summarizes the notation used throughout the theoretical analysis. We first review the Hamiltonian dynamical formulation Arnold (1989b); Marsden and Ratiu (1999); Sanz-Serna (1992); Leimkuhler and Reich (2004) and characterize how a joint state evolves through coupled dynamics generated by a shared scalar function. Motivated by this formulation, we apply the same construction to the dynamic refinement of tensor embeddings, where the retrieved mode representations are assembled into a joint latent state and evolved jointly before reconstruction. The remainder of the appendix then develops the structural properties of this interaction field and provides the theoretical analysis underlying the perturbation, mediated-transfer, and compatibility experiments in the main paper.

Table 3: **Notation used in the theoretical analysis.**

Symbol	Definition
$q^{(m)}$	Representation associated with mode m, $q^{(m)} \in \mathbb{R}^{d_m}$.
q	Joint representation $\mathrm{col}(q^{(1)}, \dots, q^{(M)}) \in \mathbb{R}^d$.
p	Single auxiliary coordinate for the joint representation, $p \in \mathbb{R}^d$.
z	Joint latent state $[q^\top, p^\top]^\top \in \mathbb{R}^{2d}$.
J	Canonical matrix $\left[\begin{smallmatrix} 0 & I_d \\ -I_d & 0 \end{smallmatrix}\right]$, with $J^\top = -J$ and $J^{-1} = -J$.
H_θ	Shared scalar generator $H_\theta : \Omega \subset \mathbb{R}^{2d} \to \mathbb{R}$.
f_θ	Joint field $f_\theta(z) = J\nabla H_\theta(z)$.
$\mathcal{H}_{ab}$	Mixed curvature block associated with modes a and b.
$\mathcal{C}_{a \leftarrow b}$	Directed local interaction from mode b to mode a.

We next explain how the quantities summarized in Table 3 enter the mathematical formulation below. For each queried tensor entry, the retrieved mode embeddings $q^{(m)} \in \mathbb{R}^{d_m}$ are first concatenated into the joint representation

$$q = \mathrm{col}(q^{(1)}, \dots, q^{(M)}) \in \mathbb{R}^d, \qquad d = \sum_{m=1}^{M} d_m.$$

A single auxiliary vector $p \in \mathbb{R}^d$ is then introduced for the joint representation, giving the latent state

$$z = [q^\top, p^\top]^\top \in \mathbb{R}^{2d}.$$

Thus, the mode-specific quantities are the retrieved representation blocks $q^{(m)}$, whereas p is one additional joint coordinate of the same total dimension as q.

Following the canonical Hamiltonian formulation Arnold (1989b); Marsden and Ratiu (1999), the time derivative of the state is obtained by applying the canonical structure matrix J to the gradient of the scalar Hamiltonian, $\dot{z} = J\nabla H(z)$. Accordingly, for $z = [q^\top, p^\top]^\top$, we use

$$J = \begin{bmatrix} 0 & I_d \\ -I_d & 0 \end{bmatrix}, \qquad J^\top = -J, \qquad J^{-1} = -J.$$

A shared scalar function $H_\theta : \Omega \subset \mathbb{R}^{2d} \to \mathbb{R}$ generates the joint field

$$f_\theta(z) = J\nabla H_\theta(z) = \begin{bmatrix} \nabla_p H_\theta(q,p) \\ -\nabla_q H_\theta(q,p) \end{bmatrix}. \tag{17}$$

We assume $H_\theta \in C^3(\Omega)$, meaning that its partial derivatives up to third order exist and are continuous on the open domain Ω; this regularity supports the Hessian and local Taylor arguments used below.

To expose mode-wise interactions, we partition the single auxiliary vector p according to the same dimensions as q and form the corresponding mode-aligned blocks:

$$p = \mathrm{col}(p_{[1]}, \dots, p_{[M]}), \quad p_{[m]} \in \mathbb{R}^{d_m}, \qquad \xi^{(m)} = \begin{bmatrix} q^{(m)} \\ p_{[m]} \end{bmatrix}.$$

Here $p_{[m]}$ is the coordinate block of the joint vector p aligned with mode m, rather than a separately parameterized auxiliary embedding.

Under the fixed permutation that groups the state into these mode-aligned blocks, the canonical matrix becomes $\mathrm{diag}(J_1, \ldots, J_M)$, where

$$J_m = \begin{bmatrix} 0 & I_{d_m} \\ -I_{d_m} & 0 \end{bmatrix}.$$

For modes a and b, we define the mixed-curvature block and its induced directed interaction jointly as

$$\mathcal{H}_{ab}(z) := \nabla^2_{\xi^{(a)}\xi^{(b)}} H_\theta(z), \qquad \mathcal{C}_{a\leftarrow b}(z) := J_a \mathcal{H}_{ab}(z). \tag{18}$$

By Schwarz's theorem for C^2 scalar functions Rudin (1976), the Hessian is symmetric; hence

$$\mathcal{H}_{ba}(z) = \mathcal{H}_{ab}(z)^\top.$$

Thus, $\mathcal{H}_{ab}$ captures the mixed curvature between two mode-aligned state blocks, while $\mathcal{C}_{a\leftarrow b}$ gives the corresponding local response from mode b to mode a.

Finally, the continuous field is realized through the partitioned refinement

$$p_{k+1} = p_k - h\nabla_q H_\theta(q_k, p_k), \quad q_{k+1} = q_k + h\nabla_p H_\theta(q_k, p_{k+1}), \qquad z_{k+1} = \Phi_h(z_k). \tag{19}$$

Here Φ_h denotes the implemented one-step refinement map. The structural results below concern the shared generator and its induced interaction field; properties specific to the discrete update are stated separately when needed.

A.1 Hamiltonian Dynamics

Hamiltonian dynamics describes the evolution of a conservative system in phase space through generalized coordinates q and their conjugate momenta p Arnold (1989b); Marsden and Ratiu (1999), with the system state written as $z = [q, p]$. and its evolution is governed by Hamilton's equations

$$\dot{q} = \nabla_p H(q, p), \qquad \dot{p} = -\nabla_q H(q, p), \tag{20}$$

or, equivalently,

$$\dot{z} = J\nabla H(z), \qquad J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}. \tag{21}$$

Given an initial condition $z_0 = (q_0, p_0)$, Hamilton's equations determine the subsequent trajectory in phase space. For an autonomous Hamiltonian, H remains constant along the exact continuous trajectory,

$$H(z(t)) = H(z_0), \tag{22}$$

and the resulting flow preserves the canonical symplectic structure (Sanz-Serna, 1992; Leimkuhler and Reich, 2004). These properties provide the classical geometric description of Hamiltonian evolution.

B Structural Constraints of the Shared Generator

The Hamiltonian formulation suggests that a single scalar function can govern the evolution of a joint latent state. Following this principle, our model uses the shared generator H_θ to organize the refinement of all latent coordinates (Gruver et al., 2022; Zhong et al., 2021). We next characterize the structural restriction induced by this choice.

Let $A_\theta(z) := Df_\theta(z)$ denote the local Jacobian of the field. Since $f_\theta(z) = J\nabla H_\theta(z)$, differentiation gives $A_\theta(z) = J\nabla^2 H_\theta(z)$. The symmetry of $\nabla^2 H_\theta(z)$, together with $J^\top J = I$ and $J^2 = -I$, therefore yields

$$A_\theta(z)^\top J + JA_\theta(z) = 0, \qquad A_\theta(z) \in \mathfrak{sp}(2d) := \left\{ A \in \mathbb{R}^{2d\times 2d} \,\middle|\, A^\top J + JA = 0 \right\}. \tag{23}$$

This is the standard Hamiltonian-matrix condition Cannas da Silva (2001).

Local degrees of freedom. The structural restriction in Eq. (23) can be quantified through the dimension of the admissible local operator class. An unrestricted field on $\mathbb{R}^{2d}$ may have an arbitrary Jacobian $A_U \in \mathbb{R}^{2d\times 2d}$ and therefore has

$$P_U = (2d)^2 = 4d^2$$

local degrees of freedom. By contrast, every Hamiltonian-compatible operator can be written as

$$A_H = JS, \qquad S = S^\top,$$

where $S \in \mathbb{R}^{2d\times 2d}$ is symmetric. Since multiplication by the invertible matrix J is one-to-one, A_H has the same number of free components as S, namely

$$P_H = \frac{(2d)(2d+1)}{2} = d(2d+1), \qquad P_U - P_H = d(2d-1). \tag{24}$$

This reduction does not imply that opposite directional responses are identical. Under the mode-wise block partition, symmetry of S gives $S_{ba} = S_{ab}^\top$, while the corresponding response blocks satisfy

$$A_{ab} = J_a S_{ab}, \qquad A_{ba} = J_b S_{ab}^\top = J_b A_{ab}^\top J_a. \tag{25}$$

Thus, the responses in the two directions are generally different, but they are not independent: once one cross-block response is fixed, the reverse response is constrained by the same underlying curvature block. The reduction in Eq. (24) therefore concerns the degrees of freedom of the *local response operator*, rather than the number of trainable neural-network parameters.

This restricted local response class is directly related to the sparse- and noisy-observation experiments in the main paper: when supervision is limited or corrupted, estimating fewer independent local responses can reduce estimation variance. The corresponding trade-off between this variance reduction and the bias caused by structural mismatch is analyzed formally in Section G.

C Training Objective and Parameter Optimization

Table 4 summarizes the model parameters and optimization settings of the implemented three-mode model. The parameter collection is $\Theta = \{Q^{(1)}, Q^{(2)}, Q^{(3)}, \theta, \mathcal{G}, (W_m, b_m)_{m=1}^3, \bar{p}, \alpha\}$.

Overall objective. For each observed entry with target y_i, preprocessing computes $\ell_i = \log(\max(y_i, 10^{-5}) + 10^{-8})$ and $y_i^{\mathrm{n}} = (\ell_i - \ell_{\min})/(\ell_{\max} - \ell_{\min} + 10^{-8})$. The extrema are estimated from the training split and reused for validation and testing. Given a mini-batch $\mathcal{B}$, the model minimizes the mean squared prediction error

$$\mathcal{L}_{\mathcal{B}}(\Theta) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \left(\widehat{y}_i(\Theta) - y_i^{\mathrm{n}}\right)^2. \tag{26}$$

This is the complete training objective; no auxiliary loss or additional regularization term is included.

Forward computation. For a queried triple $x_i = (i_1, i_2, i_3)$, the model retrieves $q_{i_m}^{(m)}$ from $Q^{(m)}$ and constructs $q_{0,i} = \mathrm{col}(q_{i_1}^{(1)}, q_{i_2}^{(2)}, q_{i_3}^{(3)})$. The auxiliary coordinate $p_{0,i} = \alpha\bar{p}$ is broadcast across the batch. Suppressing the sample index, the shared generator determines the K refinement steps:

$$\begin{aligned} p_{k+1} &= p_k - h\nabla_q H_\theta(q_k, p_k), \\ q_{k+1} &= q_k + h\nabla_p H_\theta(q_k, p_{k+1}), \qquad k = 0, \dots, K-1. \end{aligned} \tag{27}$$

Automatic differentiation supplies both partial derivatives (Baydin et al., 2018). The second subupdate reevaluates H_θ at (q_k, p_{k+1}), using the updated auxiliary coordinate. The parameters θ are shared across all refinement steps.

After refinement, each mode representation is computed by $\tilde{q}_i^{(m)} = W_m q_{K,i} + b_m$. Thus each readout acts on the entire joint representation, rather than selecting a coordinate slice. The prediction is obtained through the Tucker core and a logistic sigmoid:

$$\widehat{y}_i = \sigma\left(\sum_{a=1}^{d_1}\sum_{b=1}^{d_2}\sum_{c=1}^{d_3} \mathcal{G}_{abc}\, \tilde{q}_{i,a}^{(1)} \tilde{q}_{i,b}^{(2)} \tilde{q}_{i,c}^{(3)}\right). \tag{28}$$

Table 4: Model parameters and optimization configuration. Here $d = d_1 + d_2 + d_3$.

Parameter or setting	Definition or value
Parameters registered with the optimizer	
Mode embedding tables	$Q^{(m)} \in \mathbb{R}^{n_m \times d_m}$, $m = 1, 2, 3$; entries initialized uniformly on $[0, 1]$.
Scalar generator	θ: weights and biases of $H_\theta : \mathbb{R}^{2d} \to \mathbb{R}$, with two width-128 $\tanh$ hidden layers and a scalar linear output.
Auxiliary vector	$\bar{p} \in \mathbb{R}^d$, initialized to zero and shared across queries.
Auxiliary scale	$\alpha \in \mathbb{R}$, initialized to 0.1; the initial auxiliary coordinate is $p_{0,i} = \alpha\bar{p}$.
Mode-specific readouts	$W_m \in \mathbb{R}^{d_m \times d}$ and $b_m \in \mathbb{R}^{d_m}$, $m = 1, 2, 3$; standard `nn.Linear` initialization.
Tucker core	$\mathcal{G} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$; entries initialized from $\mathcal{N}(0, 0.01^2)$.
Fixed hyperparameters and training settings	
Embedding dimensions	$d_1 = d_2 = d_3 = 5$.
Refinement step size	$h = 0.1$.
Refinement depth	$K \in \{1, 2, 3\}$ for the depth comparison; $K = 1$ for repeated single-step runs.
Optimizer	Adam (Kingma and Ba, 2015).
Learning rate	5×10^{-4}.
Adam coefficients	$(\beta_1, \beta_2) = (0.9, 0.999)$; $\epsilon_{\text{Adam}} = 10^{-8}$.
Weight decay	0.
Batch size	64; training batches are shuffled.
Maximum epochs	200.
Early stopping	Stop after 21 consecutive epochs without a strict improvement in normalized validation RMSE.
Random seeds	42 for the depth comparison; 40–44 for repeated single-step runs.

Backward computation and parameter updates. The prediction loss first differentiates through the sigmoid and Tucker contraction, yielding gradients for $\mathcal{G}$ and the mode representations. Backpropagation through the affine readouts updates (W_m, b_m) and propagates the remaining gradient to $q_{K,i}$. It then traverses all K partitioned updates to reach θ, the retrieved embedding rows, and the auxiliary initialization parameters $(\bar{p}, \alpha)$. Because θ is reused at every step, its gradient accumulates contributions from all field evaluations. Likewise, repeated occurrences of an entity accumulate gradients in the same embedding row.

The internal gradient operations use `create_graph=True`, preserving the computational graph throughout the training trajectory. Differentiating the loss through ∇H_θ therefore involves second-order input derivatives and mixed input–parameter derivatives, evaluated through automatic differentiation without explicitly constructing a full Hessian (Baydin et al., 2018; Pearlmutter, 1994). The generator is optimized through its contribution to the final prediction, with no direct supervision of its scalar value. Its final additive output bias disappears under input differentiation and consequently receives no prediction-loss gradient.

For each mini-batch, accumulated gradients are cleared, Eq. (26) is backpropagated once, and a single Adam step jointly updates the parameters receiving gradients. The ranks, h, and K remain fixed. Normalized validation RMSE is evaluated after each epoch to update the best-epoch record and apply the early-stopping rule in Table 4.

D Directional Compatibility of the Shared Interaction Field

Section B established that a field generated by a single scalar H_θ has a constrained local Jacobian. We next present the structural constraint we find between the two reciprocal interaction directions.

For modes $a \neq b$, recall that

$$\mathcal{C}_{a \leftarrow b} = J_a \mathcal{H}_{ab}, \qquad \mathcal{C}_{b \leftarrow a} = J_b \mathcal{H}_{ba}.$$

By Schwarz's theorem, $\mathcal{H}_{ba} = \mathcal{H}_{ab}^\top$. The two directional responses therefore originate from the same mixed-curvature block rather than from two independently specified local operators.

Theorem 1 (Cross-directional compatibility)**.** *For any two distinct modes a and b, the directed local interaction operators induced by the shared scalar generator satisfy*

$$\boxed{\mathcal{C}_{b\leftarrow a}(z) = J_b \mathcal{C}_{a\leftarrow b}(z)^\top J_a}. \tag{29}$$

Proof. From $\mathcal{C}_{a\leftarrow b} = J_a \mathcal{H}_{ab}$ and $J_a^{-1} = -J_a$,

$$\mathcal{H}_{ab}^\top = \mathcal{C}_{a\leftarrow b}^\top J_a.$$

Using $\mathcal{H}_{ba} = \mathcal{H}_{ab}^\top$,

$$\mathcal{C}_{b\leftarrow a} = J_b \mathcal{H}_{ba} = J_b \mathcal{C}_{a\leftarrow b}^\top J_a,$$

which proves the result. □

Equation (29) does not imply equal responses in the two directions. The canonical blocks J_a and J_b generally produce different directed operators. The constraint is instead that the two directions cannot vary independently: both are determined by the same underlying mixed curvature.

D.1 When does a field admit a shared scalar generator?

The preceding result follows from the shared generator. The converse question is whether the same compatibility condition is sufficient for a general vector field to admit such a representation.

Theorem 2 (Existence of a shared scalar generator)**.** *Let $f : \Omega \to \mathbb{R}^{2d}$ be continuously differentiable on an open domain Ω. If*

$$Df(z)^\top J + JDf(z) = 0, \qquad z \in \Omega, \tag{30}$$

then every point $z_0 \in \Omega$ has a neighborhood U on which there exists $H_U \in C^2(U)$ satisfying

$$f = J\nabla H_U.$$

If Ω is additionally connected and simply connected, there exists a single $H \in C^2(\Omega)$, unique up to an additive constant, such that $f = J\nabla H$ throughout Ω.

Proof. Define $g = -Jf$. Since J is constant,

$$Dg = -JDf.$$

Equation (30) gives

$$Dg^\top = Df^\top J = -JDf = Dg.$$

Hence the differential form associated with g is closed. The local Poincaré lemma (Lee, 2013) therefore gives $g = \nabla H_U$ in a neighborhood of every point, and consequently $f = J\nabla H_U$. On a simply connected domain, these local potentials extend to a single global potential H, unique up to an additive constant. □

Thus, Eq. (30) is the local integrability condition associated with a scalar-generated field. The mode-wise relation in Eq. (29) is its corresponding cross-block consequence.

E Mixed Curvature and Local Perturbation Transfer

The preceding analysis establishes how the shared scalar generator constrains the local interaction field. The next question is behavioral: whether the resulting mixed curvature actually describes how the trained field responds when one mode is changed. We therefore introduce a small perturbation to a source representation and compare the induced response of another mode with the prediction given by the corresponding interaction operator.

Let z_ε denote the state obtained by perturbing the representation coordinates of mode b as

$$\xi^{(b)} \;\mapsto\; \xi^{(b)} + \varepsilon v, \qquad v = \mathrm{col}(v_q, 0),$$

so that the auxiliary coordinates are unchanged at initialization. For two distinct modes a and b, a first-order expansion gives

$$\begin{aligned} f_\theta^{(a)}(z_\varepsilon) - f_\theta^{(a)}(z) &= \varepsilon \mathcal{C}_{a\leftarrow b}(z)v + O(\varepsilon^2) \\ &= \varepsilon J_a \mathcal{H}_{ab}(z)v + O(\varepsilon^2). \end{aligned} \tag{31}$$

Thus the mixed-curvature block does not enter the response as an isolated matrix; its local effect is the directional action $J_a\mathcal{H}_{ab}(z)v$ on the applied perturbation.

The first-order approximation also determines the residual used in the local validation experiment:

$$R(\varepsilon) = \left\| f_\theta^{(a)}(z_\varepsilon) - f_\theta^{(a)}(z) - \varepsilon \mathcal{C}_{a\leftarrow b}(z)v \right\|_2 = O(\varepsilon^2). \tag{32}$$

When the leading quadratic term is nonzero, the corresponding log–log slope approaches 2 as $\varepsilon \to 0$. This gives the reference behavior for the measured-versus-predicted response and Taylor-residual experiments in Sec. 4.2. The required directional products can be evaluated through Jacobian– or Hessian–vector products without constructing the full Hessian explicitly Griewank and Walther (2008); Pearlmutter (1994).

From field response to representation refinement. The relation above concerns the instantaneous field, whereas tensor completion uses the representation produced by the discrete refinement map. For the implemented update Φ_h in Eq. (19), first-order consistency gives

$$\Phi_h(z) = z + hf_\theta(z) + O(h^2), \qquad [D\Phi_h(z)]_{ab} = h\mathcal{C}_{a\leftarrow b}(z) + O(h^2), \quad a \neq b. \tag{33}$$

Consequently, the same source perturbation produces the target representation change

$$\Delta q_+^{(a)} = h\varepsilon\, \Pi_{q^{(a)}} \mathcal{C}_{a\leftarrow b}(z)v + O(h^2\varepsilon + h\varepsilon^2), \tag{34}$$

where $\Pi_{q^{(a)}}$ selects the representation coordinates of the target mode.

Equations (31) and (34) provide the connection between the structural quantity and the behavior observed after refinement: the mixed curvature determines the local field response, and the refinement map transfers this response to the tuple-specific representation. A nonzero $\mathcal{H}_{ab}$ alone is not sufficient for every perturbation to produce a visible target response; the action can vanish along a particular direction v or after projection onto $q^{(a)}$. For this reason, the empirical comparison uses the same perturbation direction and the same representation projection for the measured and predicted responses.

F Mediated Propagation through Ordered Refinement

The preceding section describes the local response from one mode to another. The path experiment asks a different question: whether a response generated in one part of the state can be used as an intermediate signal before the current refinement step is completed. This distinction follows directly from the ordering of the implemented updates.

In Full HSF, the auxiliary state is updated before the representation. Therefore q_{k+1} is evaluated using the already updated p_{k+1}. For a source mode b, an intermediate mode m, and a target mode a, the computation graph can contain the within-step dependency

$$q_k^{(b)} \longrightarrow p_{k+1,[m]} \longrightarrow q_{k+1}^{(a)}. \tag{35}$$

Its contribution to the source-to-target Jacobian follows the ordinary chain rule,

$$\left.\frac{\partial q_{k+1}^{(a)}}{\partial q_k^{(b)}}\right|_{b\to p_{[m]}\to a} = \frac{\partial q_{k+1}^{(a)}}{\partial p_{k+1,[m]}} \frac{\partial p_{k+1,[m]}}{\partial q_k^{(b)}}. \tag{36}$$

Hence Full refinement can transmit a source perturbation through an updated intermediate auxiliary block within a single nominal step. Here $p_{[m]}$ denotes the mode-aligned coordinates of the single joint auxiliary state, rather than a separately parameterized momentum embedding.

The Q-only condition removes this intermediate update. With the auxiliary state fixed, its representation update can be written as

$$q_{k+1} = q_k + h\, g(q_k). \tag{37}$$

All components of q_{k+1} are therefore computed from the same pre-update representation q_k. An updated mediator $q_{k+1}^{(m)}$ cannot be consumed by $q_{k+1}^{(a)}$ during that same synchronous update. The updated mediator becomes available only to a subsequent refinement step, so the corresponding mediated route can emerge across steps rather than within the current one. This distinction does not imply that Q-only lacks a direct $b \to a$ interaction at $K = 1$; it concerns specifically the use of an *updated intermediate state*.

The intervention experiment is designed around this difference. In Full, resetting the mediator-aligned auxiliary block after the p update and before the q update removes the selected within-step route in Eq. (35). Q-only has no corresponding within-step mediator to remove. The comparison therefore tests whether the ordered auxiliary update provides a genuine intermediate transmission channel beyond direct cross-mode dependence.

G When Does Hamiltonian Compatibility Help?

The previous sections describe what the compatibility structure constrains and how those interactions appear in the refinement dynamics. They do not imply that the restriction is uniformly beneficial. Statistically, compatibility trades flexibility for a reduction in the number of independently estimated local responses. Its value therefore depends on the balance between estimation variance and structural mismatch (Geman et al., 1992).

Let $A_\star = Df_\star(z_0)$ denote the target local operator at a fixed state, and let $\mathcal{A}_H$ be the Hamiltonian-compatible class defined in Section B. We measure the part of the target operator that cannot be represented within this class by

$$\eta^2 = \|A_\star - \Pi_H A_\star\|_F^2\,, \tag{38}$$

where Π_H is the Frobenius-orthogonal projection onto $\mathcal{A}_H$. Thus η^2 represents the local approximation cost of enforcing compatibility. It is an induced operator mismatch and need not coincide numerically with the mixing coefficient used to construct the synthetic teacher.

To expose the statistical trade-off, consider the local observation model

$$\widetilde{A} = A_\star + \frac{\sigma}{\sqrt{N}} Z, \qquad \mathrm{vec}(Z) \sim \mathcal{N}(0, I), \tag{39}$$

where N represents the effective amount of supervision and σ^2 controls the noise level. Let P_U and P_H denote the local degrees of freedom of the unrestricted and compatible classes derived in Section B. For the unrestricted estimate $\widehat{A}_U = \widetilde{A}$ and the compatible estimate $\widehat{A}_H = \Pi_H \widetilde{A}$, the corresponding risks are

$$\mathcal{R}_U = \frac{\sigma^2 P_U}{N}, \qquad \mathcal{R}_H = \eta^2 + \frac{\sigma^2 P_H}{N}. \tag{40}$$

Their difference is therefore

$$\boxed{\mathcal{R}_H - \mathcal{R}_U = \eta^2 - \frac{\sigma^2 (P_U - P_H)}{N}}. \tag{41}$$

The two terms in Eq. (41) have distinct roles. The first is the approximation cost introduced when the true local interaction is incompatible with the Hamiltonian class. The second is the variance removed by estimating fewer independent local responses. Compatibility is beneficial in this local model when

$$\eta^2 < \frac{\sigma^2 (P_U - P_H)}{N}. \tag{42}$$

This boundary directly motivates the three controlled factors examined in the experiments. Reducing the amount of supervision increases the variance term; increasing noise has the same effect. Both make the coupling between directional responses more valuable. Increasing structural mismatch instead raises the approximation cost and eventually favors the less constrained model.

Equation (41) is a local statistical characterization, not a universal guarantee for the nonlinear network. In the trained model, both $A_\star$ and its mismatch vary with the latent state, and the observations need not be isotropic. The role of the controlled experiments is therefore to test whether the predicted sample-size, noise, and mismatch trends remain visible under the full refinement model.

H Additional Theory-Driven Validation Protocols

The preceding identities suggest the following additional diagnostics. They are proposed checks, not additional experimental results. All comparisons should use the same checkpoint and states, smooth derivatives, and declared floating-point precision.

Field and directional compatibility. For $A = Df_\theta(z)$ and a small positive normalization constant τ, report

$$r_{\mathrm{H}}(z) = \frac{\|A^\top J + JA\|_F}{2\|A\|_F + \tau}, \qquad r_{ab}(z) = \frac{\|\mathcal{C}_{b\leftarrow a} - J_b \mathcal{C}_{a\leftarrow b}^\top J_a\|_F}{\|\mathcal{C}_{b\leftarrow a}\|_F + \|\mathcal{C}_{a\leftarrow b}\|_F + \tau}. \tag{43}$$

For a correctly differentiated scalar-generated field, both residuals vanish in exact arithmetic. The reciprocal relation in Eq. (29) provides a complementary full-block compatibility check. Divergence alone is insufficient: a trace-zero matrix need not satisfy Hamiltonian compatibility when $d > 1$. These tests verify implementation structure rather than predictive accuracy or statistical generalization.

Perturbation and step-size sweeps. For fixed unit v, evaluate $R(\varepsilon)$ in Eq. (32) over a logarithmic perturbation grid. If its quadratic coefficient is nonzero, then $R(\varepsilon)/R(\varepsilon/2) \to 4$. Separately sweep h at fixed states and measure

$$e_H(h) = |H_\theta(\Phi_h(z)) - H_\theta(z)|, \quad e_J(h) = \|D\Phi_h(z)^\top J D\Phi_h(z) - J\|_F, \quad e_V(h) = |\det D\Phi_h(z) - 1|.$$

For a first-order refinement map, these one-step defects are expected to follow an $O(h^2)$ local-error reference when the corresponding leading coefficient is nonzero; this does not require a slope exactly equal to two when that coefficient vanishes. Automatic differentiation avoids finite-difference truncation in the derivative itself Griewank and Walther (2008), but subtraction, determinant evaluation, and norm calculations still incur roundoff. Fit slopes only before the cancellation or roundoff floor dominates Higham (2002).

Finite-time sensitivity. For the exact flow,

$$\|D\varphi_t(z)\|_2 \le \exp\left(\int_0^t \|Df_\theta(\varphi_s(z))\|_2 \, ds \right), \qquad t \ge 0. \tag{44}$$

This follows by applying the integral inequality to $\dot{Y} = Df_\theta(\varphi_t(z))Y$, $Y(0) = I$. Because J is orthogonal, $\|Df_\theta\|_2 = \|\nabla^2 H_\theta\|_2$. For K implemented updates, the exact derivative is instead

$$D\Phi_h^K(z_0) = D\Phi_h(z_{K-1}) \cdots D\Phi_h(z_0).$$

Reporting its largest singular value together with prediction error tests whether sensitivities remain controlled at the depths actually used. Energy or compatibility checks alone cannot establish that.

Anisotropic-noise extension. To examine the scope of the isotropic risk formula, let $\mathrm{vec}(\widetilde{A} - A_\star) = N^{-1/2}\zeta$ with $\mathbb{E}\zeta = 0$ and $\mathbb{E}\zeta\zeta^\top = \Sigma$. Let P represent Π_H after vectorization, so $P = P^\top = P^2$. Orthogonality gives

$$\mathcal{R}_H = \eta^2 + N^{-1}\,\mathrm{tr}(P\Sigma), \qquad \mathcal{R}_U = N^{-1}\,\mathrm{tr}(\Sigma). \tag{45}$$

Thus the variance removed is $N^{-1}\,\mathrm{tr}((I - P)\Sigma)$, which depends on where the noise lies, not only on the dimension difference. This direct extension of the projection argument clarifies why observation geometry matters in structured estimation Wainwright (2019). Synthetic operator experiments can check this identity because $A_\star$ and Σ are known; observed tensor entries alone do not provide them.

I Datasets and Models

The experiments use five datasets. The corresponding data settings and model variants are summarized below.

I.1 Datasets and Evaluation Protocol

Pacific Temperature (PCTemp). PCTemp contains sea-surface temperature measurements over the Pacific Ocean (Reynolds et al., 2007). We organize the data as spatial–spatial–temporal tuples (i, j, τ), where i and j index the two spatial coordinates and τ indexes the observation time. Each entry records the temperature at the corresponding location and time.

New York City Taxi (Taxi). Taxi contains urban mobility records released by the New York City Taxi and Limousine Commission (New York City Taxi and Limousine Commission). We organize the data as origin–destination–time tuples (i, j, τ), where i indexes the pickup location, j indexes the drop-off location, and τ indexes the time interval. Each entry records the aggregated mobility measurement for the corresponding origin–destination pair and interval.

Network. Network contains time-varying directed communication interactions from a terminal interaction pattern analysis system (TIPAS) (Luo et al., 2022). We organize the data as source–destination–time tuples (i, j, τ), where i and j index the source and destination terminals, respectively, and τ indexes the time slot. Each entry records the directed interaction weight between the two terminals during that slot.

Quality of Service (QoS). QoS contains temporal web-service performance measurements from WS-DREAM (Zhang et al., 2011; Zheng et al., 2014), using the benchmark employed in AN-TucF (Tang et al., 2025a). We organize the data as user–service–time tuples (i, j, τ), where i indexes the service user, j indexes the web service, and τ indexes the measurement time slice. Each entry records the selected QoS measurement for a user's invocation of a service at that time.

Soil. Soil contains measurements of coarse particulate organic matter (cPOM) and mineral-associated organic matter (MAOM) (Ferdous, 2026). We organize the data as location–fraction–spot tuples (i, j, k), where i indexes the sampling location, j identifies the organic matter fraction (cPOM or MAOM), and k indexes a sampling spot within location i. Each entry records the measurement of the selected fraction at the corresponding sampling spot.

Data partitions and evaluation. All methods use identical training, validation, and test partitions within each setting. PCTemp-1/2 and Taxi-1/2 denote alternative partitions of the respective datasets. Preprocessing statistics are fitted exclusively on training observations and applied unchanged to validation and test data. We select checkpoints using validation RMSE and report test MAE, MRE, and RMSE after transforming predictions back to the original target scale. Lower values indicate better performance for all three metrics.

I.2 External Baselines

The main comparison includes CP, 3DA-NTC, NeAT, and DEMOTE. They cover multilinear factorization, neural reconstruction, and graph-coupled dynamic embeddings.

CP. Canonical polyadic factorization represents an entry through a sum of rank-one interactions (Kolda and Bader, 2009):

$$s_{ij\tau} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{\tau r}. \tag{46}$$

The mode factors are shared across entries, and no tuple-dependent refinement is applied before reconstruction. This is the classical CP baseline named in the main text.

3DA-NTC (3D). 3DA-NTC combines inner-product tensor factorization with outer-product interaction construction, followed by a three-dimensional channel-attention convolutional network (Kang et al., 2024b). The interaction tensor exposes multiway feature combinations, while convolution and channel attention extract and reweight higher-order interaction features for missing-value prediction. The main table abbreviates this method as 3D.

NeAT. NeAT uses a component-wise additive neural decomposition (Ahn et al., 2024). Separately parameterized neural networks model the interactions within individual latent components, and their outputs are added to reconstruct an entry. This architecture allows nonlinear component responses while preserving the additive organization of the decomposition. It provides a comparison with nonlinear reconstruction without the proposed Hamiltonian refinement.

DEMOTE. DEMOTE learns time-dependent entity embeddings through a neural diffusion–reaction process (Wang et al., 2023). A multi-partite graph encodes relations between entities; graph diffusion couples their embedding trajectories, and a neural reaction term models entity-specific changes. A neural decoder predicts entry values from the resulting embeddings. Unlike HSF's tuple-specific latent refinement, DEMOTE explicitly models graph-coupled embedding trajectories over observed time.

I.3 HSNTF Architecture and HSF Variants

Full HSF. For each queried tuple, the retrieved mode embeddings are concatenated into $q_0 \in \mathbb{R}^d$. A learnable auxiliary coordinate $p_0 = \alpha\bar{p}$ gives the initial state $z_0 = (q_0, p_0)$. The scalar network $H_\theta : \mathbb{R}^{2d} \to \mathbb{R}$ generates both updates:

$$p_{k+1} = p_k - h\nabla_q H_\theta(q_k, p_k), \qquad q_{k+1} = q_k + h\nabla_p H_\theta(q_k, p_{k+1}). \tag{47}$$

The joint representation q_K is mapped to the corresponding mode-specific factor spaces:

$$\widetilde{e}^{(m)} = W_m q_K + b_m, \qquad m = 1, 2, 3, \tag{48}$$

where $W_m \in \mathbb{R}^{r_m \times d}$ and $b_m \in \mathbb{R}^{r_m}$, followed by Tucker reconstruction:

$$\widehat{y}_{\text{norm}} = \sigma_{\text{out}}\left(\mathcal{G} \times_1 \widetilde{e}^{(1)} \times_2 \widetilde{e}^{(2)} \times_3 \widetilde{e}^{(3)}\right). \tag{49}$$

Here $\mathcal{G}$ is the learnable core and σ_{out} is the output link. The main-text configuration uses Tucker rank $(5, 5, 5)$, two Tanh hidden layers of width 128 in H_θ, and $h = 0.1$. The Hamiltonian structure organizes learned latent dynamics; it does not assume physical Hamiltonian dynamics in the observations.

Static Tucker. Static Tucker removes the refinement block and applies the mode projections and Tucker readout directly to q_0. It retains the reconstruction architecture, including its cross-mode interactions, while removing dynamic adaptation of the queried representations.

Separable HSF. Separable HSF retains auxiliary and representation updates but restricts the generator to

$$H_{\text{sep}}(q, p) = \sum_{m=1}^{3} H_m(q^{(m)}, p_{[m]}). \tag{50}$$

Its refinement field is independent across modes because all cross-mode mixed Hessian blocks vanish. Cross-mode interactions remain in the final Tucker readout. Comparing this variant with Full HSF therefore tests joint refinement beyond within-mode dynamics, as discussed in Sec. 4.2.

I.4 Neural ODE and the Q-only Update

Direct vector-field architecture. A representation-only Neural ODE learns a vector-valued map $g_\phi : \mathbb{R}^d \to \mathbb{R}^d$ from the concatenated representation to its derivative (Chen et al., 2018). An MLP receives q and outputs $\dot{q}$ directly, after which integration produces the refined representation for Tucker readout. The continuous dynamics and a forward-Euler discretization are

$$\frac{dq(s)}{ds} = g_\phi(q(s)), \qquad q_{k+1} = q_k + h\, g_\phi(q_k). \tag{51}$$

Here s is latent integration time. This construction has no evolving auxiliary state and does not derive paired updates from a shared scalar generator. It can still model cross-mode dependence, since every output block may depend on the full q.

Connection to Q-only. Q-only has the same single-state update organization as Eq. (51): a representation is mapped to a vector derivative and then updated. In the fixed-checkpoint experiment, however, this derivative is supplied by the trained HSF generator with its auxiliary state frozen:

$$g_{\mathrm{Q}}(q) = \nabla_p H_\theta(q, p_0), \qquad q_{k+1} = q_k + h\, g_{\mathrm{Q}}(q_k), \qquad p_k = p_0. \tag{52}$$

Thus the two designs agree at the level of the representation-only update, but differ in field parameterization: a standalone Neural ODE directly learns g_ϕ, whereas Q-only reuses derivatives of the same H_θ as Full HSF. The Full–Q-only intervention therefore removes auxiliary-state evolution without retraining or replacing the learned generator.

In Q-only, all representation blocks are updated synchronously from q_k. A source can directly influence another mode in one step, but a route through a distinct representation mediator requires successive updates. Full HSF additionally admits $q_k^{(b)} \to p_{k+1,[m]} \to q_{k+1}^{(a)}$ within one ordered step. This is the update-level distinction analyzed in Appendix F; it is not a claim that generic ODE vector fields lack cross-mode interactions.

J Experimental Protocol and Fairness

J.1 Data Partitions and Preprocessing

Data partitions. We partition the available observations into training, validation, and test sets. PCTemp-1 and Taxi-1 use a $70\%/15\%/15\%$ split, whereas PCTemp-2 and Taxi-2 use a $30\%/20\%/50\%$ split. The suffixes denote alternative partitions of the same dataset, rather than different datasets. Within each setting, the three partitions contain disjoint tensor coordinates, and competing methods use the same designated partitions. Training observations are used for parameter optimization, validation observations for model selection, and test observations for final evaluation. Originally missing entries are not treated as observed zeros.

Training density and sparsity. For a tensor of dimensions $I_1 \times I_2 \times I_3$, let Ω_{train} denote the set of observed training coordinates. We define

$$\rho_{\mathrm{train}} = \frac{|\Omega_{\mathrm{train}}|}{I_1 I_2 I_3}, \qquad s_{\mathrm{train}} = 1 - \rho_{\mathrm{train}}. \tag{53}$$

Unlike the split ratios, which describe the allocation of available observations, training density measures the fraction of all possible tensor entries available during training. Validation, test, and originally missing entries are excluded from Ω_{train}. In sparsity experiments, the training fraction instead denotes the proportion retained from a designated training partition. Table 5 summarizes the tensor dimensions and reported training observation statistics.

Table 5: Tensor dimensions and training observation statistics. Density is the percentage of all possible tensor entries observed during training; sparsity is its complement.

Dataset	Dimensions	Density (%)	Sparsity (%)
PCTemp	$30 \times 84 \times 399$	18.225	81.775
Taxi	$30 \times 30 \times 1464$	26.043	73.957
Network	$7089 \times 5924 \times 1148$	0.0002	99.9998
QoS	$142 \times 4500 \times 64$	2.000	98.000
Soil	$152 \times 2 \times 4$	28.618	71.382

Training-only preprocessing. Each observation consists of three integer mode indices and a scalar target y. We apply a fixed lower cutoff followed by a logarithmic transformation,

$$\ell(y) = \log\big(\max(y, 10^{-5}) + \epsilon\big), \qquad \epsilon = 10^{-8}. \tag{54}$$

Let a and b denote the minimum and maximum transformed targets over training observations. We normalize targets as

$$\widetilde{y} = \frac{\ell(y) - a}{b - a + \epsilon}. \tag{55}$$

Validation and test observations use the same training statistics without refitting, and their normalized values are not clipped to $[0, 1]$. Mode index ranges are determined from the supplied coordinates to size the embedding tables; held-out target values are not used to estimate preprocessing statistics.

Evaluation scale. Predictions are mapped back to the target scale through

$$\widehat{y} = \exp\Big(\widehat{\widetilde{y}}(b - a + \epsilon) + a\Big) - \epsilon. \tag{56}$$

MAE, MRE, and RMSE are computed after this inverse transformation. When reference targets are recovered from their normalized values, they retain the lower cutoff applied during preprocessing; for targets above 10^{-5}, this leaves the original values unchanged.

J.2 Training, Model Selection, and Repeated Runs

Main comparison. Under the common main-experiment configuration, all methods use five-dimensional mode embeddings, Adam with learning rate 10^{-3}, a batch size of 64, and at most 100 epochs. Model parameters are fitted on training observations, and checkpoint selection minimizes validation RMSE in the normalized log-target space (Cawley and Talbot, 2010; Varma and Simon, 2006). Test MAE, MRE, and RMSE are computed after checkpoint selection and inverse transformation. The HSF scalar generator has two Tanh hidden layers of width 128, and its refinement uses $K = 3$ and step size $h = 0.1$. The current HSNTF stopping rule terminates after more than 20 consecutive non-improving validation evaluations.

Shared settings and model-specific components. The comparison aligns the available observations, embedding dimensions, training protocol, and evaluation targets. Model-specific reconstruction networks and graph parameters retain their respective implementations; we do not impose equal parameter counts or introduce refinement steps into static models. The inspected graph-dynamics baseline uses three updates with step size 0.1, while its learned graph-attention and diffusion–reaction parameters remain specific to that model. These controls apply to the main tensor-completion comparison (Lucic et al., 2018; Melis et al., 2018).

Parameter-counting protocol. We count all trainable parameters instantiated for the PCTemp comparison. For the static CP model, a fair comparison only requires that its mode factors lie in the same five-dimensional embedding space used by the other methods. The remaining baselines retain their model-specific reconstruction or dynamics modules, so equalizing the total number of parameters would alter their intended architectures. Consequently, the reported differences reflect the intrinsic complexity of each model rather than inconsistent embedding dimensions. This design preserves the original modeling mechanisms while providing a common latent representation for the comparison.

Table 6: Trainable-parameter comparison of the baseline models on PCTemp.

Method	Rank/dim.	Parameters
CP	5	2,565
3D	5	20,838
NeAT	5	23,650
HSNTF	5	23,555
DEMOTE	5	22,333

Randomness and summary statistics. The common training protocol uses seeds $42, 43, 44, 45, 46$ (Reimers and Gurevych, 2017; Bouthillier et al., 2019; Dodge et al., 2019). Each run initializes a fresh model and uses a seed-specific generator for training-data shuffling, while the designated data partitions remain fixed. For a metric e_s from seed s, we report

$$\bar{e} = \frac{1}{5}\sum_{s=1}^{5} e_s, \qquad \mathrm{SD}(e) = \sqrt{\frac{1}{4}\sum_{s=1}^{5}(e_s - \bar{e})^2}. \tag{57}$$

This is the sample standard deviation across runs, not the standard error or a confidence interval (Demšar, 2006). Fixed-checkpoint mechanism analyses have a separate sampling seed; repeated perturbations of one checkpoint are not counted as independent model-training runs.

J.3 Mechanism Experiments

Structural ablation and local response analysis. The structural ablation compares separately trained Static Tucker, Separable HSF, and Full HSF under the designated completion protocol. The local-response analysis instead fixes a trained checkpoint and compares finite perturbation responses with the corresponding mixed-curvature prediction on held-out tuples. The inspected Taylor pipeline selects the checkpoint using normalized validation RMSE; its diagnostic seed controls tuple and perturbation-direction sampling. Perturbation magnitudes are specified by the diagnostic configuration, rather than chosen to maximize the downstream agreement statistic.

Propagation interventions. Full-field and intervened rollouts are evaluated using the same trained weights, queried tuples, initial states, and perturbation directions. The intervention changes the specified update rule without retraining the model. The saved fixed-path experiment evaluates $K = 0, 1, 2, 3$ from a checkpoint trained with $K = 3$. Its path-response statistics characterize perturbation propagation and must be distinguished from completion errors, whose target scale is set by the evaluation routine.

J.4 Synthetic Reciprocal Transfer and Statistical Boundary

Directional teacher and observations. The reciprocal-transfer benchmark uses $H_0(z) = \frac{1}{2} z^\top Q z + \sum_r c_r \cos(w_r^\top z)$, with $Q = Q^\top$, and mixes $J\nabla H_0$ with JKz, where $K = -K^\top$:

$$f_\delta(z) = \sqrt{1-\delta^2}\, \bar{f}_H(z) + \delta\, \bar{f}_\perp(z). \tag{58}$$

The component scales are calibrated through their Jacobian Frobenius norms, using independent calibration states for the nonlinear component. They are not calibrated by the target test responses. For target direction $a \leftarrow b$, a unit perturbation supported on mode b produces the central-difference response in mode a; the reciprocal direction is $b \leftarrow a$. The formal experiment uses three modes with rank 5, phase dimension 30, states sampled from $\mathcal{N}(0, 0.25I)$, and perturbation magnitude 10^{-3}. All training, validation, and test labels in the saved reciprocal experiment are noise-free.

Reciprocal pretraining and target adaptation. For each target direction, pretraining uses the other five directed mode pairs, including its reciprocal, with 256 training responses and 128 validation responses per pair. The target direction is absent from both optimization and model selection at this stage. For each $\rho \in \{0, 0.05, 0.10, 0.20\}$, adaptation starts from the same selected pretrained checkpoint; positive fractions use only the retained target-direction training and validation responses. The subsets are nested, and the fractions are independent branches rather than successive stages of fine-tuning. Reciprocal observations are retained during pretraining but are not replayed in target-only adaptation. The two neural models use matched parameter budgets and the same directional observations. Hyperparameters and checkpoints are selected by auxiliary-direction validation MSE during pretraining and target-direction validation MSE during adaptation.

Directional evaluation and randomness. Both models are evaluated on the same 512 clean target probes using the analytic target Jacobian response as reference. We report $\mathrm{NMSE} = \sum_i \|\widehat{r}_i - r_i\|_2^2 / \sum_i \|r_i\|_2^2$ for $\delta \in \{0, 0.05, 0.10\}$ and seeds 42–46. Teacher and sampling seeds are deterministically derived from each run seed; model comparisons within a condition share the teacher and observations. The adaptation summary averages over six target directions and five seeds. Its saved standard deviation is the sample standard deviation across these 30 direction–seed results, not the standard deviation of five direction-averaged seed scores.

Nonlinear field boundary. The statistical-boundary experiment uses a separate nonlinear basis $\phi_k(z) = \mathrm{sech}^2(a_k^\top z + b_k)$, corresponding to the Hamiltonian $H(z) = \sum_k w_k \tanh(a_k^\top z + b_k)$. For each basis function, the compatible vector coefficient is $w_k J a_k$; an incompatible coefficient is constructed orthogonal to Ja_k. The incompatible field is rescaled to match the compatible field RMS on the training pool before mixing as $f_H + \delta f_\perp$. The resulting field is divided by its training-pool RMS. This teacher differs in both its basis and normalization from the directional-response teacher.

Sample size, noise, and estimation. The saved boundary datasets use phase dimension 12, 32 nonlinear basis functions, a training pool of 4096 states and 10000 test states, generated with seed 42

from coordinate-wise clipped standard Gaussian samples in $[-2.5, 2.5]$. Within each replicate, a fixed random permutation supplies nested training subsets of size N. Independent standard Gaussian noise is added to each normalized training-field coordinate: $y_i = \bar{f}_\delta(z_i) + \sigma\xi_i$, $\xi_i \sim \mathcal{N}(0, I)$. The same noise realization is shared by both estimators and rescaled across noise levels; test fields remain clean. We compare a Hamiltonian coefficient estimator with an unrestricted coefficient estimator using the same known features and exact least squares. There is no neural-network checkpoint selection. The saved experiments use $\delta \in \{0, 0.2, 0.4\}$, $\sigma \in \{0, 0.05, 0.1, 0.2, 0.4\}$ and 50 sampling/noise replicates. The common training-size grid is $N \in \{8, 16, 24, 48, 64, 96, 128, 256\}$; the saved $\delta = 0$ results additionally contain $N = 32$. Performance is measured by clean test-field MSE, with preference score $\log_{10}(E_U/E_H)$; positive values favor the constrained estimator. These 50 replicates are distinct from the five-seed neural-training protocol.

K Additional Experimental Results

K.1 Computational Efficiency under Growing Graph Scale

We compare the computational scaling of HSNTF with a controlled DEMOTE-style diffusion–reaction baseline. The purpose is to isolate the cost of *tuple-local refinement* versus *global graph propagation*, rather than to establish a predictive advantage on a particular benchmark.

Controlled protocol. We generate synthetic tensors with $N \in \{10\text{k}, 20\text{k}, 50\text{k}, 100\text{k}\}$ training observations while fixing both validation and test sets to 2,000 entries. Tensor coordinates are sampled without replacement before splitting, ensuring disjoint train, validation, and test partitions (Morris, 1991). All scales share a single rank-5 teacher,

$$g(u, s, t) = \langle \mathcal{G}, u \circ s \circ t \rangle + 0.25 \sin\langle u, s \rangle + 0.20 \cos\langle s, t \rangle + 0.15 \sin\langle u, t \rangle,$$

followed by variance normalization, a sigmoid transform, and Gaussian noise with standard deviation 0.01. Thus, increasing N changes the computational scale while preserving the underlying prediction mechanism.

We use the *node-stress* setting to increase the graph together with the number of observations. The user, service, and time cardinalities are set to $0.25N$, $0.20N$, and $0.05N$, respectively, yielding approximately $|V| = 0.5N$ entities. Each observed triple induces bidirectional user–service, user–time, and service–time edges, with duplicate edges coalesced. Time remains the third tensor mode in this controlled comparison; no timestamp-specific ODE sampling acceleration is used.

Both models use rank $r = 5$, Adam with learning rate 10^{-3}, batch size $B = 128$, step size $h = 0.1$, and $K = 3$ refinement steps, together with the same Tucker readout. To match neural dynamic capacity, the scalar Hamiltonian network of HSNTF contains 20,609 trainable parameters, while the three mode-specific DEMOTE reaction networks jointly contain 20,805. DEMOTE additionally learns one strength for each coalesced graph edge. These graph-specific parameters are excluded from the neural-capacity match, but are fully included in runtime, memory, backward propagation, and optimizer updates.

Complexity. Let C_H denote the cost of one Hamiltonian-network evaluation, including the gradient required to construct its vector field. For a fixed mini-batch size, HSNTF performs two field evaluations per refinement step only on the queried tuples, giving

$$T_{\text{HSNTF}} = \mathcal{O}(2NKC_H)\,.$$

By contrast, every DEMOTE mini-batch evolves the complete graph. Let

$$C_R = (r+1)h_R + h_R^2 + h_R r$$

denote the per-node cost of a mode-specific reaction network with hidden width h_R. Its epoch-level dynamic cost is approximately

$$T_{\text{DEMOTE}} = \mathcal{O}\left[\frac{N}{B} K \left(|E|r + |V|C_R\right)\right].$$

The principal memory distinction is similarly

$$M_{\text{HSNTF}} = \mathcal{O}(Vr) + M_{\text{local}}, \qquad M_{\text{DEMOTE}} = \mathcal{O}(Vr + |E| + K|V|A_R)\,,$$

where A_R denotes the activations retained by the reaction dynamics. Thus, both methods store entity representations, but DEMOTE additionally materializes graph structure and repeatedly propagates the global node state.

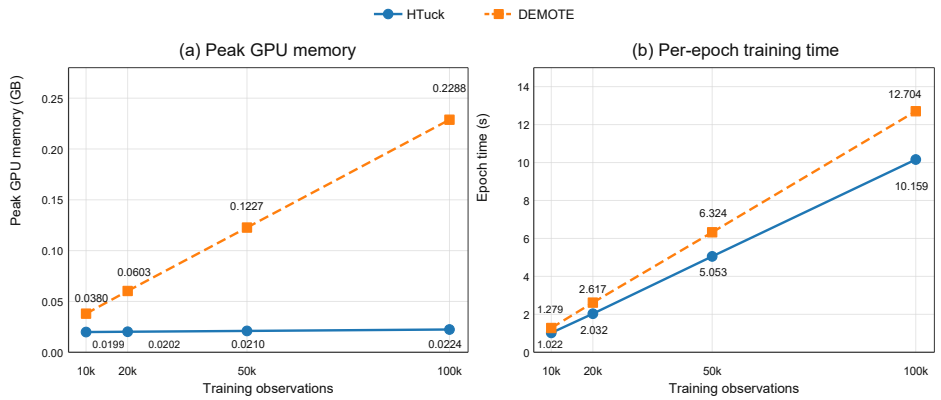


Figure 6: **Efficiency scaling with increasing training observations. (a)** Peak GPU memory. **(b)** Mean per-epoch training time. Values are averaged over two seeds; error bars denote standard deviation where visible.

Observed scaling. Figure 6 shows a clear difference in how the two architectures consume resources as the problem grows. The peak GPU memory of HSNTF remains nearly flat, increasing only from 0.0199 GB at $N = 10$k to 0.0224 GB at $N = 100$k. DEMOTE instead increases from 0.0380 GB to 0.2288 GB over the same range. Consequently, its memory requirement grows from approximately $1.9\times$ that of HSNTF at 10k observations to approximately $10.2\times$ at 100k.

Training time exhibits the same ordering, although the difference is less pronounced than for memory. Across the four scales, HSNTF reduces mean epoch time by approximately 20.0%–22.3% relative to DEMOTE. At $N = 100$k, the corresponding times are 10.159 s and 12.704 s. This difference arises despite the higher-order differentiation required to construct the Hamiltonian field.

The increasing DEMOTE cost is consistent with its global propagation mechanism. At $N = 100$k, the generated graph contains 50k nodes and approximately 599.8k learned directed edges. Each forward pass performs three complete diffusion–reaction evaluations, corresponding to approximately 1.80M edge messages and 150k node-level reaction evaluations. In contrast, HSNTF constructs its interaction field directly from the queried tuple and therefore avoids both graph materialization and whole-graph state evolution.

Importantly, this resource difference does not result from a substantial loss in predictive accuracy. Across all four scales, the test RMSEs of the two models differ by less than 5×10^{-4}. The experiment therefore isolates the main computational distinction: explicit graph propagation provides a global relational mechanism, but its storage and propagation cost grows with the graph, whereas Hamiltonian refinement remains tuple-local and yields substantially more favorable memory scaling with lower per-epoch training cost.

K.2 Context-dependent Embedding Visualization

We examine how the representation of a fixed entity changes when it is queried with different entities from the other two modes. This comparison isolates context dependence within a trained Full HSF model: the initial lookup vector is shared across queries containing the same entity, whereas its refined representation can depend on the complete tuple.

Data and trained model. We use PCTemp and Soil with the full training partitions of their respective completion settings, without additional subsampling for model training. For PCTemp, we use the $30\%/20\%/50\%$ training/validation/test split, denoted PCTemp-2 in the main experiments; Soil uses its designated benchmark split. Target preprocessing and checkpoint selection follow

Appendix J.1 and Appendix J.2: targets are clipped below 10^{-5}, log-transformed, and normalized using training-set statistics, and checkpoints are selected by normalized validation RMSE. The visualization uses a saved Full HSF model with $K = 3$ refinement steps. Model dimensions and weights are restored from the checkpoint, and the rollout uses the loaded model's step size. For Soil, the script loads the first saved run, with training seed 42. No parameters are updated during visualization.

Only held-out tensor coordinates are needed to extract the embeddings. We retain the original integer indices, remove duplicate coordinate triples, and check that all indices lie within the checkpoint's embedding tables. Target values are not used to select entities, sample contexts, or compute the two-dimensional layout. For each queried tuple, we initialize (q_0, p_0) as in Eqs. (2)–(4) and apply the trained p-then-q updates in Eqs. (10)–(11). We extract the relevant mode block at q_0 and q_3, before the affine projection and Tucker readout. Thus, the static and dynamic points come from the same model, rather than from separately trained static and dynamic baselines.

Entity and context selection. Within each mode, the automatic selection procedure ranks entities by their occurrence counts in the deduplicated test coordinates. It selects up to three entities, preferring those with at least 30 available contexts.

Table 7: **Entity indices for embedding visualization.** Indices are zero-based.

Dataset	Mode 1	Mode 2	Mode 3
PCTemp	$5, 14, 23$	$9, 10, 72$	$62, 155, 279$
Soil	$34, 93, 130$	$0, 1$	$3, 2, 0$

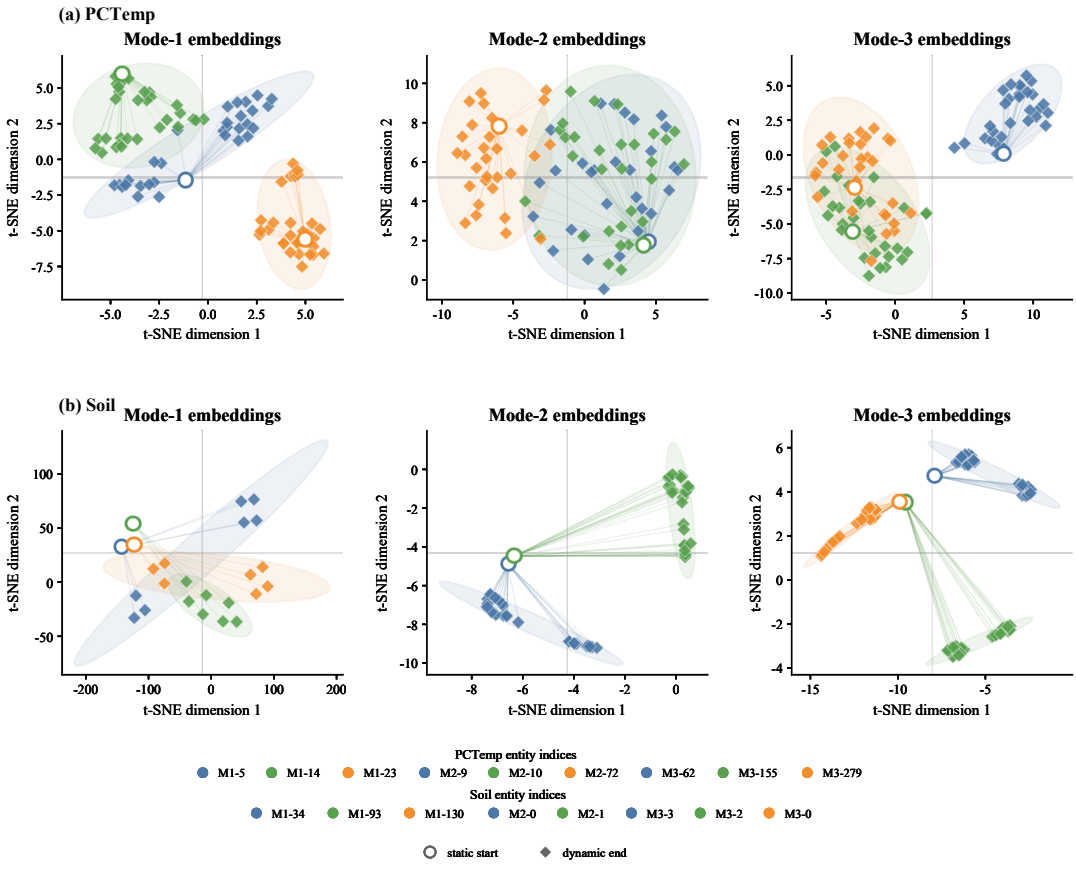


Figure 7: **Context-dependent refinement of entity embeddings.** Initial lookup vectors (circles) and context-specific representations after $K = 3$ updates (diamonds) for PCTemp and Soil. Lines link each initial point to its endpoints; shaded regions summarize endpoint dispersion. Panels are fitted independently.

If too few entities meet this threshold, it uses the most frequent entities in that mode. If the mode contains fewer than three entities, it uses all available entities. Table 7 lists the zero-based indices

shown in the figure. In particular, both entities of Soil's second mode are included. For each selected entity, we sample up to 30 distinct held-out tuples without replacement, keeping the focal index fixed and varying the other two indices. All available tuples are used when fewer than 30 are present. This is necessary for Soil's first mode, for which the other two modes permit at most $2 \times 4 = 8$ contexts. The sampling seed is $42 + 1000(r + 1) + 100m$, where r is the entity's zero-based position in the selected list and $m \in \{0, 1, 2\}$ is the mode index. Selection depends on coordinate coverage, not on target values, embedding distances, or the appearance of the refined points.

Joint t-SNE visualization. We fit a separate two-dimensional t-SNE map for each dataset–mode panel (van der Maaten and Hinton, 2008). The input contains one initial vector per selected entity and one final vector per sampled context. Initial vectors are included only once because they are identical across contexts for a fixed entity; the extraction code verifies this equality to a tolerance of 10^{-6}. All initial and final vectors in a panel are standardized together, coordinate by coordinate, and embedded in a single fit. Coordinates with standard deviation below 10^{-12} use a scaling factor of one.

t-SNE matches neighborhood similarities in the input space to similarities defined by a Student-t kernel in two dimensions. We use Euclidean input distances, PCA initialization, an automatic learning rate, and 2,000 optimization iterations. For a panel containing n points, the effective perplexity is $\min\{25, \max(2, (n-1)/3), n-1\}$. The layout seeds are 42, 43, and 44 for the three modes. Open circles denote initial vectors and diamonds denote final vectors; color identifies the focal entity within each panel. Faint segments connect each initial point to its context-specific endpoints, while shaded ellipses summarize endpoint dispersion.

Results and interpretation. Figure 7 shows that a single lookup representation gives rise to multiple refined representations when the surrounding tuple changes. On PCTemp, the endpoint distributions differ across modes: the first mode shows distinct entity-associated regions, whereas the second mode has substantial overlap between the blue and green point clouds. In the third mode, the blue endpoints occupy a region separate from most orange and green endpoints. On Soil, nearby initial points lead to different endpoint patterns, particularly in the second and third modes, where the refined representations form elongated groups. The smaller number of endpoints in Soil's first mode reflects its limited number of available contexts.

These observations illustrate that Full HSF adjusts an entity's representation according to the other indices in the query rather than assigning it a single final vector for all contexts. They do not imply that greater visual separation is preferable: t-SNE is a nonlinear neighborhood visualization, and distances, angles, and cloud sizes cannot be compared across independently fitted panels. The figure therefore provides qualitative evidence of context dependence; predictive utility and the role of the shared field are assessed by the completion and intervention experiments in the main text.