HAMILTONIAN JEPA: SYMPLECTIC LATENT DYNAMICS FOR WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-embedding predictive architectures (JEPAs) learn world models directly in latent space, but unconstrained autoregressive predictors may contract the latent distribution over long rollouts. We introduce Hamiltonian JEPA (**LeHami**), which replaces the standard predictor with an action-conditioned Hamiltonian flow over a structured latent state $(\mathbf{q}, \mathbf{p})$ integrated using an exactly symplectic leapfrog step. This construction preserves phase-space volume at every rollout step and rules out uniform contraction, asymptotically stable fixed points, and merging of distinct latent states, independently of the learned parameters or action sequence. We obtain the momentum-like component from temporal derivatives of the learned position-like representation, providing an explicit second-order state for prediction and control. On conservative and actuated visual pendulum environments, LeHami substantially improves long-horizon velocity prediction over an unconstrained JEPA world model and empirically maintains constant latent volume where the baseline contracts. The learned flow also supports accurate integration at unseen sub-frame step sizes and numerical time reversal. These results show how geometric structure can provide inference-time guarantees that are complementary to representation regularization in latent world models. Code for LeHami is available at `https://anonymous.4open.science/r/hamiltonian-jepa-7F08`.

1 INTRODUCTION

Joint Embedding Predictive Architecture (JEPA) (LeCun et al., 2022) has emerged as a promising framework for action-conditioned latent world modeling. While existing JEPA formulations successfully prevent *encoder collapse:* mapping distinct inputs to constant representations, via training-time regularization (Bardes et al., 2021; Balestriero & LeCun, 2025) or frozen backbones (Assran et al., 2025; Zhou et al., 2024), maintaining consistent dynamics over extended evaluation rollouts remains a persistent challenge (Maes et al., 2026a; Zhang et al., 2026). Long-horizon stability is further hindered by *predictor contraction*: in an autoregressive rollout each prediction is fed back as the next input, so even a slight per-step contraction of the transition map compounds over the horizon and progressively shrinks the latent distribution.

Physics avoids this contraction through geometric *structure*. Hamiltonian mechanics governs dynamics via a single scalar H, constraining conservative trajectories to energy level sets and preserving phase-space volume at every discrete step (Arnold, 1989). Crucially, volume preservation persists under actuation, provided the action enters the Hamiltonian. Port-Hamiltonian extensions further cover dissipative systems (Marsden et al., 1999; Zhong et al., 2020; Duong & Atanasov, 2021), making it a physically grounded prior for latent spaces rather than an idealization. While H is traditionally derived analytically (Goldstein, 2011), recent methods parameterize a learnable H_ϕ directly from trajectories or pixel inputs (Greydanus et al., 2019; Chen et al., 2019; Toth et al., 2019).

We propose **LeHami** (Hamiltonian JEPA), a framework that embeds the latent representation space onto a symplectic manifold where evolution is driven by a learnable action-conditioned Hamiltonian flow $H_\phi(\mathbf{q}, \mathbf{p}, \mathbf{u})$ over a canonical latent pair $(\mathbf{q}, \mathbf{p})$, where $\mathbf{u}$ denotes the control action vector, integrated by an exact symplectic leapfrog step (Hairer et al., 2003) and trained end-to-end without

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

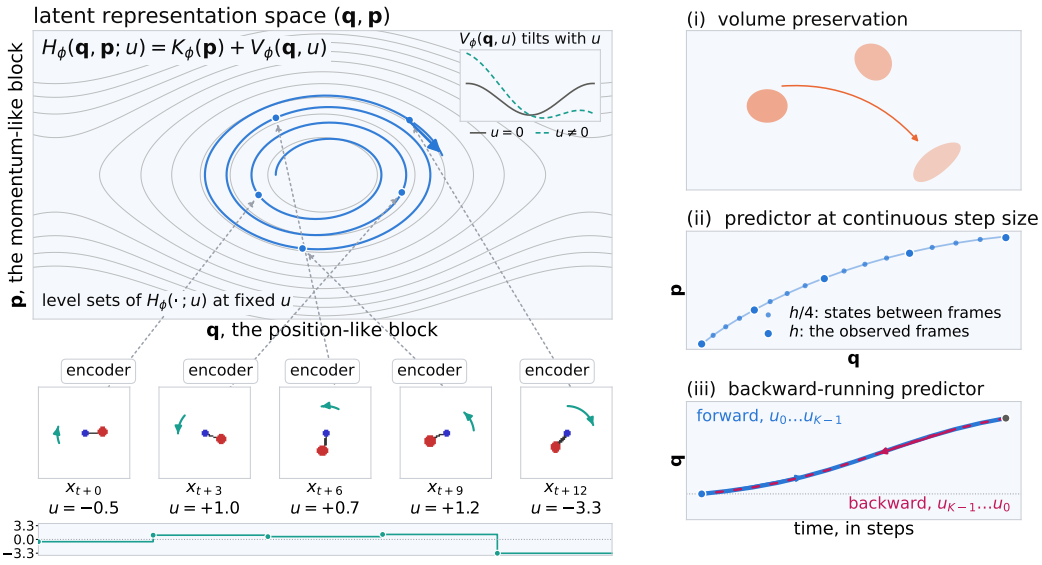

Figure 1: **LeHami: Learning symplectic world models end-to-end. Bottom left:** Actuated pendulum frames and applied torque sequence u_t (green). **Upper left:** Latent phase space $(\mathbf{q}_t, \mathbf{p}_t)$, where transitions are generated by a learned scalar $H_\phi(\mathbf{q}_t, \mathbf{p}_t; u_t) = K_\phi(\mathbf{p}_t) + V_\phi(\mathbf{q}_t, u_t)$ via a symplectic step. Grey lines indicate level sets; the blue curve shows the rollout. **Right:** Three properties of the Hamiltonian flow: **(i) Volume preservation:** Latent volume is exactly preserved under any action sequence. **(ii) Continuous step size:** H_ϕ integrates at arbitrary time intervals, including between observed frames. **(iii) Time-reversibility:** Integrates backward by inverting time $(t \to -t)$ and reversing actions.

decoders or supervised coordinates. LeHami inherits three structural properties from the Hamiltonian flow, illustrated in Fig. 1 (right):

(i) **Exact volume preservation:** Symplectic integration preserves phase-space volume by construction, provably eliminating some representation collapse modes during evaluation rollouts for arbitrary action sequences (Proposition 1).

(ii) **Continuous-time step size:** Because H_ϕ parameterizes a continuous vector field, a single trained model can be integrated at arbitrary step sizes Δt without retraining, facilitating training on multi-frequency data (Doshi et al., 2024) and non-uniform planning grids (Tan et al., 2016).

(iii) **Time-Reversibility:** Flipping time $(t \to -t)$ allows backward-in-time rollouts, serving as an essential primitive for computing backward reachable tubes and online safety guarantees in optimal control (Mitchell et al., 2005; Herbert et al., 2017).

Moreover, a central objective in physical world modeling is learning representations that capture the *complete physical state* of an environment, especially for control tasks like object interception (Kim et al., 2014) and precision throwing (Zeng et al., 2020). In physics, uniquely determining future evolution requires canonical momentum alongside position within a structured phase space. Without this geometric prior, standard JEPA architectures with unstructured predictors resort to architectural design, such as expanding the encoder's temporal window or context-conditioned predictors to capture velocity, lacking guarantees. In contrast, LeHami leverages this prior to model the latent space as a canonical phase space, enabling complete physical representations.

Contributions. We summarize our key contributions as follows:

- **Symplectic latent world model (LeHami):** We introduce an end-to-end trainable JEPA framework equipped with a learned separable Hamiltonian predictor over a canonical latent split $(\mathbf{q}, \mathbf{p})$. We prove that its volume-preserving step eliminates three specific collapse modes during evaluation rollouts across arbitrary horizons and actions (Proposition 1).

- **Continuous-time & time reversibility:** We show that a single trained predictor H_ϕ acts as a continuous vector field, enabling sub-frame time integration without performance degradation and achieving numerical time-reversal with round-trip error below $10^{-4}\%$ without architectural modifications.
- **Complete physics representation & principled planning:** By construction rather than by architectural tuning, LeHami represents the full state through exact geometric structure. Across conservative and actuated pendulum environments, we empirically show that LeHami accurately recovers full system dynamics. This paves the way towards compute-efficient planning to goals that specify the full physical state: position and velocity.

2 HAMILTONIAN JEPA

LeHami makes the latent dynamics of a JEPA world model Hamiltonian. Its latent state is a canonical pair $(\mathbf{q}_t, \mathbf{p}_t)$: the encoder outputs the position-like $\mathbf{q}_t$, and the momentum-like $\mathbf{p}_t$ is its time derivative, obtained by zero-parameter numerical differentiation of the encoder trajectory. The predictor advances this pair by one leapfrog step of a learned, action-conditioned Hamiltonian, so every step of the rollout is a symplectic map. We first state what this structure guarantees (Proposition 1), then give the flow, the multi-step objective and the design choices.

A symplectic map preserves the canonical two-form $\Omega = \sum_i dq_i \wedge dp_i$, the oriented area spanned in each position–momentum plane (Appendix B), and with it phase-space volume: a cloud of latent states can be stretched, rotated and sheared by the rollout, but never squeezed. Constraining the predictor to be such a map therefore rules out, at every step, the contraction available to an unconstrained rollout (Proposition 1).

Proposition 1 (Symplectic volume preservation and the rollout failure modes it removes). *Let Ψ_h be the leapfrog step of Eq. 4 for step size h generated by a separable $H_\phi = K_{\phi_1}(\mathbf{p}) + V_{\phi_2}(\mathbf{q}, \mathbf{u})$ with $\mathbf{u}$ held fixed within the step. Then Ψ_h is an exactly symplectic diffeomorphism of $\mathbb{R}^{2d}$ for every step size h and every parameter ϕ; it then preserves phase-space volume exactly at every horizon. Here h is the integrator's step, which equals the observation interval Δt during training but need not at inference (Section 3.3). Consequently, three collapse modes are ruled out: uniform contraction, convergence to an asymptotically stable fixed point, and the merging of distinct states. They are stated and proved in Appendix C.1.*

Crucially, symplecticity in the latent representation space is a *structural* guarantee as it holds at every inference step by construction, not only in expectation over training batches. By endowing the rollout with symplectic geometry, it may also improve long-horizon consistency and planning performance. Moreover, it has certain properties, as discussed in Section 1, that are of interest for many applications and that no unstructured predictor can have. We chose the Hamiltonian framework rather than the Lagrangian one; the reasons are given in Appendix C.3.

Scope and limitations of the guarantee. Three immediate remarks follow Proposition 1. We state them here in short form; Appendix C.2 gives the arguments in full. *(i) Shape, not size.* Volume preservation forbids the degenerate failures but it does not bound the growth of rollout error. *(ii) Regularization is not made redundant.* Symplecticity preserves volume, not distributional shape. The two act at different places: The regularizer acts on the encoder distribution in expectation over batches, while symplecticity on every inference step. *(iii) Geometric prior, not physical modeling.* Symplecticity acts purely as an inductive bias on the latent space rather than a physical simulator (Greydanus et al., 2019; Chen et al., 2019; Toth et al., 2019).

2.1 THE ARCHITECTURE

We train on unannotated trajectories of observations $[\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}]$ and actions $[\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}]$. LeHami has two learned components, an encoder Enc_θ and a Hamiltonian flow (Ham flow_ϕ) that replaces the predictor, joined by the zero-parameter Savitzky–Golay (SG) stencil (Gorry, 1990) that reads the momentum off the encoder trajectory (Fig. 2):

$$\begin{aligned} \text{Encoder + SG:} \quad & \mathbf{q}_t = \text{Enc}_\theta(\mathbf{x}_{\mathcal{W}(t)}), \qquad \mathbf{p}_t = \text{SG}(\mathbf{q}_{t-3}, \ldots, \mathbf{q}_t), \\ \text{Hamiltonian flow:} \quad & (\tilde{\mathbf{q}}_{t+1}, \tilde{\mathbf{p}}_{t+1}) = \text{Ham flow}_\phi(\mathbf{q}_t, \mathbf{p}_t, \mathbf{u}_t), \end{aligned} \tag{1}$$

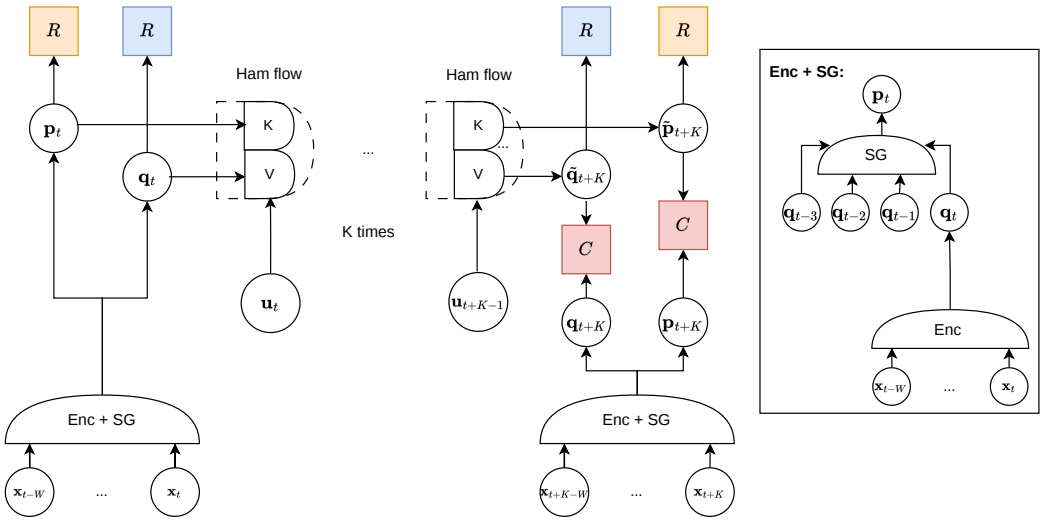

Figure 2: **LeHami multi-step training. Left:** at the anchor, the encoder block (Enc + SG) maps a window of $W+1$ frames, drawn here as the causal window $\mathbf{x}_{t-W}, \ldots, \mathbf{x}_t$, to the latent state $(\mathbf{q}_t, \mathbf{p}_t)$. The Hamiltonian flow (Ham flow) advances it by one leapfrog step of the separable Hamiltonian $H_\phi = K_{\phi_1}(\mathbf{p}) + V_{\phi_2}(\mathbf{q}, \mathbf{u})$, with the action entering the potential, and is applied K times to its own outputs. **Middle:** each prediction $(\tilde{\mathbf{q}}_{t+k}, \tilde{\mathbf{p}}_{t+k})$ is scored by the prediction loss (C) against the encoding of the corresponding frames, shown here for the last step $k = K$. The regularizer (R) acts on the encoder's embeddings of $\mathbf{q}$ (blue) and $\mathbf{p}$ (orange). **Right:** the encoder block as instantiated in this work: a pixel encoder (Enc) outputs $\mathbf{q}_t$, and a fixed Savitzky–Golay stencil (SG) derives $\mathbf{p}_t$ from $\mathbf{q}_{t-3}, \ldots, \mathbf{q}_t$.

where $\mathcal{W}(t)$ is a window of $W+1$ that includes $\mathbf{x}_t$ ($W{=}0$ reads $\mathbf{x}_t$ alone; Appendix D.2). The flow advances the state by one step of Hamilton's equations for a learned scalar $H_\phi(\mathbf{q}, \mathbf{p}, \mathbf{u})$,

$$\dot{\mathbf{q}} = \frac{\partial H_\phi}{\partial \mathbf{p}}, \qquad \dot{\mathbf{p}} = -\frac{\partial H_\phi}{\partial \mathbf{q}}, \tag{2}$$

and θ and ϕ are trained end-to-end by backpropagating through the rollout, so the encoder is shaped to produce latent states that admit this structured evolution.

The integration. In principle, different integration techniques can be used. We leverage the leapfrog (Störmer–Verlet) integrator (Hairer et al., 2003), which is second-order accurate and, crucially, exactly symplectic. For a separable Hamiltonian

$$H_\phi(\mathbf{q}, \mathbf{p}, \mathbf{u}) = K_\phi\big(\mathbf{S}^{-1}\mathbf{p}\big) + V_\phi(\mathbf{q}, \mathbf{u}), \qquad \mathbf{S} = \mathrm{diag}(e^{\boldsymbol{\sigma}})\,, \tag{3}$$

the update over a time step Δt is

$$\begin{aligned} \tilde{\mathbf{p}}_{t+\frac{1}{2}} &= \mathbf{p}_t - \tfrac{\Delta t}{2}\,\nabla_{\mathbf{q}} V_\phi(\mathbf{q}_t, \mathbf{u}_t), \\ \tilde{\mathbf{q}}_{t+1} &= \mathbf{q}_t + \Delta t\,\mathbf{S}^{-1}\,\nabla K_\phi\Big(\mathbf{S}^{-1}\tilde{\mathbf{p}}_{t+\frac{1}{2}}\Big)\,, \\ \tilde{\mathbf{p}}_{t+1} &= \tilde{\mathbf{p}}_{t+\frac{1}{2}} - \tfrac{\Delta t}{2}\,\nabla_{\mathbf{q}} V_\phi(\tilde{\mathbf{q}}_{t+1}, \mathbf{u}_t), \end{aligned} \tag{4}$$

where $\boldsymbol{\sigma} \in \mathbb{R}^d$ is learned and absorbs the relative scale of $\mathbf{q}$ and $\mathbf{p}$, which the self-supervised objective leaves undetermined. Because $\mathbf{S}$ is a constant linear rescaling, the flow remains exactly symplectic.

Action-conditioned Hamiltonian. Under actuation the controller does work and H_ϕ is no longer constant, but symplecticity is a property of the map. We condition the potential on the action so $V_{\phi_2} = V_{\phi_2}(\mathbf{q}, \mathbf{u})$ and the generalized force $\mathbf{F}(\mathbf{q}, \mathbf{u}) = -\partial V_{\phi_2}/\partial \mathbf{q}$ is a $\mathbf{q}$-gradient for every action by construction. Separability is untouched and the leapfrog of Eq. 4 stays *exactly* symplectic for every action sequence, provided only that $\mathbf{u}_t$ is constant within a step.

Multi-step learning. We consider multi-step prediction over a K-step horizon by iteratively applying the Hamiltonian flow. The loss is the average MSE across the trajectory. Given an anchor of A time steps used to form the embedding of the first time step of the episode, and K self-fed leapfrog steps, the loss is

$$\mathcal{L} = \frac{1}{K}\sum_{j=1}^{K} \left\|\tilde{\mathbf{q}}_{A+j} - \mathbf{q}_{A+j}\right\|^2 + \lambda_p \frac{1}{K}\sum_{j=1}^{K} \left\|\tilde{\mathbf{p}}_{A+j} - \mathbf{p}_{A+j}\right\|^2 + \lambda_{\mathcal{S}}\mathcal{S}, \tag{5}$$

where the first term is the prediction loss in the position-like embedding, the second is the prediction loss in the momentum-like embedding weighted by λ_p and the last term $\mathcal{S}$ is the regularizer, which, in principle, can be any (e.g., SIGReg (Balestriero & LeCun, 2025)) and it can be applied to the embeddings $\mathbf{q}$ and $\mathbf{p}$ independently, to only $\mathbf{q}$, or to both embeddings concatenated. It acts on ground-truth embeddings, never on the rollout. We compare the performance across the three placements in Table 8 in Appendix E.3.

Design choices. *The shape of H_ϕ.* While the core LeHami framework is general and agnostic to the choice of Hamiltonian, in this work we adopt a separable structure. This design choice enables fast, explicit leapfrog integration without requiring iterative numerical solvers. The guarantee is not specific to separable H: for a non-separable Hamiltonian, e.g., exact symplecticity is retained by replacing leapfrog with a non-separable symplectic integrator (Tao, 2016). *How $\mathbf{p}$ is obtained.* We enforce the canonical relation $\mathbf{p} = \dot{\mathbf{q}}$ by computing $\mathbf{p}$ as the temporal derivative of generalized coordinates $\mathbf{q}$ via a fixed Savitzky–Golay (SG) stencil (Gorry, 1990) on the encoder trajectory. This is a least-squares quadratic fit over $\mathbf{q}$ that is then differentiated at time step t. The stencil is linear with constant coefficients, so it adds no learned parameters and cannot itself collapse. More details in Appendix C.4.

2.2 Related work

Recent work imposes Hamiltonian structure on a learned latent so that predictions remain physically meaningful and useful for action. Cui & Ma (2026) propose Hamiltonian World Models, a perspective that decodes predicted latent trajectories into future observations and plans over the resulting rollouts; as a position paper, it reports no experiments. Tang et al. (2026) decompose the latent into a canonical and a context subspace and train it with a JEPA-style objective, and add the action as an external control term, which makes the update non-symplectic. Luan & Shi (2026) use port-Hamiltonian dynamics as a structural regularizer alongside the recurrent transition of a decoder-based Dreamer-style agent (Hafner et al., 2019; 2020). LeHami differs in that the action enters the Hamiltonian rather than an external control term, and every rollout step is exactly symplectic, so phase-space volume is preserved by construction, learned end to end from pixels without a decoder.

3 Empirical validation of Hamiltonian JEPA

We evaluate our approach on two pendulum environments (Fig. 3) rendered as 64×64 RGB images. In the conservative pendulum there is no actuation, and episodes vary across three rod lengths. In the actuated pendulum, a recorded sequence of scalar torques $(u_t)_t$, drawn uniformly, does work on the system, so energy is no longer conserved; episodes vary across the same three rod lengths and three masses, since under actuation the mass sets the response to the torque. In both environments the true dynamics preserve phase-space volume, which is the property LeHami builds in; the actuated pendulum tests it once energy conservation no longer holds. Appendix D.3 describes each in full.

3.1 Experimental setup

Methodology. LeHami is designed as a drop-in replacement for the predictor of a JEPA world model, so we compare against LeWorldModel (LeWM) (Maes et al., 2026a), the state-of-the-art among JEPA world models trained end-to-end from pixels, which reported gains over PLDM (Sobal et al., 2025) and DINO-WM (Zhou et al., 2024). LeWM's predictor is unstructured: a causal transformer that takes three context embeddings to predict the next one. LeHami is encoder-agnostic and it replaces only this predictor, plus the fixed SG stencil. The main text uses a temporal ConvNet (Tran et al., 2018) (Appendix Fig. 6) and Appendix E.2 repeats the comparison with a tiny ViT (Wu

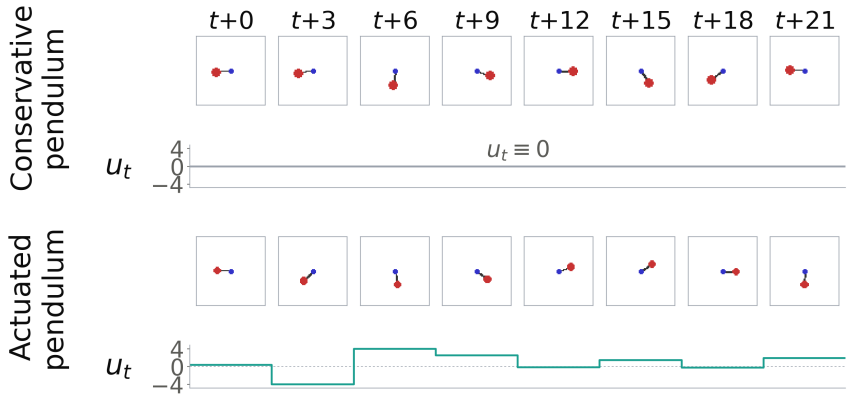

Figure 3: **The two pendulum environments. Top:** the conservative pendulum, with no actuation. **Bottom:** the actuated pendulum. Beneath each row is its respective torque u_t, drawn as the piecewise-constant signal, held constant for three frames.

et al., 2022). Moreover, LeHami is plugged on top of a frozen pre-trained encoder with DINOv2 features (Oquab et al., 2023), yielding DINO-LeHami. We do not compare against reward-driven agents such as Dreamer (Hafner et al., 2019) or TD-MPC (Hansen et al., 2022) variants, since they require task rewards whereas our setting, like LeWM's, is reward-free, goal-conditioned planning. More details on methodology in Appendix D.1.

Probing. The pendulum's state is fully described by the angle θ and the angular velocity ω, and we score their prediction along the rollout by MAE against ground truth; for θ we use MAE rather than R^2, whose normalising variance is undefined on the circle. Three probe tiers of increasing capacity read the latent: a ridge, a k-NN ($k = 25$) and a two-layer MLP. Each is fitted once, either on the encoder's own features (the anchor, K=0) or on the model's one-step output (K=1), and applied frozen. We report the results in the test split of episodes. The k-NN fitted at K=1 puts the models on equal footing because LeWM's encoder is per-frame, and only its predictor sees more than one time step.

Planning. For the actuated pendulum we perform, at inference time, optimization of the action sequence to minimize a latent goal objective $\mathcal{C}(\tilde{z}_H) = \left\|\tilde{\mathbf{z}}_H - \mathbf{z}_g\right\|^2$, where $\mathbf{z}_g$ is the goal state and $\tilde{\mathbf{z}}_H$ is the latent prediction at the horizon, which in the case of LeHami is $(\tilde{\mathbf{q}}_H, \tilde{\mathbf{p}}_H)$. The optimal sequence of actions is $\mathbf{a}^*_{1:H} = \texttt{argmin}_{\mathbf{a}_{1:H}} \mathcal{C}(\tilde{z}_H)$ and is obtained through the Cross-Entropy Method (CEM). We adopt the Model Predictive Control (MPC) strategy in which only the first R steps are executed before replanning from the updated observation. We measure the TFLOPs incurred during planning. Refer to Appendix D.4.2 for the full description of the planning protocol.

3.2 Evaluating Complete Phase-Space Physics Capture

For a mechanical system, position alone does not determine the future: the state is the pair of position and momentum. A world model that is evaluated only on position is therefore tested on half of the state. In this section we probe for both, the angle θ and the angular velocity ω, at the anchor and along the rollout, and ask whether each model's latent carries the *complete physical state*. Because LeHami enforces second-order phase-space structure by construction, its latent carries the velocity that a per-frame baseline cannot.

On the conservative pendulum, at the anchor, the kNN probe reads the angular velocity significantly better (see the anchor row in Table 7). One image does not show motion, so a per-frame encoder, as in baseline, cannot supply it. While LeHami also encodes per frame, it reads velocity from the Savitzky–Golay channel, which does not require any training. During rollout, as shown in Fig. 4 **(left)**, the two are level early and LeHami pulls ahead from $K \approx 11$ on the angle's prediction. Regarding the prediction of velocity, Fig. 4 **(middle)** shows that LeHami leads by $2.8\times$ over the

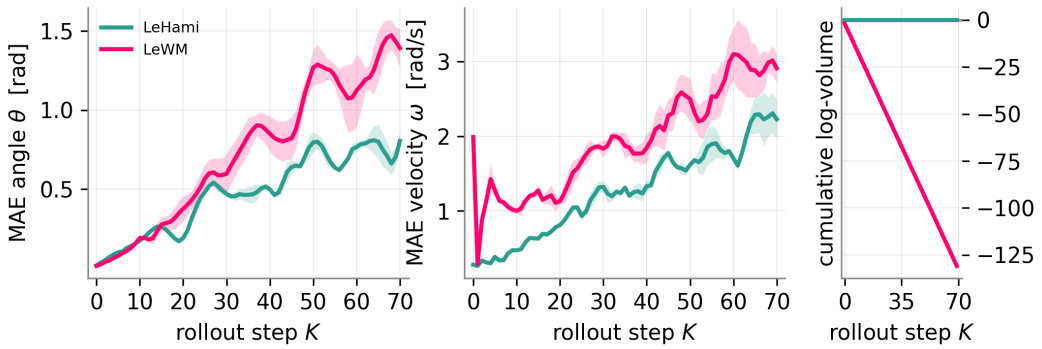

Figure 4: **Prediction performance during conservative pendulum rollouts.** Wrapped MAE of θ **(left)** and ω **(middle)** across the test rollout steps (K). Metrics are computed using a k-NN probe ($k = 25$) fitted at $K = 1$ and kept frozen. Shaded areas indicate variance across three training seeds, averaged over test episodes. Cumulative log-volume of the one-step latent map, averaged over test episodes, for one training seed **(right)**.

first ten steps and $1.3\times$ at the end of the horizon. The learned H_ϕ makes that each rollout stays on its own level set of H_ϕ, as the true energy confines the physical system, while LeWM's rollouts cross them (Appendix Figs. 10–11).

This advantage does not come from a larger model or a larger training budget: LeHami has lower θ and ω error than LeWM at every matched training budget, matching LeWM's best error with at most one-eighth of the training FLOPs (Appendix Table 4). The advantage also holds with a frozen pre-trained encoder: DINO-LeHami outperforms DINO-WM (Zhou et al., 2024) (Appendix Fig. 12).

To empirically assess whether LeHami avoids representation collapse over long horizons, we directly evaluate the phase-space volume contraction of the latent predictor during rollouts. Fig. 4 **(right)** tracks the normalized cumulative log-determinant across a K-step rollout, $\frac{1}{n}\sum_{i=0}^{K-1}\log|\det J(\mathbf{z}_i)|$, where $J(\mathbf{z}_i)$ denotes the Jacobian of the one-step predictor and n is the latent dimension. This quantity directly measures phase-space volume change under iterated predictions. As shown, LeHami maintains an exactly constant value of 0 ($|\det J(\mathbf{z}_i)| = 1, \forall \mathbf{z}_i$), avoiding rollout volume contraction. Conversely, LeWM exhibits a steady linear decline in cumulative log-volume, indicating continuous contraction during rollout; the same holds for the full phase-space volume of its three-embedding context (Appendix Fig. 13).

Table 1: **Performance in physics prediction of the actuated pendulum at the anchor K=0**. Mean $\pm$ s.d.over three training seeds, test split. Bold marks the better model within a probe tier. A negative R^2 is below the predict-the-mean floor.

model	probe	MAE θ [10^{-2} rad]	MAE ω [rad/s]	$R^2(\omega)$
LeHami	ridge	23.0 ± 2.0	$\mathbf{1.168 \pm 0.001}$	$\mathbf{+0.491 \pm 0.005}$
	25-NN	6.8 ± 1.8	$\mathbf{0.419 \pm 0.045}$	$\mathbf{+0.889 \pm 0.024}$
	MLP	2.9 ± 0.4	$\mathbf{0.184 \pm 0.019}$	$\mathbf{+0.987 \pm 0.004}$
LeWM	ridge	$\mathbf{1.4 \pm 0.1}$	2.422 ± 0.001	-0.014 ± 0.001
	25-NN	$\mathbf{1.1 \pm 0.0}$	2.508 ± 0.002	-0.103 ± 0.001
	MLP	$\mathbf{1.9 \pm 0.1}$	2.419 ± 0.001	-0.011 ± 0.001

When incorporating actions. The actuated pendulum shows the same pattern at the anchor. Table 1 evaluates the representation across three probe tiers of increasing capacity. LeHami's velocity is decoded well by the non-linear probes (k-NN, MLP) while for LeWM, all three probe tiers remain below the predict-the-mean floor. The angle exhibits the opposite trend: LeWM decodes θ more accurately, though the gap narrows at higher probe capacities. This is the expected consequence of a per-frame encoder as it contains only position, so the anchor latent encodes the angle well but carries

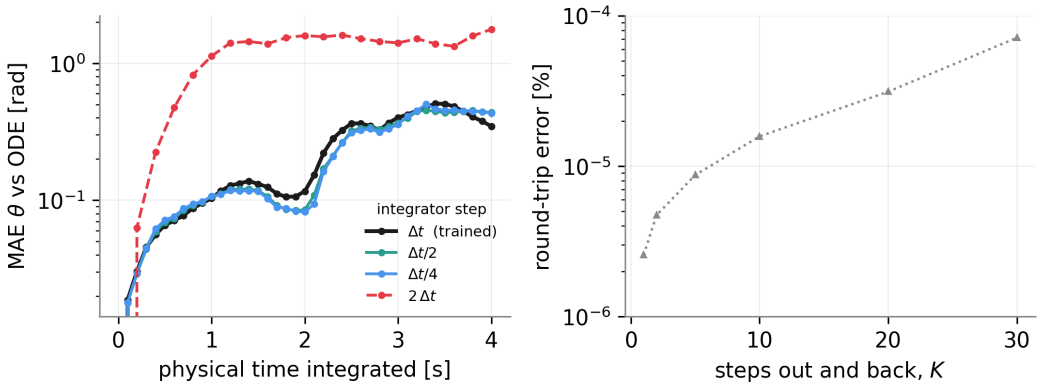

Figure 5: **The learned flow behaves like a continuous-time vector field, and is approximately reversible, showcased with the conservative pendulum. Left:** MAE of θ integrated with step sizes never seen in training (to the ground-truth from the integration of the ODE). The model is trained only at $\Delta t = 0.1$ s; halving and quartering the step (2 and 4 sub-steps per frame, respectively) traces the same trajectory, while doubling it breaks. **Right:** Round-trip error after rolling K steps out and back by sign flip $\Delta t \to -\Delta t$ in the H_ϕ.

no velocity. By contrast, LeHami structurally represents both canonical variables $(\mathbf{q}, \mathbf{p})$, reading velocity from the SG channel (Appendix Fig.8). Along the rollout these respective advantages decay (Appendix Fig. 15): LeHami maintains superior velocity prediction until $K \approx 10$, at which point the baseline's predictor has ingested enough context frames to implicitly infer motion; conversely, LeWM's initial advantage on θ decays by $K \approx 40$ as it suffers from compounding contraction.

3.3 LeHami is a world model with time re-stepping and reversal

As discussed, LeHami is a world model that runs at any rate, and in either direction, because the Hamiltonian predictor H_ϕ is a continuous-time vector field. We showcase this capability by holding the learned H_ϕ fixed and integrating it at $s\Delta t$, taking $1/s$ sub-steps per observed frame, and compare against a Runge–Kutta integration of the true ODE from the same initial condition. Refining below the observation rate leaves the predicted trajectory invariant (Fig. 5 (left)); coarsening breaks it, as an explicit symplectic integrator leaves its stability band. The same property makes the rollout reversible. Fig. 5 (right) shows the round-trip error of the latent state after rolling K steps forward and back with $\Delta t \to -\Delta t$, which stays below $10^{-4}\%$.

3.4 LeHami brings computational efficiency and structural priors in velocity planning

We evaluate goal-reaching performance on goals with position and velocity clauses, separately and together, under a strict computational budget of 15 TFLOPs per episode. Under this constraint, models must balance their per-solve computational cost against their replanning interval (R). Both LeWM and LeHami accurately decode the rod's mass and length along the whole rollout (Appendix Table 11). Since the torque's effect depends on mass and length (Appendix Eq.14), both models have the information needed to predict the response to an action. Differences in planning therefore do not stem from failed system identification.

Capturing velocity dynamics within standard JEPA frameworks requires explicit architectural asymmetry. As shown in Table 2, the baseline LeWM with ConvNet without time window (W=1) fails in the velocity clause because its per-frame goal embedding carries no velocity, even though its predictor infers it from context. Expanding the window with a symmetric (tied) kernel only captures speed while remaining direction-blind since it maps $\pm\omega$ to the same latent (Appendix Table 9). As 92% of the goals require the correct sign (Appendix Fig. 16), it performs below floor on the sign-resolved criterion of Appendix Table 10. In contrast, expanding with untied weights breaks temporal parity, allowing the model to capture signed velocity and achieve the highest success rate. However, this

Table 2: **Goal-reaching success on the actuated pendulum, by clause.** Best success probability per model at a matched budget of at most 15 TFLOP/episode. Mean ± s.d. over three model seeds. The floor probabilities (corresponding to $u = 0$) are 0.060, 0.240 and 0.040 on each of the three clauses respectively. Bold marks the better model within a clause. The planner re-solves every R steps towards the same goal at horizon 30, so it makes $\lceil 30/R \rceil$ solves, the last of which covers only the remaining steps.

Clause	Model	Success	TFLOP	Replanning interval
Position	LeWM W=1	$\mathbf{0.800 \pm 0.019}$	12.87	$R = 9$, 4 solves
	LeWM W=3 (tied)	0.633 ± 0.020	13.65	$R = 9$, 4 solves
	LeWM W=3 (untied)	0.696 ± 0.016	13.65	$R = 9$, 4 solves
	LeHami	0.480 ± 0.084	8.05	$R = 9$, 4 solves
Velocity	LeWM W=1	0.184 ± 0.027	8.78	$R = 15$, 2 solves
	LeWM W=3 (tied)	0.473 ± 0.014	13.65	$R = 9$, 4 solves
	LeWM W=3 (untied)	$\mathbf{0.800 \pm 0.058}$	13.65	$R = 9$, 4 solves
	LeHami	0.527 ± 0.019	10.98	$R = 6$, 5 solves
Position and Velocity	LeWM W=1	0.142 ± 0.033	12.87	$R = 9$, 4 solves
	LeWM W=3 (tied)	0.400 ± 0.037	9.31	$R = 15$, 2 solves
	LeWM W=3 (untied)	$\mathbf{0.636 \pm 0.025}$	13.65	$R = 9$, 4 solves
	LeHami	0.304 ± 0.088	10.98	$R = 6$, 5 solves

unrestricted temporal formulation incurs a high computational cost per solve. To remain under the 15 TFLOP budget, LeWM W=3 (untied) is forced into larger replanning intervals.

In contrast, LeHami structurally guarantees the representation of signed velocity by embedding the system's physics. Momentum-like embedding is read off the encoder trajectory by a zero-parameter SG stencil, and the Hamiltonian flow itself has 2.23M parameters, against LeWM's transformer and its predictor projection with 3.31M parameters (Appendix Table 5). This computational efficiency allows LeHami to evaluate more trajectories and replan more frequently (R=6 steps with 5 solves) on the velocity and joint clauses, consuming less TFLOPs. While LeWM W=3 (untied) achieves a higher success rate, LeHami provides a structurally rigorous alternative. Replacing the fixed SG stencil with a jointly learned $\mathbf{p}$ embedding offers a clear path to closing this performance gap while maintaining the volume-preserving guarantees.

4 Conclusion

We introduced LeHami, a world model architecture that rigorously preserves the geometric properties of second-order physical systems. Its exact symplectic step provides structural guarantees: exact volume preservation that provably eliminates some rollout collapse modes and continuous-time integration with time-reversal. We have shown empirically on the conservative pendulum that LeHami preserves latent volume along the rollout, whereas LeWM contracts it steadily, losing log-volume linearly with the horizon. Consistent with this, LeHami predicts velocity better throughout and angle better at long horizons. With actuation, LeWM reaches the goal angle more often, but it captures signed velocity only when its temporal window is explicitly untied. LeHami instead captures the full state, angle and signed velocity alike, by construction, paving the way towards velocity-constrained planning and FLOP-efficient replanning.

Limitations and future work. First, while our latent architecture maintains exact volume preservation under actuation, strongly dissipative physical systems (e.g., heavy friction) exhibit phase-space volume contraction. Thus, extending LeHami to these regimes e.g., via port-Hamiltonian extensions (Appendix C.4) is a natural direction for future work. Second, LeHami specifically targets velocity-aware tasks, whereas purely static pose tasks are served well by spatial-focused, unconstrained predictors. Finally, replacing our zero-overhead SG stencil with a learned momentum-like embedding offers a clear path to potentially improve performance without sacrificing long-horizon phase-space stability.

AI USE STATEMENT

AI was used for the writing of some parts of the manuscript and for some parts of the code development used to run the experiments. Moreover, AI was used to check the proof of Proposition 2. The authors take full responsibility for the scientific contributions and all written content.

ETHICS STATEMENT

This work aims to advance fundamental machine learning research, specifically in the domain of world models. As foundational research, we do not foresee any immediate societal impacts associated with this work.

REPRODUCIBILITY STATEMENT

The claims of this paper are supported by the appendices and by an anonymous code release at `https://anonymous.4open.science/r/hamiltonian-jepa-7F08`. *Theory.* Proposition 1 is stated with its assumptions in Section 2 and proved in full in Appendix C.1; Appendix C.2 sets out what the guarantee does and does not imply. *Models and training.* What is held fixed between LeHami and the baselines, and how checkpoints are selected, is in Appendix D.1. The architecture as implemented and the parameter counts measured from the checkpoints are in Appendix D.2 and Table 5, and the remaining design choices (the SG stencil, the scale gauge and the SIGReg placements) are in Appendix C.4. Every trained model corresponds to one configuration file in the release, which fixes all its hyperparameters, and the resolved configuration is saved with each checkpoint. The third-party code we build on is pinned to the exact commits we used. *Data.* Both environments are specified by their equations of motion, rendering and generation parameters in Appendix D.3. The episode-level train/validation/test protocol, including the generator seed and the check that no episode is shared across splits, is in Table 3. The release contains the seeded generators and the split manifests that training and evaluation both read, and each manifest records the hash of the dataset file it was cut from, so a regenerated dataset can be verified. *Evaluation.* The probes are described in Appendix D.4.1, and the planning protocol, with its goal construction, CEM optimiser, leakage checks and success criterion, is in Appendix D.4.2. Results are reported as the mean and standard deviation over three training seeds unless stated otherwise. *Code.* One runner script per experimental block trains and evaluates every model and seed reported in the paper, and the planning and compute analyses reuse its checkpoints. A single notebook regenerates every figure and table from the frozen numbers shipped with it, without a GPU, the datasets or retraining.

REFERENCES

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

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. URL `https://arxiv.org/abs/2511.08544`.

Randall Balestriero, Hugues Van Assel, Sami BuGhanem, and Lucas Maes. stable-pretraining-v1: Foundation model research made simple. *arXiv preprint arXiv:2511.19484*, 2025.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *arXiv preprint arXiv:1909.13334*, 2019.

Sen Cui and Jingheng Ma. Physically native world models: A hamiltonian perspective on generative world modeling. *arXiv preprint arXiv:2605.00412*, 2026.

Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

Thai Duong and Nikolay Atanasov. Hamiltonian-based neural ode networks on the se (3) manifold for dynamics learning and control. *arXiv preprint arXiv:2106.12782*, 2021.

Herbert Goldstein. *Classical mechanics*. Pearson Education India, 2011.

Peter A Gorry. General least-squares smoothing and differentiation by the convolution (savitzky-golay) method. *Analytical Chemistry*, 62(6):570–573, 1990.

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

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the störmer–verlet method. *Acta numerica*, 12:399–450, 2003.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522. IEEE, 2017.

Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Xueyu Luan and Chenwei Shi. Ph-dreamer: A physics-driven world model via port-hamiltonian generative dynamics. *arXiv preprint arXiv:2605.18303*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026a. URL https://arxiv.org/abs/2603.19312.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation, 2026b. URL https://arxiv.org/abs/2605.21800.

Jerrold E Marsden, Tudor S Ratiu, and M Golubitsky. *Introduction to mechanics and symmetry: a basic exposition of classical mechanical systems*, volume 17. Springer, 1999.

Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

Chee Keong Tan, Michael James Tippett, and Jie Bao. Model predictive control with non-uniformly spaced optimization horizon for multi-timescale processes. *Computers & Chemical Engineering*, 84:162–170, 2016.

Haoyun Tang, Haodong Cui, Keyao Xu, Kun Wang, and Zhandong Mei. Ham-world: Soft-hamiltonian world models with selective memory for planning. *arXiv preprint arXiv:2605.05951*, 2026.

Molei Tao. Explicit symplectic approximation of nonseparable hamiltonians: Algorithm and long time performance. *Physical Review E*, 94(4):043303, 2016.

Peter Toth, Danilo J Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International conference on learning representations*, 2019.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 6450–6459. IEEE, 2018.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, pp. 68–85. Springer, 2022.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319, 2020.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative SymODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Appendix organization

Appendix B collects the background on the Hamiltonian mechanics we use, including phase-space density and the symplectic form. Appendix C contains the theory: the formal statement and proof that the leapfrog step is exactly symplectic (§C.1); the scope of that guarantee, in three remarks on what volume preservation does and does not imply (§C.2); why we build on the Hamiltonian rather than the Lagrangian formulation (§C.3); and the design choices in the flow and the encoder (§C.4): the shape of the Hamiltonian, the SG stencil and its causal and centred windows, why no mass is learned, the learned scale gauge, and the SIGReg placements. Appendix D gives the experimental set-up in full: what is held fixed across models and how checkpoints are selected (§D.1); the implementation of LeHami (§D.2), covering the Hamiltonian flow, the per-frame and temporal encoders, the stencil at training and inference, and an extension to training across recording rates that the architecture supports but that we do not exercise here; the two datasets and their split protocol (§D.3); and the evaluation, namely the probes (§D.4.1) and the planning protocol (§D.4.2), with its goal construction, the frames each latent state is read from, the optimiser, the leakage checks and the success criterion. Appendix E reports supplementary results. It first shows that the Hamiltonian structure replaces manual design choices at lower training compute and without extra parameters (§E.1). It then gives the full results on the conservative pendulum (§E.2): the geometry of the latent space against the learned and physical energies, rollout prediction across probe tiers, independence from the encoder with a frozen DINOv2, and the phase-space volume of the one-step map. Finally, it gives the full results on the actuated pendulum (§E.3): the learned energy under actuation, where to apply the regularizer, rollout prediction, how a temporal window

in the encoder captures velocity, the goal distribution, the planning compute, a decomposition of the velocity clause of the planning criterion into speed and direction, and the prediction of the rod's length and mass.

B Background in Hamiltonian mechanics

A physical Hamiltonian system with n degrees of freedom evolves on a phase space $\mathcal{M} = \mathbb{R}^{2n}$, coordinatized by generalized positions $\mathbf{q} \in \mathbb{R}^n$ and conjugate momenta $\mathbf{p} \in \mathbb{R}^n$ (Arnold, 1989). Its evolution is generated by the Hamiltonian, which is a scalar function $H : \mathbb{R}^{2n} \to \mathbb{R}$ that represents the system's total energy, through Hamilton's equations

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}, \qquad \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}. \tag{6}$$

The two main consequences of this that we use in this work are the following. The flow ϕ_t of Eq. 6 is *symplectic*: it preserves $\Omega = \sum_i dq_i \wedge dp_i$, and hence the phase-space volume due to Liouville's theorem (Arnold, 1989). Thus, a phase-space density ρ is transported without distortion, $\partial_t \rho = -\{\rho, H\}$ with $\{\cdot, \cdot\}$ the Poisson bracket.

Phase-space density. Rather than a single initial condition one works with a density $\rho(\mathbf{q}, \mathbf{p}, t)$ normalised to unit mass. Its evolution obeys the Liouville equation $\partial_t \rho = -\{\rho, H\}$, where $\{\rho, H\} = \partial_{\mathbf{q}} \rho \cdot \partial_{\mathbf{p}} H - \partial_{\mathbf{p}} \rho \cdot \partial_{\mathbf{q}} H$. Equivalently $d\rho/dt = 0$: the density seen by an observer moving with the flow is constant, so the flow is an incompressible deformation of phase space.

Symplectic form. The two-form $\Omega = \sum_{i=1}^{n} dq_i \wedge dp_i$ is closed and non-degenerate, and the Hamiltonian flow ϕ_t preserves it exactly. The wedge product records oriented area in the (q_i, p_i) plane and changes sign under exchange of its arguments, which is the canonical asymmetry between position and momentum. Preservation of Ω implies preservation of the volume form $\Omega^n/n!$ (Liouville's theorem): a cloud of initial conditions may stretch, rotate and shear, but its volume is invariant. For N particles the form extends additively, $\Omega = \sum_{j=1}^{N} \sum_{i=1}^{n} dq_i^{(j)} \wedge dp_i^{(j)}$, and the statement holds on the full $2nN$-dimensional space.

C Theory

C.1 Symplecticity of the leapfrog step: formal statement and proof

Proposition 1 is stated in the main text in geometric terms. Here we give the regularity hypotheses, render the three consequences in explicit form. Throughout, $\mathbf{z} = (\mathbf{q}, \mathbf{p}) \in \mathbb{R}^{2d}$ and

$$\mathbf{J} \;=\; \begin{pmatrix} \mathbf{0} & \mathbf{I}_d \\ -\mathbf{I}_d & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{2d \times 2d}, \qquad \mathbf{J}^\top = -\mathbf{J} = \mathbf{J}^{-1}.$$

A C^1 map $\Psi : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ is *symplectic* if $D\Psi(\mathbf{z})^\top \mathbf{J}\, D\Psi(\mathbf{z}) = \mathbf{J}$ for every $\mathbf{z}$, where $D\Psi(\mathbf{z}) \in \mathbb{R}^{2d \times 2d}$ denotes the Jacobian of Ψ at $\mathbf{z}$, i.e., $[D\Psi(\mathbf{z})]_{ij} = \partial \Psi_i / \partial z_j$.

Proposition 2 (Formal form of Proposition 1)**.** *Let $K_{\phi_1} \in C^2(\mathbb{R}^d)$ and $V_{\phi_2} \in C^2(\mathbb{R}^d \times \mathcal{U})$, and let Ψ_h be the leapfrog step of Eq. 4 generated by $H_\phi = K_{\phi_1}(\mathbf{p}) + V_{\phi_2}(\mathbf{q}, \mathbf{u})$ with the control $\mathbf{u}$ held fixed within the step. This means that the leapfrog step can be written as $\Psi_h = A_{h/2} \circ B_h \circ A_{h/2}$ with*

$$A_{h/2}(\mathbf{q}, \mathbf{p}) = \big(\mathbf{q},\ \mathbf{p} - \tfrac{h}{2} \nabla_{\mathbf{q}} V_{\phi_2}(\mathbf{q}, \mathbf{u})\big), \qquad B_h(\mathbf{q}, \mathbf{p}) = \big(\mathbf{q} + h \nabla_{\mathbf{p}} K_{\phi_1}(\mathbf{p}),\ \mathbf{p}\big). \tag{7}$$

Then Ψ_h is a C^1 diffeomorphism of $\mathbb{R}^{2d}$ that is exactly symplectic, which means that

$$D\Psi_h(\mathbf{z})^\top \mathbf{J}\, D\Psi_h(\mathbf{z}) \;=\; \mathbf{J} \qquad \textit{for every } \mathbf{z} \in \mathbb{R}^{2d}, \tag{8}$$

for every step size $h > 0$, every control $\mathbf{u} \in \mathcal{U}$ and every parameter $\phi = (\phi_1, \phi_2)$. Consequently $\det D\Psi_h \equiv 1$, Lebesgue measure volume on $\mathbb{R}^{2d}$ is Ψ_h-invariant, and for every horizon $K \in \mathbb{N}$ the three below conditions are met:

(i) (no uniform contraction) *the singular values of* $D(\Psi_h^K)(\mathbf{z})$ *with* $\Psi_h^K := \Psi_h \circ \cdots \circ \Psi_h$ *satisfy* $\prod_{i=1}^{2d} \sigma_i = 1$*, hence* $\sigma_{\max} \geq 1 \geq \sigma_{\min}$ *at every* $\mathbf{z} \in \mathbb{R}^{2d}$*;*

(ii) (no asymptotically stable fixed point) Ψ_h *admits no asymptotically stable fixed point, where "asymptotically stable" means: a fixed point* $\mathbf{z}^*$ *that is Lyapunov stable and attracts every point of some open neighbourhood of itself;*

(iii) (no merging) Ψ_h^K *is a composition of diffeomorphisms, hence itself a diffeomorphism of* $\mathbb{R}^{2d}$*, in particular injective. Consequently, if* $\mathbf{z}_1 \neq \mathbf{z}_2$ *then* $\Psi_h^K(\mathbf{z}_1) \neq \Psi_h^K(\mathbf{z}_2)$ *for every* $K \in \mathbb{N}$*: two latents that differ at the anchor remain distinct at every horizon.*

Proof. We start by showing that each of the half-steps are symplectic shears. For this, we fix $\mathbf{u} \in \mathcal{U}$ and from Eq. 7 we compute their Jacobians as

$$DA_{h/2}(\mathbf{q}, \mathbf{p}) = \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ -\frac{h}{2} \nabla^2_{\mathbf{qq}} V_{\phi_2}(\mathbf{q}, \mathbf{u}) & \mathbf{I}_d \end{pmatrix}, \qquad DB_h(\mathbf{q}, \mathbf{p}) = \begin{pmatrix} \mathbf{I}_d & h \nabla^2_{\mathbf{pp}} K_{\phi_1}(\mathbf{p}) \\ \mathbf{0} & \mathbf{I}_d \end{pmatrix},$$

which are well defined and continuous since $K_{\phi_1}, V_{\phi_2} \in C^2$. Both are unipotent block-triangular, so $\det DA_{h/2} = \det DB_h = 1$. By Clairaut's theorem, the C^2 regularity implies that the Hessians $\nabla^2_{\mathbf{qq}} V_{\phi_2}$ and $\nabla^2_{\mathbf{pp}} K_{\phi_1}$ are symmetric. This symmetry gives $DA_{h/2}^\top \mathbf{J}\, DA_{h/2} = \mathbf{J}$ and $DB_h^\top \mathbf{J}\, DB_h = \mathbf{J}$. Thus, both maps are symplectic. Separately, each map has an explicit global C^1 inverse: $A_{h/2}^{-1} = A_{-h/2}$ and $B_h^{-1} = B_{-h}$. Hence $A_{h/2}$ and B_h are C^1 diffeomorphisms of $\mathbb{R}^{2d}$. Together, these facts show that both maps are exactly symplectic C^1 diffeomorphisms with unit Jacobian determinant, for every $h > 0$, $\mathbf{u} \in \mathcal{U}$ and ϕ.

Now we show that these properties survive composition: given Ψ, Φ C^1 diffeomorphisms of $\mathbb{R}^{2d}$, exactly symplectic, thus $\Psi \circ \Phi$ is. The chain rule shows that $D(\Psi \circ \Phi)^\top \mathbf{J}\, D(\Psi \circ \Phi) = D\Phi^\top (D\Psi^\top \mathbf{J}\, D\Psi) D\Phi = D\Phi^\top \mathbf{J}\, D\Phi = \mathbf{J}$, and $\det D(\Psi \circ \Phi) = \det D\Psi \cdot \det D\Phi = 1$. Thus, a composition of diffeomorphisms is a diffeomorphism.

With these two properties we can affirm that $\Psi_h = A_{h/2} \circ B_h \circ A_{h/2}$ is thus C^1 diffeomorphism of $\mathbb{R}^{2d}$ and exactly symplectic. Likewise, every iterate $\Psi_h^K := \Psi_h \circ \cdots \circ \Psi_h$ (K times) is a composition of C^1 diffeomorphisms of $\mathbb{R}^{2d}$, exactly symplectic. Thus, Ψ_h^K also is, with $\det D(\Psi_h^K) \equiv 1$, for every $h > 0$, $K \in \mathbb{N}$, $\mathbf{u} \in \mathcal{U}$, ϕ. With this, the proof for the three conditions are straightforward.

Proof of (i). $\prod_{i=1}^{2d} \sigma_i(M) = |\det M|$ for any square matrix M. By taking $M = D(\Psi_h^k)(\mathbf{z})$ and using $\det D(\Psi_h^k) \equiv 1$, it follows that $\prod_{i=1}^{2d} \sigma_i = 1$, which means that $\sigma_{\max} \geq 1 \geq \sigma_{\min}$ at every $\mathbf{z} \in \mathbb{R}^{2d}$.

Proof of (ii). Suppose $\mathbf{z}^*$ is an asymptotically stable fixed point of Ψ_h, attracting an open neighbourhood U. Choose a closed ball $C = \overline{B_r(\mathbf{z}^*)} \subset U$ with $r > 0$. Lyapunov stability and attraction imply uniform convergence on C: for any $\varepsilon > 0$, stability gives an open neighbourhood W of $\mathbf{z}^*$ whose forward iterates stay in $B_\varepsilon(\mathbf{z}^*)$. Each point of C enters W after finitely many steps; by continuity, a neighbourhood of that point enters W at the same step. Compactness gives a finite cover of C by such neighbourhoods, so taking the largest of their entry times gives K_ε such that

$$\Psi_h^K(C) \subseteq B_\varepsilon(\mathbf{z}^*) \qquad \text{for all } K \geq K_\varepsilon.$$

Volume preservation then gives

$$0 < \mathrm{vol}(C) = \mathrm{vol}(\Psi_h^K(C)) \leq \mathrm{vol}(B_\varepsilon(\mathbf{z}^*)) = c_{2d} \varepsilon^{2d},$$

where c_{2d} is the volume of the unit ball in $\mathbb{R}^{2d}$. Letting $\varepsilon \to 0$ yields a contradiction. Hence no such fixed point exists.

Proof of (iii). As shown, Ψ_h^k is a diffeomorphism of $\mathbb{R}^{2d}$, which means bijective, and hence injective. As a consequence, $\mathbf{z}_1 \neq \mathbf{z}_2 \implies \Psi_h^k(\mathbf{z}_1) \neq \Psi_h^k(\mathbf{z}_2)$ for every $k \in \mathbb{N}$. □

C.2 Scope and limitations of the guarantee

Remark 3 (Shape, not size)**.** A cloud that cannot be squeezed can still be sheared and stretched arbitrarily, so symplecticity places no bound on the growth of rollout error and does not make the

rollout accurate. What it removes is the degenerate failure in its three forms: the rollout that shrinks in every direction at once, the one that settles onto an attractor and stops moving, and the one that merges states which started apart.

An unconstrained predictor F_θ is under no such constraint, and the one-step objective pushes it the other way. Under any observation or latent noise, the minimiser of $\mathbb{E}\|F_\theta(\mathbf{z}_t) - \mathbf{z}_{t+1}\|^2$ is the conditional mean $\mathbb{E}[\mathbf{z}_{t+1} \mid \mathbf{z}_t]$, which averages away whatever the anchor leaves undetermined: it contracts volume, and it sends latents that differ only in the averaged-out directions to a common image. Volume loss and merging are therefore the trained optimum for such a map and iterating it drives the rollout onto exactly the attracting set the proposition forbids for Ψ_h. The proposition says the Hamiltonian rollout cannot enter that regime at any step size h, any parameter ϕ or any horizon.

Remark 4 (Regularization is not made redundant)**.** Proposition 1 states that the flow carries phase-space volume exactly and its does not state nothing about the shape of the distribution that volume belongs to. Due to this, regularization techniques are not made redundant. In this work, we leverage the SIGReg (Balestriero & LeCun, 2025) as regularizer, as it is the one used in LeWM and has shown state-of-the-art performance (Maes et al., 2026a). For the isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ installed by SIGReg at the encoder to be invariant under the flow, the radius $\|(\mathbf{q}, \mathbf{p})\|^2$ would itself have to be a conserved quantity of the generator, that is $\{\|(\mathbf{q}, \mathbf{p})\|^2, H_\phi\} = 0$. This is a genuine restriction: it is a functional constraint on H_ϕ, not a consequence of symplecticity, and a general learned Hamiltonian, which is action-conditioned, and parameterised by free-form networks as we do here, has no reason to satisfy it.

The two mechanisms are therefore complementary rather than substitutes. SIGReg shapes the encoder distribution, during training and in expectation over batches, which is where a collapse of the representation would originate. Symplecticity carries the volume of whatever distribution it is handed, at every inference step and for every input, which is where an iterated map would otherwise contract or explode.

Remark 5 (Geometric prior, no physical modeling)**.** We are not modeling the physics of the observed system. H_ϕ is never fit to a known Hamiltonian and is not claimed to recover one: the true equations of motion enter only as evaluation targets for frozen read-out probes, never as a training signal. This is the point of departure from Hamiltonian neural networks (Greydanus et al., 2019) and SRNN (Chen et al., 2019), where the learned H_ϕ is a surrogate for the physical Hamiltonian and recovering it is the goal.

The prior models the latent representation space. Writing $\mathcal{F}$ for the maps a JEPA predictor may realize, we restrict it to the symplectomorphisms reachable as one leapfrog step of a learned separable generator $H_\phi(\mathbf{q}_t, \mathbf{p}_t, \mathbf{u}_t)$; the content of that restriction is geometric (Proposition 1), and it governs the dynamics, or evolution, latent space. The canonical split is structural in the same sense: the encoder emits $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{p}_t)$ with $\mathbf{q}_t, \mathbf{p}_t \in \mathbb{R}^d$ and the flow operates on that split by construction, but $\mathbf{q}_t$ is not assumed to encode position, nor $\mathbf{p}_t$ momentum, in any observation-space sense.

C.3 Why Hamiltonian and not Lagrangian

The two formalisms describe the same mechanics in different coordinates, so the choice is made on what each buys a predictor, and two things separate them here. First, the guarantee we rely on is a statement about phase space: symplecticity, and the volume preservation it implies (Proposition 1), are properties of a flow on $(\mathbf{q}, \mathbf{p})$ and have no equally cheap counterpart on the tangent bundle $(\mathbf{q}, \dot{\mathbf{q}})$ where a Lagrangian lives. Second, the Euler–Lagrange equations are second order in $\mathbf{q}$ while Hamilton's equations are first order on phase space and a separable H_ϕ makes the leapfrog step explicit, which is what keeps the map exactly symplectic at any h rather than approximately so.

C.4 Design choices in the flow and the encoder

The shape of the Hamiltonian. In principle, there are multiple ways to structure the Hamiltonian. The simplest Hamiltonian is a separable one, where the kinetic term K_ϕ depends only on the momentum-like coordinates and the potential term V_ϕ depends only on the position-like coordinates and the action. This is a common choice in classical mechanics, and it allows for efficient integration using the leapfrog method, as discussed before. In this work, we focus on separable Hamiltonians, but more general forms can be considered as well.

We consider an action-conditioned Hamiltonian parametrized as H_ϕ, where the action $\mathbf{u}_t$ represents an external control input that influences the evolution of the latent state. This can be interpreted as inducing an action-dependent vector field, where the action modulates the dynamics through the Hamiltonian. We condition the potential on the action:

$$H_\phi(\mathbf{q}, \mathbf{p}, \mathbf{u}) \;=\; K_{\phi_1}(\mathbf{p}) \;+\; V_{\phi_2}(\mathbf{q}, \mathbf{u}), \tag{9}$$

so that the generalized force is $\mathbf{F}(\mathbf{q}, \mathbf{u}) = -\partial V_{\phi_2}/\partial \mathbf{q}$ and the dynamics remain in strict Hamiltonian form for every fixed action.

Let the action be piecewise constant, taking value $\mathbf{u}_t$ on step t, and write $\mathbf{z}_t = (\mathbf{q}_t, \mathbf{p}_t)$. Decompose the change in the Hamiltonian across a step into the part due to the action changing and the part due to the flow,

$$H_\phi(\mathbf{z}_{t+1}, \mathbf{u}_{t+1}) - H_\phi(\mathbf{z}_t, \mathbf{u}_t) \;=\; \underbrace{H_\phi(\mathbf{z}_{t+1}, \mathbf{u}_{t+1}) - H_\phi(\mathbf{z}_{t+1}, \mathbf{u}_t)}_{W_t} \;+\; \underbrace{H_\phi(\mathbf{z}_{t+1}, \mathbf{u}_t) - H_\phi(\mathbf{z}_t, \mathbf{u}_t)}_{r_t}, \tag{10}$$

where W_t is the work performed by the change of action, which is a physical transfer of energy that any actuated model must reproduce, and r_t is the integrator residual. Summing, $H_\phi(\mathbf{z}_K, \mathbf{u}_K) - H_\phi(\mathbf{z}_0, \mathbf{u}_0) = \sum_t W_t + \sum_t r_t$, so the model's energy tracks the accumulated work up to $\sum_t r_t$.

Placing the prior on the latent map rather than on the observational world (see Remark 5) does not make it vacuous. If the observed dynamics contract phase-space volume, no dimensionally-matched re-coordinatization admits a symplectic latent map that reproduces them. While an over-parameterized symplectic space could theoretically embed volume-contracting data by expanding orthogonal auxiliary dimensions to satisfy Liouville's theorem, doing so forces the model to learn unphysical, exponentially diverging latent variables. Without such artificial dimensional expansion, dissipation is the primary property of the observed system that a purely symplectic representation cannot absorb.

Because both systems studied in this work preserve volume, we leave dissipative modeling to future work. This extension takes the same structured form as the rest of the model: introducing a learned matrix $D_{\phi_3} = L_{\phi_3} L_{\phi_3}^\top \succeq 0$ into the momentum equation, which is positive semi-definite by construction so that it can only dissipate, smoothly recovering the symplectic model at $L_{\phi_3} = \mathbf{0}$.

The encoder and the SG stencil. Our proposal to model the dynamics in the latent representation space with a Hamiltonian flow is independent of the encoder architecture. However, in the self-supervised setting, the embeddings are being learned, together with its Hamiltonian evolution. For this reason, the encoder has to be sufficiently expressive to encode canonical coordinates. Encoding $(\mathbf{q}, \mathbf{p})$ is not just separating the embeddings as $(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{z}$, in which we just split the embedding into two.

We enforce $\mathbf{q}$ and $\mathbf{p}$ to be canonical by enforcing that the momentum-like coordinates are the time derivative of the position-like coordinates ($p \equiv \dot{q}$ with unit mass). This is a common assumption in classical mechanics, where the momentum is defined as the derivative of the Lagrangian with respect to the velocity. We can enforce this by making the encoder produce the momentum-like coordinates as the differentiation of the position-like coordinates. Specifically, we can do this with numerical methods. In this work, we use a Savitzky–Golay filter (Gorry, 1990) to first approximate the position-like coordinates with a polynomial and then compute the momentum-like coordinates as the derivative.

Concretely, $\mathbf{p}_t$ is a fixed linear functional of the encoder trajectory around frame t. We fit a quadratic in time to the encoded positions in a window by least squares whose coefficients are fixed in closed form by the window offsets and polynomial order alone. The least-squares fit sees the encoder values only as its right-hand side, never in its design matrix. Then, its derivative is evaluated at t, which for a fixed window is a constant stencil. Three properties of this choice matter. The estimate is at frame t rather than half a step behind it, unlike a backward difference, so $\mathbf{q}_t$ and $\mathbf{p}_t$ refer to the same instant and the pair is canonical in the only sense we can enforce. The stencil is linear in the encoder output with constant coefficients (i.e., no training involved), so it adds no parameters and cannot itself collapse: whatever the momentum block carries, it carries because the encoder trajectory moves. And a least-squares fit over several frames attenuates the high-frequency amplification that differentiation applies to a noisy signal, which a two-point difference does not, at the price of a truncation bias.

At the rollout anchor, in training and at inference alike, we use a four-frame causal window, since a centred window would read the frames the model is about to predict; the flow then evolves with no further stencil. The momentum targets used in training are computed with a five-frame centred window, $t-2,\dots,t+2$, on the encoded $\mathbf{q}$'s. The causality constraint does not apply to the targets: they are read off the encoder trajectory only to be compared with the rollout, never fed into it, so frames on both sides of t are available and using them leaks nothing. Given that choice, the centred window is the better estimator on both counts of the bias–variance trade-off above. Its weights, $(-2,-1,0,1,2)/10$, are antisymmetric in time, which cancels every even-order error term, so the truncation bias starts at $\frac{3.4}{6}\,\dddot{\mathbf{q}}\,\Delta t^2$ with no Δt^3 term, against $\frac{4.7}{6}\,\dddot{\mathbf{q}}\,\Delta t^2$ plus a Δt^3 term for the four-frame causal window. It also amplifies independent per-frame encoder noise $24.5\times$ less (a variance gain of $\sum_m w_m^2 = 0.10$ against 2.45). Since the targets enter the loss at every rollout step, a noisier target would inject that variance into every gradient. The antisymmetry has one further consequence: reversing time maps $\mathbf{p}_t \mapsto -\mathbf{p}_t$ exactly, the transformation law of a momentum under time reversal, which a one-sided window does not satisfy.

Why there is no mass. In classical mechanics the momentum is $\mathbf{p} = \mathbf{M}\dot{\mathbf{q}}$ and the kinetic energy is $\frac{1}{2}\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p}$, so a physical model must know the mass matrix $\mathbf{M}$. LeHami learns no mass, and the reason is the stencil: defining $\mathbf{p} \equiv \dot{\mathbf{q}}$ already fixes $\mathbf{M} = \mathbf{I}$ in the learned coordinates. To see it, combine the definition with Hamilton's first equation, $\dot{\mathbf{q}} = \nabla_{\mathbf{p}} H_\phi$. On the data, a flow that reproduces the encoder trajectory must satisfy

$$\nabla_{\mathbf{p}} H_\phi(\mathbf{q},\mathbf{p},\mathbf{u}) \;=\; \mathbf{S}^{-1}\nabla K_\phi\big(\mathbf{S}^{-1}\mathbf{p}\big) \;=\; \mathbf{p}, \tag{11}$$

and integrating in $\mathbf{p}$ gives

$$K_\phi\big(\mathbf{S}^{-1}\mathbf{p}\big) \;=\; \tfrac{1}{2}\|\mathbf{p}\|^2 + C_\phi, \tag{12}$$

the kinetic energy of unit mass, whatever $\mathbf{S}$ is. A learned $\mathbf{M}$ would therefore be redundant rather than missing. For a system of constant mass the encoder can absorb $\mathbf{M}$ through the linear re-coordinatization $\mathbf{q} \mapsto \mathbf{M}^{1/2}\mathbf{q}$, under which $\frac{1}{2}\dot{\mathbf{q}}^\top\mathbf{M}\dot{\mathbf{q}}$ becomes $\frac{1}{2}\|\dot{\mathbf{q}}\|^2$; this is a case of the latent-map freedom of Remark 5. This is also why the scale $\mathbf{S}$ below stays close to the identity. Two caveats. First, the argument constrains K_ϕ only on the support of the data and only to the accuracy of the stencil and of the leapfrog position step (which uses ∇K_ϕ at the half-step momentum), so we keep K_ϕ as a 3-layer MLP, following Hamiltonian neural networks (Greydanus et al., 2019). The quadratic is what that MLP must approximate at convergence, not a constraint we impose. Second, the re-coordinatization exists only when $\mathbf{M}$ is constant. When the mass matrix depends on configuration, $\mathbf{M}(\mathbf{q})$ as in a double pendulum, no global linear change of coordinates removes it, $\mathbf{p} = \dot{\mathbf{q}}$ is a velocity rather than a canonical momentum, and a separable $K_\phi(\mathbf{p})$ is mis-specified. Lifting this requires learning $\mathbf{p}$ jointly instead of reading it off with a fixed stencil.

Adaptation to the self-supervised setting. In the supervised setting, $(\mathbf{q},\mathbf{p})$ are the canonical coordinates of the physical system. Here, in the self-supervised setting, these coordinate-like embeddings are learned jointly with their evolution, so nothing fixes the relative scale of $\mathbf{q}$ and $\mathbf{p}$. Rescaling the momentum, $\mathbf{p} \mapsto \mathbf{S}\mathbf{p}$, together with $K_\phi(\cdot) \mapsto K_\phi(\mathbf{S}^{-1}\,\cdot)$, leaves the trajectory of $\mathbf{q}$ unchanged, so this scale is a gauge freedom that the objective cannot determine. This scaling is learned: $\mathbf{S} = \mathrm{diag}(e^{\boldsymbol{\sigma}})$ with $\boldsymbol{\sigma}$ learned (Eq. 3) and the leapfrog of H_ϕ in Eq. 4 remains exactly symplectic; $\mathbf{S}$ enters only the position step, through $\nabla_{\mathbf{p}} H_\phi = \mathbf{S}^{-1}\nabla K_\phi(\mathbf{S}^{-1}\mathbf{p})$.

$\boldsymbol{\sigma}$ is initialised at $\mathbf{0}$, so $\mathbf{S} = \mathbf{I}$, and is learned jointly with all other parameters (encoder, projector, K_ϕ and V_ϕ) by back-propagating the multi-step rollout loss through the leapfrog updates of Eq. 4. Since $\mathbf{S}$ enters only through K_ϕ, its gradient arrives through the position steps, via $\nabla_{\mathbf{p}} H_\phi = \mathbf{S}^{-1}\nabla K_\phi(\mathbf{S}^{-1}\mathbf{p})$, and hence through every later prediction of the rollout. In practice $\mathbf{S}$ remains close to the identity (diagonal entries between 0.95 and 1.09 across our trained models), since a per-dimension rescaling of $\mathbf{p}$ can equally be absorbed by the input weights of the MLP K_ϕ. Because $\mathbf{S}$ enters through H_ϕ rather than being inserted into the standard leapfrog of H_ϕ (Eq. 4), $\mathbf{S}$ appears only in the position step because $\nabla_{\mathbf{p}} H_\phi = \mathbf{S}^{-1}\nabla K_\phi(\mathbf{S}^{-1}\mathbf{p})$ while $\nabla_{\mathbf{q}} H_\phi = \nabla_{\mathbf{q}} V_\phi$. This makes the map to remain exactly symplectic.

Regularization. As discussed in the main text, in principle, the regularization can be applied in different ways over the embeddings produced by LeHami: $(\mathbf{q},\mathbf{p})$. Which is best is an empirical question our experimental results discuss. We apply step-wise SIGReg implemented by LeWM (Maes et al., 2026a); they differ only in which embedding block contains. We show each of them below and they are the $\mathcal{S}$ in the training loss (Eq. 5).

We index the frames of a training clip by $0, \dots, T-1$, as in Eq. 5. The anchor sits at frame A, so its causal window is frames $0, \dots, A$ ($A = 3$, four frames). The rollout then predicts frames $A+1, \dots, A+K$ ($K = 8$). Each target $\mathbf{p}_{A+j}$ is built from the centred window of half-width $h = 2$, so the last target reads frames up to $A + K + h$, and a clip has $T = A + K + h + 1 = 14$ encoded frames. SIGReg is applied at every frame where the centred momentum is defined,

$$\mathcal{T} = \{h, \dots, T-1-h\} = \{h, \dots, A+K\}, \qquad |\mathcal{T}| = 10,$$

which covers the anchor and all K targets, plus the frames $h, \dots, A-1$ inside the anchor's causal window. For the temporal encoders of window $W > 1$ these indices are in latent time. The encoder consumes $W-1$ lead-in pixel frames, so a clip holds $T + W - 1$ pixel frames. One way is to apply SIGReg over the joint embedding. Given each $\mathbf{q}$ and $\mathbf{p}$ of size d and a batch of B embeddings, $\mathbf{Z}_t$ stacks the concatenated phase-space embedding, step-wise over the full interior:

$$\mathbf{Z}_t = \left[(q_t^{(i)}, p_t^{(i)})\right]_{i=1}^{B} \in \mathbb{R}^{B \times 2d}, \qquad \mathcal{S} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathrm{SIGReg}(\mathbf{Z}_t).$$

Another way is to split the application. Same $\mathcal{T}$, but $\mathbf{q}$ and $\mathbf{p}$ are never concatenated and step-wise Gaussianity is tested on each block separately and summed:

$$\mathbf{Q}_t = \left[q_t^{(i)}\right]_{i=1}^{B}, \ \mathbf{P}_t = \left[p_t^{(i)}\right]_{i=1}^{B} \in \mathbb{R}^{B \times d}, \qquad \mathcal{S} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \Big[\mathrm{SIGReg}(\mathbf{Q}_t) + \mathrm{SIGReg}(\mathbf{P}_t)\Big].$$

We apply SIGReg across all the time steps in the batch because, as discussed in Section 2 in the main text and in Remark 4, the preservation of volume, which we obtain through the symplectic structure, is not redundant to what SIGReg enforces.

D Description of the experimental set up

D.1 Methodology

What is held fixed across the models. LeHami is implemented on top of the `stable-worldmodel` framework (Maes et al., 2026b) and its `stable-pretraining` training harness (Balestriero et al., 2025), the same stack LeWorldModel (Maes et al., 2026a) is built on. Encoder, projector, action encoder, optimiser and data pipeline are therefore shared with the baseline, and the only component that differs between the two models is the predictor and SG stencil that LeHami implements. Both models of every comparison see identical data, identical clip construction and an identical multi-step objective, the same number of epochs and the same optimizer and learning-rate schedule, so a difference in rollout behavior is not attributable to the training scheme.

The data split and checkpoint selection. We split the datasets into training, validation and test. Checkpoints are selected on the validation split. Both models use the same total latent dimension (64), split by the LeHami into $\mathbf{q}$ and $\mathbf{p}$ of half that width each (32) and used whole by the baseline, which has no momentum-like block. We assess performance as the predictor is unrolled $K = 8$ self-fed steps from an anchor. For LeHami we construct the anchor from the causal Savitzky–Golay stencil (Appendix C.4), so it sits at latent index $A = 3$, with full backpropagation through time, the momentum term in the loss (Eq. 5) carries $\lambda_p = 1.0$ and the distributional term $\lambda_S = 0.25$ is common among the models, and $\|\cdot\|_2^2$ denotes the per-dimension mean rather than the sum, which differs from the sum by a constant $1/d$ absorbed into those weights. LeWM's predictor is a causal, action-conditioned transformer over the three most recent latents while in LeHami it is the Hamiltonian flow that evolves one state at the time. Regarding the SIGReg regularizer, while it is applied on $\mathbf{z}$ for the baseline, in LeHami we tried out different places. If not said otherwise, SIGReg is applied on $\mathbf{q}$ only. Refer to the discussion on this for the actuated pendulum in Appendix E.3.

D.2 Implementation of LeHami

The Hamiltonian flow. It a separable Hamiltonian $H_\phi(\mathbf{q}, \mathbf{p}, \mathbf{u}) = K_{\phi_1}(\mathbf{p}) + V_{\phi_2}(\mathbf{q}, \mathbf{u})$ with two independent three-layer $\tanh$ MLPs of width 1024, integrated by one exactly symplectic kick–drift–kick leapfrog step per frame. For LeWM the projector output is a single $z_t \in \mathbb{R}^{64}$ and the predictor is the baseline's causal autoregressive transformer with a sliding context of three latents. For LeHami, the Hamiltonian autograd path is pinned to fp32.

The encoder. We implemented both a tiny ViT (Wu et al., 2022) and a temporal ConvNet (Tran et al., 2018). For this temporal encoder, when there is no time window (W=1) it means that the encoder takes one pixel frame $\mathbf{x}_t$ and produces its respective embedding $\mathbf{q}_t$. Then, with these embeddings $\{\mathbf{q}_{t-3}, \mathbf{q}_{t-2}, \mathbf{q}_{t-1}, \mathbf{q}_t$ corresponding to multiple time steps, the $\mathbf{p}_t$ is estimated. We also consider the temporal ConvNet with a temporal window in which pixel frames corresponding to multiple time steps are encoded to produce the embedding corresponding to a single time step. The general form of this architecture embedded into the LeHami framework is shown in Fig. 6. The encoder is a *(2+1)D* network, meaning the spatiotemporal convolution is factored into two spatial dimensions and one temporal one instead of a joint 3D kernel: a per-frame convolutional stem, one temporal mix, then the unchanged spatial trunk.

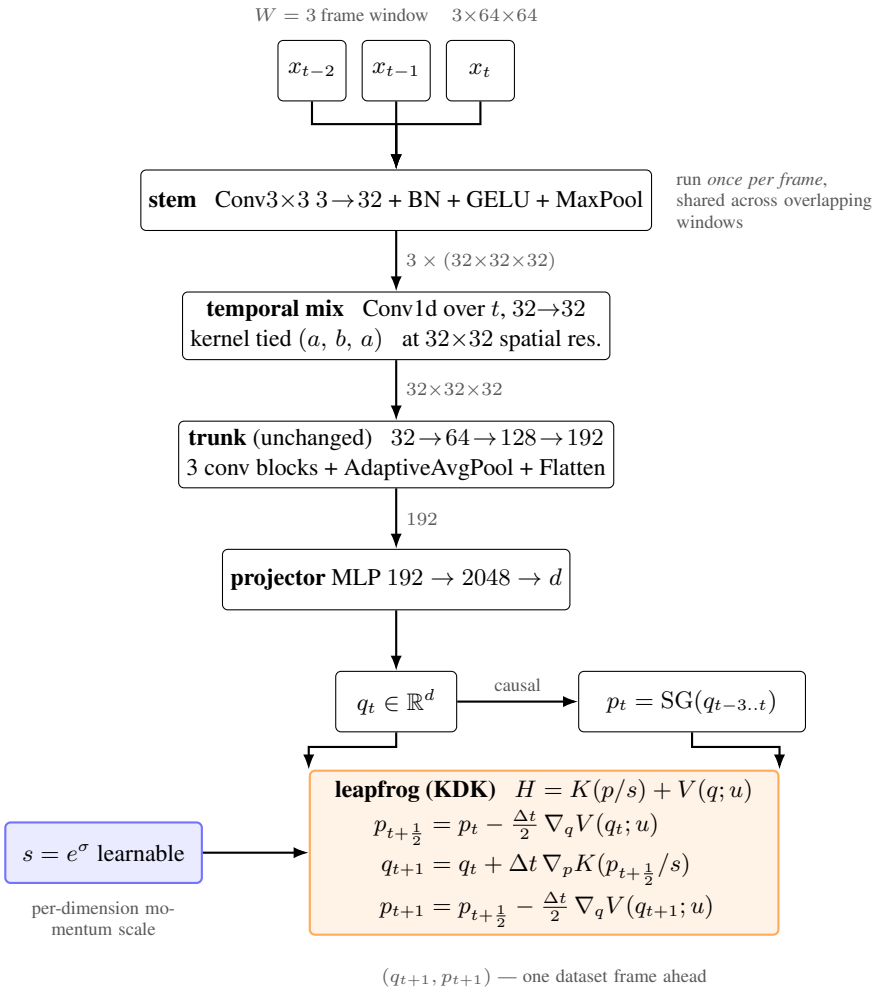

Figure 6: **The temporal ConvNet encoder inside the LeHami framework**, drawn at $W = 3$. A window of frames goes through a stem applied once per frame, then a single Conv1d over time with a symmetric tied kernel (a, b, a), then the baseline's spatial trunk and projector, unchanged, to give $\mathbf{q}_t$. The same encoder is used by the W=3 LeWM baselines; the main LeHami model uses W=1, i.e. no temporal mix. The rollout is initialised with the causal Savitzky–Golay momentum $\mathbf{p}_t$, computed from the current and three previous positions only, and advanced one frame at a time by the leapfrog step (orange) of $H_\phi = K_\phi(\mathbf{p}/s) + V_\phi(\mathbf{q}, \mathbf{u})$, where the learnable per-dimension scale s (blue) rescales the momentum and therefore appears only in the position update.

This encoder implementation allows to train across recording rates. The step size is an argument to the Hamiltonian flow rather than a property of its weights, so one learned H_ϕ can in principle be trained on episodes recorded at different rates. We describe the mechanism here for completeness, but no experiment in this paper uses it: every model is trained and evaluated at a single rate, and the

conservative dataset is restricted to its $\Delta t = 0.1\,\mathrm{s}$ slice for exactly this reason (Table 3). We leave the mixed-rate comparison to future work.

A dataset that mixes recording rates stores each episode's frame spacing in a `dt_frame` column; write Δt_{ep} for an episode's value and Δt_{ref} for a single rate fixed once at config time (`model.dt_ref`). The leapfrog step carries a Δt^2 factor, so a $V_\phi(\mathbf{q})$ fitted at one rate is wrong by $(\Delta t/\Delta t')^2$ at another: training directly on mixed-rate episodes would hand K_ϕ, V_ϕ a target that shifts with which episode a clip happens to come from. Dividing by Δt_{ref} removes this factor, so one learned H_ϕ stays valid across every rate. We therefore integrate not in Δt_{ep} but in the dimensionless ratio $h = \Delta t_{\text{ep}} \,/\, \Delta t_{\text{ref}}$

Within an episode there is a single frame spacing and advancing one frame is one step. As LeHami parametrizes the time step inside the Hamiltonian flow, allowing to input, in principle, any arbitrary time step, episodes recorded at different rates could be trained on together. To do so, a recording rate is set as the reference, Δt_{ref}, and let the integrator step be that episode's spacing measured in units of it. An episode recorded at the reference rate is integrated with $h = 1$, one recorded at twice the spacing with $h = 2$, and so on. The learned H_ϕ is then the same object for every episode, expressed in reference time, and all the rate information the model needs travels with the clip as a single scalar. This is available to a Hamiltonian predictor precisely because its step size is an argument; an unconstrained predictor has no slot to put h in, so the asymmetry is structural. We do not report a mixed-rate head-to-head, because we did not run one. Note that the momentum must ride the same clock as the position update, so it is likewise divided by h: $p_t = \mathrm{SG}(q_{t-2..t+2})/h$, so that p and the h-scaled position update share one clock.

The SG stencil to compute the momentum-like embedding. The momentum-like embedding is never learned. It is always read off the encoder trajectory by one of two fixed Savitzky–Golay stencils, least-squares quadratic fits differentiated at t (Appendix C.4):

- the *causal* stencil, over $\mathbf{q}_{t-3}, \ldots, \mathbf{q}_t$, gives the momentum that *initialises* a rollout (the anchor);
- the *centred* stencil, over $\mathbf{q}_{t-2}, \ldots, \mathbf{q}_{t+2}$, gives the momentum *targets* of the loss (Eq. 5), the $\mathbf{p}$ seen by SIGReg, and the reference $\mathbf{p}$ in evaluation.

The anchor needs the causal stencil because the frames after t are the ones the model is about to predict, and reading them would leak the targets. The anchor therefore sits at latent index $A = 3$, the first index with a full causal window.

There is no train–test shift in the momentum estimate. The state the flow is fed is built by the same causal stencil everywhere: in training, in the rollout evaluation, and in planning, both at the start state and at the goal (Appendix D.4.2). The centred stencil is used only to build targets, never an input. After the anchor, $\mathbf{p}$ is a state variable of the flow, and no stencil is applied to predicted latents.

The two stencils estimate the same derivative to the same order but differ in time parity. The centred weights are antisymmetric in the offset, so reversing the frame order sends $\mathbf{p}_t$ to $-\mathbf{p}_t$ exactly. The causal stencil has no such symmetry: under reversal it becomes a forward stencil rather than minus itself. The reversal diagnostic starts from an anchor whose $\mathbf{p}$ comes from the causal stencil, which is one reason it is reported as approximate rather than exact, even for a model whose K_ϕ were perfectly even.

D.3 Datasets and their architectures

D.3.1 Conservative pendulum.

The dynamics of a conservative pendulum of length L and angle θ is described by

$$\ddot{\theta} = -\frac{g}{L}\sin\theta, \tag{13}$$

with $g = 9.81\,\mathrm{m\,s^{-2}}$.

Our environment integrates this differential equation with a fourth-order Runge–Kutta scheme. We consider three different rod lengths: $L \in \{0.7, 1.0, 1.4\}\,\mathrm{m}$ rendered to scale, as shown in Fig.7, 300

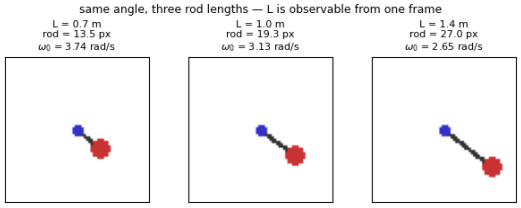

Figure 7: Frames from the three rod lengths of ($L \in \{0.7, 1.0, 1.4\}$ m), rendered to scale at 64×64. The rod length is therefore readable from a single frame, which is what makes it a positive control for the probes rather than a prediction target.

episodes of 80 frames with a time step of $\Delta t = 0.1$ s which are split into $210/45/45$ train/val/test at the episode level. Initial conditions are drawn independently per episode: $\theta_0 \sim \mathcal{U}(-\pi, \pi)$ and the initial velocity is drawn in dimensionless units so that the phase-space coverage is matched across rod lengths such that $\dot{\theta}_0 = \tilde{\omega}\sqrt{g/L}$ with $\tilde{\omega} \sim \mathcal{U}(-a, a)$ and $a = 2\,\mathrm{rad\,s^{-1}}/\sqrt{g/L_0} \approx 0.639$ at the reference length $L_0 = 1$ m, so that the bound reads $|\dot{\theta}_0| \leq 2\,\mathrm{rad\,s^{-1}}$ at L_0 and scales as $\sqrt{g/L}$ across rods.

The range $|\dot{\theta}_0| \leq 2\,\mathrm{rad\,s^{-1}}$ is well below the separatrix velocity $\dot{\theta}_{\mathrm{sep}} = 2\sqrt{g/L} \approx 6.26\,\mathrm{rad\,s^{-1}}$, so trajectories starting near the bottom ($\theta_0 \approx 0$) always oscillate. However, because θ_0 is drawn from the full $[-\pi, \pi]$, trajectories starting near the unstable equilibrium ($\theta_0 \approx \pm\pi$) carry large potential energy and rotate with any nonzero $\dot{\theta}_0$. The dataset therefore spans both the oscillating and rotating regimes.

Each episode corresponds to a given rod's length. The natural frequency of the pendulum is $\omega_0 = \sqrt{g/L}$. Thus, it spans a $1.41\times$ range and a single learned V_ϕ must cover all three, which makes the rod length a system-identification target as well as a control.

D.3.2 Action as a force: actuated pendulum.

Here we consider an actuated pendulum whose rod's length is L and mass m, whose dynamics are as the conservative pendulum in Eq. 13 but with an additional torque term, giving

$$\ddot{\theta} = -\frac{g}{L}\sin\theta + \frac{u}{mL^2}, \tag{14}$$

where u is the applied torque.

Episodes are 64×64 RGB, the manifest splits $630/135/135$ episodes at the episode level, and every rollout reported here is driven by the recorded torque. Energy is not conserved and the potential energy $V_{\phi_2}(q, u)$ carries a time-varying input, so H_ϕ is explicitly time-dependent and drift along the rollout measures how much the torque moves the learned energy. Rod length again varies across episodes ($0.7, 1.0, 1.4$ m), as well as the rod mass ($0.5, 1.0, 2.0$ kg). In this dynamics, in contrast to the conservative pendulum, the mass impacts the torque term. Thus, a model whose latent does not carry signal of the mass, cannot know what a given torque will do.

The torque. The torque is held constant over $h = 3$ frames (0.3 s), a zero-order hold at 3.3 Hz: this is 13–18 % of the small-oscillation period $T = 2\pi\sqrt{L/g}$ (1.68–2.37 s over our lengths), so the drive is band-limited to roughly $3\times$ the natural frequency and injects or removes energy coherently over a fraction of a swing. A per-frame torque would instead be white noise on the dynamical timescale as averaging out over a period would leave trajectories close to the unforced pendulum. This choice also makes the torque observable from pixels. The physical time step is dt $= 0.1s$, so the torque is held constant through a block of 0.3 s against periods of 1.68 s (L=0.7), 2.01 s (L=1.0), 2.37 s (L=1.4), which correspond to 18%, 15%, 13% of a period, with 3.4-4.8 swings per episode, which last 8s.

Table 3: The two environments and their episode-level splits. Both use the same protocol, described in Appendix D.3.

	conservative pendulum	actuated pendulum
episodes used	300	900
frames per episode	80	80
Δt	0.10 s	0.10 s
split, in episodes (train / val / test):		
episodes	$\mathbf{210/45/45}$	$\mathbf{630/135/135}$
design cells	3 rod lengths	9 ($3L \times 3m$)
episodes per cell	100	100
train episodes per cell	70	70

D.3.3 THE DATA SPLIT PROTOCOL

Both environments are split into training, validation and test with the same protocol, summarised in Table 3. The split is $70/15/15$ and is made *at the episode level*: a whole episode belongs to exactly one split, so no frame is shared between splits, and we verified that the episode indices of the three splits do not overlap in either dataset. The split is also stratified by design cell (rod length for the conservative pendulum; rod length and mass for the actuated one), so every cell contributes the same number of episodes to each split. Training clips are cut with stride 1 from the first 80 frames of each training episode, and the split is drawn with seed 42.

For the conservative pendulum we use 300 of the 1200 generated episodes. The generator sweeps four frame spacings, 300 episodes each, and we keep only the $\Delta t = 0.1\,\mathrm{s}$ slice, because every model in this paper is trained and evaluated at a single step size (see the paragraph on training across recording rates in Appendix D.2).

D.4 EVALUATION

In this subsection we describe the evaluation we perform to benchmark the models.

D.4.1 THE PROBES

Targets are θ and ω. For the first one, we probe $(\sin\theta, \cos\theta)$, which are recombined by `atan2` so the reported angular error is wrapped. As described in main text, we study the performance across multiple probes and we lead with the KNN probe trained at the first predictor's output ($K = 1$). This corresponds to equal-footing as both LeWM and LeHami capture relative differences among pixel frames. While LeHami does this structurally with the embedding $\mathbf{p}$, which is the approximated derivative of $\mathbf{q}$ through the SG stencil, LeWM processes the history (three time steps) through the predictor.

D.4.2 PLANNING

Planning is evaluated on the actuated pendulum only, since the conservative pendulum has no action to plan. Everything in this subsection therefore refers to the actuated pendulum, and it completes the description in Section 3.4. Goals are read off held-out episodes, each model is scored in its own latent coordinates, and the action sequence is optimised by CEM under MPC. Nothing in the loop observes the future.

Goal construction. Goals are drawn from test episodes, never from training data. For each of the episodes we sample a goal frame g and set the start frame to $s = g - K$, so the target is the state the trajectory actually occupied K steps after the start. For the actuated pendulum this is a recorded (θ_g, ω_g) that is in general neither upright nor at rest. Building the goal this way guarantees reachability, which the model-free pre-flight below confirms, so a failure is attributable to the world model rather than to an impossible target.

Latent states, and the frames they are read from. Each model is given the state its own architecture defines, encoded from real frames at two instants only. At the start, the 5 recorded frames ending at s are encoded; LeHami forms $\mathbf{q}$ from the W frames and $\mathbf{p} = \mathrm{SG}(\mathbf{q})$ with the same Savitzky–Golay stencil used during training, while the baseline encodes only $\mathbf{z}$ in the same way as $\mathbf{q}$ (with W frames) and it forms its context of $h = 3$ latents and pass through the (unstructured) predictor. At the goal, LeHami encodes the $(\mathbf{q}_g, \mathbf{p}_g)$ in the same way using the causal stencil with embeddings corresponding to four time steps while LeWM $\mathbf{z}$ is coming from W frames to produce the goal state.

Optimisation. The cost is evaluated in each model's own coordinates, so no ground-truth state enters the objective. A state-space cost would measure the probe rather than the world model. Actions are held over blocks of three frames, matching the hold of the dataset, and are constrained to to $\|\mathbf{a}\| \leq u_{\max}$. CEM uses 300 samples, 30 iterations and 30 elites. Under MPC the first R actions of $\mathbf{a}^*$ context is re-encoded from the frames the controller has by then observed, and the problem is re-solved. The horizon shrinks to the frames remaining before the goal instant, so the target does not recede as the episode advances; a fixed horizon would leave the goal perpetually H steps away and success rate meaningless.

No leakage. The rollout $\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_H$ is produced by self-feeding the model: one predictor call per step, with its output fed back as the next input. After the anchor, $\mathbf{p}$ evolves under the learned flow as a state variable of the predictor; the Savitzky-Golay stencil is applied only to encoded real frames, at the start context and at the goal, never to predicted latents. Pixels therefore enter at two disjoint instants, the start context and the goal window, and no frame between them is ever encoded. No ground-truth state enters the rollout, the cost or the ranking of candidates. Under replanning, every re-encoded context is built from frames the controller has actually observed, all at or before the current step. The recorded actions of the episode are used only to fill the torques already applied at the start context, never as a candidate or as an initialisation of the search. Success is evaluated by the simulator on the executed trajectory, outside the planner.

Scoring success. Success is scored at the goal's own timestep $s + K$, on the angle and the velocity with separate tolerances, $|\mathrm{wrap}(\theta - \theta_g)| < \tau_\theta$ and $|\omega - \omega_g| < \tau_\omega$. Raw errors are written for every episode, so success as a function of tolerance is recoverable without re-running. Scoring at the goal's timestamp rather than at any step within the budget is necessary rather than cosmetic: under an "ever passes near the goal" criterion, the best of 300 random action sequences under exact dynamics reaches every goal, and the measurement saturates for every model at every tolerance.

Both models use the same temporal ConvNet encoder, with W=1 for LeHami and $W \in \{1, 3\}$ for LeWM, and the same latent dimension d=64, with LeHami using q and p, each of dimension 32, and LeWM using only z of dimension 64. SIGReg in LeHami is applied to q only, while in LeWM it is applied to z only. The weight of the SIGReg loss is 0.25 in both cases, and the same number of training epochs (35) was used for both models. Both models were trained with the same optimizer and learning rate schedule. For planning, both utilized the same CEM planner. When training, the temporal ConvNet encoder was centered in the window, to guarantee the embeddings of LeHami are canonical. For planning, in which we deploy online, we have to adapt. At wall-clock frame t, having observed frames up to t, the encoder can produce latents for instants up to $t -$ lag so at planning time the context and goal latents are shifted by the encoder lag, and the rollout runs that many extra steps to reach the goal instant.

We integrate θ without wrapping, so a policy that completes n revolutions ends near $\theta_0 + 2\pi n$. Because the goal specifies a pose, and poses a full turn apart are indistinguishable, we score the angle on the circle:

$$\Delta\theta = \mathrm{wrap}(\theta_T - \theta_g), \qquad \mathrm{wrap}(a) = \big((a + \pi) \bmod 2\pi\big) - \pi, \tag{15}$$

evaluated at the goal's own timestamp T. Wrapping the *difference* is essential: wrapping each angle first and subtracting introduces a spurious 2π jump at the branch cut, reporting a near-perfect match as a maximal error. An episode succeeds when $|\Delta\theta| < 0.1\,\mathrm{rad}$ and $|\omega_T - \omega_g| < 0.5\,\mathrm{rad\,s^{-1}}$. As $\Delta\theta$ is circular, we summarise it by the mean resultant length $\bar{R} = |N^{-1} \sum_i e^{\mathrm{i}\Delta\theta_i}|$ (0 = uniform, 1 = concentrated) rather than by a linear mean and standard deviation.

Settings. Goals are placed $K = 30$ frames after the start, and the first solve plans over the full horizon $H = 30$. The replanning interval is swept over $R \in \{30, 15, 9, 6, 3\}$: $R = H$ is a single open-loop solve, and the smaller values are MPC at increasing feedback rates from the same first-solve horizon. Actions are bounded by $u_{\max} = 4.0$ and held for blocks of 3 frames, as in the dataset. CEM uses 300 samples, 30 iterations and 30 elites. Each configuration is run on $N = 50$ test episodes, for 3 model seeds $\times$ 3 planner seeds, with tolerances $(\tau_\theta, \tau_\omega) = (0.1\,\mathrm{rad},\ 0.5\,\mathrm{rad\,s^{-1}})$.

E SUPPLEMENTARY EXPERIMENTAL RESULTS

In this section we showcase supplementary experimental results. First, we comment on some general properties of the proposed Hamiltonian JEPA. To exemplify we use results corresponding to one of the two environments but the claim is in general. Then, in the following subsections we share results for each of the two environments studied.

E.1 LEHAMI REPLACES MANUAL DESIGN CHOICES WITH STRUCTURAL PRIORS AND LOWER COMPUTE

Given a sequence of pixel frames, our goal is to train a world model that captures the system's physics, specifically its higher-order dynamic states from position to its time derivatives, the velocity and the momentum it determines. In JEPA architectures, capturing these spatiotemporal differences efficiently is a central design consideration. The default implementation of LeWM (Maes et al., 2026a) employs a single-frame encoder (W=1) and delegates the task of learning representations for higher-order moments entirely to a predictor that receives a three-step frame history. A direct limitation of encoding frames individually is the inability to represent goals containing velocity clauses at the single-latent level.

We assess these different design choices empirically. Fig. 8 shows the R^2 of the angle θ and angular velocity ω decoded from anchor latents ($K = 0$) using a k-NN probe. As expected, LeWM with a single-frame encoder (W=1) fails to capture velocity, yielding an R^2 below zero. However, simply expanding the encoder to operate over a time window (W=3) does not inherently improve the representation of velocity and can actually deteriorate it if executed without architectural care.

Specifically, a temporal ConvNet with $W{=}3$ employs a symmetric kernel by construction when weights are tied (e.g., $[a, b, a]$ where outer taps share parameters). Consequently, the tied encoder is invariant to reversing the temporal order of the three frames in its window. Reversing time leaves the central angle unchanged while flipping $\omega \to -\omega$; the states $(\theta, +\omega)$ and $(\theta, -\omega)$ are therefore mapped to the exact same latent representation, rendering the sign of the velocity unrecoverable by any probe (see Appendix E.3 for a detailed breakdown). Only even functions of velocity survive: absolute velocity $|\omega|$ remains highly decodable ($R^2 = 0.92$), while signed velocity ω sits at the predict-the-mean floor.

While the parity of the kernel can be manually broken—either by untying parameters or feeding explicit frame increments $(x_t, x_t - x_{t-1})$—nothing in the standard JEPA objective forces the model to retain directional information downstream. The encoder is trained merely to make future embeddings predictable, and a representation blind to the direction of motion predicts its own sign-blind future embeddings perfectly well. The architectural choice is therefore made implicitly, with its failure modes appearing only when evaluating single-latent state readouts.

Our proposed Hamiltonian JEPA provides a structural solution rather than an empirical workaround. Instead of requiring the temporal encoder to learn velocity implicitly, LeHami partitions the representation into canonical pairs $(\mathbf{q}, \mathbf{p})$ and extracts momentum via $\mathbf{p} = \mathrm{SG}(\mathbf{q})$, an explicitly odd stencil across latents whose parity is fixed by construction. There is no kernel to tune; the Hamiltonian latent structure dictates where ω resides, and encoder parity follows naturally.

As shown in Fig. 8, the signal for predicting angular velocity ω originates almost entirely from the momentum block $\mathbf{p}$. Conversely, the angle θ is predominantly captured by the position block $\mathbf{q}$, with minor signal leakage into $\mathbf{p}$ arising because $\mathbf{p}$ is constructed as a temporal finite difference of $\mathbf{q}$.

Furthermore, LeHami is cheaper to train. At every matched training budget on the conservative pendulum (Table 4), LeHami's error is 1.45–$4.2\times$ lower on θ and 1.4–$2.3\times$ lower on ω. Its 0.75 PFLOP

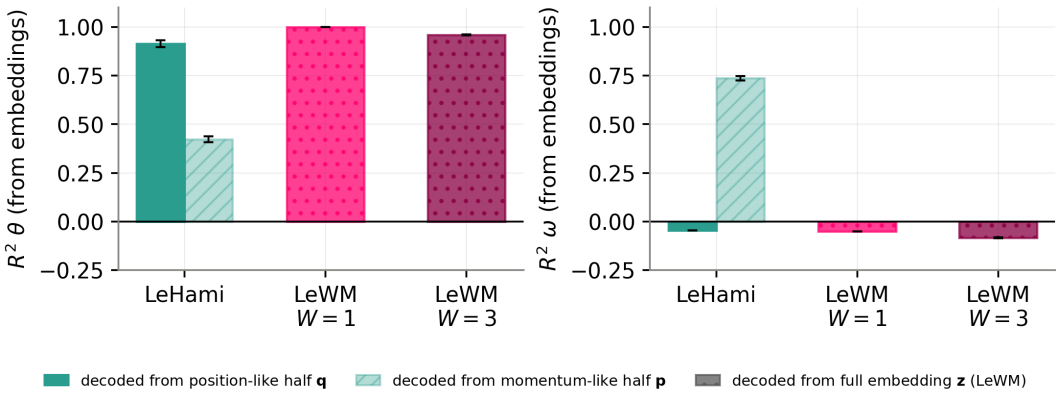

Figure 8: **The LeHami embeddings on the actuated pendulum are canonical.** R^2 of the pendulum angle θ **(left)** and angular velocity ω **(right)** decoded from the position half q (solid) and momentum half p (hatched) of the frozen encoder output. Probes are 25-NN under five episode-grouped folds fitted on embedded anchor frames ($K = 0$) pooled across test episodes. R^2 of θ is evaluated over $(\sin\theta, \cos\theta)$. Bars indicate the mean over three training seeds; error bars denote standard deviation. LeWM has no partitioned momentum block and is reported as a single joint latent z.

Table 4: **MAE at a fixed training budget for the conservative pendulum.** Test error at the checkpoint with the lowest validation prediction loss within each compute budget (epoch in parentheses), decoded through a frozen k-NN probe at $k = 20$ to ensure a shared scale. Checkpoints were saved every 10 epochs; evaluated on one seed per model.

Metric	Model	0.75 PFLOP	1.5 PFLOP	3 PFLOP	6 PFLOP
MAE θ [rad]	LeHami W=1	**0.283** (ep 9)	**0.207** (ep 19)	**0.252** (ep 39)	**0.222** (ep 79)
	LeWM W=1	0.590 (ep 9)	0.871 (ep 19)	0.487 (ep 39)	0.321 (ep 79)
MAE ω [rad/s]	LeHami W=1	**1.002** (ep 9)	**0.929** (ep 19)	**0.934** (ep 39)	**0.902** (ep 79)
	LeWM W=1	1.625 (ep 9)	2.146 (ep 19)	1.620 (ep 39)	1.270 (ep 79)

checkpoint already has lower error on both quantities than LeWM's best within 6 PFLOP, so LeHami matches the baseline with at most one-eighth of the training compute. Both models cost about the same per 100-epoch run (6.0–6.5 PFLOP, 0.7–1.1 GPU-h), so the gap reflects how efficiently each converts compute into prediction accuracy, not a difference in budget. LeHami also has fewer parameters (Table 5), so the advantage is not explained by capacity. These results are from one seed per model.

Model parameters. The compute advantage above is not bought with capacity. Table 5 counts each model's parameters from its checkpoint. The encoder is identical across the two families, and the latent projector differs only in its output width (32 for $\mathbf{q}$ against 64 for $\mathbf{z}$), and the SG stencil is not trainable. Thus, the difference sits in the predictor. LeHami's Hamiltonian flow, the two MLPs K_ϕ and V_ϕ, has 2.23M parameters, against 3.31M for LeWM's autoregressive transformer, which also needs a 0.27M predictor projection head. In total LeHami has about 32% fewer parameters (3.03M against 4.44M), and widening LeWM's encoder to W=3 adds only ∼3k. The same predictor gap drives planning cost, since over 99% of a planning episode's arithmetic is spent rolling the predictor out (Appendix E.3, *Planning compute*).

E.2 Conservative pendulum

Visualization of the latent representation space LeHami constructs. We take the encoded ground-truth embeddings across the frames in the episodes and across episodes and perform PCA, shown in Fig. 9. It can be seen that the trajectory is constrained to the learned H_ϕ. This is possible because this Hamiltonian describes a conservative system, without any action. This Fig. shows

Table 5: **Parameter counts of the models trained on the conservative pendulum environment.** The values are shown in millions, measured from the checkpoints. The predictor is where the two families differ: LeHami's separable Hamiltonian flow ($K(p)+V(q)$ MLPs, `hidden_dim` $= 1024$) against LeWM's 6-layer autoregressive transformer. *pred. proj.* is LeWM's predictor projection head, which LeHami has no counterpart for; the encoder and the latent projector are matched across families, and the $W{=}3$ encoder adds only $\sim$3k parameters over $W{=}1$.

Model	Total	Encoder	Predictor	Projector	Pred. proj.
LeHami $W{=}1$	**3.03**	0.315	**2.234**	0.465	—
LeWM $W{=}1$	**4.44**	0.315	**3.310**	0.530	0.268
LeWM $W{=}3$	4.45	0.318	3.310	0.530	0.268

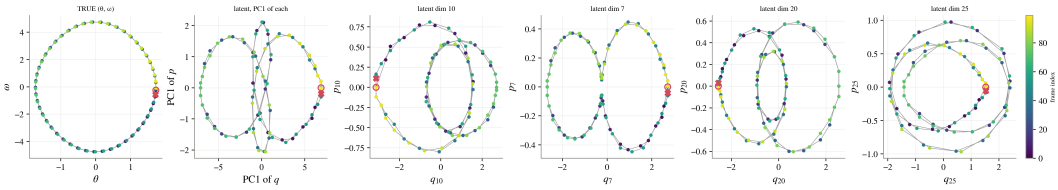

Figure 9: **Conservative pendulum: rollout latents projected onto pairs of principal components.** Each panel is one pair of the leading PCs of the encoder's $(\mathbf{q}, \mathbf{p})$ over the test episodes; colour is the episode. The learned H_ϕ is conserved along each rollout, so an episode traces a closed curve in every projection rather than drifting across them.

empirically how the proposed LeHami imposes structure in the latent representation space that sets clear trajectories for the evolution.

Similarly, Fig. 10 illustrates how the LeHami's rollout stays on a single low-dimensional closed set per episode, stratified by its learned H_ϕ. The projection is obtained by mean-centring the latents per dimension and taking the SVD; the leading right-singular vectors are the components, with no whitening and no rescaling. Centring matters here because otherwise the first component points at where the cloud sits rather than at how it spreads. Otherwise PC1 just points at where the data is, not at how it spreads. Refer to Fig.14 to see how the manifold in the latent representation space is modified by the action while still a surface of conserved energy minus work determines the trajectories. Each of the 60 test trajectories have been rolled out using the LeHami model, starting the anchor. In Fig. 11 shows that the learned energy is conserved and that determines a horizontal plane in which the trajectory takes place without crossing different places. We compare with the ground-truth physical system on the left in which each energy level determines a clear shape for different values of energy, restricting the trajectories to those.

Rollout prediction across probes. Linear tier underperforms non-linear: ridge recovers ω from the Hamiltonian phase space at only $R^2 = 0.45$ against 25-NN's 0.975 (Table 7), so velocity is encoded non-linearly in $(\mathbf{q}, \mathbf{p})$ and every linear number here is a lower bound. Moreover, the LeHami's own momentum block degrades faster than its position block: R^2_p against the reference momentum falls to 0.613 by $K = 10$ and below zero by $K = 25$, while R^2_q is still 0.950 at $K = 10$ (Table 6). Both R^2_q and R^2_p are computed in the latent space against the encoder's own reference embeddings, so no angle is wrapped and the caveat that applies to $R^2(\theta)$ does not apply here. The velocity advantage is therefore inherited from the representation.

Encoder independence. The encoder in the head-to-head above is a ConvNet. To check that the effect is a property of the predictor rather than of that choice, we compare the tiny ViT (Wu et al., 2022) and ConvNet encoder pairs on a single-rod dataset ($L = 1.0$ m, $\Delta t = 0.05$ s, 500 episodes of 100 steps), holding everything else fixed. At the anchor, the LeHami decodes ω at $R^2 = 0.929$ with a ViT encoder and 0.939 with a ConvNet; the unconstrained baseline scores -0.031 (ViT) and -0.018 (ConvNet). The ordering, its sign and its magnitude are unchanged by the encoder family, so the velocity content is attributable to the phase-space split and the flow, not to the vision backbone. Note that this check uses the single-rod dataset, for this reason its anchor values differ from those in Table 7.

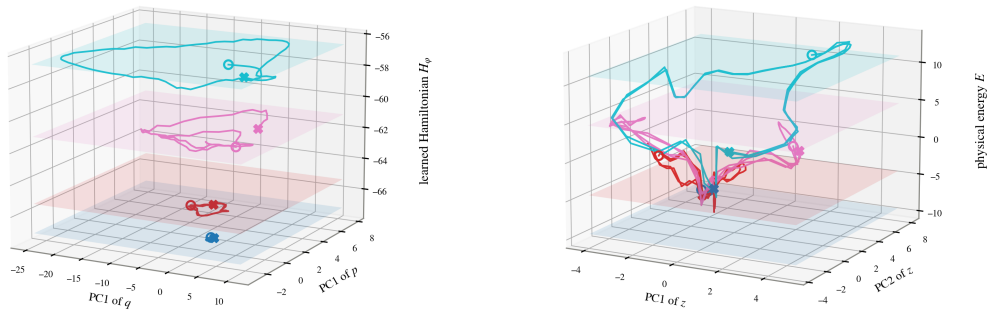

Figure 10: **Latent representation space projected into the first two principal components (PCs).** The embeddings correspond to LeHami **(left)** and unstuctured LeWM **(right)**, shown as a function (z-axis) of the Hamiltonian learned (left) and the physical energy corresponding to the episodes. The embeddings of four episodes are shown in colors. The planes with the same color correspond to the learned Hamiltonian and energy, respectively. While the rollout episode's embeddings in LeHami are restrictive in each of the planes corresponding to the (constant) respective learned Hamiltonian, in LeWM, the rollout episode's representations move across planes of energy, mixing up.

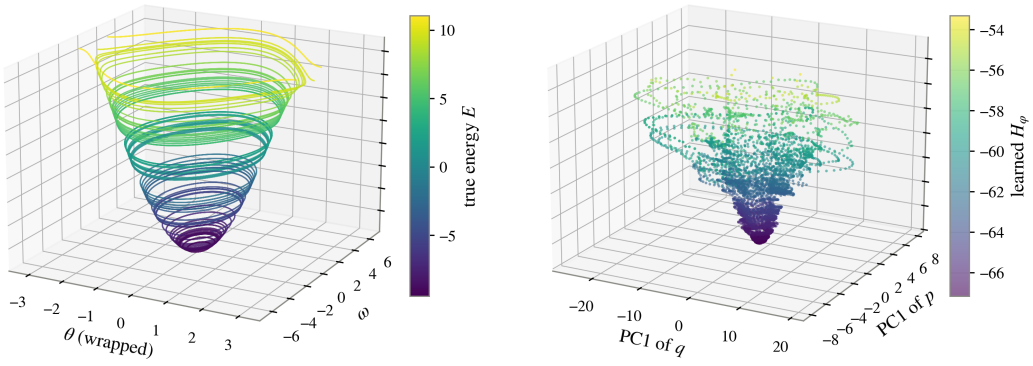

Figure 11: **The learned energy landscape against the physical one. (left)** level sets of the true energy of the conservative pendulum; **(right)** level sets of the learned H_ϕ over the same 60 test rollouts, each started from its own anchor. Distinct energies give disjoint closed curves in both, so the learned generator stratifies the latent space the way the physical energy stratifies phase space.

Table 6: **Where the LeHami's two blocks lose the reference, conservative pendulum.** R^2 of the predicted $\mathbf{q}_k$ and $\mathbf{p}_k$ against the encoded ground-truth $\mathbf{q}$ and $\mathbf{p}$ at the same frame, test split. These are latent-space quantities in the model's own coordinates, not physical ones: they measure how far the flow drifts from the encoder trajectory it was trained to reproduce, and being self-referential they cannot by themselves establish that a physical quantity survives. What survives decoding is in Table 7..

rollout step k	0	1	5	10	20	25	50
R^2_q (position block)	1.000	0.997	0.985	0.950	0.929	0.646	0.373
R^2_p (momentum block)	1.000	0.935	0.867	0.613	0.586	-0.324	-1.165

Table 7: **Performance in physics prediction of the conservative pendulum.** Wrapped MAE of angle θ and ω predicted by the k-NN probe (with $k = 25$) fitted on each model's own $K = 1$ output and applied frozen along the rollout, averaged over the rollout band. Ratio is LeWM $\div$ LeHami, so > 1 favours the LeHami.

	MAE θ [rad, wrapped]			MAE ω [rad/s]		
band	LeHami	LeWM	ratio	LeHami	LeWM	ratio
$k = 1$–10	0.107	0.100	0.93	**0.364**	1.020	**2.80**
$k = 11$–20	**0.225**	0.287	**1.27**	**0.669**	1.169	**1.75**
$k = 21$–70	**0.596**	0.981	**1.65**	**1.534**	2.331	**1.52**
anchor ($K = 0$), $R^2(\omega)$ from the frozen encoder features:				LeHami $+0.975$	LeWM -0.667	
same anchor, LeHami read by the linear tier:				ridge $+0.448$, against the 25-NN's $+0.975$ above		

We showcase that the proposed LeHami can leverage a pre-trained encoder to be a world model on pre-trained visual features as DINO-WM (Zhou et al., 2024) does. For this, the pre-trained encoder, DINO-v2 (Oquab et al., 2023), is followed by a projector to output $\mathbf{q} \in \mathbb{R}^32$, and all the other components of LeHami remain the same. Similarly, DINO-WM has a projector to project to 64-dimension space. This is DINO-LeHami. The loss for this model, as well as DINO-WM, is the same as the one used for both LeHami and LeWM for the training of the models we discussed their performance in Fig. 4. The evaluation is also exactly the same. Fig. 12 shows that MAE in both θ and ω is reduced in comparison to their respective implementations without the pre-trained encoder. The advantage of LeHami in the angle MAE to LeWM seems to be reduced while for the velocity ω it continues to outperform, while for K larger than 60 both models start to deviate significantly.

Phase-space volume of the one-step map. For a one-step predictor F, $\mathbf{z}_{i+1} = F(\mathbf{z}_i, \mathbf{u}_i)$, the Jacobian $J(\mathbf{z}_i) = \partial F/\partial \mathbf{z}|_{\mathbf{z}_i}$ linearises F around $\mathbf{z}_i$, and $|\det J(\mathbf{z}_i)|$ is the factor by which an infinitesimal volume of states around $\mathbf{z}_i$ is scaled. By multiplicativity of the determinant, $\sum_{i<K} \log|\det J(\mathbf{z}_i)|$ is the log volume factor of the K-step map. We divide by the state dimension n so that maps of different sizes are comparable. A value of 0 means volume is preserved and a negative value means contraction. The true pendulum flow is symplectic, so 0 is also the ground truth. For LeHami, $\mathbf{z}_i = (\mathbf{q}_i, \mathbf{p}_i)$ and F is one leapfrog step, so $\det J = 1$ exactly.

LeWM's predictor takes three context embeddings and returns one, so it has no square map from a state to the next state. We therefore report two quantities. The full map acts on the context, $(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t) \mapsto (\mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{z}_{t+1})$ so its Jacobian is block-companion, and its determinant equals $\pm\det(\partial \mathbf{z}_{t+1}/\partial \mathbf{z}_{t-2})$. The newest block, $\partial \mathbf{z}_{t+1}/\partial \mathbf{z}_t$, has the dimension of a single embedding and measures how the most recent state is rescaled and it is not a volume. Fig. 13 shows both measures at the anchor of every test episode: LeWM contracts under the full map and under the newest block alike, with no overlap with LeHami, which stays at 0 for every state. The contraction is therefore not an artefact of which Jacobian is chosen for LeWM's three-embedding context.

E.3 Actuated pendulum

Visualization of the latent representation space LeHami constructs. In this dataset, work is performed by the action. While in the conservative pendulum the energy determines the orbit as we

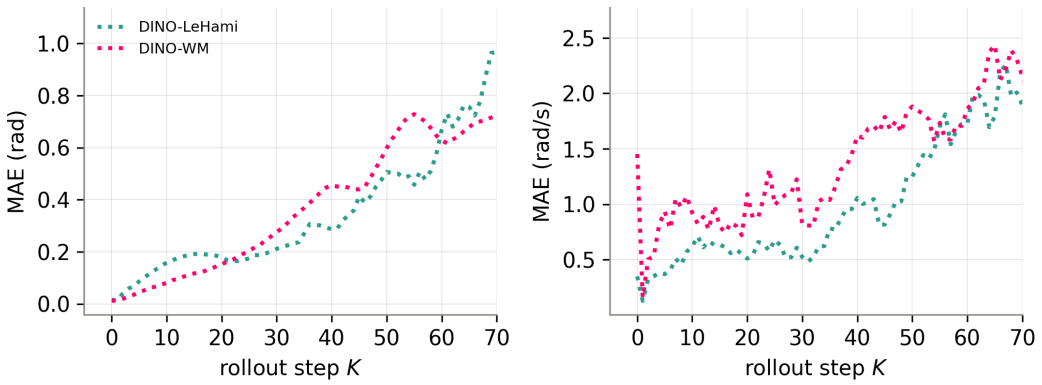

Figure 12: **DINO-LeHami vs. DINO-WM on the conservative pendulum.** Both models use a frozen DINOv2 encoder followed by a learned projector; Wrapped MAE of θ **(left)** and ω **(right)** along the test rollout (K steps, $\Delta t = 0.05$ s), read by a k-NN probe ($k = 25$) fitted at $K = 1$ and kept frozen. One training seed per model, both at epoch 60.

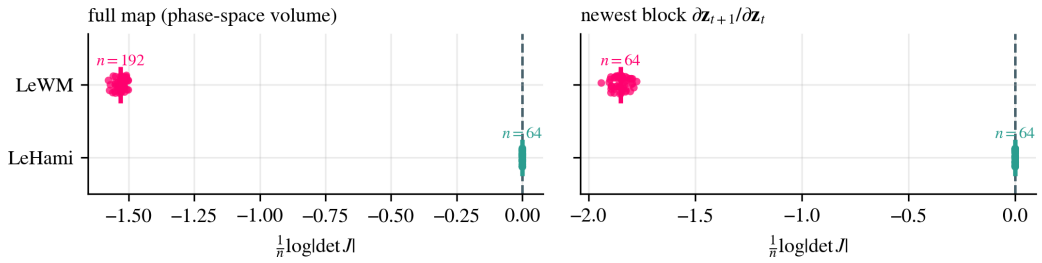

Figure 13: **Log-volume factor of the one-step latent map on the conservative pendulum.** One Jacobian per test episode at the anchor (45 states; bars mark the mean). **Left:** the full map, $n = 64$ for LeHami on $(\mathbf{q}, \mathbf{p})$ and $n = 192$ for LeWM on its three-embedding context. **Right:** the newest block $\partial \mathbf{z}_{t+1}/\partial \mathbf{z}_t$, $n = 64$ for both. The dashed line is the true flow. One training seed per model.

observe a closed rings in Fig. 11, in the actuated case, the trajectory is free to wander over a large region in the phase-space, so the surface of $E - W$ is a horizontal plane spanning the whole space, not a ring bounding anything. Different episode's planes overlap in the phase-space because they transverse similar regions, making the latent representation space to look as in Fig. 14, projected over the main PC of $\mathbf{q}$ and $\mathbf{p}$, respectively. The planes are now curves, and each one correspond to a single value of torque (action), and the trajectories can move between these planes. However, given a torque, position in the space is determined.

Where to apply regularizer in LeHami. Given the ground-truth embeddings (produced by the encoder) $(\mathbf{q}, \mathbf{p})$, with $\mathbf{p}$ the Savitzky–Golay derivative of $\mathbf{q}$ on the centred-valid window, the three placements to apply SIGReg are: $\lambda_S \,\mathrm{SIGReg}(\mathbf{q}\|\mathbf{p}])$ (`joint`), $\lambda_S \,\mathrm{SIGReg}(\mathbf{q})$ (`q`), and $\lambda_S \,\mathrm{SIGReg}(\mathbf{q}) + \lambda_S \,\mathrm{SIGReg}(\mathbf{p})$ (`split`). We study the variance of the embeddings within-episode and between-episode in Table 8. A random unit direction in the 64-wide concatenation puts on average half its squared length in the q block, so with $p \approx 0$ the projected variance is $\approx \frac{1}{2}\,\mathrm{Var}(q)$ and satisfying a standard-normal test (as with SIGReg) requires $\mathrm{Var}_{\text{per-dim}}(q) \approx 2$ as we measure $\mathrm{std}(q) = 1.36$ against a predicted $\sqrt{2} = 1.41$ Under `q`-mode, where the statistic sees only the 32 real dimensions, the variance lands at 0.941 ± 0.010, the target. The joint statistic can be blind to the momentum block and indeed it is satisfied by killing it and inflating the position block to compensate as we see with the largest ratios of $\mathrm{std}(p)/\mathrm{std}(q)$. When splitting, the within-episode variance increases significantly. However, the standard deviation of p though we apply the regularizer over, it remains far from unit. This is because $\mathbf{p}$ is not free, it is conditioned on $\mathbf{q}$ as it is estimated as its differentiation.

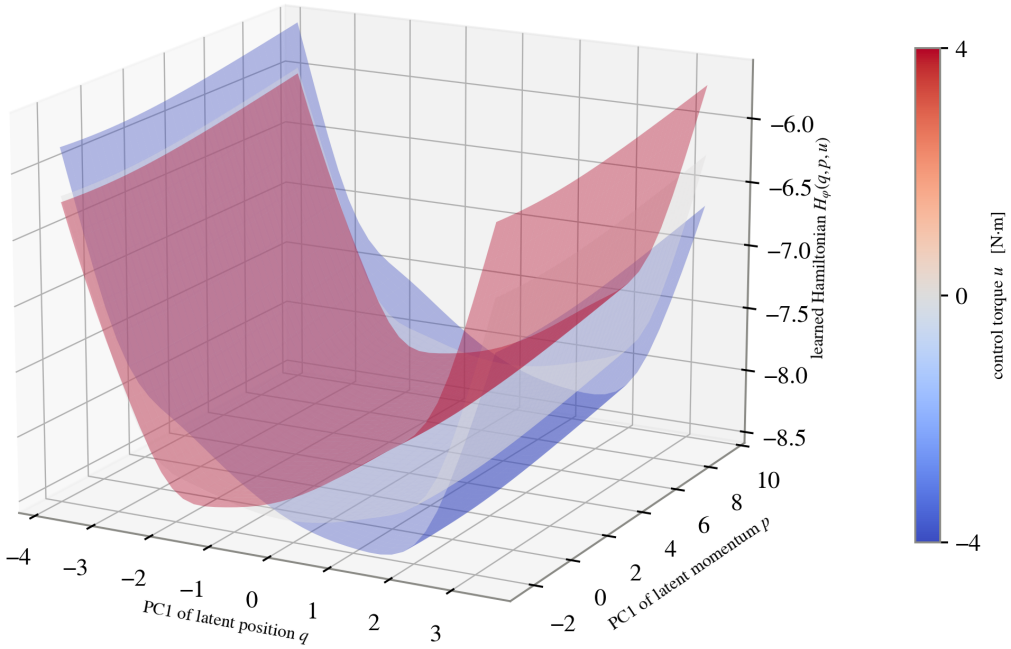

Figure 14: **The learned energy landscape under actuation.** The same projection as Fig. 11, for the actuated pendulum. The torque does work, so H_ϕ is not constant along a rollout and the trajectories cross the level sets instead of nesting into disjoint shells. Symplecticity is unaffected: it constrains the map, not the value of H_ϕ.

Table 8: **Within- versus between-episode variance of the encoder's latent coordinates, by SIGReg placement.** For each latent dimension, V_{within} is the mean squared deviation of an episode's trajectory about its own mean and V_{between} is the variance of those per-episode means; the column reports $V_{\text{within}}/(V_{\text{within}} + V_{\text{between}})$, averaged over dimensions. A value near zero means the latent is an episode identifier rather than a state. Measured on the encoder's own latents (`gt_q`, `gt_p`) over the test episodes at a common horizon $K = 0 \dots 74$; mean $\pm$ standard deviation over three training seeds. Bold marks the largest within-episode share per window. The unstructured baseline attains 42.8% on this measure but cannot be tabulated here, as its evaluation path stores latents only at $K = 0$.

window	SIGReg	within-episode %	std(q)	std(p)	std(p)/std(q)
$W = 1$	`joint`	0.01 ± 0.00	1.356	0.006	0.0043
	`q`	15.32 ± 0.29	0.980	0.101	0.1031
	`split`	$\mathbf{48.89 \pm 16.39}$	1.174	0.236	0.2014
$W = 3$	`joint`	4.90 ± 0.30	1.405	0.051	0.0362
	`q`	8.87 ± 0.54	0.979	0.051	0.0519
	`split`	$\mathbf{47.06 \pm 2.56}$	1.124	0.201	0.1784

Physics prediction in rollout. We studied the performance of LeHami applying SIGReg on three placements: $\lambda_S\,\mathrm{SIGReg}(\mathbf{q}\|\mathbf{p}])$ (`joint`), $\lambda_S\,\mathrm{SIGReg}(\mathbf{q})$ (`q`), and $\lambda_S\,\mathrm{SIGReg}(\mathbf{q}) + \lambda_S\,\mathrm{SIGReg}(\mathbf{p})$ (`split`). We study the variance of the embeddings within-episode and between-episode in Table 8. Based on this analysis, and on the performance in predicting the physical quantities of interest, applying SIGReg on the joint $[\mathbf{q}\|\mathbf{p}]$ has very bad performance. For this reason, we compare the performance in physics prediction of the other two variants. Table 1 reports the `q` placement, the model used for planning; the split placement is analysed in Table 8. LeHami significantly improves the reconstruction of the velocity, over a LeWM that performs worse than predicting the average. Regarding angle, the MAE achieved by LeWM is smaller than LeHami's at every probe tier, though the gap narrows as the probe gains capacity. This is attributed to the fact that LeHami splits the capacity of the latent representation space into $\mathbf{q}$ and $\mathbf{p}$, each (32) having the half the dimension of LeWM (64). Thus, an improvement in expressivity in one seems to deteriorate the other one.

The results corresponding to the rollouts are shown in Fig. 15. The probes are trained at the anchor on training data, and the evaluated frozen. While LeWM has significantly lower MAE in angle for the smaller rollout steps, after $K \sim 40$, the MAE overlaps with the one of LeHami. Regarding the velocity, for small K show significantly lower MAE. As rollout progresses, after $K \sim 10$, the performance is on par. This has to do, as commented before, that the predictor takes three context frames, and after rollouts, the embeddings start to have a better representation of velocity though it suffers the expected decay due to error accumulation through rollout.

Capturing velocity through the encoder and unstructured predictor. While expanding the time window improves the prediction of the angle θ of LeWM, it deteriorates the performance in predicting ω unless the temporal weights are explicitly decoupled. The reason is that with no time window ($W = 1$), the velocity is captured only by the predictor, which takes three context frames to perform the next prediction. In the "tied" $W = 3$ variant, the kernel of the temporal ConvNet is symmetric by construction, with weights $[a, b, a]$. Thus, the tied encoder is invariant to reversing the order of the three frames in its window. Reversing time leaves the centre angle unchanged but flips $\omega \to -\omega$; the states $(\theta, +\omega)$ and $(\theta, -\omega)$ are therefore mapped to the same latent, and no readout of that latent can recover the sign. For a pendulum at θ with rate ω, the window holds $\theta - \omega dt,\ \theta,\ \theta + \omega dt$, and the symmetric average is

$$\frac{b + 2a\cos(\omega\, dt)}{b + 2a}\,(\sin\theta,\ \cos\theta), \tag{16}$$

which corresponds to the middle frame (at time step t), shrunk by that scalar factor between 0 and 1. Note that this contraction is even because the cosine function is even (i.e., $\cos(\omega dt) = \cos(-\omega dt)$), so ω and $-\omega$ give the identical output. Conversely, the "untied" $W = 3$ variant removes this parameter sharing, learning independent weights $[w_1, w_2, w_3]$. This unconstrained asymmetry allows the untied encoder to natively distinguish forward from backward motion.

Table 9: **The comparison of the R^2 due to the time window in the encoder.** They correspond to the k-NN (k=25) probe applied on encoded anchor latents over held-out episodes, across 3 seeds. The n refers to the history size of the predictor, which is the same in both. Note that $W = 3$ refers here to the tied variant, which exacerbates the sign-blindness gap.

Model	n	$R^2(\omega)$	$R^2(\lvert\omega\rvert)$	gap
LeWM W=1	3	−0.048	+0.032	+0.080
LeWM W=3	3	−0.115	+0.921	+1.036

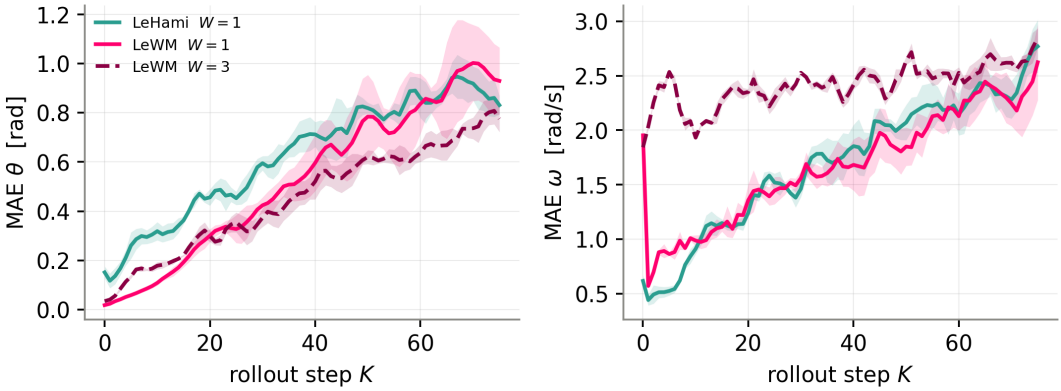

Figure 15: **Performance in physics predictions in rollout of the actuated pendulum.** Wrapped MAE of θ **(left)** and ω **(right)** along the test rollout, 25-NN probe fitted at $K = 1$ and applied frozen. Same encoder, same schedule, same regularizer weight. Shaded area corresponds to the standard deviation across the three architecture seeds.

This is measured empirically in Table 9 where it is shown that while the $R^2(|\omega|)$ is significantly improved with expanding the time window in the tied encoder, the variance explained by the vector ω is lower with a lower $R^2(\omega)$, and the gap between the two R^2 is amplified, in comparison to $W = 1$.

Both LeWM models use three frames for history so the predictor sees three consecutive latents to predict the next one. Ordering is preserved at the predictor even though the tied encoder discarded it, and the sign of ω is recoverable there. For $W = 1$, LeWM encodes no ω at all at $K = 0$ yet reads it at $R^2 \approx 0.90$ after a single predictor step, so that information can only have come from the ordering of its context. At tied W=3 that route survives but is degraded. With stride 1, latents j and j+1 are built from frames $\{j, j+1, j+2\}$ and $\{j+1, j+2, j+3\}$, respectively. Because these adjacent latents share two out of three frames, consecutive-frame motion is attenuated by 34% compared to $W = 1$. Since it is exactly this frame-to-frame change that carries the directional sign at the predictor level, the tied time window hurts twice over. It hands the model an easy, sign-blind estimate of speed from the encoder, and it attenuates the one signal coming from the change between consecutive latents, from which the direction could still have been read.

On the contrary, LeHami is on the other side of this symmetry. The Hamiltonian flow does not have any history; it is capturing velocity solely via p computed with a Savitzky–Golay finite difference of q. This finite difference is an odd operator, hence sign-aware by construction. Any consistent first-derivative estimator satisfies $\sum_m w_m = 0$ and $\sum_m w_m m = 1$, where m indexes frames within the stencil window; reversing time sends $m \to -m$ and hence the estimate to -1 times itself. The Hamiltonian model gets velocity not by seeing more frames or learning unrestricted asymmetric weights, but by applying an antisymmetric structural operator to them.

The distribution of the angles and velocities across the goals. As we perform a discussion about the models being able to capture the sign of the velocity, we show the distribution of the angles and the velocities across the goals for which we assess planning performance in Fig.16. Although the distribution of velocities is not centered around zero, and thus they are not balanced in sign, the

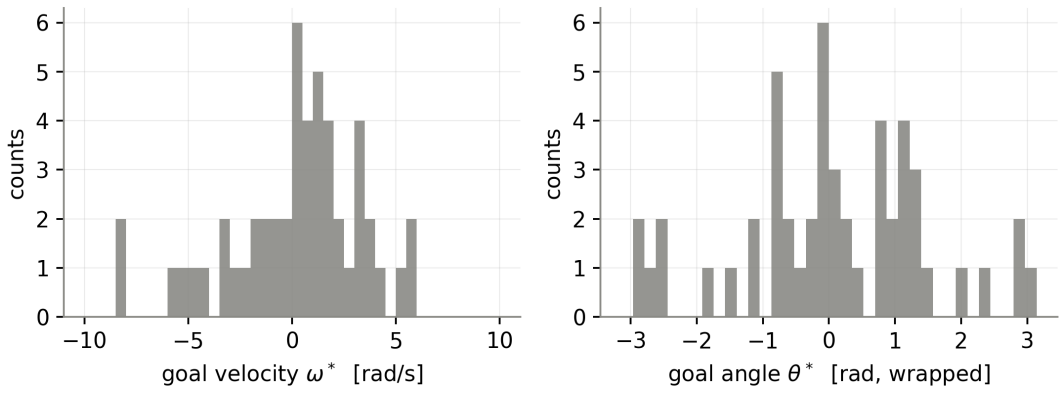

Figure 16: The 50 planning goals. Left: the goal velocity ω^*; dashed lines mark $\pm\tau_\omega/2$, beyond which a sign error alone fails the velocity clause. 92% of the goals lie outside them. Right: the goal angle θ^*, wrapped; the green lines mark upright, which no goal is within τ_θ of. Goals are recorded states of held-out test episodes at $K = 30$ frames after the start, shared identically by every model, rung and seed.

difference is not sufficient to back up a claim that the goals favored the planning performance in terms of absolute value.

Planning compute. At any fixed rung a planning episode costs LeHami $0.63\times$ the baseline's FLOPs. The saving is in the predictor rather than the encoder: over 99% of an episode's arithmetic is the CEM rollout, where LeHami's MLP flow costs 13.6 MFLOP per step per candidate against 21.7 for LeWM. LeWM's predictor is a causal, action-conditioned transformer over the three most recent latents, in which self-attention is the only operation that mixes them; LeHami instead recovers the momentum-like coordinate from a causal Savitzky–Golay stencil, so its predictor never has to form that mixture.

Decomposing the velocity clause in planning: speed vs. direction. To understand the representational failure modes during planning, the task's velocity clause $|\omega - \omega^*| < \tau_\omega$ can be decomposed into two distinct sub-components: a magnitude (speed) clause $\big||\omega| - |\omega^*|\big| < \tau_\omega$ and a directional sign clause $\mathrm{sign}(\omega) = \mathrm{sign}(\omega^*)$.

On this domain, directional sign is the decisive factor: 92% of goals satisfy $2|\omega^*| \geq \tau_\omega$, meaning an agent arriving at the goal state spinning in the wrong direction cannot satisfy the velocity clause regardless of speed precision. Table 10 reports performance on both components.

The magnitude clause is where temporal windowing assists the baseline: LeWM $W{=}3$ reaches 0.638 tied and 0.813 untied, compared to LeHami's 0.547. On the sign clause, LeWM $W{=}1$ (0.656) and tied LeWM $W{=}3$ (0.649) both sit slightly below the passive $u = 0$ coasting floor (0.680), so neither planner gets the sign right more often than passive dynamics do. For tied LeWM $W{=}3$ this is expected: its symmetric encoder maps opposite velocity directions to the same latent. Untying the kernel breaks this temporal parity, and untied LeWM $W{=}3$ gets the sign right most often (0.962). LeHami and the untied LeWM $W{=}3$ are the only models that improve over coasting on the sign clause; LeHami (0.800) does so without a learned temporal kernel and at $0.59\times$ the compute (8.05 against 13.65 TFLOP).

Conditioning on the goal angle, $P(\theta \wedge \omega) = P(\theta)\, P(\omega \mid \theta)$, separates the two bottlenecks. LeHami's advantage is on the sign, and it holds at every replanning interval: among plans that reach the goal angle, LeHami's final velocity has the goal's sign in 0.91–0.95 of cases, against 0.65–0.69 for tied LeWM $W{=}3$ and 0.55–0.67 for LeWM $W{=}1$. Only untied LeWM $W{=}3$ does better (0.95–0.99). The full velocity clause given the angle does not consistently favour LeHami.

What limits LeHami's joint success is reaching the goal angle. At $R = 9$ it reaches it on $P(\theta) = 0.48$ of plans, against 0.63 for tied and 0.70 for untied LeWM $W{=}3$, so $0.48 \times 0.55 = 0.26 < 0.63 \times 0.61 = 0.39$ for tied LeWM $W{=}3$. On this task the bottleneck is positional goal-reaching, not the direction of the velocity.

Table 10: **Velocity clause decomposition into magnitude and directional sign components.** Evaluated under the same computational budget ($\leq$ 15 TFLOP/episode) and protocol as Table 2.

Clause	Model	Success	TFLOP	Replanning Interval (R)
Velocity, magnitude	LeWM W=1	0.251 ± 0.051	8.78	$R = 15$, 2 solves
	LeWM W=3 (tied)	0.638 ± 0.008	13.65	$R = 9$, 4 solves
	LeWM W=3 (untied)	$\mathbf{0.813 \pm 0.052}$	13.65	$R = 9$, 4 solves
	LeHami	0.547 ± 0.022	10.98	$R = 6$, 5 solves
	$u = 0$ *coast*	*0.300*	—	—
Velocity, sign	LeWM W=1	0.656 ± 0.017	5.85	$R = 30$, 1 solve
	LeWM W=3 (tied)	0.649 ± 0.022	13.65	$R = 9$, 4 solves
	LeWM W=3 (untied)	$\mathbf{0.962 \pm 0.017}$	13.65	$R = 9$, 4 solves
	LeHami	0.800 ± 0.034	8.05	$R = 9$, 4 solves
	$u = 0$ *coast*	*0.680*	—	—

Prediction of the rod's length and mass. A symplectic flow conserves the Hamiltonian it has learned and phase-space volume, and nothing else by construction. The actuated pendulum has two further quantities that are exactly constant within an episode and visible in every frame: the rod length $L \in \{0.7, 1.0, 1.4\}$ m, drawn at its true length, and the bob mass $m \in \{0.5, 1.0, 2.0\}$ kg, drawn as a constant-density sphere of radius $\propto m^{1/3}$. Neither the architecture nor the loss asks the latent to keep them. They matter for control because the torque enters the dynamics only through $u/(mL^2)$ (Eq. 14), so a model that loses m cannot know what a given torque will do. We regress rather than classify: m and L are continuous parameters that happen to be sampled at three values, and R^2 registers the gradual drift of a frozen readout that a three-way accuracy would saturate over. Ridge and 25-NN probes are fitted on the anchor ($K = 0$) latents of the 630 training episodes to predict (m, L) jointly, then applied frozen to each model's own rollout and scored on the 135 test episodes.

Table 11 shows that both models carry both parameters. At the anchor every cell reads $R^2 \geq 0.999$. After 76 rollout steps the 25-NN probe is still essentially exact ($R^2 \geq 0.997$) for both models, while the frozen ridge probe drifts, more for LeHami (0.900 on L and 0.977 on m) than for LeWM (0.983 and 0.993). The drift is an affine deformation rather than a loss of information: by $K = 76$ the linear readout of L has over-dispersed by 3–6% and picked up a small offset, which a nearest-neighbour readout, depending only on which training points are closest, does not see. A local readout that keeps working while a fixed affine one loses calibration is the signature of a parameter that has been transported within the latent rather than destroyed. For control the drift is inconsequential: assigning the ridge prediction of m to the nearest of $\{0.5, 1.0, 2.0\}$ kg is correct on 99.3% of episodes at $K = 76$ for both models, against a majority-class floor of 34.8%. The planning results of Section 3.4 therefore cannot be attributed to either model being blind to the mass.

Table 11: **Performance in prediction of rod's length and mass in the actuated pendulum.** Both episode-constant parameters are recovered from the latent by probes fitted at anchor ($K = 0$) on training episodes and applied unchanged along each model's own rollout over 135 held-out episodes produced by LeHami and LeWM. R^2 is reported at the anchor and at the end of the horizon ($K = 76$). Three seeds for LeHami (min–max in brackets) for both ridge and 25-NN probes. Both models carry both parameters. While the k-NN (k=25) probe is exact at every K rollout, it is observed some drift for the ridge, affecting more LeHami. Note that The R^2 columns therefore measure how far the frozen readout drifts, not whether the parameter is still identifiable.

		rod length L		bob mass m	
probe	model	$K = 0$	$K = 76$	$K = 0$	$K = 76$
ridge	LeHami	0.999	0.900 [0.871, 0.951]	0.999	0.977 [0.958, 0.993]
	LeWM	0.999	0.983 [0.976, 0.990]	0.999	0.993
25-NN	LeHami	1.000	0.997 [0.992, 1.000]	1.000	1.000
	LeWM	1.000	1.000	1.000	1.000