RETAIN KNOWLEDGE, ADAPT TO CHANGE: WHEN DOES FITTING AN ENVIRONMENT CODE HELP?

Anonymous authors
Paper under double-blind review

ABSTRACT

Fitting a small environment code can improve a frozen world model after conditions change. The same update can also improve prediction when the dynamics have not changed, because the initial code was estimated imperfectly. We study how to distinguish these effects. Our evaluation combines provably inert physical interventions with *matched code replay*: codes updated on unchanged and changed probe trajectories are both scored on the same held-out changed trajectories. This defines a signed, change-specific prediction gain without subtracting errors from different test distributions. In a controlled comparison of discrete-time CAVIA and CoDA implementations and a jointly trained context model, ten training seeds and three shift amplitudes expose judgments that aggregate gains hide. On the rolling ball, all three obtain positive change-specific gains; a 6.5-percentage-point total-gain advantage of the joint context model over CoDA becomes an unresolved -0.8-point difference in change-specific gain. On the pendulum, the joint context model's change-informed update is harmful even after control transfer is accounted for. Complementary experiments across nine systems show when code fitting helps prediction and test whether modular structure is needed. Dynamically equivalent interventions also show why counting changed physical parameters cannot by itself explain adaptation difficulty. The contribution is a controlled way to measure what real-change probe observations are worth to an update under a specified intervention, code parameterisation, loss and update schedule, alongside the prediction accuracy that matters at deployment.

1 INTRODUCTION

A world model must keep useful knowledge while responding to changing conditions. Friction can drop, a payload can change, or a contact can enter a different regime, and updating a compact environment representation while holding the shared dynamics fixed is an economical response. But an improvement after a change does not show that the update learned anything about that change: a code fitted from a short prefix may merely be poorly calibrated, and further observations improve it even when the environment stays exactly the same.

This ambiguity matters when evaluating adaptation methods. A method can obtain a large relative improvement because its initial code is weak, while another starts from an accurate predictor and has little room to improve. Conversely, an update may use changed observations productively but still lose overall because its ordinary recalibration is harmful. Prediction accuracy, total update gain, and the value of change-specific information answer different questions. An evaluation that reports only one of them can obscure the design choice it is meant to inform.

We make the distinction experimentally testable in two stages. First, a provably inert intervention changes a simulator parameter while leaving the observation process unchanged (Proposition 3, with the closed-form argument in Appendix C). Improvement under this intervention detects recalibration without recovery from changed dynamics. Second, we fit codes on paired inert and real-change probe branches and replay both code sequences on the *same* real-change query trajectories. Their loss difference measures the predictive value of updating from changed rather than unchanged observations. All predictions use causal prefixes, and probe and query trajectories are disjoint. Figure 1 shows the comparison.

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.

The shared-model/fitted-code design has established precedents in system identification (Yu et al., 2017; Wang & Johnson, 2021), context-conditioned dynamics (Lee et al., 2020; Kirchmeyer et al., 2022) and meta-learning (Zintgraf et al., 2019); we use that design to study a measurement problem. A useful code need not name a physical constant, and a changed parameter name need not imply changed dynamics, an old distinction (Laozi, 1963) with a concrete counterpart here, since both claims are judged through the predictions they permit.

These measurements change how familiar methods are assessed. On a rolling-ball shift the joint context model has the larger total gain over CoDA while the change-specific contribution does not resolve a winner, and on the pendulum all three implementations lose aggregate accuracy while the replay comparison separates nearly neutral from clearly harmful updates.

Our contributions are:

- **A controlled estimand for the value of changed observations.** Provable no-op interventions detect recalibration; matched code replay measures the extra prediction gain on a shared query distribution, with an exact signed decomposition and explicit interpretation.
- **A comparison across adaptation strategies.** Ten-seed experiments with CAVIA, CoDA and joint context learning show how total gains, change-specific gains and absolute errors lead to different judgments, including harmful updates hidden by a single aggregate.
- **Tests of what must change.** Complementary experiments across nine systems examine code fitting, update scope and model structure; matched physical interventions isolate the failure of raw parameter counts as a measure of dynamics change.

2 Measuring adaptation with matched controls

2.1 Shared predictor and deployment update

We factor a predictor into shared parameters θ and an environment code c:

$$\hat{o}_{t+1} = F_\theta(H_t, c), \qquad H_t = (o_0, \ldots, o_t). \tag{1}$$

The synthetic systems are autonomous, so actions are omitted in our experiments. Shared parameters are trained across physical environments and frozen at deployment. Only the code is fitted, using recent observed transitions D_t:

$$c_t = \mathrm{Update}_K\left(c_{t^-}, \quad \text{the code entering the update}, \frac{1}{|D_t|}\sum_{(H_i, o_{i+1})\in D_t} \|F_\theta(H_i, c) - o_{i+1}\|_2^2\right). \tag{2}$$

The update takes a fixed number of optimisation steps rather than solving to convergence. In the online comparisons, a supplied change time starts periodic updates; the model receives no physical parameter labels. A prediction is scored before the next update, and only observations already available at that time enter the fit. We evaluate prediction under this schedule, not an autonomous change detector or a learned control policy.

2.2 What a no-op control identifies

An inert intervention changes physical parameters without changing the observation law. For a solid sphere rolling without slip on an incline, acceleration is $a = \frac{5}{7} g \sin\vartheta$ for $I = \frac{2}{5} mR^2$, independently of the static-friction coefficient μ while the no-slip condition holds. (A solid cylinder has $I = \frac{1}{2} mR^2$ and therefore $a = \frac{2}{3} g \sin\vartheta$, which is why the two coefficients must not be conflated.) Shifting μ within that regime therefore preserves trajectories. The pendulum provides a second construction: $\dot{\omega} = -(g/L)\sin\varphi$, so scaling g and L by the same factor preserves the dynamics. The simulator also preserves its observation noise law. With paired initial conditions and noise, the inert and unchanged branches are numerically identical. Appendix C gives the physical proofs.

Any update benefit under an inert intervention is evidence of recalibration. This control alone does not determine the recalibration contribution on a real-change trajectory: the changed dynamics alter the states visited, the fitting data, and the baseline error. In particular, subtracting two within-arm

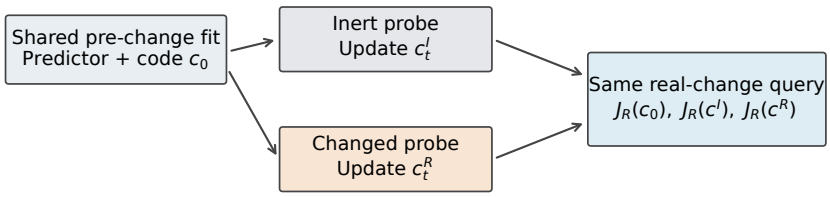

Figure 1: Matched code replay. Both branches start from the same pre-change code. Only the probe observations used for subsequent updates differ. The retained code, inert-branch code and real-branch code are scored on identical real-change query trajectories. Shared weights remain fixed, and query observations never enter code fitting.

improvement ratios uses different denominators and different query distributions. A same-system pair improves comparability, but does not remove these differences.

2.3 Matched code replay and its interpretation

We fork a simulator into an inert branch I and a real-change branch R, starting from matched initial conditions and random streams. Independent probe and query streams are forked in this way. Let c_0 be fitted on the shared pre-change probe prefix. The same update rule produces code sequences c_t^I and c_t^R from the respective probe branches. Write $J_R(c)$ for cumulative prediction error on the real-change query branch using code sequence c and the same observed query history. We record three losses and form

$$G = J_R(c_0) - J_R(c^R), \tag{3}$$

$$C = J_R(c_0) - J_R(c^I), \tag{4}$$

$$S = J_R(c^I) - J_R(c^R), \qquad G = C + S. \tag{5}$$

G is total update gain, C is the gain from transferring the inert-branch update, and S is *change-specific gain*: the predictive value of using changed rather than unchanged probe observations under the specified coupling and update rule. Dividing all three quantities by the same positive $J_R(c_0)$ preserves the identity. Positive values mean lower loss.

This decomposition is operational. It requires a simulatable matched branch, a common predictor and initial code, a fixed update schedule, and a common query trajectory. It does not assume that recalibration transfers unchanged across environments, because C is measured on the real query itself. Its terms are signed: a control update can be harmful, and S can exceed G. Consequently they are not generally percentages of a latent causal "adaptation component". We report absolute prediction error alongside these gains; a method with a larger relative gain need not be the more accurate predictor. Appendix A proves the identity and states the limits of its interpretation.

2.4 Statistical unit and physical comparability

For the new replay comparison, the independent replication unit is the training seed. Within each seed we average the predefined amplitudes and the paired probe/query replicates. We then compute means and 95% Student-t intervals across seeds; method contrasts pair the same seeds and test trajectories. Identical inert and unchanged trajectories count once in aggregate estimates. The amplitudes are fixed test conditions, not additional independent samples.

Raw parameter count is not a measure of effective dynamics change. For a pendulum, the one-parameter shift $(g, L) \mapsto (fg, L)$ and the two-parameter shift $(g, L) \mapsto (f^2 g, fL)$ give the same new ratio fg/L. We compare these at $f \in \{1.15, 1.30, 1.50\}$, while the inert pair (fg, fL) retains the original ratio. The rolling ball supplies an analogous comparison: a gravity shift with or without an additional inert friction shift. These controls vary the number of changed constants while holding the effective transition law fixed.

3 EXPERIMENTS

3.1 CONTROLLED COMPARISON OF ADAPTATION STRATEGIES

We instantiate three established adaptation strategies under one discrete state-prediction task. **CAVIA** concatenates a context vector to the state and meta-trains shared weights through five context-only inner gradient steps (Zintgraf et al., 2019). **CoDA** uses an affine hypernetwork $\theta(c) = \theta_0 + Wc$ with a low-rank basis and ℓ_2 regularisation (Kirchmeyer et al., 2022). **Joint context** uses CAVIA's architecture and jointly trains one code per environment. All three are documented discrete-time implementations, not reproductions of the original benchmark scores or continuous-time solvers.

All three receive the same 64 training trajectories of 100 frames, 16 validation trajectories, an eight-dimensional code and 60 epochs; CAVIA and joint context carry $10{,}562$ shared parameters against $10{,}962$ for CoDA, and checkpoints are selected by validation error without inspecting targets. Evaluation uses 50 Adam steps at rate 0.01, a ten-frame window, updates every five steps, two systems, three amplitudes and ten seeds, giving 480 paired physical settings. Training objectives, seed construction, checks and all absolute errors are in Appendix A.

The systems differ in what is observable: gravity sets the ball's constant acceleration, while the ratio g/L sets the pendulum's state-dependent restoring acceleration. Training samples pendulum $g \in [0.5, 2]$, $L \in [1, 3]$ and ball $g \in [8, 11]$, $\mu \in [0.1, 0.6]$. The pendulum's named test constants lie outside those ranges while its effective ratio does not, so we do not read the comparison as a uniform test of out-of-distribution dynamics.

Table 1: Matched replay gains, in percentage points of the same real-arm frozen-code loss ($G = C + S$; higher is better). Each cell averages amplitudes and probe/query pairs within seed, then ten seeds. The interval is for S. Negative values are retained.

System	Method	Total G	Control C	Specific S	95% CI for S
Pendulum	CAVIA	-1.9	-1.7	-0.2	$[-0.8, +0.3]$
Pendulum	CoDA	-26.3	-31.1	$+4.9$	$[-5.9, +15.6]$
Pendulum	Joint context	-27.1	-4.7	-22.3	$[-31.0, -13.7]$
Rolling ball	CAVIA	$+46.0$	$+2.6$	$+43.4$	$[+38.8, +48.1]$
Rolling ball	CoDA	$+42.2$	-6.5	$+48.7$	$[+42.7, +54.7]$
Rolling ball	Joint context	$+48.6$	$+0.7$	$+47.9$	$[+46.8, +49.0]$

The control changes the judgment of an update. On the rolling ball, all three methods have positive change-specific gains, with intervals excluding zero (Table 1). Joint context obtains 48.6% total gain and CoDA 42.2%. The paired CoDA-minus-joint difference is -6.5 points, with interval $[-11.0, -2.0]$. For change-specific gain it becomes $+0.8$ points, with interval $[-5.1, +6.6]$. Thus the total-gain advantage does not establish a stronger response to changed observations. The point-estimate order changes, but the change-specific comparison does not resolve a winner. Absolute adapted one-step MSE still favours joint context (3.58×10^{-4}) over CoDA (4.18×10^{-4}) and CAVIA (5.48×10^{-4}); the protocol retains this practical distinction.

The distinction is not created solely by normalising each method's gain. In absolute one-step MSE units, the paired CoDA-minus-joint total-gain difference is -6.20×10^{-5} with interval $[-10.63, -1.77] \times 10^{-5}$, whereas the change-specific difference is -0.63×10^{-5} with interval $[-6.04, 4.78] \times 10^{-5}$. The raw-loss comparison likewise changes from a resolved total-gain advantage to an unresolved change-specific contrast.

On the pendulum, all three methods lose accuracy after the scheduled update, but for different reasons. CAVIA's change-specific effect is close to zero, CoDA's mean effect is positive with a wide interval, and joint context has a negative effect of -22.3 points with interval $[-31.0, -13.7]$. For CoDA, the inert-code transfer costs 31.1 points while the changed-data contribution recovers 4.9 points on average. Calling this entire loss increase a failure to use changed observations would conceal the distinction. Figure 2 displays both terms, including their signs.

Longer prediction and equivalent interventions. We also evaluate five-step open-loop predictions, holding each code fixed during a rollout. On the rolling ball, adapted MSE is 0.00495 for

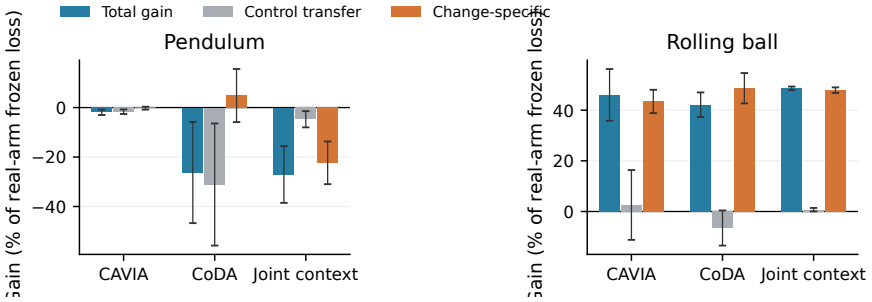

Figure 2: A total gain can contain opposing effects. Matched replay evaluates every code on the same real query, so the three bars obey $G = C + S$. Whiskers are 95% intervals across ten training seeds, after averaging repeated test conditions inside seed. Relative gain measures an update's effect on its own predictor; absolute prediction errors are reported separately.

joint context, 0.00672 for CoDA and 0.00914 for CAVIA; on the pendulum it is 0.00264, 0.00306 and 0.00402 for CAVIA, CoDA and joint context, respectively. These extend the prediction horizon without establishing a control-return benefit. Across all matched physical settings, inert branches reproduce no-change observations exactly, and one- versus two-constant interventions with equal effective dynamics agree within 10^{-6}. This demonstrates why parameter count alone cannot explain the observed differences.

3.2 Sensitivity to initial fitting and update budget

An adaptation gain is a property of an update and its starting point, not just of a trained predictor. We re-evaluate all 60 saved checkpoints with initial fitting budgets of 10, 50 and 200 steps, holding online fitting at 50 steps, and with online budgets of 10, 50 and 100 steps, holding the initial fit at 50. All other data, windows and random streams remain fixed. These four alternatives add 5,760 method-setting records; every setting is reported and none is used to select the main result.

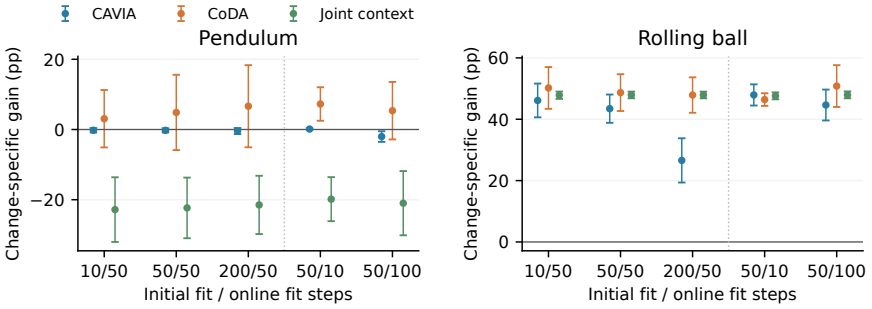

Figure 3: Fixed-checkpoint budget sensitivity. Points show change-specific gain; whiskers are pointwise 95% intervals over ten training seeds. The first three settings vary initial fitting, and the last two vary online fitting around the shared 50/50 reference. All alternatives use the same checkpoints and paired trajectories.

On the rolling ball, CAVIA's mean total gain rises from 25.7% to 46.0% to 72.8% as initial fitting increases, whereas its change-specific gain falls from 46.1% to 43.4% to 26.6%. More initial optimisation therefore does not simply make the subsequent update more responsive to change. The starting error and the control-transfer term also move. CoDA's total gains remain near 42% and its change-specific gains near 48–50%; joint context remains near 49% and 48%, respectively. The different sensitivities would be obscured by reporting only a favourable total-gain setting.

Varying online steps changes the magnitude of harm on the pendulum: CoDA's total gain ranges from −1.7% at ten steps to −52.4% at 100 steps, while its change-specific point estimate stays positive. Joint context's change-specific mean remains negative in every tested budget. Figure 3 and Appendix A.5 report the uncertainty and all settings. These results motivate reporting initial cali-

Table 2: Original contact taxonomy: post-change loss divided by the retained-code loss (lower is better), mean ± sample SD over ten training seeds. The first two rows are a paired identity check and count as one condition in aggregate statistics.

Scenario	Dynamics change?	Re-encode	Gradient
No change	no	0.941 ± 0.083	0.844 ± 0.223
Friction on rolling ball	no	0.941 ± 0.083	0.844 ± 0.223
Gravity on rolling ball	yes	0.935 ± 0.088	0.790 ± 0.249
Friction on sliding block	yes	1.076 ± 0.151	0.978 ± 0.073
Rolling/sliding threshold	yes	0.937 ± 0.097	0.856 ± 0.193

bration and update budget alongside a controlled gain; they do not select one universal deployment budget.

3.3 What the original no-op taxonomy establishes

The modular predictor provides a complementary test of re-encoding versus gradient refinement. It uses per-variable encoders, recurrent dynamics with message passing and a 32-dimensional code for two-parameter systems. A retained-code control and two update policies start from the same pre-change fit. Five contact scenarios include unchanged dynamics, inert friction on a rolling ball, changed gravity, effective friction on a sliding block and a rolling-to-sliding switch. The last occurs below the cylinder threshold $\mu_c = \tan(0.3\,\mathrm{rad})/3 = 0.1031$.

Gradient refinement has lower loss at every scenario mean (Table 2), and its improvement on the inert row demonstrates recalibration directly. After merging the duplicate control and giving the four remaining conditions equal weight, the re-encoding-minus-refinement advantage is 0.105. Its ten seed-level means range from 0.005 to 0.308 and are positive for nine of the ten, giving a paired t interval of $[0.037, 0.174]$ that excludes zero. Refinement therefore improves on re-encoding by a margin this design resolves.

The five-seed pendulum amplitude sweep supplies a same-system extension of the inert control. Scaling g and L together preserves its observations, while scaling g alone changes them. The inert-arm gain is 0.213 at each amplitude, and the real-minus-inert *within-arm ratio* contrasts are 0.021, 0.037 and 0.044. Their seed-level intervals all include zero (Appendix B). These contrasts motivate matched replay, but are not interpreted as identified recovery shares: they score different query trajectories and use different frozen-code denominators.

3.4 When a fitted representation is useful

The offline study trains on nine physical systems, fits one code on five probe trajectories and scores eleven disjoint query trajectories. It measures representation fitting at fixed environment parameters; it is not by itself evidence of recovery after a change. Across 45 system-seed runs, 39 improve, and mean per-run error reduction is 8.6%. Per-system aggregate reductions range from 0.8% to 34.1%. Persistence remains a strong comparator on smooth systems and beats the adapted model on the coupled oscillator and pendulum. Appendix F reports both absolute errors and all baselines, rather than using an underperforming RSSM as the sole reference.

Two larger systems test the limits of the same update. The coupled oscillator has four state variables and three physical parameters; Lotka–Volterra has two state variables and four parameters. Changing one parameter helps both the modular and context-GRU candidates at the tested settings. Changing all parameters is approximately neutral for the modular model and harmful for the context GRU on Lotka–Volterra. These configurations vary both the number of changed constants and their effective perturbation. We treat the pattern as a boundary observation; the equivalent-intervention test above shows why the count alone cannot explain it.

The boundary can then be located rather than only observed. Varying the shift over seven settings on both sides of no change moves the change-specific gain monotonically on each side, and the two sides are not mirror images: the rolling ball's boundary sits effectively at zero, while on the pendulum the two directions of change diverge, and the mechanism rather than the system decides which direction is harmful. How far the changed branch of the code separates from the unchanged one accounts for the sign across the six system-mechanism pairs, and the condition follows in first order:

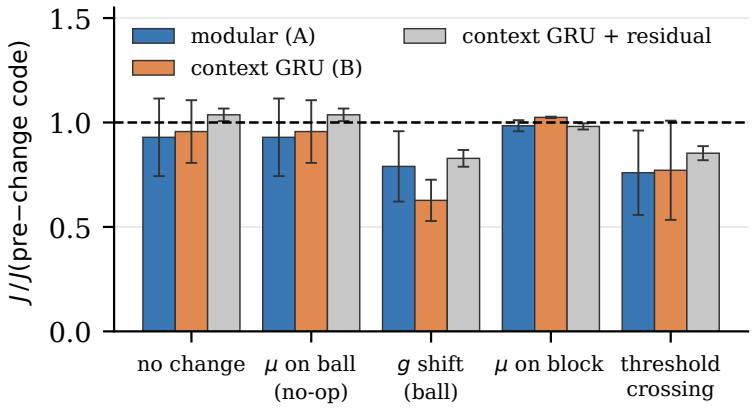

Figure 4: The adaptation pattern survives a change of architecture. Bars are within-model loss ratios after gradient fitting; lower is better. Whiskers are sample SD over three seeds. A context GRU, its residual variant and the modular model share the evaluation data and update budget.

writing g_I for the query-loss gradient at the code the unchanged branch reaches, $S \approx -\langle g_I, c^R - c^I \rangle$, so an observed change helps exactly when the update it induces moves the code down the query's own loss surface. Appendix P gives the amplitudes, correlations and verification.

3.5 Does the effect require modular structure?

A capacity-matched context GRU recovers the qualitative adaptation pattern of the modular model. On the three real-change contact scenarios, its mean within-model loss ratio is 0.808, compared with 0.845 for the modular candidate. Because each ratio uses its own frozen-model denominator, this tests whether adaptation survives a different architecture, rather than establishing an absolute prediction ranking. Figure 4 shows the individual scenarios.

The structural controls favour simple explanations of the gain. A learned sparse graph does not improve consistently on a fixed dense graph across free fall, pendulum and spring–mass. Removing all cross-variable paths impairs prediction, but also blocks the code from reaching the output, so that variant is an information-path ablation. Likewise, selecting designated code coordinates provides little advantage over a random equal-size subset in the selective-update study. These results support retaining communication and testing a simple context model before adding learned structure; they do not identify a universally optimal architecture. Full ablations and all update cases are in Appendices G and H.

The decomposition is not specific to gradient fitting: it holds for a capacity-matched adapter, for LoRA up to rank four, and for an amortised in-context baseline (Appendix O).

A higher-dimensional check, and what an inert intervention requires. The protocol asks one thing of a system: a parameter direction that leaves the observation law unchanged. The existing simulator family supplies two such directions and both are two-dimensional, so we built a third. The coupled oscillator's spring parameters map one-to-one onto the quantities that enter its forces, with a damping term fixed by construction, and Lotka–Volterra scaling is a time reparameterisation; neither admits an inert direction.

The multi-pendulum is three independently parameterised pendulums advanced on one shared clock and observed jointly, giving six state dimensions and twelve causal variables. Each block obeys $\dot{\omega}_i = -(g_i/L_i)\sin\theta_i$, so scaling g_i and L_i together inside one block leaves that block's trajectory unchanged, while scaling g_i alone changes the ratio by the same factor. The inert arm reproduces the unchanged arm bitwise before the change point, and the runner asserts this identity along with $G = C + S$ on every row.

Refinement is harmful on this system in both components and at every displacement. Over ten seeds, total gain is -0.189 with interval $[-0.240, -0.136]$, splitting into -0.099 of control trans-

fer and -0.090 of change-specific gain, and all three intervals exclude zero. The damage grows monotonically with displacement, from -0.142 total and -0.040 change-specific at 15 percent to -0.240 and -0.144 at 50 percent. Where the rolling ball gave a gain that aggregate reporting would overstate, this system gives one it would understate, so the decomposition informs in both directions rather than only where the total gain looks favourable.

A uniform budget can be the confound. Holding the update rule fixed is what makes the mechanisms comparable, but it can penalise one whose training assumed a different rule. CAVIA is meta-trained with five inner-loop steps of size 0.1, so we repeated its evaluation with that same rule at deployment. On the pendulum the uniform rule gives a total gain of -0.019 with interval $[-0.028, -0.011]$, while the native rule gives $+0.0006$ with interval $[+0.0001, +0.0013]$: the apparent harm is more than thirty times the residual and does not survive the change, and the change-specific component follows, -0.003 becoming $+0.0002$. The direction of the original reading survives but its magnitude does not, so under a matched optimiser CAVIA neither harms nor materially improves this system.

We report this because the decomposition is what exposed it: scoring every method under one budget keeps the comparison honest, and separating the two components turns a confound into a visible reading.

4 Discussion

The useful distinction is between improving a predictor and using information about a change productively. A deployment system needs low error, so absolute accuracy and total gain stay essential; a study of adaptation must additionally ask what the changed observations supplied. Matched replay asks that while holding the query trajectory fixed, and it surfaces both apparent advantages that the change-specific comparison does not resolve and updates that are actively harmful. Subsection 3.4 locates that boundary in shift amplitude and gives it a mechanism.

The measurement is conditional on a specified simulator coupling and update procedure: an inert branch must preserve the observation process as well as the transition law, and the two code sequences must be usable by the same frozen predictor. Portability therefore lies in the evaluation design, not in an assumption that every environment supplies such a control, and the family marks a failure case too: the spring-mass damping coefficient is fixed at construction rather than exposed as a parameter, so scaling its stiffness and mass together changes the damping ratio and is not inert.

The recurrent state-space model in the nine-system tables is a reference, not a tuned baseline, and does worse than persistence on several systems; nothing above depends on it, since the change-specific contrast is computed inside one system against a code that same model fits. Secondary comparisons run at three seeds and are reported as descriptive.

The evidence concerns state observations of two to six dimensions, supplied change times, and horizons of one and five steps. The archived MuJoCo experiments condition on a named friction scalar and remain diagnostics that do not validate the same unsupervised-code method; closed-loop return and learned change triggers would establish a different part of the argument.

A compact code is not the only parameter-efficient choice, so the protocol is run with a rank-1 adapter carrying twelve times as many free parameters reaches a change-specific gain of $+0.015$ on the rolling ball against $+0.479$ for the code, so the extra capacity buys nothing on these systems (Appendix K).

A detector built by comparing the kept code's loss against a refitted code's loss separates real changes from provable no-ops on the ball but not on the pendulum, where the best threshold still misses most real changes (Appendix N). That is the same difficulty seen from the other side, and it is why the change time is supplied rather than inferred. The reading also survives observation noise five and ten times the training level, and holds across adaptation windows of five to twenty steps and learning rates from 0.003 to 0.03 (Appendices J and L).

5 Related work

Evaluating adaptation. TTAB identifies the sensitivity of test-time adaptation to model selection, initial model quality and shift type (Zhao et al., 2023). Our measurement question is narrower: what part of an update's prediction gain depends on changed observations? The inert intervention and common-query replay make this comparison executable in a physical simulator. The arithmetic identity is simple; the experimental content is the controlled construction of its three losses.

World models and changing dynamics. World models support prediction and planning through learned dynamics (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2025). ContextWM, contextual recurrent state-space models and DALI make environmental context explicit (Wu et al., 2023; Prasanna et al., 2024; Röder et al., 2025); reconstruction conditions delimit what compact histories can represent (Ross et al., 2025). AdaWM adapts parts of a driving agent, AdaWorld learns transferable latent actions, and ReDRAW uses latent dynamics residuals (Wang et al., 2025; Gao et al., 2025; Lanier et al., 2025). In-context world models learn or recognise environments from trajectories (Wang et al., 2026). Our controlled setting measures which prediction gains depend on changed observations, independently of a particular world-model design.

Context adaptation and system identification. Meta-learning prepares models for a small number of updates (Finn et al., 2017; Nagabandi et al., 2019). CAVIA restricts task adaptation to a context vector, PEARL infers a probabilistic context, and CoDA maps context into a low-dimensional subspace of weights (Zintgraf et al., 2019; Rakelly et al., 2019; Kirchmeyer et al., 2022). Related dynamics models and identification methods separate shared structure from environment information (Lee et al., 2020; Wang & Johnson, 2021; Evans et al., 2022; Forgione et al., 2025). Universal policies and rapid motor adaptation infer physical context for control (Yu et al., 2017; Kumar et al., 2021); online model updates use filtering, compensation or causality-guided representations (Abuduweili & Liu, 2020; 2023; Yang et al., 2025). Uncertainty-aware context models further address when to trust a fit (Rufolo et al., 2026). We build on these adaptation mechanisms and evaluate the added value of changed data using a common query distribution. Meta-learned in-context models can also be adapted after training, and adapting them helps most when the target lies outside the training class (Piga et al., 2023); physically-constrained meta-learning reaches similar conclusions while showing that fine-tuning subnetworks beats fine-tuning everything (Chakrabarty et al., 2025).

Structure in learned dynamics. Interaction networks, neural relational inference and graph simulators represent interactions between variables or objects (Battaglia et al., 2016; Kipf et al., 2018; Sanchez-Gonzalez et al., 2020; Allen et al., 2023). Causal dynamics learning targets transferable state abstractions (Wang et al., 2022). Our ablations distinguish the ability to communicate across variables from the benefit of estimating a sparse adjacency. This complements the replay analysis: useful prediction does not establish either a semantic code or a learned causal graph. Variational causal dynamics learns a modular transition model and uses interventions to identify which parts of the dynamics changed (Lei et al., 2022), which is the structural counterpart of the update-scope question we measure.

6 Conclusion

Fitting an environment code can retain useful knowledge while improving a world model, but the meaning of the improvement requires a control. Provably inert interventions detect recalibration; matched code replay measures what changed observations add on a shared query distribution. The resulting signed decomposition changes concrete judgments of CAVIA, CoDA and joint context updates and exposes the limits of raw parameter counts as a measure of change. Reporting this comparison beside absolute error makes adaptation claims more informative about both the source of a gain and the predictor that would actually be deployed.

Reproducibility statement

The supplement gives the analytical controls, exact estimands, all configurations and seed-level summaries. The accompanying code provides experiment runners, raw per-case losses, checkpoint

selection records and a deterministic analysis script. Tests cover no-op identity, equivalent interventions, disjoint probe/query streams, frozen shared weights, CAVIA's meta-gradient and CoDA's affine parameterisation. Appendix A maps these artifacts to the new comparison; Appendix E covers the complementary experiments.

AI USE STATEMENT

Generative AI tools were used for a limited set of supporting tasks: checking and reformulating parts of the authors' text, verifying that quantities reported in the manuscript match the result files, and consistency checks across the manuscript. All of that output was reviewed by the authors. The tools were not used to implement the method, to design the experiments, to generate data sets, to formulate or prove the mathematical claims, or to propose the hypotheses under study; every trajectory comes from the simulators in the repository, and the methods and conclusions reported here are the authors' own. We take responsibility for the final content of this work, including text, claims and artifacts.

See the ICLR 2027 AI Policy for Authors for more details.

ETHICS STATEMENT

The experiments use simulated physical systems and contain no human-subject or personal data. The reported prediction results do not establish safe autonomous deployment or improved control returns.

REFERENCES

Abulikemu Abuduweili and Changliu Liu. Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120, pp. 65–74, 2020. URL https://proceedings.mlr.press/v120/abuduweili20a.html.

Abulikemu Abuduweili and Changliu Liu. Online model adaptation with feedforward compensation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pp. 3687–3709, 2023. URL https://proceedings.mlr.press/v229/abuduweili23a.html.

Kelsey R. Allen, Tatiana Lopez-Guevara, Yulia Rubanova, Kimberly Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pp. 1157–1167, 2023. URL https://proceedings.mlr.press/v205/allen23a.html.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/3147da8ab4a0437c15ef51a5cc7f2dc4-Abstract.html.

Ankush Chakrabarty, Gordon Wichern, Vedang M. Deshpande, Abraham P. Vinod, Karl Berntorp, and Christopher R. Laughman. Meta-learning for physically-constrained neural system identification. *arXiv preprint arXiv:2501.06167*, 2025.

Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context is everything: Implicit identification for dynamics adaptation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Marco Forgione, Ankush Chakrabarty, Dario Piga, Matteo Rufolo, and Alberto Bemporad. Manifold meta-learning for reduced-complexity neural system identification. *arXiv preprint arXiv:2504.11811*, 2025.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 18744–18771, 2025. URL https://proceedings.mlr.press/v267/gao25u.html.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 2688–2697, 2018. URL https://proceedings.mlr.press/v80/kipf18a.html.

Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In *International Conference on Machine Learning*, 2022. URL https://arxiv.org/abs/2202.01889.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Proceedings of Robotics: Science and Systems*, 2021. doi: 10.15607/RSS.2021.XVII.011.

JB Lanier, Kyungmin Kim, Armin Karamzade, Yifei Liu, Ankita Sinha, Kat He, Davide Corsi, and Roy Fox. Adapting world models with latent-state dynamics residuals. *arXiv preprint arXiv:2504.02252*, 2025. URL https://arxiv.org/abs/2504.02252.

Laozi. *Tao Te Ching*. Penguin Classics, 1963.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 5757–5766, 2020. URL https://proceedings.mlr.press/v119/lee20g.html.

Anson Lei, Bernhard Schölkopf, and Ingmar Posner. Variational causal dynamics: Discovering modular world models from interventions. *arXiv preprint arXiv:2206.11131*, 2022.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1803.11347.

Dario Piga, Filippo Pura, and Marco Forgione. On the adaptation of in-context learners for system identification. *arXiv preprint arXiv:2312.04083*, 2023.

Sai Prasanna, Karim Farid, Raghu Rajan, and André Biedenkapp. Dreaming of many worlds: Learning contextual world models aids zero-shot generalization. In *Reinforcement Learning Conference*, 2024.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5331–5340, 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Frank Röder, Jan Benad, Manfred Eppe, and Pradeep Kr. Banerjee. Dynamics-aligned latent imagination in contextual world models for zero-shot generalization. In *Advances in Neural Information Processing Systems*, 2025.

Edmund Ross, Claudia Drygala, Leonhard Schwarz, Samir Kaiser, Francesca di Mare, Tobias Breiten, and Hanno Gottschalk. When do world models successfully learn dynamical systems? *arXiv preprint arXiv:2507.04898*, 2025.

Matteo Rufolo, Dario Piga, and Marco Forgione. Variational meta-learning inference for low dimensional neural system identification. *arXiv preprint arXiv:2607.18965*, 2026.

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

Fan Wang, Zhiyuan Chen, Yuxuan Zhong, Sunjian Zheng, Pengtao Shao, Bo Yu, Shaoshan Liu, Jianan Wang, Ning Ding, Yang Cao, and Yu Kang. Context and diversity matter: The emergence of in-context learning in world models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a46b9dcef23e9160f46d800a49a931d0-Abstract-Conference.html.

Hang Wang, Xin Ye, Feng Tao, Chenbin Pan, Abhirup Mallik, Burhan Yaman, Liu Ren, and Junshan Zhang. AdaWM: Adaptive world model based planning for autonomous driving. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d4c745dcbaf8ef0d7e145754e31b1516-Abstract-Conference.html.

Sean J. Wang and Aaron M. Johnson. Domain adaptation using system invariant dynamics models. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144, pp. 1130–1141, 2021. URL https://proceedings.mlr.press/v144/wang21c.html.

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 23151–23180, 2022. URL https://proceedings.mlr.press/v162/wang22ae.html.

Jialong Wu, Haoyu Ma, Chaoyi Deng, and Mingsheng Long. Pre-training contextualized world models with in-the-wild videos for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Yupei Yang, Biwei Huang, Fan Feng, Xinyue Wang, Shikui Tu, and Lei Xu. Towards generalizable reinforcement learning via causality-guided self-adaptive representations. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/83c230118e9f6688ba8f20bfef99e6da-Abstract-Conference.html.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017. URL https://www.roboticsproceedings.org/rss13/p48.pdf.

Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 42058–42080, 2023. URL https://proceedings.mlr.press/v202/zhao23d.html.

Luisa Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. CAVIA: Fast context adaptation via meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 7693–7702, 2019. URL https://proceedings.mlr.press/v97/zintgraf19a.html.

A MATCHED REPLAY: DEFINITIONS, IMPLEMENTATION AND COMPLETE RESULTS

A.1 THE COMPARISON AND ITS ASSUMPTIONS

For each trained predictor, physical condition and paired random stream, let Q^R be the real-change query trajectory. Define

$$J_R(c) = \sum_{t=30}^{58} \frac{1}{d} \left\| F_\theta(H_t^R, c_t) - o_{t+1}^R \right\|_2^2 .$$

The query history H_t^R is identical in every term of the comparison. The retained code c_0 is fitted on the first 30 probe frames. The inert and real code sequences are fitted on their respective probe histories, using the same optimiser and schedule. All shared weights remain unchanged, including when the CoDA context generates different effective predictor weights through its frozen hypernetwork.

Proposition 1 (Common-query gain identity)**.** *Fix a predictor, initial code, paired probe streams and real query stream. The gains in Equations 3–5 satisfy $G = C + S$ for every realised stream and hence under any common averaging distribution. The identity also holds after dividing all three terms by the same positive reference loss.*

Proof. Insert and subtract $J_R(c^I)$ in $J_R(c_0) - J_R(c^R)$. No exchange of expectations, distributional invariance of losses, or assumption about the optimisation landscape is required. Linearity of expectation preserves the equality. Division by a common nonzero scalar preserves it samplewise, which also permits averaging ratios rather than taking a ratio of averages. □

What is identified. The experiment identifies the loss contrast between two executable code-update procedures under the chosen simulator coupling. It holds the predictor, query state sequence, initial code and optimisation budget fixed, and changes the observations supplied to the code updater. In a recurrent predictor, hidden states induced by the same query history must also be handled identically across the comparison. The new reference implementations are Markov state predictors, making this condition immediate. No query observation is used for code optimisation.

What is not identified. C is control-transfer gain on the real query, not the inert arm's own improvement $J_I(c_0) - J_I(c^I)$. These quantities need not coincide. S includes all consequences of supplying changed observations, including changes in the probe-state distribution and measurement-noise response; it is not an identified physical parameter or a pure latent mechanism-recovery term. Its value depends on the coupling, update budget and initial code. It can be negative, and no non-negative "recalibration share" or "recovery share" is assumed. Outside a simulator capable of the matched fork, this contrast need not be observable.

Proposition 2 (Invariance to a change of physical description)**.** *If two interventions produce the same observed probe and query trajectories and the update rule has the same initial state and randomness, they produce the same fitted codes and predictions. The number of named constants modified by the interventions cannot distinguish them.*

Proof. At initialisation the update inputs agree. If their codes agree at an update time, the identical observed fitting data and update randomness produce the same next code. Induction over updates gives identical code sequences. Identical query histories then give identical predictions. The argument uses observation equivalence, not a physical interpretation of the code coordinates. □

For the pendulum, $g'/L' = fg/L$ for both $(g', L') = (fg, L)$ and $(f^2 g, fL)$. For the rolling ball, the acceleration depends on g but not on μ within the rolling regime. Thus the two-constant real interventions have the same transition law as their one-constant counterparts. The number of *effective* mechanisms changed is not increased by merely modifying an additional redundant or inert constant.

A.2 PHYSICAL CONDITIONS AND INDEPENDENT RANDOM STREAMS

Table 3: Predeclared matched interventions. $f \in \{1.15, 1.30, 1.50\}$. Inert and unchanged branches have equal dynamics; real and equivalent branches have equal changed dynamics.

System	Source	Inert target	Real / equivalent targets
Pendulum	$(g, L) = (3.48, 3.93)$	(fg, fL)	(fg, L) / $(f^2 g, fL)$
Rolling ball	$(\mu, g) = (0.30, 9.5)$	$(f\mu, g)$	(μ, fg) / $(f\mu, fg)$

Training seeds are 42–51. For seed s and replicate $r \in \{0, \ldots, 7\}$, probe randomness uses seed $200000 + 100s + r$ and query randomness uses $300000 + 100s + r$. The two offsets make probe and query streams disjoint; each intervention branch reuses its stream's seed to pair initial conditions and observation noise. The simulator runs for 60 frames, with the parameter assignment changed at step 30 and no state reset. The observation noise standard deviation is 0.01, as in the original environments. Every episode asserts identical pre-change histories, exact no-op identity, nonidentity of the real arm and agreement within 10^{-6} for equivalent one- and two-constant interventions.

Training and source validation sample each physical constant uniformly and independently from the ranges stated in Section 3.1; the validation random stream is separate. Both simulators use step size 0.05 with semi-implicit Euler updates (velocity before position). Pendulum initial angle is uniform on $[-\pi/4, \pi/4]$ and angular velocity on $[-1, 1]$. The ball starts at position zero with velocity uniform on $[0, 1]$, on an incline of 0.3 radians. Observations are angle/angular velocity or position/velocity with independent Gaussian noise of standard deviation 0.01 per coordinate. Physical states evolve without process noise. For the ball, the smallest sampled friction 0.1 exceeds the required no-slip threshold $\frac{2}{7}\tan(0.3) \simeq 0.0884$; the tested inert changes remain in this regime. The test source and target ratio range for the pendulum is approximately $[0.885, 1.328]$, inside the training ratio support $[1/6, 2]$, despite its individually out-of-range constants. These fixed physical choices are controlled test cases rather than a random sample of all possible environmental changes.

There are $2 \times 10 \times 3 \times 8 = 480$ physical settings. The three methods produce $1{,}440$ result records from 60 trained models. Reusing a physical setting across methods creates a paired comparison, not another independent sample. The eight test pairs and three amplitudes are averaged within each training seed. Systems are reported separately.

A.3 REFERENCE METHODS AND SOURCE-ONLY MODEL SELECTION

All reference methods predict a normalized state increment, then transform it back to observation units and add it to the current state. State and increment means and standard deviations are estimated from training data only; each standard deviation is lower bounded by 0.01. The normalization is fixed for source validation and every target branch. Every code has dimension eight. No physical parameter labels are supplied to training or fitting.

CAVIA implementation. The predictor concatenates the state and context, followed by two 96-unit tanh hidden layers and a two-dimensional linear output. For each training trajectory, the context starts at zero and takes five SGD steps of size 0.1 on transitions among its first 30 frames. The outer loss uses transitions 50–98 of the same trajectory, which do not overlap the context-fitting frames. The outer gradient differentiates through every inner step with second-order derivatives. This implements context-only adaptation and shared-weight meta-learning; it does not replace the meta-objective with ordinary joint training. Deployment fitting uses the common Adam rule to isolate evaluation under one fixed update budget.

CoDA implementation. The base predictor has two 32-unit tanh layers. An affine hypernetwork generates every predictor weight and bias as $\theta_0 + Wc$, with W of rank at most eight. Training jointly optimises θ_0, W and one zero-initialised context per training trajectory. The mean per-task normalised increment MSE is regularised by $10^{-4}\|c\|_2^2 + 10^{-6}\|W\|_F^2$. At adaptation only c changes; the context penalty is retained and the basis penalty is constant. This is a discrete one-step implementation of affine context-conditioned parameter adaptation, rather than CoDA's original continuous-time trajectory solver and experimental architecture.

Joint-context implementation. This method uses CAVIA's architecture but jointly learns shared weights and a code for each training trajectory with ordinary one-step prediction loss. It has no meta-gradient. New environments still start from a zero code and use gradient fitting. The comparison therefore includes meta-learned input context, joint input context and low-rank parameter context.

Table 4: Reference-method configuration. Equal training data and evaluation budgets do not imply equal training compute: CAVIA differentiates through its inner updates.

Setting	Value
Training / validation	64 / 16 independent 100-frame trajectories per system and seed
Training optimiser	Adam, learning rate 10^{-3}, 60 epochs, task batch size 8
Gradient clipping	Shared-training gradient norm capped at 5
Checkpoint candidates	Epochs $0, 10, 20, 30, 40, 50, 60$
Selection score	Validation one-step MSE on frames 50–99 after fitting on frames 0–29
Validation seed	Training seed plus 10000; no target data used
Shared parameters	10,562 (CAVIA and joint context); 10,962 (CoDA)
Training-only contexts	64 vectors of dimension 8 for joint context and CoDA
Initial deployment fit	50 Adam steps, learning rate 0.01, first 30 probe frames, zero code
Post-change fit	50 Adam steps, learning rate 0.01, latest ten probe frames, warm-started code
Update times	After scoring predictions at $t = 35, 40, 45, 50, 55$; use probe frames through t
One-step score	Mean over two observation coordinates, summed for $t = 30, \ldots, 58$
Five-step score	Free rollout from query state at $t = 30, \ldots, 54$; code held fixed during rollout

The initial smoke run used seed 41, ten training epochs and two test pairs solely to exercise the implementation. It is excluded from every reported analysis. The full comparison used the settings above without choosing hyperparameters from target scores. The goal is a controlled test of evaluation judgments across documented implementations, not an optimally tuned leaderboard for the named methods.

A.4 Absolute errors and uncertainty

Table 5: Absolute query MSE, averaged over the same conditions as Table 1; lower is better. 0, I and R denote retained, inert-updated and real-updated codes, always scored on the real query. The one-step losses divide J by 29.

		One step			Five-step rollout		
System	Method	0	I	R	0	I	R
Pendulum	CAVIA	0.000289	0.000294	0.000295	0.00251	0.00262	0.00264
Pendulum	CoDA	0.000243	0.000336	0.000322	0.00128	0.00344	0.00306
Pendulum	Joint context	0.000273	0.000286	0.000351	0.00206	0.0024	0.00402
Rolling ball	CAVIA	0.00112	0.00106	0.000548	0.024	0.0235	0.00914
Rolling ball	CoDA	0.000817	0.000867	0.000418	0.0161	0.0173	0.00672
Rolling ball	Joint context	0.000819	0.000813	0.000358	0.0161	0.0159	0.00495

For a reported gain x, let $\bar{x}_s$ be its mean over the 24 fixed test settings inside seed s. We report $\bar{x} \pm t_{9,0.975}\ \mathrm{SD}(\bar{x}_s)/\sqrt{10}$. Pairwise method intervals first subtract seed-level means and then apply the same rule. These are pointwise intervals, not simultaneous confidence statements over all comparisons. The analysis file additionally reports exact two-sided sign-flip statistics for the ten paired seed differences; these are descriptive secondary checks rather than a multiple-testing-adjusted claim of universal method superiority. The CAVIA–CoDA change-specific contrast on the rolling ball, for example, has interval $[-12.8, +2.4]$ percentage points, so its point-estimate ordering is not a resolved ranking.

Table 6: Pendulum: seed-level relative gains in percentage points. Each row averages three amplitudes and eight probe/query pairs.

	Total gain G			Change-specific gain S		
Seed	CAVIA	CoDA	Joint	CAVIA	CoDA	Joint
42	-2.9	-22.8	-12.2	$+0.3$	-6.3	-9.9
43	-0.3	-0.5	-4.9	$+0.3$	$+2.3$	-5.1
44	-2.4	-28.7	-59.0	-0.3	-10.2	-44.0
45	-0.8	-6.9	-27.6	-0.1	$+10.1$	-24.3
46	-2.4	-4.4	-32.7	-0.7	-1.0	-27.1
47	$+0.4$	-37.8	-34.2	$+1.2$	$+43.1$	-32.5
48	-0.9	-2.6	-15.6	-0.0	$+2.0$	-12.8
49	-3.2	-8.8	-15.2	-1.9	$+0.8$	-13.7
50	-2.1	-81.3	-27.2	-1.0	$+11.3$	-21.8
51	-4.7	-68.7	-42.1	-0.1	-3.7	-32.1

Table 7: Rolling ball: seed-level relative gains in percentage points. Each row averages three amplitudes and eight probe/query pairs.

	Total gain G			Change-specific gain S		
Seed	CAVIA	CoDA	Joint	CAVIA	CoDA	Joint
42	$+45.9$	$+43.6$	$+47.7$	$+45.3$	$+45.4$	$+48.7$
43	$+29.9$	$+33.2$	$+47.6$	$+43.6$	$+52.5$	$+46.0$
44	$+31.0$	$+43.5$	$+49.3$	$+39.9$	$+46.6$	$+48.7$
45	$+43.2$	$+47.3$	$+48.0$	$+49.6$	$+48.6$	$+46.3$
46	$+40.5$	$+35.6$	$+48.8$	$+45.7$	$+64.5$	$+47.2$
47	$+54.1$	$+49.6$	$+50.2$	$+42.1$	$+54.8$	$+49.7$
48	$+66.0$	$+45.6$	$+49.9$	$+37.1$	$+49.3$	$+50.2$
49	$+71.5$	$+45.8$	$+49.2$	$+30.3$	$+44.0$	$+48.2$
50	$+31.9$	$+47.5$	$+48.1$	$+51.6$	$+49.5$	$+48.3$
51	$+46.4$	$+29.8$	$+47.5$	$+49.3$	$+31.6$	$+45.8$

A.5 Fixed-checkpoint budget sensitivity

The sensitivity runner restores all 60 main-experiment checkpoints without retraining. It first reproduces the eight pendulum CAVIA cases for seed 42 and factor 1.3 at the original budget, checking every one-step loss to absolute tolerance 10^{-12}. It then runs four alternative initial/online fitting budgets: $10/50$, $200/50$, $50/10$ and $50/100$. All seeds, amplitudes, probe/query streams, learning rates and window lengths are unchanged. The resulting $5{,}760$ records are stored separately from the main $1{,}440$ records. The alternatives were fixed before their evaluation and all are reported; no target score selects a checkpoint, budget or subset of cases.

Table 8: All fitting budgets. Each cell gives total / change-specific gain in percentage points, averaged within seed and then over ten seeds. The $50/50$ row is the main experiment. Figure 3 gives seed-level uncertainty for the change-specific comparison; the analysis file retains both metrics' per-seed values and intervals.

System	Initial/online steps	CAVIA	CoDA	Joint context
Pendulum	10/50	$-2.1/-0.2$	$-25.2/+3.1$	$-29.7/-22.8$
Pendulum	50/50	$-1.9/-0.2$	$-26.3/+4.9$	$-27.1/-22.3$
Pendulum	200/50	$-1.2/-0.4$	$-23.9/+6.6$	$-25.1/-21.5$
Pendulum	50/10	$+0.0/+0.1$	$-1.7/+7.3$	$-23.3/-19.8$
Pendulum	50/100	$-6.4/-2.0$	$-52.4/+5.4$	$-29.4/-21.0$
Rolling ball	10/50	$+25.7/+46.1$	$+41.8/+50.2$	$+48.7/+47.8$
Rolling ball	50/50	$+46.0/+43.4$	$+42.2/+48.7$	$+48.6/+47.9$
Rolling ball	200/50	$+72.8/+26.6$	$+42.0/+47.9$	$+48.6/+47.9$
Rolling ball	50/10	$+47.8/+47.9$	$+44.9/+46.4$	$+48.3/+47.6$
Rolling ball	50/100	$+47.3/+44.6$	$+41.6/+50.8$	$+48.7/+47.9$

The runner is `run_replay_sensitivity_v1.py`; `summarize_replay_sensitivity_v1.py` produces the table, plot and machine-readable seed-level statistics. Because more optimisation can also overfit a noisy prefix, the number of initial fitting steps is not assumed to order initial-code quality.

A.6 Executable checks and result provenance

The runner `run_matched_controls_v1.py` writes its command-line configuration, environment versions, Git revision and source SHA-256 before any training. Each saved checkpoint records its source-validation score and selected epoch. The raw `per_case.json` contains all five one-step losses $J_R(c_0), J_R(c^I), J_R(c^R), J_I(c_0), J_I(c^I)$ and the three five-step errors. The analysis script `summarize_matched_controls_v1.py` generates both manuscript tables and figures from these records; all underlying conditions, including harmful updates, are retained.

The test suite `test_matched_controls_v1.py` checks the physical no-op and equivalent-change identities, disjoint probe/query streams, CoDA's batched affine network against an explicit weight-by-weight forward pass, CAVIA's second-order meta-gradient against finite differences, invariance of shared weights during fitting, and equality between batched and individual code fits. The runner additionally checks unchanged shared weights after every evaluation and verifies that no post-change update affects the first six scored predictions. These tests validate the mechanics of the comparison; they do not substitute for external benchmarks.

B Reanalysis of the original taxonomy and amplitude sweep

The main text reports this contrast recomputed on ten seeds. This section keeps the original three-seed reanalysis, because the feature that motivates the larger design is visible only in the old table's structure: it has 15 displayed rows but only four distinct physical data conditions per seed, the unchanged and inert-friction conditions being bit-identical. We merge these conditions and average the four re-encode-minus-gradient differences inside each seed. The resulting values are 0.009178, 0.020870 and 0.129465, with mean 0.053171 and 95% paired t interval $[-0.111604, 0.217946]$. An exact two-sided sign-flip test on three seed-level means has minimum attainable $p = 0.25$, which is attained here. An interval obtained by resampling the 12 or 15 rows independently does not represent training-seed uncertainty. A three-cluster percentile bootstrap also has very limited support, so it is not used as the principal uncertainty statement.

Table 9: Original five-seed pendulum sweep. The contrast subtracts two within-arm relative gains; it is not the common-query S defined in this paper. Each inert row equals the unchanged condition and is not counted as another independent control.

Gravity increase	Inert-arm gain	Real-minus-inert contrast	95% paired t interval
15%	0.213	0.021	$[-0.028, +0.071]$
30%	0.213	0.037	$[-0.055, +0.129]$
50%	0.213	0.044	$[-0.083, +0.171]$

The interval uses $t_{4,0.975}$ over seeds 42–46. The results come from `results/pendulum_sweep_v1/per_seed.json`. The partially completed `results/pendulum_mech_v1` output is not pooled into the sweep or any formal estimate. The original sweep establishes that a valid same-system inert intervention is available; the new replay experiment addresses the remaining mismatch of query distribution and denominator.

C Analytical controls

The change taxonomy uses elementary mechanics to decide whether a changed simulator parameter can alter the observations. This decision is made before looking at model error.

Proposition 3 (Friction no-op for the rolling ball)**.** *For a solid sphere rolling without slip on an incline of angle θ, the centre-of-mass acceleration is $a = \frac{5}{7} g \sin\theta$ and is independent of the coefficient of static friction μ, provided the no-slip inequality remains satisfied.*

Proof. Translation and rotation give $ma = mg\sin\theta - f$ and $fR = I\alpha$ with $I = \frac{2}{5}mR^2$ and $a = R\alpha$. Eliminating f yields $a = \frac{5}{7}g\sin\theta$. Friction only supplies the constraint required for rolling. Both values used by the no-op pair lie in that regime, so paired trajectories with the same initial condition are identical even though μ changes. □

Proposition 4 (Cylinder regime boundary)**.** *A solid cylinder rolls without slip on the same incline only if $\mu \geq \mu_c = \tan\theta/3$. At $\theta = 0.3$ rad, $\mu_c = 0.1031$.*

Proof. For $I = \frac{1}{2}mR^2$, the rolling equations give $a = \frac{2}{3}g\sin\theta$ and required friction $f = \frac{1}{3}mg\sin\theta$. Since the normal force is $N = mg\cos\theta$, the static-friction condition $f \leq \mu N$ becomes $\mu \geq \tan\theta/3$. The intervention $0.50 \to 0.05$ therefore crosses the rolling/sliding boundary rather than merely changing a coefficient inside one regime. □

Proposition 5 (Pendulum non-identifiability)**.** *Nondegenerate angle trajectories of an unforced ideal pendulum can identify the ratio g/L, but cannot identify g and L separately. A static equilibrium trajectory need not identify even the ratio.*

Proof. The observed angle obeys $\ddot{\theta} = -(g/L)\sin\theta$. Any parameter pair (g', L') satisfying $g'/L' = g/L$ induces the same differential equation and, for matched initial conditions, the same trajectory. Separate identification requires an additional measurement or intervention. □

D Implementation notes

The original modular-model measurements follow the protocol of Section 2 under the configuration of Appendix R. Five properties of the implementation bear on how they should be read.

- **Causal parameter inference.** The parameter encoder is unidirectional: the code used at step t depends only on observations up to t, so no prediction can draw on the observation it is about to predict.
- **Probe and query separation.** The code is fitted on probe trajectories and scored on disjoint query trajectories generated with the same physical parameters, so a reported score cannot benefit from having been fitted on its own data.
- **Slots carry timing, not semantics.** Parameter slots are constant by construction, so a ranking of slots reflects the temporal constancy of the code rather than a discovered physical identity. Such rankings are reported as a privileged-information diagnostic, and no semantic-identification claim is made from them.
- **Initialisation at persistence.** Every decoder output head is zero-initialised, so training begins from the persistence predictor.
- **Ablation wiring.** The no-graph variant removes every cross-variable path, which makes it an information-path ablation rather than a usable adaptation baseline; the dense-graph variant serves as the structural control.

The Gaussian RSSM is a local implementation that follows the reverse-KL direction of its stated variational objective, and it is not a reproduction of DreamerV3.

E Evidence inventory

Table 10 maps each empirical question to its conditions and comparison. The separation matters because a seed, an environment and an update policy are not interchangeable replicates. Training seed is the replication unit for uncertainty about a trained method; conditions and policies evaluated on the same checkpoint remain paired measurements.

Table 10: Experiment inventory. "Cases" counts the primary paired settings before model or update variants are expanded. The online taxonomy's five scenario rows cover four distinct data conditions, because its no-change row and its shifted-friction row on a ball are bit-identical under the closed form.

Study	Scope	Cases	Question isolated
Offline code fitting	9 systems, 5 seeds	45	Does a probe-fitted code transfer to disjoint query trajectories?
Structure ablation	3 systems, 3 seeds	9	Message passing, graph learning, and residual parameterisation
Online taxonomy	4 distinct conditions, 3 seeds	12	Re-encoding versus gradient refinement; inert-control detection
Selective update	6 systems, 3 seeds, 2 change sizes	36	All, designated, and random equal-size update scopes
Additional systems	2 systems, 3 seeds, 3 change scopes	18	No, partial, and wholesale parameter changes under two candidates
MuJoCo diagnostic	2 agents, 2 friction targets, 3 seeds	12	Named-scalar prediction adaptation and behaviour-policy sensitivity

Table 11: Script-to-result map. Each entry lists the runner, the result directory it writes, the aggregate table it feeds and the recorded measurement unit. These are not all independent: statistical uncertainty groups measurements by training seed.

Runner	Result directory and paper output	Recorded unit
`run_protocol_v2.py`	`results/protocol_v2_9sys_*`; Table 12	system × seed
`run_change_taxonomy_v3.py`	`results/change_taxonomy_v4`; Table 2, Figure 6	scenario × seed
`run_online_change_v3.py`	`results/online_change_v3_*`; Table 14	system × seed × scope
`context_baseline_v3.py`	`results/context_baseline_v4_*`; Table 15	scenario × seed
`run_three_questions_v2.py`	`results/three_questions_v3_fair_*`; Table 18	system × seed × shift
`run_structure_v2.py`	`results/structure_v3_*`; Table 16	system × seed × variant
`run_dense_control_v2.py`	`results/protocol_v2_dense_*`; Table 16	system × seed
`dr_sensitivity.py`	`results/dr_sensitivity_grid_*`; Table 29	environment × bound × seed

F FULL TABLES

Nine-system evaluation. One-step test MSE, five seeds per system, at matched capacity. The dense RSSM is worse than persistence in eight of nine systems, by factors from $3.1\times$ (contact ball) to $487\times$ (pendulum); free fall is the exception, where the RSSM is the better predictor by $2.9\times$. Its validation error sits at persistence level, which is consistent with out-of-distribution extrapolation failure rather than an optimisation failure, although window length and hidden-state distribution are not excluded as contributing causes.

The adapted model improves on the RSSM reference but does not beat persistence on every system: it trails persistence on the coupled oscillator ($1.29\times$) and the pendulum ($1.30\times$) and only reaches $0.83\times$ on spring–mass. Persistence is a strong baseline on smooth low-dimensional systems, so the offline adaptation result is better read as catching up to it than as beating it.

Table 12: Nine-system evaluation with the dense RSSM and persistence references.

System	Persistence	RSSM	Code c_0	Adapted code $\hat{c}$
Alien physics	0.001010	0.3029	0.0007181	0.0006902
Contact ball	0.08384	0.2634	0.003040	0.002365
Contact block	0.0002164	0.003105	0.0002179	0.0001437
Contact cylinder	0.07358	0.2412	0.004213	0.003563
Coupled oscillator	0.006457	0.5498	0.008432	0.008295
Free fall	0.2205	0.07610	0.008802	0.008728
Lotka–Volterra	0.04438	0.1807	0.02870	0.02799
Pendulum	0.0006914	0.3366	0.0009262	0.0009011
Spring–mass	0.0007474	0.2812	0.0006296	0.0006165

Additional systems. Cumulative post-change loss J on systems with more latent variables and parameter slots, three seeds. Each ratio is gradient-to-fixed within the same candidate. The ratio columns r_A and r_B are interleaved with the absolute values they are computed from, and each ratio is the mean of the per-seed ratios rather than a quotient of the two adjacent means.

Table 13: Aggregate results on the two higher-dimensional systems. J_f retains the pre-change code and J_g refines it; both columns are means of per-seed absolute values, whereas r is the mean of the per-seed ratios, so r is not the quotient of the two columns beside it (lower is better).

System	Shift	A: $J_f \to J_g$	r_A	B: $J_f \to J_g$	r_B
Coupled	none	$0.0806 \to 0.0636$	0.772	$0.0279 \to 0.0147$	0.569
Coupled	one parameter	$0.1459 \to 0.1368$	0.932	$0.0421 \to 0.0298$	0.708
Coupled	all three	$0.3739 \to 0.3778$	1.010	$0.1226 \to 0.1146$	0.896
Lotka–Volterra	none	$0.9346 \to 0.5580$	0.719	$0.0788 \to 0.0069$	0.086
Lotka–Volterra	one parameter	$2.209 \to 2.128$	0.980	$0.8276 \to 0.4307$	0.479
Lotka–Volterra	all four	$2.497 \to 2.459$	0.984	$1.775 \to 2.049$	1.176

G SELECTIVE UPDATE AND ARCHITECTURE COMPARISONS

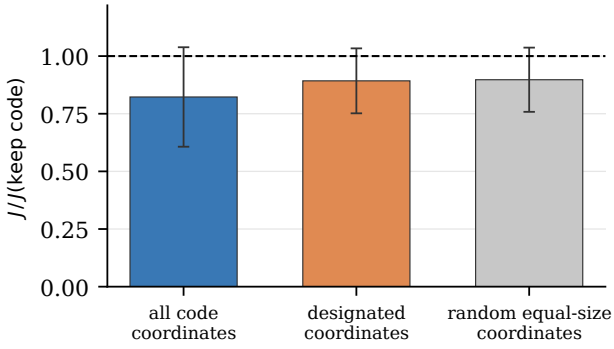

Figure 5: Update scope on the 18 partial-change cases (mean ± sample standard deviation; lower is better). Updating all code coordinates outperforms both the designated subset and a random equal-size subset.

Table 14 expands the selective-update summary in the main paper. When every parameter slot changes, the designated and random subsets both equal the full set by construction; the partial-change rows are the informative comparison; across the eighteen partial-change conditions the designated subset beats a random subset of equal size in 17% of them, by a mean of $+0.001$, so the subset choice carries no information the hypothesis assumes.

Table 14: Selective-update cases for candidate A, averaged within seed and then over ten seeds; entries are cumulative post-change loss normalised by retaining the pre-change code (lower is better), with sample SD. When every parameter slot changes, the designated and random subsets equal the full set by construction.

System	Changed	All coordinates	Designated	Random
Spring–mass	k	0.899	0.961 ± 0.082	0.957 ± 0.083
Spring–mass	m	0.649	0.720 ± 0.127	0.717 ± 0.126
Spring–mass	k, m	0.767	0.767 ± 0.146	0.767 ± 0.146
Pendulum	L	0.738	0.817 ± 0.109	0.817 ± 0.103
Pendulum	g	0.793	0.879 ± 0.187	0.878 ± 0.189
Pendulum	g, L	0.713	0.713 ± 0.142	0.713 ± 0.142
Alien physics	k	0.839	0.935 ± 0.122	0.931 ± 0.121
Alien physics	m	0.875	0.928 ± 0.083	0.928 ± 0.086
Alien physics	k, m	0.659	0.659 ± 0.124	0.659 ± 0.124
Contact block	g	0.850	0.884 ± 0.153	0.893 ± 0.135
Contact block	μ	1.049	1.026 ± 0.246	1.023 ± 0.244
Contact block	μ, g	0.996	0.996 ± 0.468	0.996 ± 0.468
Contact ball	g	0.735	0.851 ± 0.138	0.846 ± 0.147
Contact ball	μ	0.997	1.025 ± 0.058	1.025 ± 0.059
Contact ball	μ, g	0.786	0.786 ± 0.231	0.786 ± 0.231
Contact cylinder	g	0.750	0.835 ± 0.187	0.828 ± 0.201
Contact cylinder	μ	0.861	0.909 ± 0.080	0.908 ± 0.081
Contact cylinder	μ, g	0.827	0.827 ± 0.231	0.827 ± 0.231

Table 15: Candidate comparison by contact scenario. Each number is the within-model ratio of gradient-updated to retained-code loss, averaged over three seeds (lower is better).

Scenario	Dynamics changed?	Modular	Context GRU	GRU+residual
No change	no	0.929	0.957	1.037
μ on rolling ball	no (proved)	0.929	0.957	1.037
g on rolling ball	yes	0.790	0.627	0.828
μ on sliding block	yes	0.984	1.025	0.981
Rolling/sliding crossing	yes	0.759	0.771	0.853

H Complete structure ablation

Table 16: One-step test MSE, averaged over three seeds. The residual GRU is capacity-matched; the residual columns add an identity shortcut to the named modular variant.

System	Persistence	Residual GRU	Modular	Dense graph	No graph	Mod.+res.	No-graph+res.
Free fall	0.2127	0.004046	0.005968	0.01147	0.01882	0.005851	0.01670
Pendulum	0.0006978	0.0002286	0.0009948	0.0008338	0.001197	0.0006996	0.0007003
Spring–mass	0.0007199	0.0002255	0.0005575	0.0004769	0.001045	0.0007213	0.0007218

The no-graph variant is an information-path ablation: deleting all cross-variable edges also prevents the parameter code from reaching the prediction. The dense-graph comparison isolates graph learning without that wiring confound. The identity shortcut is inconsistent across systems, whereas the residual GRU is uniformly strongest.

I Complete per-case results

This section reports the seed-level values behind the aggregate plots and the parameter counts produced by capacity matching. Every row is generated from the committed result files, so the counts and ratios can be recomputed with the accompanying scripts.

The taxonomy table gives the three values behind each scenario mean, and it is what the dispersion discussion in Section 3.3 refers to: one no-change seed degrades to 1.08 while another improves to 0.72.

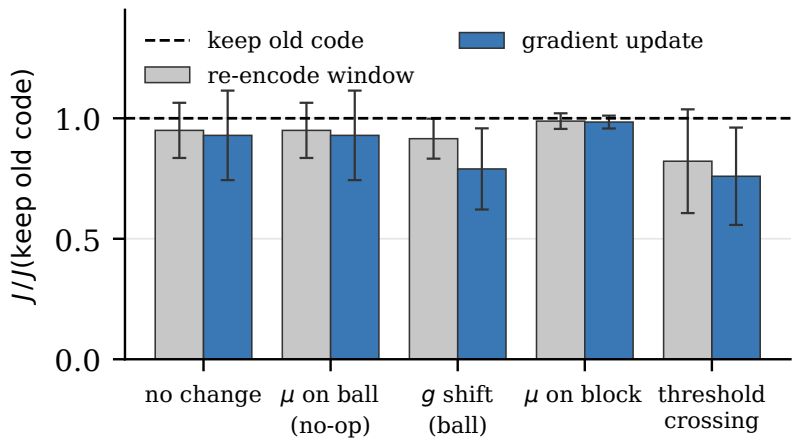

Figure 6: Original contact taxonomy. Bars are within-model post-change loss ratios and whiskers are sample SD over three seeds. The two leftmost controls are the same observation condition and are merged for the seed-level aggregate analysis.

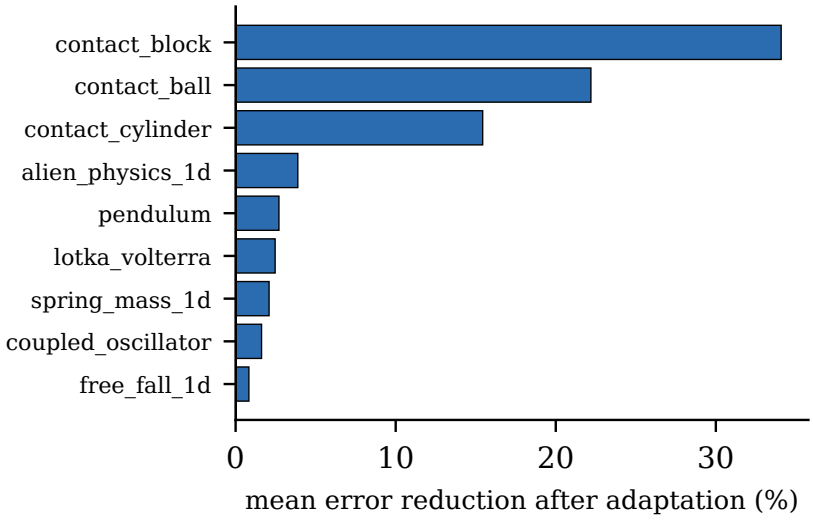

Figure 7: Original nine-system offline code-fitting study. Reductions are ratios of system-level mean errors across five seeds, distinct from the mean of per-run percentage reductions.

The parameter-count table records what capacity matching produced, so the reader can verify that the comparison between candidates was run at comparable model size rather than at comparable intent.

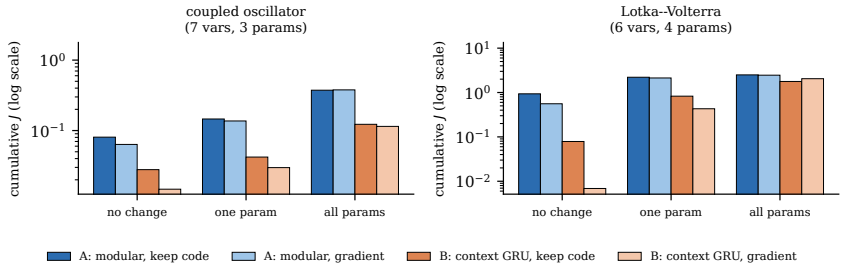

Figure 8: Original larger-system comparison: absolute cumulative losses, three seeds. Changed parameter counts and effective perturbation amplitudes covary in this experiment.

Table 17: Online taxonomy, candidate A. Cumulative post-change loss J relative to keeping the pre-change code, per seed. The μ-on-a-ball rows are bit-identical to the no-change rows, because the closed-form acceleration does not depend on μ.

Scenario	Seed	Re-encode	Gradient	Real change?
No change	42	1.003	0.986	no
No change	43	1.028	1.080	no
No change	44	0.818	0.722	no
Shift μ on a ball	42	1.003	0.986	no (provable)
Shift μ on a ball	43	1.028	1.080	no (provable)
Shift μ on a ball	44	0.818	0.722	no (provable)
Shift g on a ball	42	1.002	0.982	yes
Shift g on a ball	43	0.907	0.716	yes
Shift g on a ball	44	0.837	0.671	yes
Shift μ on a block	42	0.997	1.002	yes
Shift μ on a block	43	0.953	0.954	yes
Shift μ on a block	44	1.016	0.997	yes
Cross the rolling/sliding threshold	42	0.998	0.993	yes
Cross the rolling/sliding threshold	43	0.582	0.636	yes
Cross the rolling/sliding threshold	44	0.886	0.650	yes

Table 18: Higher-dimensional systems, per seed. J_f and J_g are absolute cumulative losses before and after gradient refinement; $r = J_g/J_f$.

		Modular			Context GRU		
System	Shift	J_f	J_g	r	J_f	J_g	r
Coupled	none	0.1268	0.1051	0.829	0.01955	0.01533	0.784
Coupled	none	0.05915	0.04508	0.762	0.03623	0.01348	0.372
Coupled	none	0.05590	0.04051	0.725	0.02785	0.01533	0.551
Coupled	one	0.1925	0.1854	0.963	0.03256	0.02476	0.760
Coupled	one	0.07481	0.06858	0.917	0.02826	0.01866	0.661
Coupled	one	0.1705	0.1564	0.917	0.06535	0.04595	0.703
Coupled	all	0.5807	0.6000	1.033	0.1429	0.1457	1.019
Coupled	all	0.1813	0.1882	1.038	0.07530	0.05250	0.697
Coupled	all	0.3597	0.3453	0.960	0.1497	0.1456	0.973
Lotka–Volterra	none	0.06331	0.05127	0.810	0.06236	0.005716	0.092
Lotka–Volterra	none	2.609	1.523	0.584	0.06521	0.004870	0.075
Lotka–Volterra	none	0.1311	0.1001	0.763	0.1088	0.01004	0.092
Lotka–Volterra	one	0.2712	0.2678	0.987	0.2730	0.1672	0.613
Lotka–Volterra	one	5.197	4.959	0.954	1.423	1.066	0.749
Lotka–Volterra	one	1.160	1.157	0.997	0.7863	0.05870	0.075
Lotka–Volterra	all	2.634	2.607	0.990	1.997	2.283	1.143
Lotka–Volterra	all	2.368	2.300	0.971	1.040	1.326	1.275
Lotka–Volterra	all	2.489	2.469	0.992	2.289	2.538	1.109

Table 19: Parameter counts. The first row covers spring–mass, pendulum and the three contact systems. Candidate B and the RSSM are matched to A before evaluation; candidate B was not run on free fall, so that entry is n/a.

System	Code dim	Modular (A)	Context GRU (B)	RSSM
Two-parameter systems	32	839,123	823,874	807,378
Free fall	16	769,820	n/a	732,642
Coupled oscillator	48	1,321,990	1,294,164	1,332,660
Lotka–Volterra	64	977,735	959,522	967,602

J Robustness to observation noise

The models are trained at the noise level used everywhere else ($\sigma = 0.01$) and evaluated at 0.05 and 0.10, so the evaluation noise lies outside the training condition. Each cell below averages ten seeds and three shift amplitudes.

Table 20: Matched-replay decomposition of the jointly trained context model on the pendulum as observation noise increases. Models are trained at $\sigma = 0.01$ and evaluated above it; lower is better, and negative values mean the update raised the loss.

Evaluation noise σ	Total gain	Control transfer	Change-specific
0.01	-0.2706	-0.0472	-0.2234
0.05	-0.0259	-0.0124	-0.0135
0.10	-0.0110	-0.0077	-0.0033

The signs and their significance do not change: total, control-transfer and change-specific gain are all negative with intervals that exclude zero at every level. The magnitude does, falling monotonically as noise rises, from -0.271 total and -0.223 change-specific at the training level to -0.026 and -0.013 at five times that noise, and to -0.011 and -0.003 at ten times it. More noise in the observations means the prediction error is increasingly set by the noise rather than by the code, leaving an update less room to move the loss in either direction. The decomposition keeps its reading under a shift in the observation channel, while the size of the effect tracks how much of the error the code can still explain.

K Comparison with a parameter-efficient adapter

Whether fitting a compact environment code is the best parameter-efficient option cannot be settled without an adapter baseline under the same online budget. The protocol accommodates one without modification, because matched replay only requires an adaptation state that can be swapped between probe streams: for a low-rank adapter that state is the factor it adapts.

We use a rank-1 increment on the first layer of the same network, with the other factor fixed at a small random value so that the increment starts at zero while its gradient does not. The adapter therefore carries 96 free parameters against 8 for the environment code, twelve times as many, and is trained under the same schedule.

Table 21: Change-specific gain for the compact code and for a rank-1 adapter that has twelve times as many free parameters, on the same systems, seeds and shift amplitudes. Higher is better.

System	Code (8 params)	Rank-1 adapter (96 params)
Rolling ball	$+0.479$	$+0.015$
Pendulum	-0.223	$+0.003$

The compact code is the stronger of the two on the rolling ball, where the code reaches $+0.479$ against $+0.015$ for the adapter: roughly 32 times as much value from one twelfth of the parameters. On the pendulum both mechanisms are close to neutral, and the adapter does not recover the negative change-specific reading of the code either. The comparison does not establish that code fitting is the

best parameter-efficient mechanism in general, since we test one adapter family, but it does show that on these systems the extra capacity buys nothing.

L SENSITIVITY TO THE UPDATE WINDOW AND LEARNING RATE

The deployment update uses a fixed window of ten steps and a learning rate of 0.01 everywhere in the paper. The reading could depend on those particular choices, so we vary each while holding the rest fixed: two shorter and longer windows, and two smaller and larger learning rates, on the pendulum at one shift amplitude with five seeds each. The default appears as the first row.

Table 22: Matched-replay decomposition of the jointly trained context model on the pendulum under five update settings. Both columns are change-specific gains, so higher is better; negative values mean the update raised the loss.

Window	Learning rate	Total gain	Change-specific
10	0.01	-0.261	-0.208
5	0.01	-0.320	-0.235
20	0.01	-0.181	-0.140
10	0.003	-0.222	-0.171
10	0.03	-0.309	-0.246

The reading is stable in sign and in ordering. Total and change-specific gain are negative at every one of the five settings, so refinement does not become beneficial when the window or the step size is tuned, and the change-specific component never exceeds zero. The magnitudes move by roughly a factor of two across the range, which is the same order as the seed-to-seed spread at a single setting, so the comparison between mechanisms is not being decided by these two constants. We also vary the number of gradient steps in Figure 3, which behaves the same way.

M REGULARISED SELECTIVE UPDATES

The main text notes that selecting designated code coordinates gives little advantage over a random equal-size subset. A natural objection is that the selection would matter if the update were regularised, so the rule was rerun with an ℓ_1 penalty on the code and with a proximal term pulling it toward its pre-change value, both at coefficient 0.01, on the same six systems, ten seeds and two change sizes.

Table 23: Effect of regularising the update. Entries are the change in normalised post-change loss relative to the unregularised rule, averaged over 18 system–change conditions; negative would mean the penalty helps.

Rule	Mean change	Median change	Conditions improved
$\ell_1 = 0.01$	$+0.158$	$+0.150$	6%
Proximal $= 0.01$	$+0.079$	$+0.072$	17%

The penalties make the outcome worse, not better: the mean change is $+0.158$ under ℓ_1 and $+0.079$ under the proximal term, and only 6% and 17% of the 18 conditions improve. Pulling the code toward zero or toward its starting point moves it away from the value that minimises the one-step loss, which is what the update is for.

The comparison that the objection concerns is unaffected. With the unregularised rule the gap between designated and random subsets averages $+0.0009$; with ℓ_1 it is $+0.0004$ and with the proximal term $+0.0005$. In all three the designated subset is better in fewer than half the conditions (17%, 44% and 33% respectively), so regularisation does not rescue the selection hypothesis – the subset choice simply does not carry the information the hypothesis assumes.

N A LOSS-COMPARISON CHANGE DETECTOR, SCORED OFFLINE

Deciding whether to update by comparing the loss under the kept code against the loss under a refitted code would remove the need to supply the change time. The replay records already contain both quantities, so the detector can be scored without a new run: for a real change the signal is the relative loss drop from refitting on that arm, and for the provable no-op it is the same quantity on the inert arm.

The signal separates the two populations but not cleanly. On the contact_ball records the real arm drops by +0.456 on average (range -0.481 to +0.792) while the inert arm drops by -0.004 (range -0.593 to +0.667), so the distributions overlap. Sweeping the decision threshold, the best joint error over the grid is 0.08 false positives and 0.02 false negatives. A detector of this kind is therefore not free: it either updates on some provable no-ops or misses some real changes.

We report this as a bound rather than a proposal. It is the same difficulty the main text identifies from the other direction, that a loss drop does not say what produced it, and it is why the paper supplies the change point and measures what the update is worth under it instead of claiming to detect change.

O ADDITIONAL MECHANISM FAMILIES

The main text states that the decomposition holds beyond the mechanisms it was built on. This appendix gives the supporting runs: a bottleneck adapter, LoRA at three ranks, and an amortised in-context baseline, each evaluated with the same matched-replay protocol, the same systems, the same ten seeds and the same three shift amplitudes.

Capacity matching. The comparison is meaningful only if the mechanisms are of comparable size. Three layers of width 96 on the two-dimensional systems give 10562 parameters for the joint context trunk and 10578 for the adapter, so those two are effectively capacity-matched. The in-context baseline has 15706 parameters, one and a half times the trunk; it was deliberately shrunk from an initial 78,218 so that a capacity advantage could not explain its behaviour. LoRA keeps a fixed 9794-parameter backbone and grows only its adaptation state, from 96 to 384 dimensions.

Change-specific gain. Higher is better, following the main text's convention that a positive S means a lower loss: a larger S means the mechanism is less harmed by the change, or better able to use it.

Table 24: Change-specific gain S for the additional mechanisms, averaged within seed and then over ten seeds. The adapter matches the joint context trunk in capacity yet lands in the same place; LoRA gains little even at rank four; the in-context baseline stays near zero on both systems.

System	Mechanism	S	SD	n
contact_ball	Joint context	$+0.479$	0.154	240
contact_ball	Bottleneck adapter	$+0.454$	0.157	240
contact_ball	LoRA rank 1	$+0.015$	0.025	240
contact_ball	LoRA rank 2	$+0.034$	0.038	240
contact_ball	LoRA rank 4	$+0.041$	0.051	240
contact_ball	In-context baseline	-0.035	0.095	240
pendulum	Joint context	-0.223	0.418	240
pendulum	Bottleneck adapter	-0.129	0.294	240
pendulum	LoRA rank 1	$+0.003$	0.014	240
pendulum	LoRA rank 2	$+0.005$	0.022	240
pendulum	LoRA rank 4	$+0.002$	0.031	240
pendulum	In-context baseline	-0.007	0.027	240

The decomposition is not an artefact of gradient fitting. The identity $G = C + S$ holds on every record of all three families: 0 violations in 960 adapter records, 0 in 1440 LoRA records and 0 in 960 in-context records. What differs between families is how far the change moves the code, not whether the split describes it. Read together with the amplitude sweep of Appendix P, this is

the sense in which the decomposition belongs to the measurement rather than to any one adaptation mechanism.

A limitation we state plainly. The in-context baseline was not tuned for this protocol. Its validation error is 0.000593 against 0.000232 for the joint context trunk, and training stops early, at a median best epoch of 10 against 60. It therefore measures what a generic amortised method does when read through this protocol, not what such a method achieves when tuned. Its near-zero S is best read as neither exploiting nor being harmed by this change, which is consistent with the main text's point that the two failure modes are distinct.

P WHERE THE BOUNDARY LIES, AND WHY IT LIES THERE

The main text characterises a boundary; it does not locate it or explain it. Two additions do both. The shift amplitude is varied across seven settings on both sides of no change for two systems and three mechanisms, and the same runs record how far apart the changed and unchanged branches leave the code.

The boundary in amplitude. Write $\delta = 1 - \text{factor}$, so $\delta = 0$ is no change, $\delta < 0$ weakens the parameter and $\delta > 0$ strengthens it. The change-specific gain is not a symmetric function of $|\delta|$. On the rolling ball all three mechanisms rise on both sides, so the boundary sits essentially at the origin. On the pendulum the two sides diverge: averaged over settings the weakened side gives $S = 0.227$ against 0.164 for the strengthened side, and the divergence is largest for the joint context model, whose weakened readings fall monotonically to 0.049 at $\delta = -0.5$ while its strengthened readings stay near zero. Which direction of change is harmful is therefore a property of the mechanism, not of the system.

Table 25: Change-specific gain S against shift amplitude. Columns run from the strongest weakening ($\delta = -0.5$) to the strongest strengthening ($\delta = +0.35$). The no-change setting $\delta = 0$ is excluded, because no change is not a reportable shift.

System	Method	-0.50	-0.30	-0.15	$+0.10$	$+0.20$	$+0.30$	$+0.35$
contact_ball	cavia	$+0.380$	$+0.357$	$+0.259$	$+0.097$	$+0.373$	$+0.501$	$+0.492$
contact_ball	coda	$+0.581$	$+0.538$	$+0.410$	$+0.187$	$+0.298$	$+0.516$	$+0.596$
contact_ball	context	$+0.606$	$+0.562$	$+0.421$	$+0.182$	$+0.291$	$+0.539$	$+0.635$
pendulum	cavia	-0.006	-0.006	-0.005	-0.000	-0.001	-0.003	-0.006
pendulum	coda	$+0.261$	$+0.226$	$+0.146$	$+0.062$	-0.057	-0.079	-0.074
pendulum	context	$+0.033$	$+0.051$	$+0.066$	$+0.047$	-0.140	-0.351	-0.579

Why the sign differs between mechanisms. If a mechanism never separates the two branches it has not parameterised the change, and S should not be positive; if it separates them while S is negative it responded in an unhelpful direction. Measuring separation as the distance between the two final codes relative to the pre-change code norm, the mechanisms differ by an order of magnitude on the same system: on the rolling ball the joint context code moves 17.1 code-lengths apart, on the pendulum only 1.30. Across the six system-mechanism pairs the separation correlates with S at $r = +0.64$, and the least-squares fit is $S \approx 0.020 \times \text{separation} + 0.135$.

Table 26: Branch separation in code space, relative to the pre-change code norm, against the change-specific gain, averaged over seeds and amplitudes.

System	Method	Separation	S
contact_ball	cavia	2.28	$+0.273$
contact_ball	coda	3.55	$+0.429$
contact_ball	context	17.11	$+0.443$
pendulum	cavia	1.99	-0.004
pendulum	coda	1.73	$+0.174$
pendulum	context	1.30	$+0.049$

A first-order reading. Both observations have one explanation. Linearising the query loss at the code the unchanged branch reaches, c^I, gives $J_R(c^R) \approx J_R(c^I) + \langle g_I,\, c^R - c^I \rangle$ with $g_I = \nabla J_R(c^I)$, so

$$S = J_R(c^I) - J_R(c^R) \approx -\langle g_I,\, c^R - c^I \rangle. \tag{6}$$

The change-specific gain is positive exactly when the displacement induced by the changed observations is negatively aligned with the query-loss gradient, that is, when those observations move the code down the problem's own loss surface, not simply that it moved. This makes the two regimes above two sides of one condition: a small $\|c^R - c^I\|$ leaves the inner product near zero and S near zero whatever the direction, while a large displacement with the wrong alignment makes S negative. It also explains why the sign is mechanism-dependent, because $c^R - c^I$ is produced by the mechanism's own update rule and need not follow the query gradient.

The reading is checkable, so we checked it. Over six (system, mechanism, amplitude) settings and 48 paired probes, S was measured directly and the right-hand side of Equation 6 formed from a central-difference gradient of the query loss at c^I. Across the settings the two agree in correlation at $r = +0.87$ and in sign on 90% of the individual pairs. We do not read (6) as an equality: the residual is largest where the displacement is largest, which is the expected behaviour of a first-order expansion, and the mean absolute difference between the two sides is 0.39 against gains of order one. What the check supports is the qualitative content the main text uses, that the sign of S is set by the alignment of the displacement with the query gradient.

Q Archived MuJoCo diagnostics and policy sensitivity

An earlier line used MuJoCo locomotion with a named friction scalar rather than a latent code. Friction changes at step 25 without a reset; five trajectories supply identification data and eleven disjoint trajectories supply evaluation data. The script chose the adaptation window from the supplied switch time and adapted regardless of detector output, with detector delay reported separately, so this line does not evaluate an autonomous trigger-to-adaptation loop. All reported quantities are world-model prediction errors, not policy returns.

Table 27: Archived rollout-error ratios (no-adaptation MSE divided by adapted MSE; larger is better) with the formal behaviour policy `ppo_42_final`, mean $\pm$ population standard deviation over three seeds.

Environment	Target μ	Estimated scalar	True scalar
Walker2d	4	0.79 ± 0.19	0.42 ± 0.12
Walker2d	6	0.92 ± 0.09	0.63 ± 0.17
HalfCheetah	4	1.46 ± 0.09	1.37 ± 0.11
HalfCheetah	6	2.03 ± 0.12	2.21 ± 0.07

A separately trained behaviour policy (`ppo_42`) changes the picture: five-seed HalfCheetah means at $\mu = 6$ are 1.019 for the estimated scalar and 0.974 for the true scalar, so the $2.03\times$ improvement is not reproduced across policy choices and the two lines are never pooled. Training-support sweeps over the domain-randomisation upper bound behave differently per environment, with rank correlation -0.96 for Walker2d over seven upper bounds and -0.54 for HalfCheetah with its optimum at the support boundary, rather than following a single monotonic law. Injecting the true scalar is a conditioning-channel diagnostic, because the model was trained with that named input; it cannot localise an identification bottleneck for an unsupervised latent coordinate.

R Training and evaluation configuration

Model and training. Per-variable latent dimension 16; dynamics GRU width 64 per variable; 60 epochs; 64 training trajectories of 100 frames plus 16 validation trajectories; batch size 128; 8-frame windows with stride 4; AdamW with learning rate 3×10^{-4} and weight decay 10^{-5}; gradient clipping at 1.0. The best validation checkpoint is restored together with the graph temperature in force at selection, and a runtime check re-evaluates the restored model and must reproduce the selected validation error.

Adaptation. Offline, five 41-frame probe trajectories initialise and fit the code for 50 Adam steps at learning rate 10^{-2}, initialised from zero, and eleven disjoint 100-frame query trajectories score it. Online, the same fit is re-applied every five steps over a sliding window of ten frames, and the shared weights are asserted unchanged across adaptation.

Candidate B and capacity matching. The context GRU uses a context vector of the same width as the modular code (16 dimensions per parameter slot) and a hidden width chosen from a grid so that its parameter count is as close as possible to candidate A's before any evaluation number is seen.

Change-taxonomy scenarios. The contact simulator is used because its acceleration is available in closed form and permits a provable no-op.

Table 28: Scenario parameters ($\theta = 0.3$ rad throughout).

Scenario	System	Source	Target	Dynamics changes?
No change	`contact_ball`	$\mu = 0.30, g = 9.5$	identical	no
μ only	`contact_ball`	$\mu = 0.30, g = 9.5$	$\mu = 1.80, g = 9.5$	no (provable)
g shift	`contact_ball`	$\mu = 0.30, g = 9.5$	$g = 12.0$	yes
μ on a block	`contact_block`	$\mu = 0.30, g = 9.5$	$\mu = 1.80, g = 9.5$	yes
Threshold crossing	`contact_cylinder`	$\mu = 0.50, g = 9.5$	$\mu = 0.05, g = 9.5$	yes (regime)

Domain-randomisation support sweep. Two agents, seven upper bounds on the training friction range, five seeds, target $\mu = 4$. Identification error falls once the target enters the training support and is not monotone in the support width, with HalfCheetah reaching its optimum exactly at the boundary.

Table 29: Domain-randomisation support sweep for the archived MuJoCo diagnostic. Values are mean absolute scalar-identification error over five seeds; lower is better.

Environment	Support bound	Mean absolute error	Std
Walker2d	2	0.801	0.742
Walker2d	2.5	0.988	0.823
Walker2d	3	0.421	0.549
Walker2d	3.5	0.364	0.448
Walker2d	4	0.362	0.310
Walker2d	5	0.276	0.249
Walker2d	6	0.251	0.263
HalfCheetah	2	1.958	1.019
HalfCheetah	2.5	0.992	0.910
HalfCheetah	3	0.788	0.579
HalfCheetah	3.5	0.551	0.488
HalfCheetah	4	0.413	0.351
HalfCheetah	5	0.600	0.365
HalfCheetah	6	0.834	0.597

Metric. For every step after $t_{\text{change}} = 30$ the model first emits a prediction and only then consumes the observation, so the cumulative loss J charges the transient. All ratios are within-seed paired comparisons against the same model keeping its pre-change code.

Table 30: Configuration for the original modular/context-GRU experiments. The new CAVIA, CoDA and joint-context replay comparison uses the separate configuration in Appendix A.

Setting	Value
Latent dimension per variable	16
Dynamics GRU width per variable	64
Training epochs	60
Training data	64 trajectories of 100 frames
Validation data	16 trajectories of 100 frames (different seeds)
Batch size	128
Window / stride	8 / 4 frames
Optimiser	AdamW, lr 3×10^{-4}, weight decay 10^{-5}
Gradient clipping	1.0
Graph temperature schedule	$3.0 \rightarrow 0.4$, linear in epoch
DAG penalty / sparsity penalty / KL weight	0.1 / 0.05 / 0.1
Reporting seeds	42–44 (three) or 42–46 (five)
Probe trajectories	5 of 41 frames, disjoint from query
Query trajectories	11 of 100 frames
Code fitting	50 Adam steps, lr 10^{-2}, zero initialisation
Online update interval / window	every 5 steps / last 10 frames
Change point	step 30 of a 60-frame episode
Candidate B hidden width	448 (two-parameter systems), matched by search otherwise

S DEPLOYMENT-TIME ADAPTATION PROCEDURE

The following procedure describes the predict–then–update schedule used by the original online experiments. It makes explicit which information is available at each point and which parameters are allowed to move. The trigger is supplied by the evaluation protocol; learning an autonomous change detector is outside the scope of the reported results.

Procedure 1: masked environment-code refinement.

Inputs: frozen predictor parameters θ; retained code c; coordinate mask m; update interval $U = 5$; history length $W = 10$; optimisation steps $K = 50$; learning rate $\eta = 10^{-2}$.

1. At time t, emit $\hat{x}_{t+1} = f_\theta(x_{\le t}, a_{\le t}, c)$ before observing x_{t+1}, and add its error to the cumulative post-change loss.
2. If t is not a scheduled update time, retain c and continue. Otherwise use only observations through x_t; the target x_{t+1} just scored is not included in this update.
3. Form the most recent window $\mathcal{W}_t$ of at most W observations and initialise $c^{(0)} \leftarrow c$.
4. For $k = 0, \dots, K-1$, evaluate the one-step prediction loss on $\mathcal{W}_t$ and take an Adam step only in the permitted coordinates: $c^{(k+1)} \leftarrow c^{(k)} - \eta\, m \odot \mathrm{Adam}(\nabla_c \mathcal{L}_{\mathcal{W}_t}(\theta, c^{(k)}))$.
5. Set $c \leftarrow c^{(K)}$. The shared parameters θ are never updated at deployment. The next prediction therefore measures the transient benefit or harm of the fit.

The retained-code control sets $K = 0$. The all-coordinate, designated-coordinate and random equal-size conditions differ only in m; data, initialisation and optimisation budget are matched.