# FLUIDJEPA: JOINT-EMBEDDING PREDICTIVE WORLD MODELS FOR CONTROLLED FLUID DYNAMICS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Modeling controlled fluid dynamics requires predicting how a flow evolves under a prescribed action sequence. Unlike forecasting under fixed forcing, the same initial state can lead to different futures as the control changes. Accurate next-field prediction, however, does not ensure that a model's internal state can support repeated action-conditioned prediction. We introduce `FluidJEPA`, a joint-embedding world model that learns physical-field prediction and latent dynamics together. Action-conditioned latent predictions are matched to encoded future observations. A jointly trained decoder retains physical-field supervision, and a distributional regularizer discourages representation collapse. We construct a benchmark of four two-dimensional flow environments spanning internal forcing and boundary actuation, with 8,000 trajectories organized into shared-initial-state control branches. Evaluation separates field-space feedback from latent rollout and tests control-induced changes and generalization to an action family withheld from training and validation. `FluidJEPA` achieves competitive field accuracy. In same-architecture ablations across the four environments, latent alignment reduces 20-step mean physical-field error under latent rollout by 73.6–96.6% relative to training without this alignment. Action-family holdout produces modest degradation in periodic Navier–Stokes latent prediction relative to training on all families. The larger degradation in Pinball highlights a remaining challenge when unseen actions also change actuator coordination.

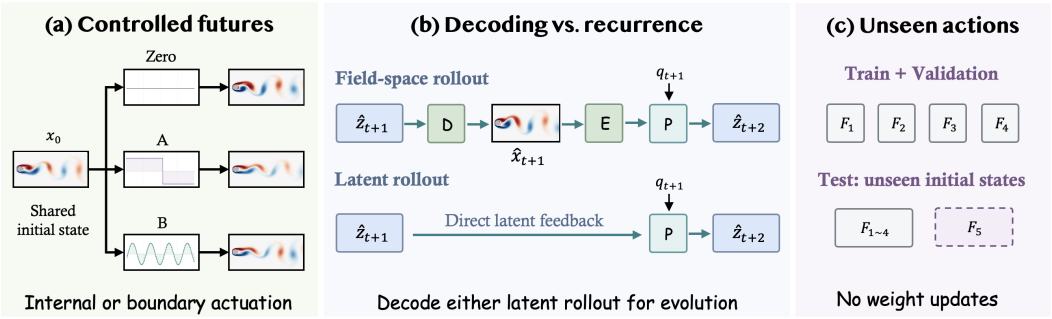



Figure 1: **Controlled futures, recurrence paths, held-out actions.** (a) Commands from one state $\boldsymbol{x}_0$ give paired futures, read against the zero branch. (b) Field rollout re-encodes each decoded state; latent rollout does not. (c) A withheld family is tested with no weight updates; sketches are schematic.

## 1 INTRODUCTION

Unsteady flows past bluff bodies shed vortices, and those vortices set the drag, lift, and mixing that engineering systems must live with. Small inputs at the boundary can reorganize this behavior: blowing or suction through a narrow slot, or the rotation of an immersed cylinder, alters the shear layer that feeds the wake, and a modest body force can redirect the evolution of a periodic flow (Lagemann et al., 2026). The governing incompressible Navier–Stokes equations are deterministic, so once the state and the inputs are fixed, the future is determined. The practical difficulty is that these inputs



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.

are ours to choose, and each choice sends the same flow down a different trajectory. Comparing candidate interventions therefore means predicting several futures from one present.

Numerical solvers answer such questions accurately but at a cost that scales with the number of candidates, which motivates learned surrogates (Rabault et al., 2019; Brunton et al., 2020; Vinuesa & Brunton, 2022). Two routes dominate. Neural operators map fields to fields, learning a discretized solution operator directly in physical space (Li et al., 2021; Rahman et al., 2023; Lu et al., 2021). Latent evolution models instead compress the state, advance it, and decode the result, so that the transition is learned on a reduced representation (Wu et al., 2022). Both routes accept a control input, and both are trained on the same signal: the error of the decoded future field.

That training signal is weaker than it appears. A decoder is a many-to-one map, so it can absorb components of a latent error that leave the output field nearly unchanged. Such components are invisible to field supervision, yet they are still present in the state, and the predictor consumes that state at the next step. Accurate decoding at one step therefore does not certify that the internal state remains usable for the step after. Joint-embedding predictive architectures address exactly this asymmetry by supervising a predicted representation against an encoded target rather than only the observable output (Assran et al., 2023; Bardes et al., 2024), a principle since extended to end-to-end action-conditioned dynamics (Maes et al., 2026) and to PDE and aerodynamic settings (Gallagher & Guglielmi, 2026; Giral et al., 2026). What remains untested is whether, in unsteady flows where each action changes the evolution to be predicted, the two capabilities actually come apart.

They can be separated by running one trained model along two feedback paths (Figure 1b). *Field rollout* decodes each prediction and re-encodes it before advancing; *latent rollout* returns the predicted representation straight to the predictor. Both start from one observation, receive identical future actions, and are scored through decoded fields, so neither is given an easier target. The gap between them measures what re-encoding silently repairs. This matters beyond diagnosis: scoring candidate action sequences or planning consumes the latent path, and decoding at every step reinstates the dense field computation that a latent model exists to avoid (Gallagher & Guglielmi, 2026).

| ***Key principle*** | *A state that decodes well is not yet a state that predicts again; field accuracy and latent recurrence must be supervised and measured apart.* |
|---|---|

We instantiate this principle in `FluidJEPA`, which trains field prediction and action-conditioned latent dynamics jointly. An encoder maps the flow to spatial tokens, a predictor advances them under the supplied control, and a jointly trained decoder returns the next field. Alongside future-field supervision, predicted latents are matched to encoded future observations, and SIGReg (Balestriero & LeCun, 2025) keeps the representation from collapsing. Because the same tokens are decoded, matched, and carried forward, with no auxiliary projection in between, the two feedback paths compare a single learned state rather than two different ones.

Measuring the principle also requires data in which control has visible consequences. We build a benchmark of four two-dimensional environments that span internal forcing and boundary actuation, namely periodic Navier–Stokes, Cylinder, Pinball, and Cavity, with 8,000 trajectories in total (Figure 1a). Five control branches share each initial state, and branches never cross a split, so the change relative to zero control is measured rather than approximated. We further withhold one action family from training and validation (Figure 1c), which separates transfer to a new action-generation mechanism from resampling within a familiar one. Across these environments `FluidJEPA` attains competitive field accuracy, while removing latent matching leaves field rollout nearly unchanged yet degrades latent rollout by up to an order of magnitude. In summary, we make the following key contributions:

- ❶ **Paired benchmark.** We organize four flow environments into shared-initial-state control branches with aligned action sequences and initial-state-disjoint splits, so control responses and full fields are both measurable.
- ❷ **Joint objective.** We combine physical-field supervision, action-conditioned latent matching, and distributional regularization over one shared state, and expose decoded accuracy and state reuse as two rollout paths of the same model.
- ❸ **Separated evaluation.** We withhold an action family from training and validation, and show across four environments that field accuracy, latent recurrence, and action-family transfer move independently.

## 2 Related Work

**Learned fluid surrogates.** Neural operators predict fields directly, parameterizing the solution operator in Fourier space (Li et al., 2021) or through multiscale operator blocks (Rahman et al., 2023), while mesh-based models propagate updates over simulation graphs (Pfaff et al., 2021). Latent evolution instead advances a compressed state before decoding (Wu et al., 2022; Sanchez-Gonzalez et al., 2020). Control environments expose the resulting dependence on input sequences (Zhang et al., 2024; Bhan et al., 2024; Lagemann et al., 2026). We reuse existing solvers and actuators rather than proposing numerics, and add an offline protocol in which branches share an initial state, so a control response is measured against a matched reference instead of an average trajectory.

**Action-conditioned world models.** World Models and Dreamer couple observation reconstruction with recurrent latent dynamics (Ha & Schmidhuber, 2018; Hafner et al., 2020), TD-MPC ties latent transitions to a value function for planning (Hansen et al., 2022), and DINO-WM predicts future features from a pretrained encoder (Zhou et al., 2025). These establish that latent transitions can be learned and then used, but their representations are shaped by reconstruction, reward, or a frozen backbone. We ask instead how dense physical-field supervision and predictive latent matching shape one state jointly, and we read that state along two rollout paths of the same trained model.

**Joint-embedding prediction.** I-JEPA supervises predicted representations against encoded targets (Assran et al., 2023; Bardes et al., 2024), LeJEPA regularizes them toward an isotropic Gaussian (Balestriero & LeCun, 2025), and LeWorldModel combines both for end-to-end action-conditioned dynamics (Maes et al., 2026). Physical instances exist: joint-embedding dynamics with latent planning for PDE control (Gallagher & Guglielmi, 2026), and coupled embedding and field supervision for steady aerodynamic fields (Giral et al., 2026; Bardes et al., 2022). We therefore claim neither the first fluid JEPA nor the first joint supervision of fields and representations. Our contribution is to test, in unsteady flows driven by time-varying actions, whether field accuracy and latent recurrence separate, and to report them apart together with transfer to a withheld action family.

## 3 Preliminaries

### 3.1 Problem Formulation.

We consider two-dimensional incompressible flows driven by prescribed controls. Let $s_t$ be the complete numerical state at time $t\Delta t$, including integration history and actuator memory, and let $\mu$ fix the geometry and flow parameters, so that

$$s_{t+1} = \mathcal{F}_\mu(s_t, \boldsymbol{A}_t), \quad \boldsymbol{x}_t = \mathcal{O}_\mu(s_t) \in \mathbb{R}^{C\times H\times W}, \quad \boldsymbol{m} \in \{0,1\}^{H\times W}, \tag{1}$$

where $\boldsymbol{x}_t$ holds $C$ physical channels on an $H \times W$ grid and $\boldsymbol{m}$ marks valid fluid pixels; since $\mathcal{O}_\mu$ may filter or crop, one observation need not be a complete Markov state. The block $\boldsymbol{A}_t \in \mathbb{R}^{S\times d_a}$ holds the $S$ ordered commands applied during one interval. Because boundary actuators are first-order filtered, a command is not the instantaneous boundary velocity, so $\boldsymbol{q}_t$ carries the command block together with the actuator state, advanced from known commands rather than read from future measurements. The learning task is $\widehat{\boldsymbol{x}}_{1:L} = \mathcal{M}_\Theta(\boldsymbol{x}_0, \boldsymbol{q}_{0:L-1}, \boldsymbol{m})$ with $L = 20$ and no ground-truth field after $\boldsymbol{x}_0$: we predict the consequences of supplied actions, not an optimized control policy.

### 3.2 Controlled Fluid Dynamics Benchmark

Here we introduce our Benchmark. Four environments (periodic Navier–Stokes (NS), Cylinder, Pinball, Cavity) cover internal forcing and boundary actuation, NS on an UNO-derived spectral solver (Rahman et al., 2023) and the others on HydroGym's Firedrake solvers (Lagemann et al., 2026). Each holds 400 initial-state groups with five control branches: 2,000 trajectories per environment, 8,000 total. Within group $i$ every branch resumes from one identical complete state, $s_0^{i,b} = s_0^{i,0}$, restoring integration history and actuator memory rather than only the visible field. Branch $b = 0$ applies no additional command, so the paired response $\Delta\boldsymbol{x}_t^{i,b} = \boldsymbol{x}_t^{i,b} - \boldsymbol{x}_t^{i,0}$ isolates the effect of control at the same physical time rather than differencing against an average trajectory; the four nonzero branches are distinct action families, resampled per group.

Table 1: **Controlled fluid benchmark.** Every learning sequence contains 21 fields, $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{20}$, and $L = 20$ prediction intervals. $\Delta t$ is the observation interval, not the solver step; $T = L\Delta t$ is the physical duration in each environment's nondimensional units. Each interval contains $S$ ordered commands of dimension $d_a$, held for $\delta t = \Delta t / S$. The listed grids are observation grids. NS records $\omega$; the others record $(\omega, u, v, p)$. Sampling and control details are given in Appendix A.

| Environment | Actuation | $C \times H \times W$ | $d_a$ | $S$ | $\Delta t$ | $T$ |
|---|---|---|---|---|---|---|
| NS | Fourier source | $1 \times 64 \times 64$ | 4 | 1 | 1 | 20 |
| Cylinder | Two wall jets | $4 \times 64 \times 128$ | 1 | 5 | 1 | 20 |
| Pinball | Three wall rotations | $4 \times 64 \times 128$ | 3 | 5 | 0.5 | 10 |
| Cavity | Leading-edge actuation | $4 \times 64 \times 128$ | 1 | 5 | 0.125 | 2.5 |

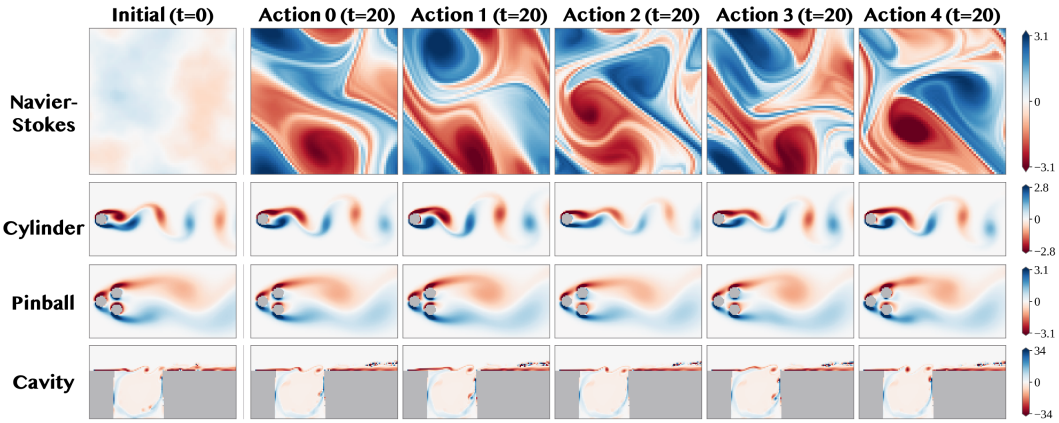



Figure 2: **Shared initial states, different controlled futures.** Each row gives one environment's shared initial vorticity, then terminal fields under all five branches.

**Splits and holdout.** Splits are disjoint by initial-state group, 320/40/40, so no test initial state appears in training under any branch, and normalization uses training-only statistics. The action-family holdout removes one family entirely, biased multisine for NS and sinusoidal control otherwise, so only branches 0 to 3 enter training, validation, normalization, and checkpoint selection. The same 40 unseen test initial states then supply 160 seen-family and 40 held-out-family trajectories, scored without weight updates. Since models receive numerical control information rather than family labels, this tests a new action-generation mechanism, not new samples from a familiar family. Solver settings, control profiles, and pre-rolls are deferred to Appendix A.

## 4 Methodology

**Overview.** `FluidJEPA` learns spatial states that both decode the future flow and support the next action-conditioned prediction. An encoder $E_\theta$, predictor $P_\phi$, and decoder $D_\psi$ are trained jointly from scratch, following the end-to-end predictive representation approach of Maes et al. (2026). One transition is:

$$\boldsymbol{z}_t = E_\theta(\widetilde{\boldsymbol{x}}_t, \boldsymbol{m}), \qquad \widehat{\boldsymbol{z}}_{t+1} = P_\phi(\boldsymbol{z}_t, \boldsymbol{q}_t, \boldsymbol{m}), \qquad \widehat{\widetilde{\boldsymbol{x}}}_{t+1} = D_\psi(\widehat{\boldsymbol{z}}_{t+1}, \boldsymbol{m}), \tag{2}$$

where $\widetilde{\boldsymbol{x}}_t$ is the channel-wise normalized field and $\boldsymbol{q}_t$ the control over the next interval. All three modules act on the same spatial token grid, share one architectural design across environments, and hold separate weights per environment. Figure 3 summarizes the training objectives.

### 4.1 Action-Conditioned Spatial Architecture

**Spatial encoding.** We normalize each physical channel by its training statistics, zero invalid pixels, and concatenate the geometry mask as a known input channel for boundary-controlled environments; periodic NS needs no mask channel. A linear embedding maps each $4 \times 4$ patch to a token, a

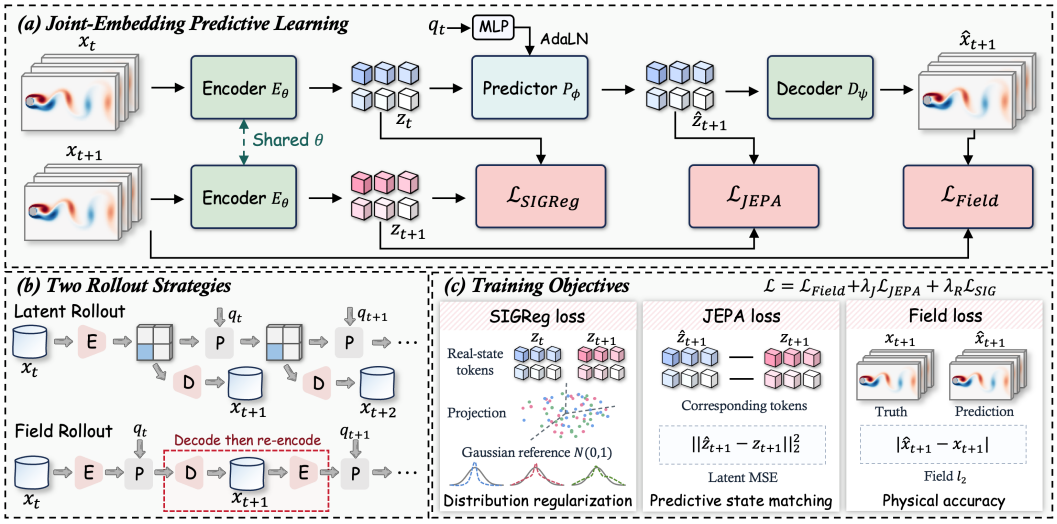



Figure 3: **Joint learning of physical predictions and recurrent representations.** (a) The predictor receives the interval control through AdaLN; its output is decoded and matched to the encoded future observation, with both encoders sharing trainable $\theta$ and no stop-gradient. (b) Latent rollout returns predicted tokens to the predictor, while field rollout decodes and re-encodes first. (c) The three objectives constrain distribution, predicted state, and physical field.

Transformer stack mixes tokens, and a two-layer state head produces:

$$\boldsymbol{z}_{t,r} = G_E(\mathrm{LN}(\mathcal{T}_E(\mathrm{Embed}_4(\widetilde{\boldsymbol{x}}_t, \boldsymbol{m}) + \boldsymbol{B}_E; \boldsymbol{m}))_r), \qquad r \in \mathcal{V}, \tag{3}$$

with learned position embeddings $\boldsymbol{B}_E$ and valid patches $\mathcal{V}$. No normalization follows $G_E$, so state coordinates can learn their own means and scales. With $d = 192$ and $N = HW/16$, the state is $256 \times 192$ for NS and $512 \times 192$ otherwise. Retaining the token grid preserves spatial indexing for dense decoding, while attention lets each token draw on other flow regions. A patch is valid if it holds at least one fluid pixel; purely invalid patches are dropped as attention keys and zeroed after each block, whereas mixed patches stay valid and the mask separates their pixels. Field losses use the finer pixel mask, so keeping a mixed patch never supervises its solid pixels.

**Action-conditioned prediction.** The control enters as $\boldsymbol{q}_t = \boldsymbol{a}_t \in \mathbb{R}^4$ for NS and $\boldsymbol{q}_t = [\mathrm{vec}(\boldsymbol{A}_t); \boldsymbol{c}_t]/s_a$ for boundary-controlled flows, where $s_a$ is the fixed command amplitude; the five commands keep their order instead of being averaged, and the actuator advances analytically through them. Hence $\boldsymbol{q}_t$ has dimension four for NS, six for Cylinder and Cavity, and eighteen for Pinball. A conditioning MLP gives $\boldsymbol{h}_t = \mathrm{MLP}_{\mathrm{cond}}(\boldsymbol{q}_t) \in \mathbb{R}^{192}$, which supplies channel-wise scale and shift to both LayerNorms of every predictor block:

$$\mathcal{A}_{\ell,s}(\boldsymbol{U}, \boldsymbol{h}_t) = \big(1 + \boldsymbol{\gamma}_{\ell,s}(\boldsymbol{h}_t)\big) \odot \mathrm{LN}(\boldsymbol{U}) + \boldsymbol{\beta}_{\ell,s}(\boldsymbol{h}_t), \qquad s \in \{1, 2\}, \tag{4}$$

so that $\boldsymbol{V}^\ell = \boldsymbol{U}^\ell + \mathrm{Attn}_\ell(\mathcal{A}_{\ell,1}(\boldsymbol{U}^\ell, \boldsymbol{h}_t); \boldsymbol{m})$ and $\boldsymbol{U}^{\ell+1} = \boldsymbol{V}^\ell + \mathrm{FFN}_\ell(\mathcal{A}_{\ell,2}(\boldsymbol{V}^\ell, \boldsymbol{h}_t))$, starting from $\boldsymbol{U}^0 = \boldsymbol{z}_t + \boldsymbol{B}_P$. Modulation is broadcast over tokens, and attention mixes only the current state. The control coefficients describe fixed actuators, while positions, geometry, and current features let the network learn their nonuniform flow response; we do not expand the control into a forcing image. After $n_P = 6$ blocks a linear head returns $\widehat{\boldsymbol{z}}_{t+1} = G_P(\mathrm{LN}(\boldsymbol{U}^{n_P})) \in \mathbb{R}^{N \times d}$, which is the next state itself rather than an increment, matching the encoder's shape and token order so it can enter the next call undecoded.

**Physical-field decoding.** The decoder adds its own position embedding, applies a Transformer stack, and unpatches a token-wise linear head into a dense field, $\widehat{\widetilde{\boldsymbol{x}}}_{t+1} = \mathrm{Unpatch}_4(G_D(\mathrm{LN}(\mathcal{T}_D(\widehat{\boldsymbol{z}}_{t+1} + \boldsymbol{B}_D; \boldsymbol{m})))) \odot \boldsymbol{m}$. NS emits one vorticity channel; the others jointly emit vorticity, two velocity components, and pressure, each denormalized for supervision. The decoder takes no separate action input and no skip connection from the observed field, so the route from observation and control to the predicted field passes entirely through $\widehat{\boldsymbol{z}}_{t+1}$. Encoder, predictor,

and decoder use six, six, and four blocks at equal width, giving 8.3–8.5M parameters. Crucially, exactly the same tokens are decoded, matched to the encoded future, and reused for recurrence: no auxiliary projection separates the supervised representation from the recurrent one.

### 4.2 Joint Physical and Predictive Objectives

A shared interface alone does not make a well-decoded state suitable for the next transition, so we supervise fields and representations together. For a window of $K$ transitions indexed from zero, we set $\widehat{\boldsymbol{z}}_0 = E_\theta(\widetilde{\boldsymbol{x}}_0, \boldsymbol{m})$, apply Eq. 2 recursively by feeding each predicted latent onward, and let $\mathcal{J}$ index supervised steps. Writing $\boldsymbol{x}_j$ for true fields and $\boldsymbol{z}_j = E_\theta(\widetilde{\boldsymbol{x}}_j, \boldsymbol{m})$ for their encodings, the field and matching terms are

$$\mathcal{L}_{\text{field}} = \mathbb{E}\left[\frac{1}{|\mathcal{J}|C}\sum_{j\in\mathcal{J}}\sum_{c=1}^{C}\frac{\|\boldsymbol{m}\odot(\widehat{\boldsymbol{x}}_{j,c}-\boldsymbol{x}_{j,c})\|_2}{\max(\|\boldsymbol{m}\odot\boldsymbol{x}_{j,c}\|_2,\varepsilon_c)}\right],\ \mathcal{L}_{\text{JEPA}} = \mathbb{E}\left[\frac{\sum_{j\in\mathcal{J}}\sum_{r\in\mathcal{V}}\|\widehat{\boldsymbol{z}}_{j,r}-\boldsymbol{z}_{j,r}\|_2^2}{|\mathcal{J}||\mathcal{V}|d}\right]. \tag{5}$$

The field term is an unsquared relative $L_2$ loss in physical units, not pixelwise MSE, and it supervises future predictions without a separate autoencoding term on $D_\psi(E_\theta(\widetilde{\boldsymbol{x}}_j))$; the floor $\varepsilon_c$ prevents division by a near-zero reference norm. Gradients pass through both the predicted and target encodings, with no EMA encoder, frozen target, or stop-gradient, and future observations never replace a predicted recurrent input.

Matching alone admits uninformative representations, so we apply SIGReg (Balestriero & LeCun, 2025) to at most $n \le 128$ valid tokens sampled from real encodings at steps $\{0\}\cup\mathcal{J}$. For $M = 1{,}024$ random unit directions $\boldsymbol{v}_\ell$ it compares empirical characteristic functions against a standard Gaussian,

$$\mathcal{L}_{\text{SIG}} = \frac{n}{M}\sum_{\ell=1}^{M}\sum_{r=0}^{16} w_r\left|\frac{1}{n}\sum_{i=1}^{n}\exp\left(\mathrm{i}\,\xi_r\,\boldsymbol{v}_\ell^\top\boldsymbol{w}_i\right) - e^{-\xi_r^2/2}\right|^2, \qquad \xi_r = 3r/16, \tag{6}$$

with quadrature weights $w_r$ and the factor $n$ both part of the implemented objective. Predicted tokens are excluded from its pool, so the regularizer constrains observed representations only; this encourages distributional diversity but certifies neither non-collapse nor stable recurrence. The full objective is $\mathcal{L} = \mathcal{L}_{\text{field}} + \lambda_{\text{J}}\mathcal{L}_{\text{JEPA}} + \lambda_{\text{R}}\mathcal{L}_{\text{SIG}}$ with $\lambda_{\text{J}} = 1$ and $\lambda_{\text{R}} = 0.09$. The main setting uses $K = 1$ and $\mathcal{J} = \{1\}$. We also study terminal-only two-step fine-tuning with $K = 2$ and $\mathcal{J} = \{2\}$, where gradients pass through both predictor calls and SIGReg uses $\boldsymbol{z}_0, \boldsymbol{z}_2$.

With identical weights, initial observation, geometry, and controls, and the mask left implicit, the two paths are

$$\widehat{\boldsymbol{z}}^{\text{L}}_{t+1} = P_\phi(\widehat{\boldsymbol{z}}^{\text{L}}_t, \boldsymbol{q}_t), \qquad \widehat{\widehat{\boldsymbol{x}}}^{\text{F}}_{t+1} = D_\psi\left(P_\phi\left(E_\theta(\widehat{\widehat{\boldsymbol{x}}}^{\text{F}}_t), \boldsymbol{q}_t\right)\right), \tag{7}$$

both initialized from $\widetilde{\boldsymbol{x}}_0$ and read out through denormalized fields. Latent rollout treats decoding as a readout that cannot affect the next transition, whereas field rollout decodes and re-encodes at every step. Their first predictions coincide, and later ones may diverge because re-encoding need not preserve a predicted latent. Latent-rollout error is therefore still a decoded forecast error, not a latent distance, and these names denote neural feedback paths rather than PDE time integrators.

## 5 Experiments

Our experiments ask whether a state that decodes accurately is also a state that can be predicted from again: whether joint training keeps field accuracy competitive while exposing a usable latent path (§5.1), whether either capability survives a withheld action family (§5.2), and which objective makes predicted representations reusable (§5.3).

**Setup.** Each environment provides 400 initial conditions (ICs) with five control branches, split 320/40/40 by IC so all branches of an IC stay together, and every model predicts 20 transitions from one initial field and the prescribed commands with no further observations. The metric is physical-field relative $L_2$ after inverse normalization, on valid pixels, averaged equally over channels, times, and test trajectories. Training uses $K$=1, 200 epochs, batch size 16, and learning rate $3 \times 10^{-4}$;

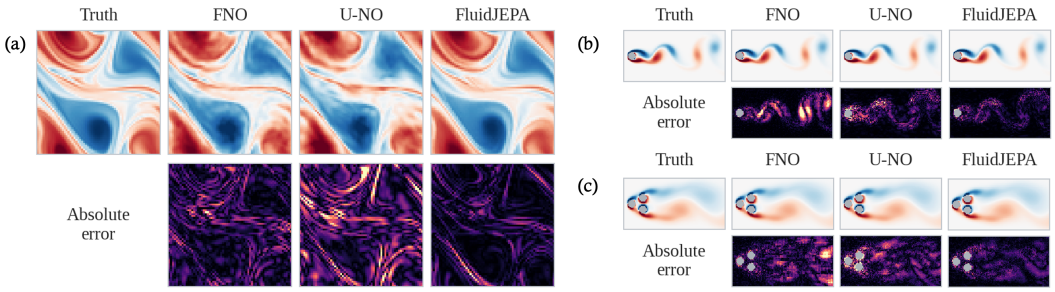



Figure 4: **Where the error lives.** Vorticity and absolute-error maps for FNO, U-NO, and `FluidJEPA` on (a) NS, (b) Cylinder, (c) Pinball; gray marks invalid pixels. These panels illustrate error structure only: their checkpoint provenance differs from Table 2 and does not establish the ranking.

`FluidJEPA` checkpoints are selected by validation latent-rollout error, operator checkpoints by validation field-rollout error. All results use a single training seed, which we treat as a limitation rather than a controlled variable (Appendix B).

### 5.1 Field Accuracy and Latent Recurrence Are Not the Same Axis

We compare `FluidJEPA` against action-conditioned FNO (Li et al., 2021) and U-NO (Rahman et al., 2023), which receive identical controls, coordinates, and geometry at approximately matched capacity (8.3 to 8.5M parameters) and training budget. Because the operators emit fields directly, only field rollout is defined for them.

Table 2: **Twenty-step forecasting on four controlled flows.** Mean physical-field relative $L_2$ over 200 test trajectories; lower is better. Bold marks the best field rollout. The latent column uses the same weights as `FluidJEPA`'s field rollout.

| | Field rollout | | | Latent rollout |
|---|---|---|---|---|
| Environment | FNO | U-NO | FluidJEPA | FluidJEPA |
| NS | 0.0436 | 0.0668 | **0.0375** | 0.0999 |
| Cylinder | 0.0077 | 0.0099 | **0.0068** | 0.0206 |
| Pinball | **0.0062** | 0.0077 | 0.0077 | 0.0246 |
| Cavity | 0.1421 | 0.1606 | **0.1059** | 0.1390 |

**Competitive, not dominant.** `FluidJEPA` attains the lowest field error on three of four environments, reducing FNO's error by 14.0% on NS, 11.6% on Cylinder, and 25.5% on Cavity (Table 2); Pinball is the exception, where FNO reaches 0.0062 against our 0.0077. We also decline to lean on the Cylinder margin, whose paired bootstrap interval over test ICs contains zero. The defensible reading is that adding latent objectives leaves decoded accuracy broadly intact, not that it yields a uniformly better forecaster: architectures and selection criteria both differ here, and only the same-architecture study in §5.3 isolates the objectives. For all three models the residual error concentrates in shear layers and vortex cores rather than the free stream (Figure 4).

**One model, two behaviors.** Reading the identical $K{=}1$ model through latent rollout multiplies its error by 1.31 to 3.22 (Table 2). Both paths issue the same first prediction under the same commands, so the gap is attributable to what each feeds forward: a re-encoded decoded field is not interchangeable with a predicted latent state. Figure 5 traces the gap over the horizon and shows it responds to training, as fine-tuning the predictor on its own two-step latent output cuts latent-rollout error by 43.6%, 53.4%, 45.8%, and 18.3% on NS, Cylinder, Pinball, and Cavity while field-rollout error rises in all four. Exposing a path during training improves that path, confirming the two are separately controllable; but the fine-tuned latent rollout still does not reach $K{=}1$ field rollout, and the stage also changes the learning rate and adds optimizer steps, so it does not isolate the horizon.

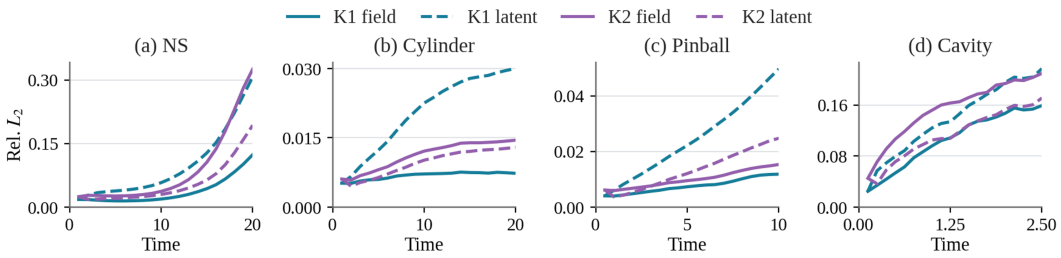



Figure 5: **Error accumulation before and after two-step fine-tuning.** Curves average 200 test trajectories and all channels over 20 predictions against physical time. Color separates $K=1$ from $K=2$; solid and dashed separate field from latent rollout. Fine-tuning lowers every latent curve and raises every field curve.

Table 3: **Matched evaluation on the fifth action family.** Both models predict the same 40 fifth-family trajectories. ID/OOD refers to that family, not to the test IC, which is unseen by both. Entries are mean full-field relative $L_2$; ratios divide four-family by five-family error within a path.

| | Five-family (ID) | | Four-family (OOD) | | OOD/ID | |
|---|---|---|---|---|---|---|
| Environment | Field | Latent | Field | Latent | Field | Latent |
| NS | 0.0486 | 0.1187 | 0.0766 | 0.1325 | 1.58 | 1.12 |
| Cylinder | 0.0078 | 0.0226 | 0.0297 | 0.0457 | 3.82 | 2.02 |
| Pinball | 0.0078 | 0.0241 | 0.0797 | 0.0947 | 10.24 | 3.93 |
| Cavity | 0.1004 | 0.1271 | 0.1957 | 0.2233 | 1.95 | 1.76 |

### 5.2 Transfer to a Withheld Action Family

A model that merely interpolates within familiar command shapes would satisfy §5.1 without learning controllable dynamics. We therefore train a second model on the first four families only, including zero control, withholding the fifth entirely: biased multisine for NS, sinusoidal control elsewhere. Both models are scored on the *same* 40 fifth-family trajectories, one per held-out IC, with no test-time adaptation, so the family is out of distribution for one model and in distribution for the other while the ICs are unseen by both.

**Structure of the shift, not its size.** The penalty is mild where the withheld family is a new waveform on a familiar actuator and severe where it also changes how actuators coordinate. NS latent error rises only from 0.1187 to 0.1325, a factor of 1.12, against latent factors of 2.02 on Cylinder, 1.76 on Cavity, and 3.93 on Pinball (Table 3). Pinball is the informative failure: its held-out sinusoidal family retimes three independent cylinder rotations relative to one another, so the model meets an unseen actuation *pattern* rather than an unseen amplitude, and field-rollout error grows by a factor of 10.24 against the latent path's 3.93. We read this as limited transfer with one clear failure mode rather than robustness to arbitrary commands: geometry, Reynolds number, and actuator interfaces stay fixed, and four-family training also sees fewer trajectories and updates.

### 5.3 Which Objective Makes a State Reusable

The baseline comparison varies architecture and selection criteria at once. To attribute the latent behavior to the objectives, we delete JEPA matching, SIGReg, or both while holding architecture, schedule, and initial learning rate fixed, and score the common epoch-200 checkpoint; a one-step evaluation supplying true context at every transition separates local prediction quality from self-fed recurrence.

**Alignment is cheap in field error, decisive in latent error.** On NS, removing JEPA moves field error only from 0.0363 to 0.0383 while latent error jumps from 0.1039 to 0.9340, a factor of nine (Table 4); across all four environments the full objective reduces latent error by 73.6% to 96.6% relative to w/o JEPA with field error essentially unchanged. The sharpest counterexample is Cavity,

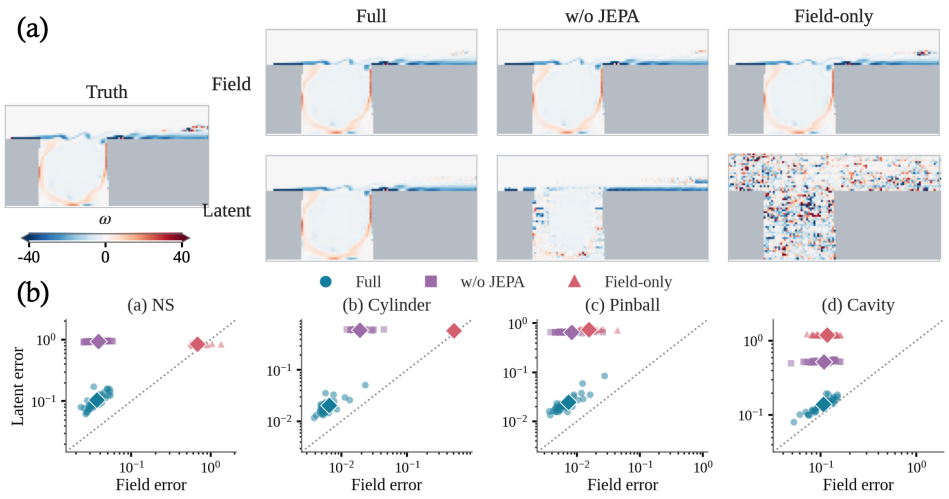



Figure 6: **Field accuracy does not determine latent-rollout accuracy.** (a) Terminal Cavity vorticity for one trajectory under Full, w/o JEPA, and Field-only along both paths, shared color scale. (b) IC-level errors on NS, Cylinder, Pinball, Cavity; each point averages one IC's five branches over 20 predictions, diamonds mark means, both log axes measure decoded field error. Epoch 200; diverged w/o SIGReg runs omitted.

where Field-only attains a *lower* one-step error than the full objective (0.0533 versus 0.0592) and a comparable field rollout, yet its latent rollout reaches 1.2041. A model can therefore be locally accurate and decode well while holding a state that is unusable one step later, which is exactly the failure field supervision cannot see; Figure 6 shows it as horizontal spread in the IC-level scatter, where points share a field error but differ by an order of magnitude in latent error.

**Regularization buys trainability.** All four w/o SIGReg runs reach nonfinite gradients between epochs 29 and 63, and we assign them no score rather than substituting an earlier checkpoint; Field-only completes training but degrades late on NS and Cylinder. We keep these outcomes distinct, numerical termination versus poor final accuracy: together they indicate the regularizer stabilizes optimization here, which is weaker than showing it necessary under any optimizer. Replacing SIGReg on NS reads the same way, as variance–covariance alternatives (Bardes et al., 2022) hold field error comparable while their latent errors reach 3.13 and 2.68 times ours.

Table 4: **Objective deletion at epoch 200 (NS, Cylinder).** One-step supplies true context at each transition; rollouts start from the initial observation only.

| Env. | Objective | 1-step | Field | Latent |
|---|---|---|---|---|
| NS | Full | 0.0179 | **0.0363** | **0.1039** |
| | w/o JEPA | **0.0170** | 0.0383 | 0.9340 |
| | Field-only | 0.3376 | 0.6848 | 0.8419 |
| | w/o SIGReg | – | – | – |
| Cyl. | Full | **0.0031** | **0.0066** | **0.0207** |
| | w/o JEPA | 0.0096 | 0.0192 | 0.6084 |
| | Field-only | 0.3016 | 0.5124 | 0.5843 |
| | w/o SIGReg | – | – | – |

## 6 Conclusion

We introduced a controlled-fluid benchmark and `FluidJEPA`, which jointly learns physical-field predictions and action-conditioned predictive representations. Across four environments, the framework retains competitive field forecasting while substantially improving latent recurrence over loss-deletion controls. Two-step fine-tuning further improves latent rollout, but not field feedback. Action-family holdouts reveal both transfer and substantial failures, particularly when actuator coordination changes. Together, these results distinguish accurate decoding from reusable predictive dynamics. The present evidence concerns offline prediction with one training seed; broader physical shifts, action selection, and closed-loop planning remain to be tested.

## AI USE STATEMENT

We used generative AI tools to help run the experiments, translate and polish the text, and search the literature; every cited paper was checked against its source by the authors. We carefully checked all code to ensure that it is correct. We take responsibility for the final content of this work, including text, claims and artifacts.

## ETHICS STATEMENT

This work does not raise ethical concerns.

## REPRODUCIBILITY STATEMENT

Section 4 specifies the architecture, training objectives, and both rollout paths. Appendix A details the numerical solvers, initial-state and action sampling, observation intervals, and preprocessing. It also defines the IC-grouped data splits and the action-family holdout, including the exclusion of the held-out family from normalization statistics. We report the training budgets, evaluation metric, and checkpoint-selection rules. Training uses AdamW with weight decay $10^{-3}$, five warmup epochs, cosine learning-rate decay, and gradient-norm clipping at one. We use seed 42 and BF16 training, with regularizer statistics computed in float32. FluidJEPA checkpoints are selected by validation latent-rollout error, whereas operator checkpoints are selected by validation field-rollout error. The objective-deletion study evaluates epoch 200 and reports early numerical termination without substituting earlier checkpoints. Per-run configurations, checkpoints, and evaluation records are retained for traceability. The reported results use one training seed and do not quantify variability across independent training runs.

## REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499. URL https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. doi: 10.48550/arXiv.2511.08544. URL https://arxiv.org/abs/2511.08544v3.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2105.04906.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. doi: 10.48550/arXiv.2404.08471. URL https://arxiv.org/abs/2404.08471.

Luke Bhan, Yuexin Bian, Miroslav Krstic, and Yuanyuan Shi. PDE Control Gym: A benchmark for data-driven boundary control of partial differential equations. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pp. 1083–1095. PMLR, 2024. URL https://proceedings.mlr.press/v242/bhan24a.html.

Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214. URL https://doi.org/10.1146/annurev-fluid-010719-060214.

Jonathan Gallagher and Roberto Guglielmi. Toward goal-agnostic joint-embedding predictive control of partial differential equations. *arXiv preprint arXiv:2607.21644*, 2026. doi: 10.48550/arXiv.2607.21644. URL https://arxiv.org/abs/2607.21644v2.

Francisco Giral, Abhijeet Vishwasrao, Andrea Arroyo Ramo, Mahmoud Golestanian, Federica Tonti, Adrian Lozano-Duran, Steven L. Brunton, Sergio Hoyas, Hector Gomez, Soledad Le Clainche, and Ricardo Vinuesa. AeroJEPA: Learning semantic latent representations for scalable 3D aerodynamic field modeling. *arXiv preprint arXiv:2605.05586*, 2026. doi: 10.48550/arXiv.2605.05586. URL https://arxiv.org/abs/2605.05586v1.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. doi: 10.48550/arXiv.1803.10122. URL https://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Christian Lagemann, Sajeda Mokbel, Miro Gondrum, Mario Rüttgers, Yuning Wang, Pol Suárez, Ludger Paehler, Deniz A. Bezgin, Aaron B. Buhendwa, Jared L. Callaham, Samuel Ahnert, Nicholas Zolman, Xiao Shao, Jean-Christophe Loiseau, Nikolaus A. Adams, Matthias Meinke, Wolfgang Schröder, Kai Lagemann, Esther Lagemann, Ricardo Vinuesa, and Steven L. Brunton. The HydroGym reinforcement learning platform for fluid dynamics. *Nature*, 657:369–376, 2026. doi: 10.1038/s41586-026-10917-6. URL https://www.nature.com/articles/s41586-026-10917-6.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

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*, 2026. doi: 10.48550/arXiv.2603.19312. URL https://arxiv.org/abs/2603.19312v3.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 2019. doi: 10.1017/jfm.2019.62. URL https://doi.org/10.1017/jfm.2019.62.

Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=j3oQF9coJd.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 2020. URL https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

Ricardo Vinuesa and Steven L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022. doi: 10.1038/s43588-022-00264-7. URL https://doi.org/10.1038/s43588-022-00264-7.

Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0163. URL https://papers.neurips.cc/paper_files/paper/2022/hash/0f817dcbad81afb21fb695f1b2e55e44-Abstract-Conference.html.

Xiangyuan Zhang, Weichao Mao, Saviz Mowlavi, Mouhacine Benosman, and Tamer Başar. ControlGym: Large-scale control environments for benchmarking reinforcement learning algorithms. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pp. 181–196. PMLR, 2024. URL https://proceedings.mlr.press/v242/zhang24b.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

# A CONTROLLED FLUID BENCHMARK: CONSTRUCTION AND PROTOCOL

This section specifies the numerical data and control interfaces summarized in Table 1. The benchmark is an offline prediction resource: a model receives prescribed commands and predicts their consequences. Neither the data-generation procedure nor the action-family holdout evaluates an optimized control policy.

## A.1 OBSERVATION TIMES AND NUMERICAL DISCRETIZATION

We distinguish three time scales: the solver step $h$, the command-hold interval $\delta t$, and the model observation interval $\Delta t = S\delta t$. The index $t \in \{0, \dots, 20\}$ labels observations; the corresponding elapsed physical time is $t\Delta t$. Each learning sequence therefore contains 21 fields and 20 prediction intervals, regardless of the number of internal solver steps. All times below use the corresponding environment's nondimensional convention, not seconds or a common physical scale across environments.

Table 5: **Simulation and recording settings.** Raw-frame spacing equals the command-hold interval $\delta t$. The stride selects observations for learning, not commands: all commands between selected frames are retained. Pre-roll time is additional to the branch duration $T$ in Table 1.

| Environment | Solver grid/mesh | $h$ | $\delta t$ | Raw frames | Stride | Pre-roll |
|---|---|---|---|---|---|---|
| NS | $256 \times 256$ | $10^{-4}$ | 1 | 21 | 1 | None |
| Cylinder | Medium FEM mesh | 0.01 | 0.2 | 101 | 5 | $[20, 40]$ |
| Pinball | Fine FEM mesh | 0.01 | 0.1 | 101 | 5 | $[10, 20]$ |
| Cavity | Medium FEM mesh | 0.0025 | 0.025 | 101 | 5 | $[2.5, 5]$ |

NS uses the spectral time-stepping implementation derived from U-NO (Rahman et al., 2023), retaining its nonlinear convention. It evolves vorticity on the periodic unit square with viscosity $\nu = 10^{-5}$. Before recording a $64 \times 64$ observation, we apply a separable periodic spectral low-pass filter on the $256 \times 256$ field and decimate by four. The filter preserves coordinate-wise wavenumbers up to 24, tapers with a raised cosine between 24 and 32, and removes modes at and beyond the output Nyquist frequency 32. Downsampling changes the observation, not the integration grid.

Cylinder, Pinball, and Cavity use HydroGym's Firedrake environments (Lagemann et al., 2026), with Reynolds numbers 100, 30, and 7,500. We use the released medium, fine, and medium meshes, respectively, and a third-order semi-implicit BDF integrator. Each branch records 101 fields. The learning interface selects raw indices $0, 5, \dots, 100$, yielding $\Delta t = 1$, 0.5, and 0.125. The command block between adjacent selected fields retains the five ordered commands; it is neither subsampled nor averaged.

The four channels are $(\omega, u, v, p)$: vorticity, two velocity components, and pressure. They are sampled from the finite-element solution onto a $64 \times 128$ observation grid. The regions of interest are $[-1, 12] \times [-3.25, 3.25]$ for Cylinder and Pinball, and $[-0.5, 2.5] \times [-1, 0.5]$ for Cavity. Points outside the fluid domain are masked. This point sampling is not a conservative remapping and does not make the observation grid the solver mesh.

## A.2 INTERNAL FORCING AND BOUNDARY ACTUATORS

**NS: an additional vorticity source.** The four-dimensional action controls fixed Fourier modes:

$$f_{\mathrm{ctrl},t}(\mathbf{r}) = 0.1 \sum_{j=1}^{4} a_{t,j}\phi_j(\mathbf{r}), \qquad |a_{t,j}| \le 0.5,$$
$$\boldsymbol{\phi}(\mathbf{r}) = \big(\sin 4\pi r_1, \cos 4\pi r_1, \sin 4\pi r_2, \cos 4\pi r_2\big)^\top. \tag{8}$$

Here $\mathbf{r} = (r_1, r_2)$ denotes position on the unit square. The additional source enters the vorticity equation directly; its coefficients are not velocity boundary values or momentum-force components. It is added to the fixed background source $f_{\mathrm{bg}}(\mathbf{r}) = 0.1[\sin(2\pi(r_1 + r_2)) + \cos(2\pi(r_1 + r_2))]$. The spatial modes and source gain are fixed across all branches and initial conditions. Commands are held constant over each interval of length one.

**HydroGym: velocity boundary conditions.** Cylinder uses one scalar command for two localized radial wall jets, centered at the top and bottom of the cylinder, each with angular width $10^\circ$. The implemented profiles have the same radial sign for a given command; this is not a zero-net-flux blowing/suction pair. Pinball uses three commands for the tangential velocities of three cylinder walls. Cavity uses one command for localized blowing or suction near the leading edge of an open cavity. It is not a moving-lid cavity. These commands modulate the environments' prescribed velocity boundary profiles, rather than adding a spatially uniform volume force.

**Commands and actuator memory.** The applied boundary amplitude $\boldsymbol{c}$ is filtered and need not equal the command $\boldsymbol{a}$. For a piecewise-constant command, the filter and its exact update are

$$\tau \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\eta} = \boldsymbol{a} - \boldsymbol{c}, \qquad \boldsymbol{c}_{t,r+1} = \boldsymbol{a}_{t,r} + (\boldsymbol{c}_{t,r} - \boldsymbol{a}_{t,r}) \exp(-\delta t/\tau), \tag{9}$$

where $\eta$ is continuous physical time and $r = 0, \ldots, S-1$ indexes commands within an observation interval. We set $\boldsymbol{c}_{t,0} = \boldsymbol{c}_t$ and $\boldsymbol{c}_{t+1} = \boldsymbol{c}_{t,S}$. The time constants are 0.0556, 0.05, and 0.075 for Cylinder, Pinball, and Cavity. The initial actuator state $\boldsymbol{c}_0$ is available with the initial observation. Subsequent actuator states are computed from it and the supplied commands using Eq. (9); no future field or actuator measurement is needed. The model condition is $\boldsymbol{q}_t = \boldsymbol{a}_t \in \mathbb{R}^4$ for NS and $\boldsymbol{q}_t = [\mathrm{vec}(\boldsymbol{A}_t); \boldsymbol{c}_t]/s_a$ for the boundary-controlled environments. Here $s_a$ is the fixed command scale, equal to 0.05, 1, and 0.08, respectively. Thus the latter condition dimensions are 6, 18, and 6, although the raw command dimensions are 1, 3, and 1.

### A.3 Control families and per-group resampling

Table 6: **Branch identities.** Branch indices agree with the Action 0–4 columns in Figure 2. Branch 4 is withheld in the action-family OOD protocol; all five branches are used in the standard protocol.

| Branch | NS | Cylinder / Pinball / Cavity |
|---|---|---|
| 0 | Zero additional command | Zero command |
| 1 | Smooth AR(1) | Constant positive |
| 2 | Traveling wave | Constant negative |
| 3 | Three sign reversals | One sign reversal |
| 4 | Biased multisine | Sinusoidal |

Each initial-state group receives new numerical sequences, not a copy of one common action trace. Sampling is deterministic given the dataset seed, environment, initial-state ID, and family-specific random stream. The mechanism stays fixed across groups; its sampled parameters change.

**NS profiles.** The smooth AR(1) family samples a coefficient bound $A \sim \mathcal{U}(0.35, 0.5)$ and correlation $\rho \sim \mathcal{U}(0.85, 0.95)$. Each initial coefficient is uniform on $[-A/2, A/2]$. The recurrence uses independent Gaussian innovations with standard deviation $0.55A\sqrt{1-\rho^2}$ and clips each update to $[-A, A]$; after clipping, it is not an exact Gaussian AR(1) process. Traveling waves use sine/cosine pairs in Eq. (8), with two amplitudes drawn from $[0.25, 0.5]$, two phases from $[0, 2\pi)$, independently sampled propagation signs, and a shared period drawn from $[14, 28]$. Phases are sampled at command times and held between updates. The reversal family samples a four-dimensional base vector with independently signed magnitudes in $[0.15, 0.5]$. It reverses the entire vector at three integer command indices drawn from $\{4, 5, 6\}$, $\{9, 10, 11\}$, and $\{14, 15, 16\}$.

Biased multisine draws a signed bias magnitude in $[0.25, 0.38]$ and a modulation amplitude in $[0.08, 0.12]$ for each coefficient. It adds two sinusoids with mixture weights 0.65 and 0.35, periods drawn from $[12, 24]$ and $[24, 40]$, and independent uniform phases. These choices preserve the bias sign and keep the coefficients within $[-0.5, 0.5]$. The bias is temporal: the Fourier source still has zero spatial mean. The four nonzero NS families use separate random streams.

**Boundary-control profiles.** The amplitude ranges are $[0.025, 0.05]$ for Cylinder, $[0.5, 1]$ for Pinball, and $[0.04, 0.08]$ for Cavity. For each group, the constant pair shares one uniformly sampled amplitude $A$: its commands are $A\mathbf{d}$ and $-A\mathbf{d}$. Here $\mathbf{d} = 1$ for Cylinder and Cavity, and $\mathbf{d} = (0, 1, -1)^\top$ for Pinball. The pair is therefore coupled, not independently sampled. The reversal branch samples

its own amplitude and switches from $A\mathbf{d}$ to $-A\mathbf{d}$ at a uniformly sampled fraction in $[0.35, 0.65]$ of the trajectory, rounded to a command boundary.

The sinusoidal branch samples its amplitude separately, its period uniformly from $[0.22T, 0.40T]$, and its phase uniformly from $[0, 2\pi)$. Cylinder and Cavity use this scalar sinusoid. Pinball uses the same amplitude and period across its three commands with relative phases $0$, $2\pi/3$, and $4\pi/3$. These are open-loop sequences, sampled at the command interval and held until the next update.

### A.4 Initial states, paired branches, and the zero reference

NS samples 400 initial vorticity fields with the UNO-derived Gaussian random-field generator, using spectral parameters $\alpha = 2.5$ and $\tau_{\mathrm{GRF}} = 7$. All five branches of a group share the same high-resolution initial field. For each HydroGym environment, groups are constructed by separate random-control pre-rolls from one released reference flow. The pre-roll duration is sampled uniformly from the range in Table 5 and rounded up to a solver step. Pre-roll commands are generated from an AR(1) process with correlation 0.9, mapped through $\tanh$, and scaled by a sampled amplitude from that environment's range. Thus the groups vary through independently sampled pre-rolls, but they do not represent different geometries or independent source simulations.

At each branching point, we save the flow variables, BDF integration history, solver time, and actuator state. Every branch restores this same numerical state before its commands are applied. Generation checks verify matching initial observations and actuator states across branches, finite fields, and consistency of the actuator update. The saved restart is also checked by advancing a step and comparing it with a restored continuation. These checks establish the intended paired numerical protocol, not a mesh-convergence guarantee.

Zero means zero additional command throughout the branch. NS retains its background source $f_{\mathrm{bg}}$. For the boundary-controlled environments, a nonzero $\boldsymbol{c}_0$ inherited from the pre-roll decays under zero command according to Eq. (9). The zero branch is therefore not necessarily instantaneously unactuated. The paired response $\Delta\boldsymbol{x}_t^{i,b} = \boldsymbol{x}_t^{i,b} - \boldsymbol{x}_t^{i,0}$ compares branches at the same elapsed physical time and with the same initial numerical state.

### A.5 Splits, preprocessing, and action-family holdout

Each environment has 400 initial-state groups and 2,000 trajectories. The group-disjoint split assigns 320, 40, and 40 groups to training, validation, and test. The standard protocol uses all five branches, giving 1,600, 200, and 200 trajectories. The OOD protocol uses branches 0–3 only for training, validation, normalization, and checkpoint selection, giving 1,280 training and 160 validation trajectories. Its test set contains 160 seen-family and 40 held-out-family trajectories from the same 40 unseen initial states. The fifth branches of training and validation groups are not moved into the test set.

Normalization uses per-channel statistics from the training observations and valid fluid pixels only. For OOD experiments, these statistics are recomputed without branch 4. Invalid pixels are zeroed after normalization and excluded from physical-field losses and evaluation. The geometry mask is known and fixed within an environment.

The holdout changes the action-generation family, while geometry, flow parameters, actuator locations, and spatial control profiles remain fixed. It does not introduce a new physical actuator or equation. Pinball requires an additional qualification: its first four command families lie along $\mathbf{d} = (0, 1, -1)^\top$, whereas the sinusoidal family changes coordination and actuates the first cylinder as well. The initial actuator state can still reflect the multidimensional pre-roll. Consequently, this holdout tests both a new temporal profile and new command coordination, not waveform generalization alone. All reported predictions remain open-loop rollouts without test-time weight updates or feedback from future true fields.

### A.6 Reading the benchmark figure and numerical scope

Figure 2 shows simulated vorticity, not model predictions. Its initial column is $\boldsymbol{x}_0$; Action 0–4 correspond to Table 6 and show $\boldsymbol{x}_{20}$. The terminal label $t = 20$ denotes the prediction index, not a shared physical end time. The respective elapsed times are 20, 20, 10, and 2.5 for NS, Cylinder,

Pinball, and Cavity. For HydroGym, the terminal field is raw frame index 100; it is frame 21 of the selected learning sequence. Grey regions mark solid or out-of-domain pixels. Colorbars are specific to each environment, so equal colors across different rows need not represent equal vorticity.

The observation maps discard information through filtering or cropping; we do not assume that a single observed field is a closed Markov state. The benchmark also does not establish mesh or time-step convergence. In particular, the Cavity corpus uses $h = 0.0025$, inherited from the generation protocol, rather than the environment class default of $10^{-4}$. The pre-rolls start from released reference flows whose earlier integration histories are not reconstructed. Our paired restart procedure preserves the histories accumulated during each pre-roll, but it does not remove this provenance limitation. These datasets support comparisons under fixed numerical protocols, not claims of resolution-independent physical accuracy or arbitrary-control coverage.

# B Experimental Protocol and Implementation

This appendix makes the comparisons in Section 5 explicit: the models share data and control interfaces, but their feedback paths and checkpoint-selection criteria differ. We first describe the model-facing data, then the optimization and evaluation procedures.

## B.1 Tensor interfaces and valid-domain handling

Each learning sequence contains $L + 1 = 21$ physical fields and $L = 20$ control intervals. Table 7 gives the per-step model interface, omitting the batch dimension. For the boundary-controlled cases, one observation interval contains five ordered commands, and the condition also includes the actuator state at its start. The filter update in Eq. (9) advances this state using known commands only. These are model-facing shapes, not a claim about the axis order of every on-disk array.

Table 7: **Model-facing tensors per prediction interval.** Shapes omit the batch dimension. $\boldsymbol{A}_t$ contains the ordered commands between observations; $\boldsymbol{q}_t$ is the predictor condition after including actuator memory when applicable. The recurrent token grid is unchanged by prediction.

| Environment | Field $\boldsymbol{x}_t$ | Commands $\boldsymbol{A}_t$ | $\dim(\boldsymbol{q}_t)$ | Tokens $\boldsymbol{z}_t$ |
|---|---|---|---|---|
| NS | $1 \times 64 \times 64$ | $1 \times 4$ | 4 | $256 \times 192$ |
| Cylinder | $4 \times 64 \times 128$ | $5 \times 1$ | 6 | $512 \times 192$ |
| Pinball | $4 \times 64 \times 128$ | $5 \times 3$ | 18 | $512 \times 192$ |
| Cavity | $4 \times 64 \times 128$ | $5 \times 1$ | 6 | $512 \times 192$ |

Physical channels are normalized independently using training-set fluid pixels. The field tensor uses channel order $(\omega, u, v, p)$ in the boundary-controlled cases and only $\omega$ in NS. Both the loss and the metric exclude invalid pixels. A patch containing any valid fluid pixel remains an attention token, even if it also intersects a solid boundary. Its invalid pixels are not supervised. Purely invalid patches are excluded as attention keys, zeroed after each block, and omitted from JEPA matching and regularizer sampling. Thus, patch validity and pixel validity have different roles; retaining a mixed patch does not treat its solid pixels as flow observations.

## B.2 Optimization and checkpoint selection

All datasets use the IC-grouped 320/40/40 split described in Appendix A. Training statistics are fitted on training fields only, independently for each physical channel and excluding invalid pixels. NS uses all 21 stored fields; the other environments use raw indices $0, 5, \ldots, 100$. Every five-command block between retained fields remains ordered, and actuator memory is advanced with the known filter rather than with future measurements.

K1 training uses AdamW for 200 epochs, batch size 16, initial learning rate $3 \times 10^{-4}$, weight decay $10^{-3}$, five warmup epochs, and cosine decay to zero. The gradient norm is clipped at one. Training uses BF16 with float32 regularizer statistics; the reported evaluations use the same H100/BF16 backend. The model has patch size four, token width 192, six encoder blocks, six predictor blocks, and four decoder blocks. The field and JEPA coefficients are one, and SIGReg has coefficient 0.09.

Table 8: Validation-selected checkpoint epochs. K2 epochs are local to its additional 100-epoch stage; K1 epoch 200 supplies its initialization.

| Environment | K1 | FNO | U-NO | K2 | OOD |
|---|---|---|---|---|---|
| NS | 185 | 200 | 200 | 50 | 170 |
| Cylinder | 195 | 200 | 200 | 65 | 145 |
| Pinball | 190 | 200 | 200 | 100 | 200 |
| Cavity | 195 | 100 | 75 | 50 | 165 |

Each environment is trained separately with seed 42. The experiments do not provide repeated-training confidence intervals. The Transformer blocks use six attention heads, an MLP expansion ratio of four, and zero dropout. The objectives and SIGReg sampling settings are those in Section 4.2: at most 128 valid token rows, 1,024 projection directions, and 17 quadrature knots. All stages train end to end; there is no separate autoencoder pretraining or frozen target encoder.

Full, K2, OOD, and regularizer-replacement models select the checkpoint with lowest validation latent-rollout error. FNO and U-NO select by validation field-rollout error, their available feedback path. Thus, the operator comparison shares data and training budgets, but not a common model-selection criterion. The objective-deletion study instead compares the common epoch 200, exposing late degradation rather than replacing a degraded final model by an earlier checkpoint. Runs that terminate before epoch 200 have no final score. Table 8 records the selected checkpoint epochs.

Two-step fine-tuning starts from K1's *last* epoch-200 weights, not its validation-selected weights. It runs for 100 additional epochs at initial learning rate $3 \times 10^{-5}$ with a new optimizer and schedule. No intermediate true field is fed back. The field and JEPA terms supervise step 2 only; SIGReg samples true encodings at steps 0 and 2. Gradients pass through both predictor calls. The K1 and K2 scores in Table 15 each use the best checkpoint selected within their own stage, so the K1 comparison score need not describe the exact initialization used by K2.

### B.3 Physical-field metrics and uncertainty

Let $M_f$ be the number of valid pixels and $\sigma_c$ the training standard deviation of physical channel $c$. After inverse normalization, the per-field score is

$$e_{i,t,c} = \frac{\|\boldsymbol{m} \odot (\widehat{\boldsymbol{x}}_{i,t,c} - \boldsymbol{x}_{i,t,c})\|_2}{\max(\|\boldsymbol{m} \odot \boldsymbol{x}_{i,t,c}\|_2, \varepsilon_c)},$$
$$\varepsilon_c = \begin{cases} 10^{-8}, & \text{NS}, \\ \max(10^{-3}\sigma_c\sqrt{M_f}, 10^{-8}), & \text{boundary-controlled flows}. \end{cases} \tag{10}$$

For $N_{\text{test}} = 200$ and $L = 20$, the primary and terminal errors are

$$\bar{e} = \frac{1}{N_{\text{test}} LC} \sum_{i,t,c} e_{i,t,c}, \qquad e_{\text{final}} = \frac{1}{N_{\text{test}} C} \sum_{i,c} e_{i,L,c}. \tag{11}$$

This is an average of unsquared relative spatial norms, not pixelwise MSE, a pooled ratio, or a latent-space distance. Each channel has equal weight. The initial observed field is excluded. One-step evaluation supplies the true preceding field at each of the 20 transitions; rollout evaluation supplies only the initial observation. For an OOD subgroup, the same reduction is applied after selecting the corresponding 160 or 40 trajectories. We never normalize an OOD error by a five-family model evaluated on a different trajectory set.

The five branches of one initial condition are dependent. For test-case uncertainty, we first average within each of the 40 IC groups, then resample 40 groups with replacement 2,000 times using seed 42. Intervals use the 2.5th and 97.5th percentiles. Between-model comparisons use identical resampled groups. These intervals quantify test-case sampling conditional on the trained weights, not variability across optimizer seeds. The Cylinder interval includes zero; its point estimate alone does not establish a reliable advantage over FNO across test ICs. The positive Pinball difference favors FNO in this comparison.

Table 9: **Paired test-case uncertainty.** Difference in field-rollout mean error, FluidJEPA minus FNO. Intervals resample 40 complete IC groups 2,000 times, retaining all five branches together. They do not measure training-seed variability.

| Environment | Mean difference | 95% IC-bootstrap interval |
|---|---|---|
| NS | -0.00609 | [-0.00761, -0.00459] |
| Cylinder | -0.00090 | [-0.00369, 0.00094] |
| Pinball | 0.00149 | [0.00078, 0.00251] |
| Cavity | -0.03623 | [-0.04218, -0.03106] |

Table 10: Trainable parameters in millions. Each complex spectral coefficient counts as two real scalars. Baselines use approximately matched capacity, not exactly matched computation.

| Environment | FNO | U-NO | FluidJEPA |
|---|---|---|---|
| NS | 8.318 | 8.519 | 8.307 |
| Cylinder | 8.318 | 8.519 | 8.476 |
| Pinball | 8.319 | 8.520 | 8.478 |
| Cavity | 8.318 | 8.519 | 8.476 |

### B.4 Action-conditioned operator baselines

FNO uses four spectral blocks, width 60, and 12 modes per spatial axis. U-NO uses seven blocks with base width $d = 27$, widths $2d, 4d, 8d, 8d, 4d, 2d, d$, and mode counts $12, 6, 3, 3, 3, 6, 12$. Controls are broadcast over the grid and concatenated with the field, mask, and coordinates before lifting. The boundary-controlled environments provide the same ordered command block and current actuator state as FluidJEPA. NS uses periodic sine/cosine coordinates; the other environments use two linear coordinates and eight-pixel padding on every side, cropped from the output. Solid and exterior output pixels are masked. These are action-conditioned adaptations, not unchanged original benchmark configurations. Table 10 counts real-valued parameters, counting each complex coefficient twice. Parameter matching does not imply matched FLOPs, wall-clock cost, or memory.

### B.5 Visual and source-data conventions

The forecasting, fine-tuning, and ablation panels in the main text, together with the OOD response panel in Appendix D, use the supplied PNG assets without recoloring or rearranging their panels. The additional trajectory plates in Appendix C.5 retain the previously generated checkpoint-derived visualizations, one per environment. The fine-tuning curves, OOD responses, and ablation statistics correspond to the recorded evaluation protocols described here. The OOD examples use NS IC 6, Cylinder IC 117, Pinball IC 70, and Cavity IC 155, with the fifth control family. The Cavity ablation example uses IC 155 and that same family. These ICs were selected as the smallest IDs in the available complete preview groups, not by comparative error. The operator comparison in Figure 4 was supplied as a new inference result. Its checkpoint identifiers, trajectory identifiers, arrays, and display-scale metadata remain to be supplied and are not inferred from the earlier figure. Accordingly, no quantitative conclusion or table value is extracted from that image. The accompanying source manifest records this unresolved provenance separately from the verified numerical results. Numerical tables are linked to saved evaluation records in the accompanying provenance manifest. Training configurations and checkpoints are kept separate from visualization assets, so an individual illustration does not replace a complete-test-set evaluation.

## C Additional Prediction and Spectral Results

### C.1 Temporal, terminal, and channel-wise errors

Tables 11 and 12 supplement the main mean scores with terminal errors. Table 13 separates vorticity, velocity, and pressure instead of attributing the aggregate ranking to a single channel. The time curves in Figure 5 average physical channels, whereas the spatial panels in the main text isolate vorticity. This distinction prevents interpreting a vorticity example as a complete four-channel comparison.

Table 11: **Two feedback paths of the same K1 model.** Physical-field relative $L_2$, averaged over 20 transitions (mean) or evaluated at transition 20 (final). Each row uses one validation-selected checkpoint for both paths; lower is better.

| Environment | Field mean | Latent mean | Field final | Latent final |
|---|---|---|---|---|
| NS | 0.0375 | 0.0999 | 0.1238 | 0.3074 |
| Cylinder | 0.0068 | 0.0206 | 0.0073 | 0.0300 |
| Pinball | 0.0077 | 0.0246 | 0.0118 | 0.0497 |
| Cavity | 0.1059 | 0.1390 | 0.1592 | 0.2170 |

Table 12: **Terminal field-rollout error.** Same checkpoints and trajectories as Table 2; transition 20 only.

| Environment | FNO | U-NO | FluidJEPA |
|---|---|---|---|
| NS | 0.1565 | 0.2401 | 0.1238 |
| Cylinder | 0.0080 | 0.0096 | 0.0073 |
| Pinball | 0.0094 | 0.0094 | 0.0118 |
| Cavity | 0.2070 | 0.2498 | 0.1592 |

Table 13: **Channel-wise error for the full K1 model.** All 200 test trajectories and 20 transitions; the primary score averages these channel-wise relative errors with equal weights.

| Environment | Mode | $\omega$ | $u$ | $v$ | $p$ |
|---|---|---|---|---|---|
| Cylinder | Field | 0.0077 | 0.0017 | 0.0098 | 0.0080 |
| | Latent | 0.0252 | 0.0049 | 0.0311 | 0.0212 |
| Pinball | Field | 0.0062 | 0.0028 | 0.0093 | 0.0122 |
| | Latent | 0.0225 | 0.0103 | 0.0332 | 0.0324 |
| Cavity | Field | 0.2717 | 0.0092 | 0.0673 | 0.0754 |
| | Latent | 0.2894 | 0.0178 | 0.1044 | 0.1445 |

The channel-wise table also makes the aggregation rule visible: the mean score does not preferentially weight a channel because its dimensional magnitude is larger. The vorticity panels should therefore be read as spatial diagnostics rather than as explanations of every change in the aggregate score.

### C.2 NS SPECTRAL DIAGNOSTICS

The NS spectral diagnostics evaluate the resolved periodic $64 \times 64$ fields. We compute spectra separately on all 200 trajectories before averaging and retain complete nonzero radial shells. For the saved periodic grid, state and error enstrophy are

$$Z(\omega) = \frac{1}{2HW} \sum_{h,w} \omega_{h,w}^2, \qquad Z_{\mathrm{err}} = Z(\widehat{\omega} - \omega). \tag{12}$$

Table 14 reports terminal averages for the same validation-selected checkpoints as the forecasting comparison. At $T = 20$, mean enstrophy is 1.4920 for the simulation, 1.4906 for field feedback, and 1.5316 for latent recurrence. The corresponding error-field enstrophies are 0.0264 and 0.1573. The spectral content of a predicted state can therefore remain similar while the spatial error is much larger. The error spectrum transforms $\widehat{\omega} - \omega$ itself; it is not the difference of the two state spectra. These diagnostics concern the saved low-pass-filtered observations and establish neither exact conservation nor a recovered inertial range.

### C.3 TERMINAL-ONLY TWO-STEP FINE-TUNING

Two-step fine-tuning initializes from each $K{=}1$ model's epoch-200 weights and trains 100 further epochs at learning rate $3 \times 10^{-5}$. The predictor advances twice in latent space with gradients through both calls and no intermediate true-state input; field and JEPA losses supervise only the second prediction, and SIGReg uses true-state encodings at steps 0 and 2. Each stage contributes its own

Table 14: **Terminal NS state and error enstrophy.** Averages over the 200 test trajectories at $T = 20$, computed on the saved $64 \times 64$ observations. The simulated reference has mean enstrophy 1.4920. Similar state enstrophy does not imply a small error field.

| Model | Rollout | $Z(\widehat{\omega})$ | $Z(\widehat{\omega} - \omega)$ |
|---|---|---|---|
| FNO | Field | 1.4834 | 0.0417 |
| U-NO | Field | 1.4952 | 0.0991 |
| FluidJEPA | Field | 1.4906 | 0.0264 |
| FluidJEPA | Latent | 1.5316 | 0.1573 |

validation-selected checkpoint, so the $K{=}2$ row is not necessarily initialized from the exact $K{=}1$ checkpoint reported beside it. Table 15 gives the aggregate scores that accompany Figure 5 in the main text. Two-step fine-tuning improves latent recurrence but increases the field-feedback error in every aggregate comparison. This is compatible with exposing the predictor to its own outputs during fine-tuning; it is not a controlled causal test of recurrence length independent of the extra training and changed learning rate.

Table 15: **Two-step fine-tuning improves latent recurrence, not field feedback.** K2 adds 100 epochs from K1's epoch-200 weights, supervising only the second prediction. Each stage uses its own validation-selected checkpoint. Bold compares K1 and K2 within the same rollout path. The last column is the relative reduction in latent error.

| | Field rollout | | Latent rollout | | Latent |
|---|---|---|---|---|---|
| Environment | K1 | K2 | K1 | K2 | reduction |
| NS | **0.0375** | 0.0896 | 0.0999 | **0.0563** | 43.6% |
| Cylinder | **0.0068** | 0.0112 | 0.0206 | **0.0096** | 53.4% |
| Pinball | **0.0077** | 0.0101 | 0.0246 | **0.0133** | 45.8% |
| Cavity | **0.1059** | 0.1536 | 0.1390 | **0.1135** | 18.3% |

### C.4 Predicting the response to control

For a nonzero branch $b$, let $\Delta \boldsymbol{x}_t^{(b)} = \boldsymbol{x}_t^{(b)} - \boldsymbol{x}_t^{(0)}$ and define the predicted difference using the model's own two predicted branches. We compute their relative error jointly over 20 future times and valid pixels, then average over physical channels and the 160 nonzero/zero branch pairs. The denominator is the norm of the true *response*, not the norm of the true full field. Its floor is $10^{-8}$ on NS and $10^{-3}\sigma_c\sqrt{20M_f}$ otherwise. Table 16 reports this complementary diagnostic. It measures the predicted consequence of a supplied intervention, not the ability to choose an effective intervention. FluidJEPA has the lowest response error on NS, Cylinder, and Cavity, whereas FNO is better on Pinball. The response metric therefore preserves the important exception in the full-field comparison.

### C.5 Trajectory-level comparisons

Figures 7–10 show one test trajectory per environment under the full K1 model. Every plate compares the same initial condition and prescribed controls under field feedback and latent recurrence at prediction steps 5, 10, 15, and 20. The first three rows show vorticity; the last two show absolute vorticity error. State limits are shared across truth, both predictions, and the four displayed times within a plate; the two error rows likewise share one scale. For legibility, these limits use the 99th percentile of absolute displayed state values and the 99.5th percentile of displayed absolute errors on valid pixels. Colorbar extensions mark clipping, and grey regions are excluded from evaluation. The command inset plots the prescribed input, not the filtered actuator response. The error inset instead summarizes all 200 test trajectories, averages physical channels, and shows pointwise 95% IC-bootstrap intervals. It should not be interpreted as the error curve of the illustrated branch or as training-seed uncertainty. These examples use the five-family models and are not additional OOD evaluations, even though the illustrated action family is held out in the separate four-family experiment.

Table 16: **Error in the response to control.** Field-rollout relative space–time error of predicted branch-minus-zero fields, with 160 matched nonzero/zero pairs per environment. The denominator is the true response, not the total flow. These numbers are not directly comparable to Table 2.

| Model | NS | Cylinder | Pinball | Cavity |
|---|---|---|---|---|
| FNO | 0.1849 | 0.0264 | 0.0278 | 0.5629 |
| U-NO | 0.2885 | 0.0358 | 0.0423 | 0.6362 |
| FluidJEPA | 0.1567 | 0.0256 | 0.0348 | 0.4634 |

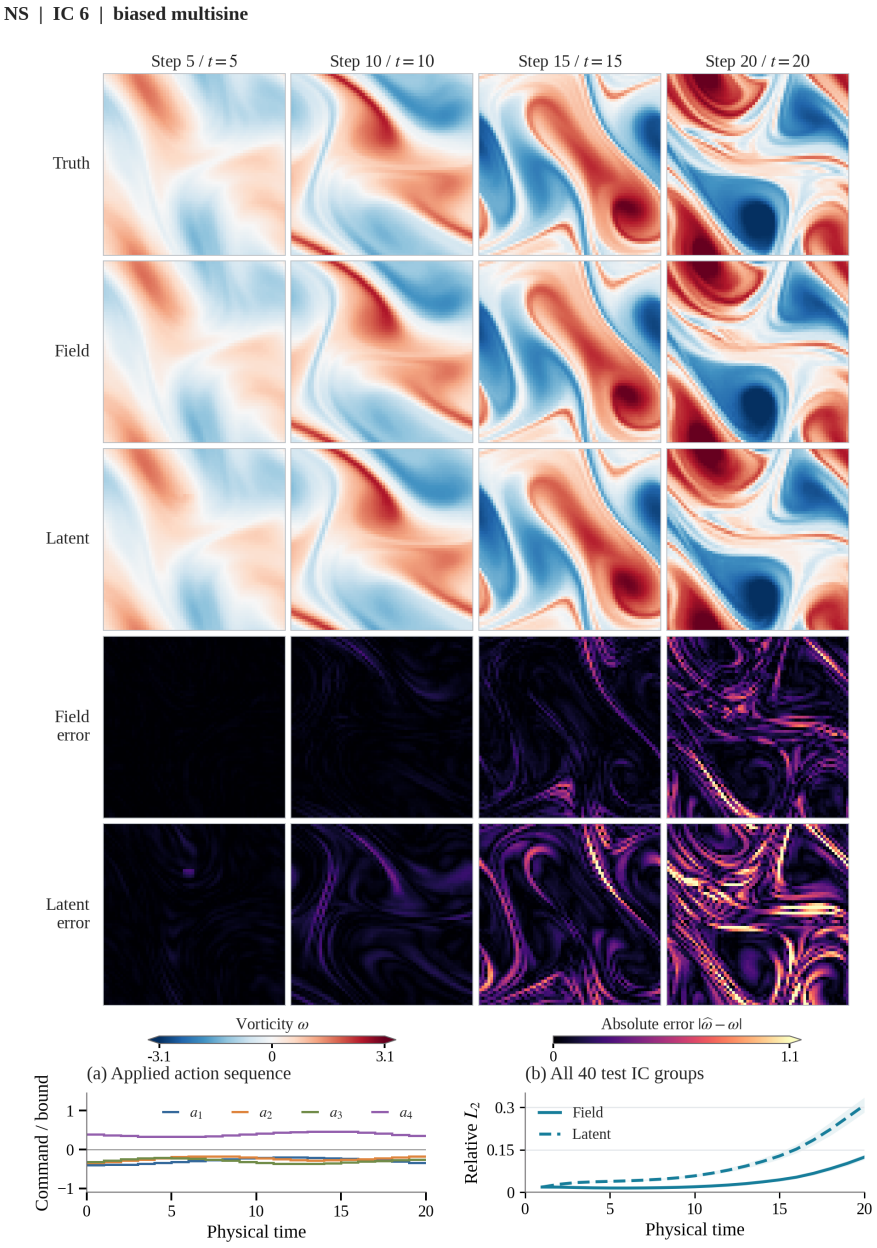



Figure 7: **NS: field and latent rollout from the same initial condition.** Test IC 6, biased-multisine branch, K1 checkpoint epoch 185. Columns show prediction steps 5, 10, 15, and 20 ($t = 5, 10, 15, 20$). Rows show true vorticity, field-rollout prediction, latent-rollout prediction, and their respective absolute errors. The lower panels give the prescribed four-component command and the complete-test-set error curves. Scales and confidence intervals follow Appendix C.5.

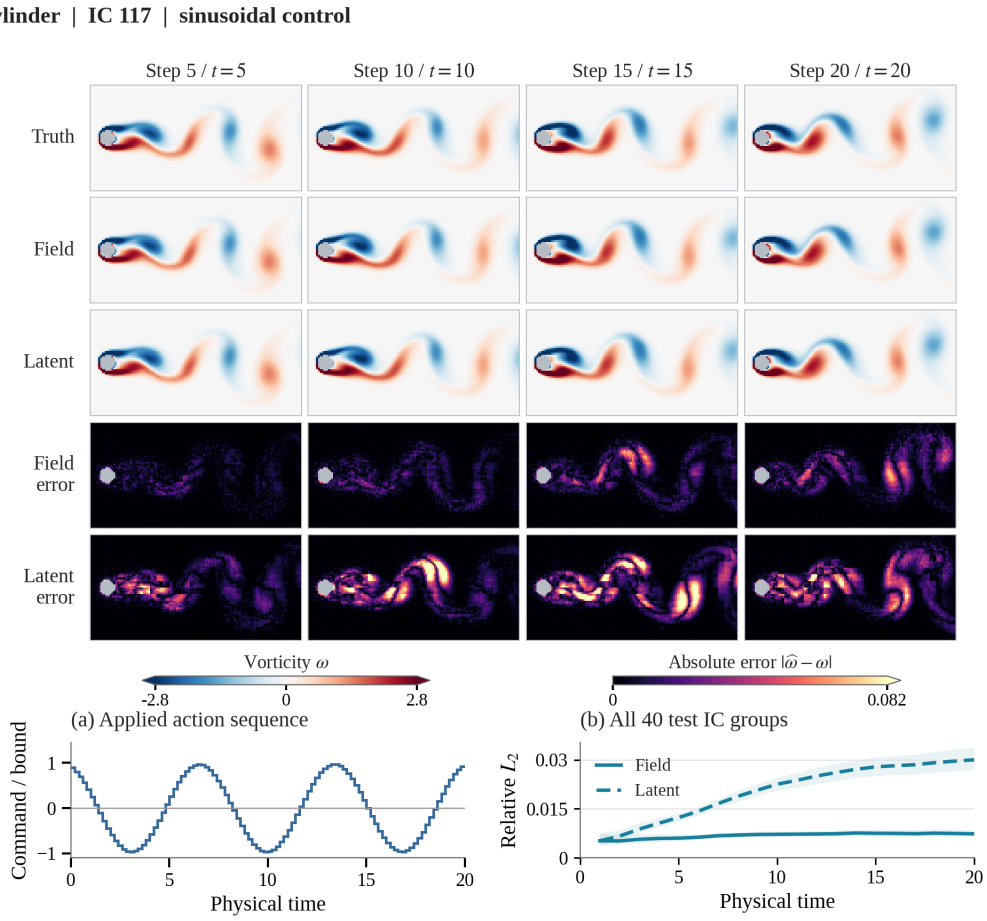



Figure 8: **Cylinder: field and latent rollout from the same initial condition.** Test IC 117, sinusoidal branch, K1 checkpoint epoch 195. Columns show prediction steps 5, 10, 15, and 20 ($t = 5, 10, 15, 20$). Rows follow the NS plate: truth, field prediction, latent prediction, and the two absolute vorticity errors. Grey pixels are outside the evaluated fluid domain. The command inset belongs to this trajectory; the error curves average all four physical channels and all 200 test trajectories, with pointwise IC-bootstrap intervals.

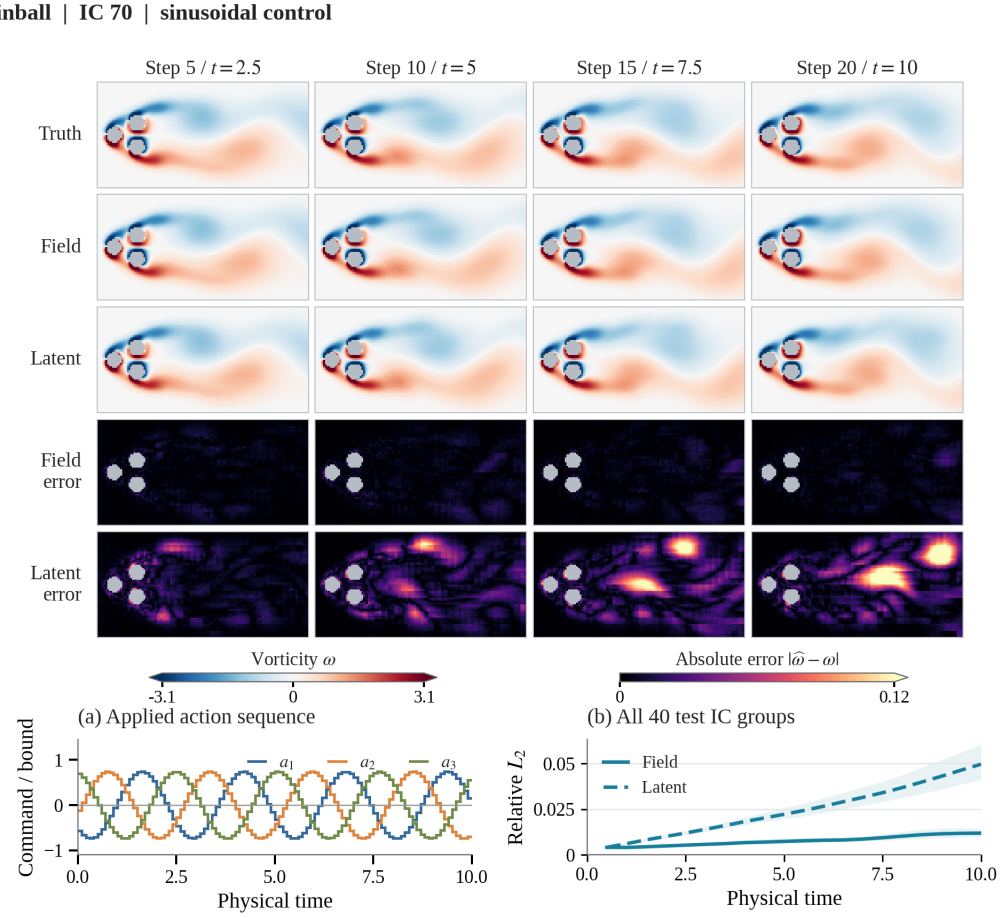



Figure 9: **Pinball: field and latent rollout from the same initial condition.** Test IC 70, phase-offset sinusoidal branch, K1 checkpoint epoch 190. Columns show prediction steps 5, 10, 15, and 20 ($t = 2.5, 5, 7.5, 10$). The three prescribed commands are shown separately below the vorticity and absolute-error rows. State and error scales are shared within their respective row groups. The lower-right curves summarize all four physical channels and all 200 test trajectories, not only the displayed sinusoidal branch.

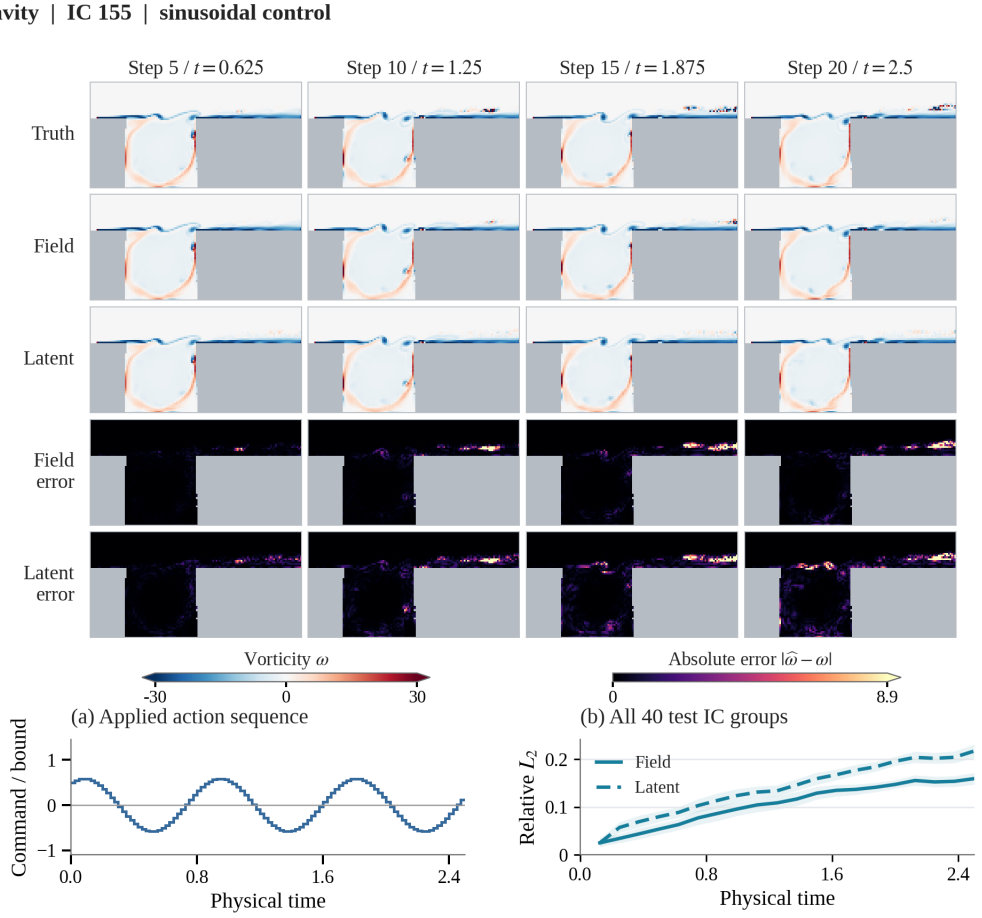



Figure 10: **Cavity: field and latent rollout from the same initial condition.** Test IC 155, sinusoidal branch, K1 checkpoint epoch 195. Columns show prediction steps 5, 10, 15, and 20 ($t = 0.625, 1.25, 1.875, 2.5$). The last two rows expose localized errors that are less apparent in the state images. Grey pixels are masked, and the command and complete-test-set error insets follow the same conventions as the other environments. The population curves average $\omega, u, v, p$ and are not vorticity-only scores.

# D UNSEEN ACTION FAMILIES: COMPLETE EVALUATION

## D.1 MATCHED SUBSETS AND ABSOLUTE ERRORS

The four-family models exclude family 4 from both training and validation, while preserving the same IC partition. The 160 seen-family test trajectories and 40 held-out-family trajectories all use test ICs unseen during training. The five-family reference is evaluated on the *same* 40 held-out-family trajectories. Table 17 separates these sets rather than comparing a held-out subset against a five-family average. There is no test-time optimization or adaptation.

Table 17: **Generalization to a withheld action family.** Four-family training is tested on 160 seen-family trajectories (seen-4) and 40 withheld-family trajectories (held-4). The five-family reference is evaluated on those same 40 trajectories (matched-5). Ratio is held-4/matched-5. Lower is better.

| Environment | Mode | Seen-4 | Held-4 | Matched-5 | Ratio |
|---|---|---|---|---|---|
| NS | Field | 0.0488 | 0.0766 | 0.0486 | 1.58× |
| | Latent | 0.0991 | 0.1325 | 0.1187 | 1.12× |
| Cylinder | Field | 0.0143 | 0.0297 | 0.0078 | 3.82× |
| | Latent | 0.0293 | 0.0457 | 0.0226 | 2.02× |
| Pinball | Field | 0.0087 | 0.0797 | 0.0078 | 10.24× |
| | Latent | 0.0299 | 0.0947 | 0.0241 | 3.93× |
| Cavity | Field | 0.1087 | 0.1957 | 0.1004 | 1.95× |
| | Latent | 0.1592 | 0.2233 | 0.1271 | 1.76× |

## D.2 WHAT CHANGES UNDER THE HOLDOUT

For NS the held-out family is biased multisine; the boundary-controlled cases hold out sinusoidal commands. The known actuator interface, geometry, and physical parameters remain unchanged. In Pinball the seen-family commands follow the actuator direction $(0, 1, -1)$, whereas the held-out family coordinates all three actuators with phase offsets $0, 2\pi/3, 4\pi/3$. This changes coordination as well as waveform and does not mean that the first actuator was never excited during IC preparation. Four-family training also uses 1,280 rather than 1,600 trajectories; keeping 200 epochs therefore reduces the number of optimizer updates. The comparison does not isolate temporal-family novelty from these other differences.

Figure 11 places the actual commands beside the terminal branch-minus-zero response. Each model subtracts its own predicted zero trajectory, not the simulated zero field. The examples explain spatial differences but do not replace the population scores. The largest shift occurs on Pinball; the result is evidence of a limitation rather than a general claim of robust transfer to arbitrary unseen forcing.

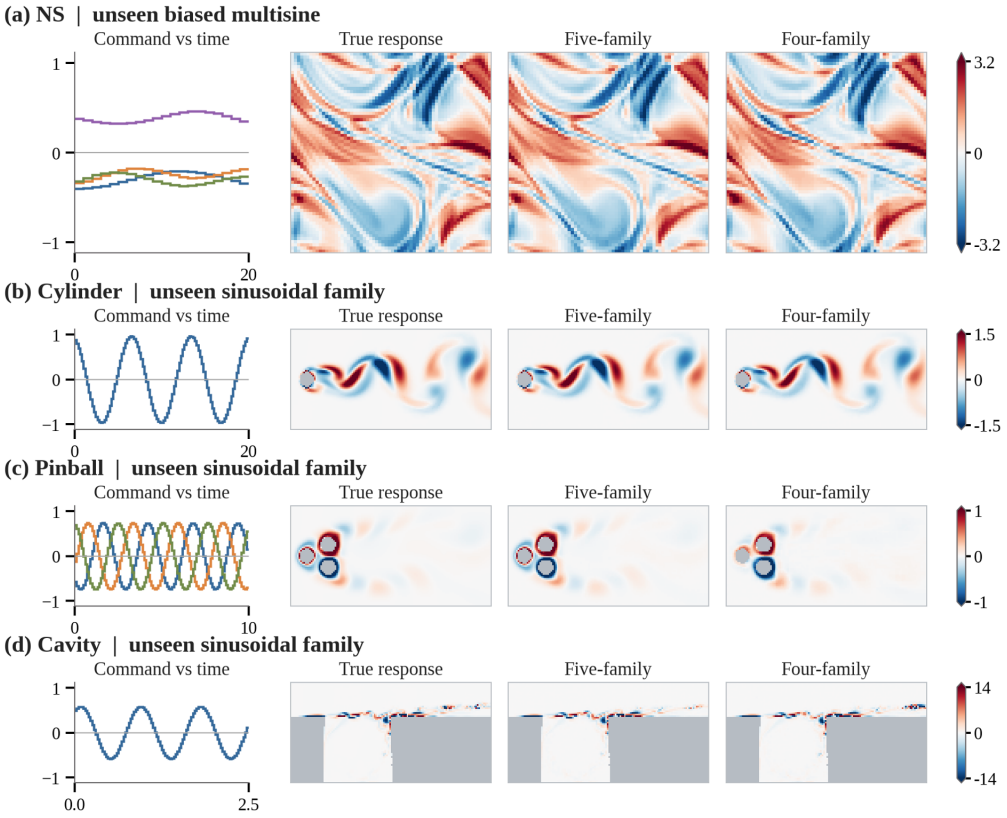



Figure 11: **Predicted response to a held-out action family.** Rows are NS, Cylinder, Pinball, and Cavity. Columns give the prescribed commands, the simulated terminal response, and the responses predicted by the five-family and four-family models under field rollout. A response is the controlled branch minus the zero branch at the same time, and each model subtracts its *own* predicted zero field. Five-family and four-family refer to the training sets, not the number of test branches. Commands are scaled by their configured bounds. Unlike these response images, Table 17 measures error in the full fields over all 20 predictions.

# E Objective Deletion, Stability, and Regularizer Replacements

## E.1 Fixed-epoch deletion protocol

w/o JEPA uses Field+SIGReg; w/o SIGReg uses Field+JEPA; Field-only removes both latent terms. All retain the same architecture and initial learning rate $3 \times 10^{-4}$. Table 18 extends the NS and Cylinder rows of Table 4 to all four environments and includes one-step scores to distinguish true-context prediction from self-fed rollout. The w/o SIGReg runs fail with nonfinite gradients after

Table 18: **Loss deletion at fixed epoch 200.** One-step evaluation supplies the true context at each transition; both rollout modes use only the initial observation. w/o SIGReg stopped after the indicated last complete epoch and has no epoch-200 score. All deletions retain the same architecture and learning rate.

| Environment | Objective | One-step | Field | Latent |
|---|---|---|---|---|
| NS | Full | 0.0179 | 0.0363 | 0.1039 |
| | w/o JEPA | 0.0170 | 0.0383 | 0.9340 |
| | Field-only | 0.3376 | 0.6848 | 0.8419 |
| | w/o SIGReg | — | — | — |
| Cylinder | Full | 0.0031 | 0.0066 | 0.0207 |
| | w/o JEPA | 0.0096 | 0.0192 | 0.6084 |
| | Field-only | 0.3016 | 0.5124 | 0.5843 |
| | w/o SIGReg | — | — | — |
| Pinball | Full | 0.0030 | 0.0073 | 0.0245 |
| | w/o JEPA | 0.0036 | 0.0082 | 0.6574 |
| | Field-only | 0.0040 | 0.0154 | 0.7449 |
| | w/o SIGReg | — | — | — |
| Cavity | Full | 0.0592 | 0.1061 | 0.1392 |
| | w/o JEPA | 0.0652 | 0.1080 | 0.5267 |
| | Field-only | 0.0533 | 0.1163 | 1.2041 |
| | w/o SIGReg | — | — | — |

complete epochs 63, 42, 36, and 29 on NS, Cylinder, Pinball, and Cavity. No score is extrapolated to epoch 200, and an earlier checkpoint is not silently substituted. Field-only finishes training, but its NS and Cylinder weights degrade late; this differs from numerical termination. Lower-learning-rate follow-up runs are not mixed into this fixed-protocol table.

## E.2 Variance–covariance alternatives

For sampled token rows $Z \in \mathbb{R}^{n \times d}$, the implemented penalties are

$$V(Z) = \frac{1}{d}\sum_{j=1}^{d} \max\left(0, 1 - \sqrt{\mathrm{Var}_{n-1}(Z_j) + 10^{-4}}\right),$$
$$C(Z) = \frac{1}{d}\sum_{j \neq k} \mathrm{Cov}_{n-1}(Z)_{jk}^2. \tag{13}$$

The observed-token replacement samples up to 128 valid rows from true encodings at times 0 and 1:

$$\mathcal{L}_{\mathrm{obsVC}} = \mathcal{L}_{\mathrm{field}} + \mathcal{L}_{\mathrm{JEPA}} + V(Z_{\mathrm{obs}}) + 0.04C(Z_{\mathrm{obs}}). \tag{14}$$

The predictive VICReg adaptation instead uses paired predicted/target rows at identical indices:

$$\mathcal{L}_{\mathrm{predVC}} = \mathcal{L}_{\mathrm{field}} + \mathcal{L}_{\mathrm{I}} + \tfrac{1}{2}[V(Z_{\mathrm{pred}}) + V(Z_{\mathrm{true}})] + 0.04[C(Z_{\mathrm{pred}}) + C(Z_{\mathrm{true}})]. \tag{15}$$

The alignment term $\mathcal{L}_{\mathrm{I}}$ is the paired latent MSE with the JEPA mask reduction; it replaces rather than duplicates that matching term. Variance penalties are averaged across branches, while covariance penalties are summed. Statistics use float32, the unbiased $n-1$ divisor, and at most 128 rows per branch, without SIGReg's additional factor $n$. No projector, stop-gradient target, EMA encoder, or

latent normalization is introduced. The coefficients are fixed rather than optimized by a regularizer-specific search. Since $n \leq 128$ and $d = 192$, centered empirical covariance has rank at most 127. These differences in statistical budget and reduction preclude treating the comparison as each regularizer's best achievable result.

Table 19: **Regularizer replacements on NS.** All runs train for 200 epochs and use the checkpoint with lowest validation latent-rollout error. These are not the fixed-epoch deletion scores. V+C acts on observed encodings; predictive VICReg acts on paired predicted/target encodings. Lower error is better.

| Objective | Selected epoch | Field | Latent |
|---|---|---|---|
| Full (SIGReg) | 185 | 0.0375 | 0.0999 |
| Field + JEPA + V+C | 185 | 0.0386 | 0.3127 |
| Field + VICReg | 200 | 0.0383 | 0.2678 |

The NS-only comparison in Table 19 uses each run's validation-selected checkpoint, unlike the fixed-epoch deletion table. The selected epochs are 185 for Full, 185 for Field+JEPA+V+C, and 200 for Field+VICReg. Both replacements retain similar NS field accuracy but worsen latent error. This is a comparison of the specified recipes on NS, not evidence that SIGReg is optimal across regularizers or environments.

### E.3 Interpreting the objective comparisons

The fixed-epoch and validation-selected tables answer different questions. The former tests what each objective produces after the common training budget, including late degradation; the latter evaluates alternative regularizers at the checkpoint chosen by the same validation criterion. Their Full rows consequently need not be numerically identical. Field-only's poor final NS and Cylinder scores do not imply that all of its earlier checkpoints are equally poor. Similarly, a nonfinite-gradient termination does not by itself identify representation collapse as its cause. The observed results support the value of predictive alignment and distribution regularization under the specified recipe, without proving that a particular regularizer is necessary under every optimization setting. These experiments evaluate supplied action sequences, not action selection, policy optimization, or closed-loop planning.