SYMBOL AND RULE DISCOVERY IN CONTINUOUS CONTROL THROUGH PROGRESSIVE INDUCTIVE ABDUCTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the discovery of symbols and rules from continuous-control trajectories, preserving characteristics of motion dynamics while supporting discrete reasoning. Jointly learning continuous factors, value boundaries, and temporal rules creates a large and coupled optimization problem. We address this problem by *DynaAbd*, a multi-stage progressive inductive abductive learning (inductive ABL) approach. In the first two stages, contrastive spectral initialization and one-step dynamics modeling organize continuous motion into cyclic and dependent factors, capturing initial features in motion dynamics. In the final stage, we introduce an inductive ABL strategy for fine-grained dynamics modeling. Logical induction and abduction refines the symbol discovery and builds the corresponding rules based on multi-step logical consistency. Alternating gradient optimization fine-tunes the symbol value boundaries. Our method is experimentally evaluated in polar motion systems and MuJoCo locomotion tasks. DynaAbd discovers latent driving and dependent symbols and motion rules whose auxiliary supervision can accelerate early-stage policy learning in the original environment. Our code and data will be open-sourced upon formal release.

1 INTRODUCTION

Continuous-control policies generate motion as sequences of continuous states and actions. They can also be formulated at the level of symbolic abstractions that describe stages of movement and other regularities. For example, the body rises during one part of a gait and falls during another. Extracting such symbols and rules would connect continuous experience with higher-level reasoning (Konidaris et al., 2018) and make learned behavior easier to interpret. They can also provide auxiliary supervision to accelerate early-stage policy learning in the original environment.

However, these symbolic abstractions usually hide inside complicated behavior dynamics, whose discovery is a challenging problem. The main difficulty is that continuous factors, symbol grounding, and rules must be learned together. From a randomly initialized representation, the learner must build the plausible symbol space, search the space to find the symbols, decide their discrete symbol values, and find the rule programs through induction simultaneously, leaving a vast coupled optimization space with little initial structure to guide optimization. This entanglement is a recognized bottleneck in joint perception and rule induction: Quality prediction could be misled by incidental correlations (Geirhos et al., 2020), and latent semantic factors cannot be recovered without introducing appropriate inductive biases (Locatello et al., 2019). In continuous control, the necessity to handle long-term dynamics results in further ambiguity: Coupled and noisy transitions make the trajectories generated by the same policy highly varied, leading to further difficulty in capturing the latent symbols from behavior data.

In this paper, we study how to tackle the above challenges, focusing on cyclic motion behaviors, such as locomotion. We propose *DynaAbd*, a progressive inductive abductive learning (inductive ABL) (Dai & Muggleton, 2021; Peng et al., 2025a) approach for continuous control. The idea is to build a multi-stage procedure to gradually disentangle symbols through key information behind motion dynamics and trajectory quality discrimination, meanwhile induce the rules as the symbols emerge. In the first two stages, we propose an initialization approach to narrow the effective search space

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.

before inductive ABL, making symbols to emerge meanwhile retain continuous coordinates and an explicit interface to discrete intervals. The first stage introduces contrastive extended dynamic mode decomposition (EDMD) (Williams et al., 2015) and spectral analysis, providing initial cyclic and dependent coordinates in low-dimensional subspaces. The second stage models one-step dynamics, i.e. differentiable one-step prediction and quality discrimination to refine these coordinates under soft motion constraints. Ordered partitions connect the resulting factors to symbols. The obtained spectral coordinates thus provide a motion-informed starting point, while the final factors, grounding boundaries, and rules are learned through subsequent refinement and program selection. In the final stage, we propose an inductive ABL strategy for fine-grained dynamics modeling. Logical induction and abduction are applied to refine the symbol discovery and build the corresponding rules based on multi-step logical consistency, which explains phase progression and phase-dependent changes. Alternating gradient optimization fine-tunes the symbol value boundaries, obtaining fine-grained constraints on continuous motion within this structure. Through this multi-stage approach, the original large and coupled optimization is decomposed into solvable subproblems under a coarse-to-fine routine.

Our method is experimentally evaluated in simulated polar motion systems with known ground-truth symbols and rules, as well as MuJoCo locomotion tasks (Todorov et al., 2012). DynaAbd discovers latent driving and dependent symbols and motion rules whose auxiliary supervision can accelerate early-stage policy learning in the original environment.

2 Related Work

Several approaches discover structure from continuous dynamics. EDMD approximates Koopman evolution (Williams et al., 2015), sparse identification of nonlinear dynamics (SINDy) identifies sparse equations (Brunton et al., 2016), and SINDy autoencoders (SINDy-AE) jointly learn coordinates and equations (Champion et al., 2019). Mechanism sparsity jointly learns latent factors and sparse temporal dependencies (Lachapelle et al., 2022). Task-oriented Koopman control learns an embedding, evolution operator, and controller using task cost (Lyu et al., 2023). These approaches provide continuous coordinates, equations, or dependency graphs; discrete phase predicates and temporal logic programs require an additional symbol and rule discovery mechanism.

Symbolic abstractions and programs can be learned using skill models and logical knowledge. From Skills to Symbols learns planning abstractions from the initiation and outcome models of available skills in continuous environments (Konidaris et al., 2018). Semantic loss turns supplied formulas into differentiable supervision (Xu et al., 2018). Popper learns recursive programs from positive and negative logical examples, pruning through failure-derived constraints (Cropper & Morel, 2021). These methods use skill models, specified formulas, or logical examples. For quality-labelled motion without skill annotations or observed predicates, learning the continuous factors and phase-dependent predicates that supply this interface remains an additional problem.

Abductive approaches offer a direct route to joint grounding and rule learning. Abductive Learning infers facts and clauses to improve perception and reasoning together (Dai et al., 2019); MetaAbd learns neural perception and recursive programs using probability-guided search (Dai & Muggleton, 2021). These frameworks can accommodate learned motion predicates in principle. The difficulty is optimization: MetaAbd reports local optima and improved learning from weak perceptual pretraining. Reasoning-shortcut analyses also identify unintended intermediate meanings and consistency objectives that reduce this risk (Yang et al., 2024). In continuous motion, discrete consistency alone leaves another ambiguity: groundings with the same logical outcomes can differ in their continuous transition consistency. Adapting joint abductive learning therefore calls for motion-informed initialization and an additional criterion for selecting groundings according to their within-symbol dynamics.

Trajectory structure also supports policy learning. ConSpec learns interpretable critical-state prototypes and intrinsic rewards from successful and unsuccessful episodes (Sun et al., 2023). Adversarial Motion Priors (AMP) combines learned transition-based motion rewards with task reward (Peng et al., 2021). Their outputs support credit assignment or imitation rather than jointly discovered temporal programs and continuous groundings; explaining which phase-dependent relation is satisfied or violated requires an additional rule-discovery component.

Our intermediate layer connects dynamics-based representation learning to abductive rule discovery. Soft motion constraints initialize temporally organized factors; coarse-to-fine feedback jointly refines their coordinates, symbol boundaries, and motion relations using discrete consistency and within-symbol dynamics. This structure provides auxiliary supervision for faster early-stage policy learning in the original environment.

3 Problem Formulation

We observe N quality-labeled episodes $\mathcal{D} = \{(\tau_e, y_e)\}_{e=1}^{N}$, where e indexes episodes. Episode $\tau_e = (o_{e,0}, \ldots, o_{e,T_e})$ contains T_e transitions; $o_{e,t} \in \mathbb{R}^{d_o}$ is its d_o-dimensional observation at time t. The label $y_e \in \{+, -\}$ denotes a higher- or lower-quality source policy, not local transition correctness. Discovery uses observations, temporal adjacency, and episode labels. Multiple sources are pooled without environment identifiers. We omit the episode index when clear; Δ denotes a one-step difference, using signed circular differences for angles.

We develop and validate the method for tasks with recurring motion cycles, such as locomotion. To express phase-dependent motion, we separate cycle progress from an associated scalar change. The cyclic Driver $z_{\mathrm{dr}}(o) \in \mathbb{S}^1$ lies on the two-dimensional unit circle; the scalar Dependent is $z_{\mathrm{de}}(o) \in \mathbb{R}$. Subscripts dr and de consistently denote these two roles. The Driver angle $\theta = \mathrm{atan2}([z_{\mathrm{dr}}]_2, [z_{\mathrm{dr}}]_1)$ and the Dependent are partitioned into phase k and ordinal code j. We learn

$$\mathcal{M} = (F_\vartheta, \boldsymbol{\beta}, H, f_A), \qquad F_\vartheta(o) = (z_{\mathrm{dr}}(o), z_{\mathrm{de}}(o)), \qquad g_{\boldsymbol{\beta}}(F_\vartheta(o)) = (k, j). \tag{1}$$

Here F_ϑ is the continuous factor map with readout parameters ϑ; $g_{\boldsymbol{\beta}}$ is the grounder with interval boundaries $\boldsymbol{\beta}$; H is a shared discrete program; and f_A is a continuous motion model with parameters ψ_A. Adjacent phases may share a change direction. Known cyclic systems assess recovery, and locomotion experiments assess control utility. Appendix D collects the notation.

4 Method

Given quality-labelled trajectories $\mathcal{D}$, DynaAbd learns $\mathcal{M}$ through three stages of progressive inductive ABL (Figure 1). Spectral initialization uses EDMD to find candidate factor spaces and initial projections, reducing the risk that subsequent dynamics optimization converges to poor local optima. One-step dynamics modeling then corrects the factors' dynamical properties: Driver predicts its own progression, and Dependent predicts its changes conditioned on Driver. Together, EDMD-based spectral initialization and one-step dynamics modeling construct the neural–symbolic intermediate layer, whose continuous factors are partitioned into intervals for logical reasoning. Finally, abduction on positive trajectory windows learns rules and refines the factor projections and interval boundaries, with continuous motion fitting supervising changes within the intervals.

4.1 Contrastive Spectral Initialization

We adapt EDMD (Williams et al., 2015) to initialize F_ϑ in restricted candidate spaces. We use a dictionary $\phi(o) \in \mathbb{R}^{d_\phi}$ of linear and quadratic terms of normalized observations. Our role-specific criteria construct fourteen-dimensional bases $B_{\mathrm{dr}}, B_{\mathrm{de}} \in \mathbb{R}^{d_\phi \times 14}$ for subsequent dynamics modeling and inductive ABL.

Driver initialization. Using positive trajectories, spectral selection favors predictable evolution while penalizing prediction errors and rapid changes. Within the resulting space, a complex eigenvector defines a complex-valued function of the observation; its real and imaginary parts initialize a unit-circle Driver. Selection favors one revolution between returns to similar observations.

Dependent initialization. Given Driver, sine/cosine features of its phase angle predict dictionary increments separately by quality class. Subtracting coefficients fitted with time-shifted phase-angle features, while keeping increments fixed, gives a corrected response. Spectral selection favors a strong positive response and positive–negative response differences, while penalizing linear redundancy with Driver. The resulting space supplies an initial scalar Dependent projection.

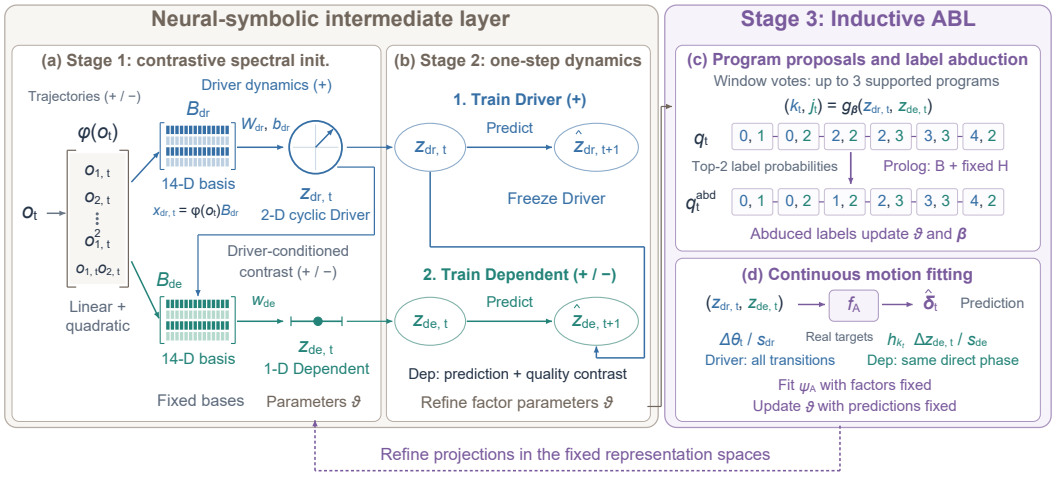


Figure 1: DynaAbd's three stages. (a) Contrastive spectral initialization constructs candidate spaces and initial factor projections. (b) One-step dynamics modeling refines Driver, then Dependent with Driver fixed. Together, (a) and (b) form the neural–symbolic intermediate layer. Inductive ABL combines (c) program proposals and label abduction with (d) continuous motion fitting. Each candidate program is refined in an independent candidate model. Both candidate bases remain fixed after initialization; the displayed labels are illustrative.

Initialized factor mappings. With fixed preprocessing included in $x_{\mathrm{dr}}, x_{\mathrm{de}}$,

$$x_{\mathrm{dr}} = \phi(o)B_{\mathrm{dr}}, \qquad x_{\mathrm{de}} = \phi(o)B_{\mathrm{de}},$$
$$z_{\mathrm{dr}} = \frac{x_{\mathrm{dr}}W_{\mathrm{dr}} + b_{\mathrm{dr}}}{\|x_{\mathrm{dr}}W_{\mathrm{dr}} + b_{\mathrm{dr}}\|_2}, \qquad z_{\mathrm{de}} = \frac{x_{\mathrm{de}}w_{\mathrm{de}}}{\sqrt{w_{\mathrm{de}}^{\top}C_{\mathrm{de},+}w_{\mathrm{de}}}}. \tag{2}$$

Here $C_{\mathrm{de},+}$ is the fixed within-episode covariance of Dependent features on positive trajectories. Subsequent stages update the factor-projection parameters $\vartheta = (W_{\mathrm{dr}}, b_{\mathrm{dr}}, w_{\mathrm{de}})$. Appendix A.1 details the objectives, closed-form subproblems, and refinements.

4.2 One-Step Dynamics Modeling

The second stage corrects the initialized factors' dynamical properties, providing a structured starting point for inductive ABL. It takes the initial F_ϑ and real one-step transitions and returns a refined factor map. We first refine Driver using positive trajectories, then freeze it and refine Dependent using both quality classes. Prediction accuracy, the gain from conditioning on Driver, and quality discrimination guide this refinement.

Motion prediction. We want Driver to predict its next value and Dependent to be predictable from Driver and its current value. At each update, a fitting bank of real transitions supplies the predictor fit, while a separate query batch measures prediction error. The next-value predictors $f_{D,\mathrm{dr}}$ and $f_{D,\mathrm{de}}$ use ridge regression; the Dependent predictor includes Fourier features of the Driver phase angle and their interactions with Dependent. The query-batch mean squared error (MSE), denoted $\mathrm{MSE}_{D,\nu}$, is normalized using a fixed scale based on the factor's changes at initialization. Keeping this scale fixed makes errors comparable as the factor projection changes. An additional $+$ restricts the loss to positive transitions:

$$L_{\mathrm{self}} = \mathrm{MSE}_{D,\mathrm{dr},+}(f_{D,\mathrm{dr}}(z_{\mathrm{dr},t}), z_{\mathrm{dr},t+1}),$$
$$L_{\mathrm{pred}} = \mathrm{MSE}_{D,\mathrm{de}}(f_{D,\mathrm{de}}(z_{\mathrm{dr},t}, z_{\mathrm{de},t}), z_{\mathrm{de},t+1}). \tag{3}$$

The fitted regression coefficients depend on the current factors. Differentiating the query-batch error through the regression solution updates the factor projections, allowing prediction accuracy to guide representation learning (Appendix A.2).

Driver-conditioned gain. To encourage the Dependent mapping to make use of Driver, compare conditional prediction with a Dependent-only predictor $f_{D,\mathrm{de},0}$:

$$\begin{aligned} L_{\mathrm{de}0} &= \mathrm{MSE}_{D,\mathrm{de}}(f_{D,\mathrm{de},0}(z_{\mathrm{de},t}), z_{\mathrm{de},t+1}), \\ L_{\mathrm{gain}} &= \frac{L_{\mathrm{pred}} + 0.02}{\mathrm{stopgrad}(L_{\mathrm{de}0}) + 0.02}. \end{aligned} \tag{4}$$

The ratio rewards a lower prediction error when Driver is available. The operator $\mathrm{stopgrad}$ holds its argument fixed during differentiation, making the denominator a fixed comparison value.

Quality discrimination. To retain quality-related motion differences, we separately fit a Gaussian distribution of Dependent increments for each class, conditioned on the current Driver and Dependent. These class-specific response models estimate both the expected increment and its variance, using the same fixed prediction-error normalization. Their likelihoods compare the two quality classes. Let $p_{\mathrm{resp},y,t}$ be the density assigned by class y to the real increment $\Delta z_{\mathrm{de},t}$ given $(z_{\mathrm{dr},t}, z_{\mathrm{de},t})$. With equal class priors, the positive-class probability and balanced classification loss are

$$\begin{aligned} p_{\mathrm{qual},t} &= \frac{p_{\mathrm{resp},+,t}}{p_{\mathrm{resp},+,t} + p_{\mathrm{resp},-,t}}, \\ L_{\mathrm{class}} &= -\frac{\mathbb{E}_+ \log p_{\mathrm{qual},t} + \mathbb{E}_- \log(1 - p_{\mathrm{qual},t})}{2 \log 2}. \end{aligned} \tag{5}$$

Here $\mathbb{E}_y$ averages query transitions from class y. This term updates Dependent; Driver supplies a fixed condition during that update.

Regularization and optimization. Prediction alone can favor nearly constant factors. Driver regularization maintains variation in both coordinates and anchors the mean squared length of its pre-normalization vector to its initial value. Dependent regularization discourages vanishing increments and large departures from its initial projection. Sparsity penalties favor fewer dictionary features. Denote these regularizers by $L_{\mathrm{reg,dr}}$ and $L_{\mathrm{reg,de}}$; their definitions and weights are in Appendix A.2. Let ℓ count Dependent updates and $\xi_\ell = \min(1, \ell/200)$ gradually increase the classification and gain weights. The objectives are

$$\begin{aligned} L_{D,\mathrm{dr}} &= 0.1 L_{\mathrm{self}} + L_{\mathrm{reg,dr}}, \\ L_{D,\mathrm{de}} &= 0.1 L_{\mathrm{pred}} + 0.5 \xi_\ell L_{\mathrm{class}} + 0.3 \xi_\ell L_{\mathrm{gain}} + L_{\mathrm{reg,de}}. \end{aligned} \tag{6}$$

The Driver objective updates $(W_{\mathrm{dr}}, b_{\mathrm{dr}})$; the subsequent Dependent objective updates w_{de} with Driver fixed. These local objectives organize continuous motion but leave the discrete explanations and shared programs undetermined. The next stage uses logical consistency across a window to learn that structure. Predictor fitting and optimization details are specified in Appendix A.2.

4.3 Inductive ABL for Fine-Grained Dynamics Modeling

The final stage learns $\mathcal{M} = (F_\vartheta, \boldsymbol{\beta}, H, f_A)$ from the factors refined by one-step dynamics modeling. Five circular Driver intervals and five ordered Dependent intervals define direct labels $q_t = (k_t, j_t) = g_{\boldsymbol{\beta}}(F_\vartheta(o_t))$, with trainable boundaries. Each case is a window from one episode. Following abductive learning and AbdGen (Dai et al., 2019; Peng et al., 2025b), label abduction uses positive windows to infer assignments that satisfy a program; these are the abduced labels.

We first propose candidate programs from positive windows. Each initializes an independent model from the same checkpoint, with its program held fixed while label abduction, motion fitting, and factor/boundary updates alternate. Finally, we calibrate each model's acceptance thresholds on positive training transitions and select the complete model with the highest validation score.

Rule induction and label abduction. Background knowledge $\mathcal{B}$ allows phase labels to stay or advance by one modulo five. Dependent labels stay or move by one in the starting phase's direction within $\{0, \ldots, 4\}$. A positive explanation must visit every phase and include strict Dependent increases and decreases. Equality between the initial and final states is optional. A program $H = (h_0, \ldots, h_4)$ assigns direction $h_k \in \{-1, +1\}$ to each phase. For example, $h_0 = +1$ gives:

```
f(A,B) :- motion(slot0,slot1,0,increase,A,C), f(C,B).
f(A,B) :- terminate(A,B).
```

Here slot0/slot1 denote Driver/Dependent; A/C are state lists before/after one transition, and B is the target remainder, empty for a complete window. The predicate motion checks the starting phase and label changes, including across phases; recursion checks the remainder, and terminate checks completion and window constraints.

Interval probabilities use a softmax of negative squared distances, circular for Driver and linear for Dependent. Retaining and renormalizing the two most probable labels per factor gives $\widetilde{p}_{\mathrm{dr},t}$ and $\widetilde{p}_{\mathrm{de},t}$. Write $\mathbf{q}^{\mathrm{abd}} = (q_t^{\mathrm{abd}})_t$ for a window's ordered abduced pairs, with $q_t^{\mathrm{abd}} = (k_t^{\mathrm{abd}}, j_t^{\mathrm{abd}})$. At proposal time, factors and boundaries are fixed, and Prolog scores feasible program–label explanations by

$$\begin{aligned} \max_{H,\mathbf{q}^{\mathrm{abd}}} \quad & S_{\mathrm{abd}}(H, \mathbf{q}^{\mathrm{abd}}) = \sum_t \left[\log \widetilde{p}_{\mathrm{dr},t}(k_t^{\mathrm{abd}}) + \log \widetilde{p}_{\mathrm{de},t}(j_t^{\mathrm{abd}})\right] + \log P(H) \\ \text{subject to} \quad & \mathbf{q}^{\mathrm{abd}} \text{ explains the positive window under } \mathcal{B} \text{ and } H. \end{aligned} \tag{7}$$

For each positive window, abduction returns its best feasible program and adds one vote to that program. We retain up to three highest-vote programs as candidates. Each candidate is then refined with its program fixed. Search details are given in Appendix A.3.

Grounding refinement under a candidate program. The inferred labels q_t^{abd} are held fixed during continuous updates. Interval-distance losses move factor values into their assigned intervals. Direction losses encourage forward Driver progress and real Dependent increments in direction $h_{k_t^{\mathrm{abd}}}$. Together with penalties limiting collapse, boundary drift, and dense dictionary combinations, they form L_A, updating ϑ and $\boldsymbol{\beta}$. Updated factors and intervals change the next query's probabilities.

Continuous motion fitting. Abduced labels constrain interval membership and direction; fine-grained fitting supervises continuous change amounts within discrete intervals. We first fix the factors and fit the motion model to real increments; we then fix prediction values and adjust factor projections so their increments agree better. Window-level logic spans multiple steps, while continuous fitting uses adjacent observations. For factors $\mathbf{z}_t = (z_{\mathrm{dr},t}, z_{\mathrm{de},t})$, signed shortest phase-angle change $\Delta\theta_t$, and fixed positive-data scales $s_{\mathrm{dr}}, s_{\mathrm{de}}$,

$$\begin{aligned} \widehat{\boldsymbol{\delta}}_t = f_A(\mathbf{z}_t) \in \mathbb{R}^2_{>0}, \qquad \boldsymbol{\delta}_t = \left(\frac{\Delta\theta_t}{s_{\mathrm{dr}}},\ \frac{h_{k_t}\Delta z_{\mathrm{de},t}}{s_{\mathrm{de}}}\right), \\ m_{\mathrm{fit},t,\mathrm{dr}} = 1, \qquad m_{\mathrm{fit},t,\mathrm{de}} = \mathbf{1}[k_t = k_{t+1}]. \end{aligned} \tag{8}$$

The direct phase k_t determines fitting directions and masks. Driver uses all real positive transitions; Dependent magnitude uses only same-phase transitions. Positive predictions encode forward Driver progress and rule-directed Dependent change; violating transitions retain signed targets. Direction constraints in L_A apply across phases.

Motion fitting uses all sampled positive windows and updates only ψ_A, with factor inputs and real-increment targets fixed. Representation updates require abduced labels. Stopping gradients through the entire prediction with sg gives

$$L_{\mathrm{feedback}} = \frac{\sum_{t,\nu} m_{\mathrm{fit},t,\nu}\left(\mathrm{sg}(\widehat{\delta}_{t,\nu}) - \delta_{t,\nu}\right)^2}{\sum_{t,\nu} m_{\mathrm{fit},t,\nu}}, \qquad L_{\mathrm{repr}} = L_A + \lambda_{\mathrm{fit}} L_{\mathrm{feedback}}, \tag{9}$$

Here $\nu \in \{\mathrm{dr}, \mathrm{de}\}$ indexes scalar components $\delta_{t,\nu} = [\boldsymbol{\delta}_t]_\nu$ and $\widehat{\delta}_{t,\nu} = [\widehat{\boldsymbol{\delta}}_t]_\nu$. Feedback updates factor projections through the real increments in $\boldsymbol{\delta}$; L_A also updates boundaries. Direct labels and masks are fixed within each update. Revised factors change label probabilities and thereby inform the next abduction query.

Shared-program selection. Validation compares each refined $\mathcal{M}_H$ using its own factor map, boundaries, motion model, and calibrated thresholds. We measure the positive acceptance rate (PA) and negative rejection rate (NR) on real transitions with direct labels, and score each model by $S_{\mathrm{val}}(\mathcal{M}_H) = 0.75\mathrm{PA} + 0.25\mathrm{NR}$. Acceptance requires discrete legality, continuous direction agreement, and calibrated fitting-residual checks; we call this criterion *Coarse+Fine*. We return the highest-scoring complete model and its fixed scales and thresholds.

5 EXPERIMENTS

We evaluate factor and rule recovery, followed by short-budget policy retraining in the original environment.

5.1 FACTOR AND RULE RECOVERY IN CONTROLLED CYCLIC ENVIRONMENTS

Setup and purpose. PolarCycle comprises synthetic continuous-control environments with known cyclic factors and rules (Figure 2). A phase angle $\theta^\star$ locates cycle position; a radial factor $R^\star$ increases or decreases according to phase. Polar3, Polar4, and Polar5 test recovery of these factors, boundaries, and directions over five seeds. Methods share observations and episode-level partitions; training and model selection use only observations and quality labels. Appendix B specifies the datasets and baseline configurations.

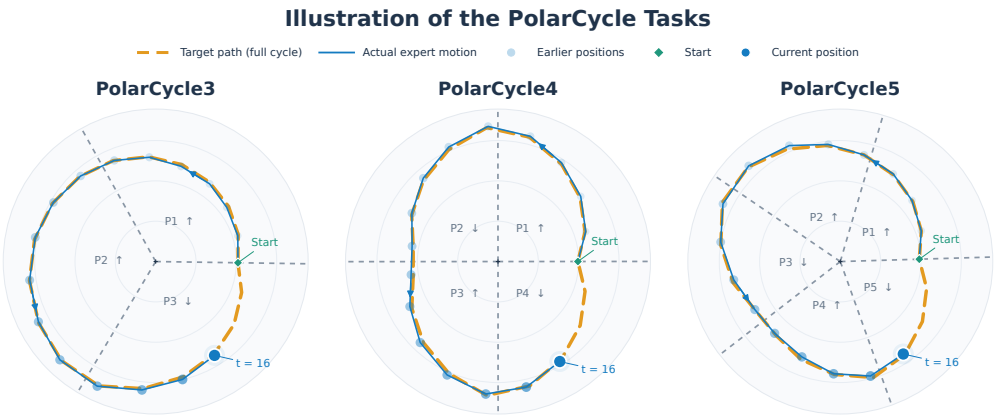


Figure 2: PolarCycle tasks with three, four, and five underlying motion phases. Dashed orange paths show the target cycle and blue traces show expert motion. Radial dashed lines delimit phases; arrows indicate increasing or decreasing radius. Discovery uses observations without these physical phase labels or boundaries.

Factor correspondence. Driver identifies position within a cycle, so we compare phase-angle values. Dependent describes how a quantity changes during motion, so we compare its real consecutive increments:

$$\rho_{\mathrm{dr}} = \left|\mathrm{Pearson}(\theta, \theta^\star)\right|, \qquad \rho_{\mathrm{de}} = \left|\mathrm{Pearson}(\Delta z_{\mathrm{de}}, \Delta R^\star)\right|. \tag{10}$$

Absolute correlations accommodate opposite orientations.

Comparisons. Table 1 compares autoencoders with sparse identification of nonlinear dynamics (SINDy-AE) (Champion et al., 2019) and mechanism sparsity (Lachapelle et al., 2022) with DynaAbd. Internal ablations remove one-step dynamics modeling (E+A), replace spectral initialization with a vector-quantized variational autoencoder (VQ-VAE; VQ+D+A) (van den Oord et al., 2017), or use random initialization followed by dynamics modeling and abductive learning (Random D+A) or abductive learning alone (Random A). DynaAbd has the highest mean Dependent correspondence, while E+A has a slightly higher mean Driver score. Dynamics modeling improves Dependent means on Polar3/4, but E+A performs better on both factors in Polar5. The comparison concerns fixed adaptations with different supervision, architectures, and training budgets.

Stage-wise refinement. Figure 3 shows equal-task mean gains in Dependent correspondence; mean Driver correspondence decreases slightly after dynamics modeling before improving after abductive learning.

Rules and discrimination. Table 2 evaluates the final models on real Evaluation transitions using labels computed directly from observations. The positive acceptance rate (PA) is the fraction of

Table 1: Native factor recovery: five-seed means with ± sample standard deviations (SD) below. Mean averages Polar3/4/5 within each seed. Spectral initialization (E); one-step dynamics modeling (D); inductive ABL with continuous fitting (A); vector-quantized variational autoencoder initialization (VQ). Random denotes random initialization.

	Polar3		Polar4		Polar5		Mean	
Method	ρ_{dr}	ρ_{de}	ρ_{dr}	ρ_{de}	ρ_{dr}	ρ_{de}	ρ_{dr}	ρ_{de}
Complete factor-discovery methods								
DynaAbd (E+D+A)	**0.938** ±0.008	**0.717** ±0.166	**0.888** ±0.027	**0.775** ±0.172	0.917 ±0.012	0.884 ±0.017	0.914 ±0.013	**0.792** ±0.050
SINDy-AE	0.057 ±0.049	0.021 ±0.013	0.066 ±0.065	0.049 ±0.035	0.125 ±0.091	0.049 ±0.078	0.083 ±0.054	0.040 ±0.028
Mechanism sparsity	0.079 ±0.032	0.076 ±0.046	0.078 ±0.044	0.014 ±0.013	0.073 ±0.058	0.080 ±0.036	0.076 ±0.041	0.057 ±0.019
Internal ablations								
E+A	0.924 ±0.019	0.666 ±0.250	0.884 ±0.028	0.656 ±0.223	**0.945** ±0.005	**0.890** ±0.038	**0.918** ±0.011	0.737 ±0.097
VQ+D+A	0.236 ±0.192	0.149 ±0.160	0.086 ±0.081	0.119 ±0.074	0.193 ±0.164	0.228 ±0.171	0.172 ±0.026	0.165 ±0.048
Random D+A	0.201 ±0.163	0.250 ±0.198	0.117 ±0.155	0.087 ±0.043	0.172 ±0.185	0.189 ±0.109	0.163 ±0.165	0.175 ±0.109
Random A	0.203 ±0.117	0.270 ±0.269	0.089 ±0.120	0.090 ±0.056	0.115 ±0.145	0.157 ±0.121	0.136 ±0.121	0.172 ±0.100

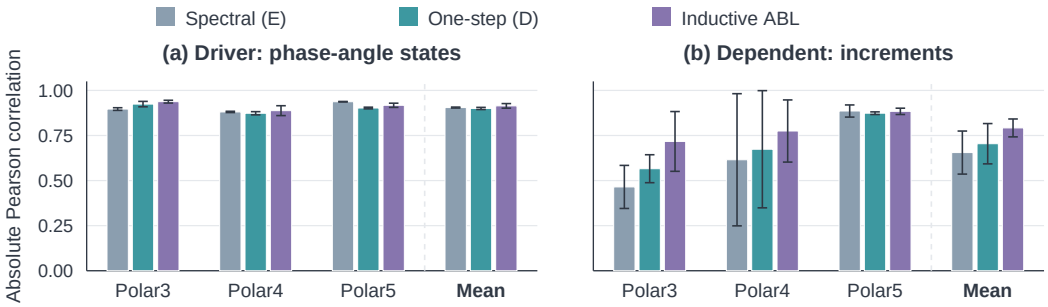


Figure 3: Stage-wise factor recovery: five-seed means ± sample SD. Mean averages Polar3/4/5 within each seed. Spectral initialization (E) and one-step dynamics modeling (D) precede inductive ABL, which includes label abduction and continuous fitting.

positive transitions accepted, and the negative rejection rate (NR) is the fraction of negative transitions rejected. *Coarse* checks discrete phase/Dependent-label transitions. *Coarse+Fine* additionally requires real continuous changes to follow the rule direction and eligible fitting residuals to pass their calibrated thresholds. This comparison measures the contribution of continuous motion constraints to quality discrimination.

Table 2: Real-transition discrimination of final DynaAbd models (five-seed mean ± sample SD). Mean averages Polar3/4/5 within each seed. Coarse+Fine combines discrete legality, continuous direction, and fitting-residual checks.

	Coarse		Coarse+Fine	
Environment	PA	NR	PA	NR
Polar3	**0.995** ± 0.002	0.026 ± 0.004	0.874 ± 0.021	**0.540** ± 0.020
Polar4	**0.985** ± 0.014	0.032 ± 0.023	0.866 ± 0.046	**0.523** ± 0.033
Polar5	**0.994** ± 0.001	0.019 ± 0.003	0.898 ± 0.003	**0.424** ± 0.018
Mean	**0.991** ± 0.004	0.026 ± 0.006	0.879 ± 0.011	**0.496** ± 0.013

5.2 Rule Supervision for Single-Environment Policy Retraining

Purpose and comparisons. We test whether the discovered factors and rules provide useful information for policy learning in the default MuJoCo Hopper, HalfCheetah, and Walker2d environments. An autoencoder (AE) learns an expanded observation representation through reconstruction; auxiliary factor-and-rule supervision then encourages it to retain the discovered motion structure. We compare raw observations (Raw), reconstruction-only AE, and AE with this auxiliary supervision (AE+rule, $\alpha_{\text{rule}} = 0.1$). The two autoencoders share initialization, architecture, and ordered minibatches. Frozen encoders supply inputs to randomly initialized soft actor-critic (SAC) (Haarnoja et al., 2018) policies trained with the original task reward. The AE+rule versus AE comparison tests the contribution of the combined supervision while preserving capacity for other observation information. Appendix C gives the task-specific settings.

Short-budget evaluation. Table 3 reports the time-normalized area under the learning curve (nAUC) over the first 30,000 environment steps. Every 5,000 steps, including step zero, ten evaluation episodes are averaged before trapezoidal integration and division by 30,000; nAUC retains return units. Each task uses ten paired training seeds.

Relative to reconstruction-only AE, factor-and-rule supervision increases mean nAUC by 6.4% in Hopper, 6.9% in HalfCheetah, and 10.8% in Walker2d. For Walker2d, the paired mean gain over AE is 27.0, with a 95% confidence interval (CI) of $[-29.34, 83.34]$, and AE+rule remains below Raw. Thus, the mean gains are consistent with useful auxiliary motion information.

Table 3: Single-environment retraining: 0–30k nAUC in return units. Each task reports ten-seed means $\pm$ sample SD. Best column means are bold. AE encodings have 22 dimensions in Hopper and 34 in HalfCheetah and Walker2d. Rule-supervision weight is $\alpha_{\text{rule}} = 0.1$.

Method	Hopper	HalfCheetah	Walker2d
Raw	281.1 ± 33.9	200.7 ± 74.7	$\mathbf{293.3} \pm 51.2$
AE	357.5 ± 84.6	794.0 ± 94.2	250.6 ± 60.6
AE + rule	$\mathbf{380.5} \pm 64.3$	$\mathbf{849.1} \pm 85.1$	277.6 ± 57.0

6 Discussion

DynaAbd addresses the coupled search over continuous factors, symbol groundings, and motion rules through progressive inductive ABL. Its first two stages organize the representation search using cyclic motion, conditional dynamics, and trajectory quality. The final stage induces shared programs and abduces compatible labels, while continuous fitting refines motion within the resulting symbolic structure. The downstream experiment evaluates rule supervision for early policy retraining while retaining full-observation reconstruction.

The assumptions are a recurring cycle, one Dependent, and a restricted phase-wise direction grammar. Multiple gaits may require additional factors or programs. Top-two label support and bounded search limit the explanations available for refinement. Native phase-angle Pearson depends on phase origin and branch cut, while absolute increment correlation discards sign and scale. Causal identification requires additional assumptions or interventions. Motion descriptions may also omit control-relevant quantities, making separate return and fidelity measurements essential.

7 Conclusion

We presented DynaAbd, a progressive inductive ABL approach for discovering symbols and rules from continuous-control trajectories. Contrastive spectral initialization and one-step dynamics modeling provide a structured starting point for joint learning. Rule induction and label abduction then refine the symbolic explanation together with its continuous factors and grounding boundaries, while continuous motion fitting supplies fine-grained supervision. Controlled cyclic systems evaluate factor and rule recovery, and MuJoCo locomotion experiments evaluate rule supervision during early policy retraining in the original environment.

Artificial intelligence (AI) use statement

Large language models assisted with literature retrieval, research discussion, experimental design, and manuscript drafting, translation, and editing. AI tools also helped implement research ideas in code, execute experiments, and analyze experimental results. The authors take responsibility for the final manuscript, claims, references, code, and reported results.

Ethics statement

The study concerns simulated continuous-control tasks. The intended use is analysis and learning of interpretable movement representations. Simulation-level rule satisfaction and task return do not establish the safety of deployment on physical robots; such deployment requires separate validation of sensing, actuation, and recovery behavior.

Reproducibility statement

Appendices A–C specify the representation, Prolog grammar and search protocol, continuous losses and gradient paths, data partitions, acceptance calibration, and downstream comparisons. Reproduction requires the corresponding background theory, model checkpoints, fixed scales, and thresholds.

References

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, 2019. doi: 10.1073/pnas.1906995116. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC6842598/.

Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110:801–856, 2021. doi: 10.1007/s10994-020-05934-z.

Wang-Zhou Dai and Stephen H. Muggleton. Abductive knowledge induction from raw data. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1845–1851, 2021. doi: 10.24963/ijcai.2021/254.

Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/9c19a2aa1d84e04b0bd4bc888792bd1e-Abstract.html.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018. doi: 10.1613/jair.5575.

Sébastien Lachapelle, Pau Rodríguez, Yash Sharma, Katie E. Everett, Rémi Le Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA. In *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pp. 428–484. PMLR, 2022. URL https://proceedings.mlr.press/v177/lachapelle22a.html.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 4114–4124. PMLR, 2019. URL https://proceedings.mlr.press/v97/locatello19a.html.

Xubo Lyu, Hanyang Hu, Seth Siriya, Ye Pu, and Mo Chen. Task-oriented Koopman-based control with contrastive encoder. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 93–105. PMLR, 2023. URL https://proceedings.mlr.press/v229/lyu23a.html.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4), 2021. doi: 10.1145/3450626.3459670. URL https://xbpeng.github.io/projects/AMP/AMP_2021.pdf.

Yifei Peng, Yaoli Liu, Enbo Xia, Yu Jin, Wang-Zhou Dai, Zhong Ren, Yao-Xiang Ding, and Kun Zhou. Abductive logical rule induction by bridging inductive logic programming and multimodal large language models. *arXiv preprint arXiv:2509.21874*, 2025a. URL https://arxiv.org/abs/2509.21874.

Yifei Peng, Zijie Zha, Yu Jin, Zhexu Luo, Wang-Zhou Dai, Zhong Ren, Yao-Xiang Ding, and Kun Zhou. Generating by understanding: Neural visual generation with logical symbol groundings. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*. Association for Computing Machinery, 2025b. doi: 10.1145/3711896.3736978. URL https://arxiv.org/abs/2310.17451v4.

Chen Sun, Wannan Yang, Thomas Jiralerspong, Dane Malenfant, Benjamin Alsbury-Nealy, Yoshua Bengio, and Blake Richards. Contrastive retrospection: Honing in on critical steps for rapid learning and generalization in RL. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015. doi: 10.1007/s00332-015-9258-5.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 5502–5511. PMLR, 2018. URL https://proceedings.mlr.press/v80/xu18h.html.

Xiao-Wen Yang, Wen-Da Wei, Jie-Jing Shao, Yu-Feng Li, and Zhi-Hua Zhou. Analysis for abductive learning and neural-symbolic reasoning shortcuts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 56524–56541. PMLR, 2024. URL https://proceedings.mlr.press/v235/yang24ac.html.

A Complete Method Specification

This appendix specifies spectral initialization, one-step dynamics modeling, and inductive ABL with continuous fitting. Experimental data and baseline settings are collected in Appendix B; the downstream supervision interface is defined in Appendix C.

A.1 Contrastive Spectral Initialization

Observations and dictionary. Collector blocks are split at real episode resets. Terminal transitions remain; reset jumps and padding are excluded. Discovery uses observations, adjacency, and policy-quality labels. Polar observations are $[x, y, v_x, v_y, x_0, y_0]$, excluding the ground-truth phase field; Hopper uses its native 11-dimensional observation. Positive Fit episodes determine fixed, episode-balanced centering and scaling. Linear and unique quadratic monomials give $d_\phi = d_o + d_o(d_o+1)/2$ features: 27 for Polar and 77 for Hopper. Intercepts are handled separately. The factor map acts on each observation independently.

Spectral subproblem and EDMD. Let X and Y contain current and successor feature rows in the centered, whitened effective space of dimension d_{eff}, and let diagonal W give each episode equal total weight. The weighted ridge fit and role-specific spectral subproblem are

$$\begin{aligned} K^\star &= \arg\min_K \|W^{1/2}(XK - Y)\|_F^2 + \eta\|K\|_F^2 \\ &= (X^\top W X + \eta I)^{-1} X^\top W Y, \\ V_\nu^\star &\in \arg\max_{V^\top N_\nu V = I_{14}} \operatorname{tr}(V^\top M_\nu V), \\ V_\nu^\star &= N_\nu^{-1/2} \operatorname{eig}_{14}\Big(N_\nu^{-1/2} M_\nu N_\nu^{-1/2}\Big). \end{aligned} \tag{11}$$

Here $\nu \in \{\mathrm{dr}, \mathrm{de}\}$ identifies the role, M_ν is its signal matrix, N_ν its positive-definite penalty metric, and eig_{14} returns the fourteen leading orthonormal eigenvectors. These are fixed-matrix solutions; subsequent sparse and temporal updates refine the resulting subspaces. Dictionary lifting, least-squares evolution, and eigenfunction reconstruction are EDMD ingredients (Williams et al., 2015). The role-specific objectives, response aggregation, and refinements below specify our initialization around that fit.

Predictable Driver space. Only positive transitions enter the Driver fit, with $\eta = 0.01$. Current and successor rows are centered separately. Define $C_{\text{xx}} = X_+^\top W_+ X_+$, $E_{\text{fit},+} = Y_+ - X_+ K_+$, and $\Delta X_+ = Y_+ - X_+$. Predictable next-state energy is rewarded, while residual error and rapid change are penalized:

$$\begin{aligned} M_{\text{dr}} &= K_+^\top C_{\text{xx}} K_+, \\ N_{\text{dr}} &= E_{\text{fit},+}^\top W_+ E_{\text{fit},+} + 0.1 \Delta X_+^\top W_+ \Delta X_+ + \varepsilon_{\text{sp,dr}} I. \end{aligned} \tag{12}$$

The stabilization includes $10^{-3}\operatorname{tr}(Y_+^\top W_+ Y_+)/d_{\text{eff}}$ and a numerical eigensolver regularizer. Fourteen generalized eigenvectors initialize 200 sparse subspace updates. Another 400 updates predict observed lags $\{1, 2, 4, 8, 16\}$ using powers of one fitted operator, with variance, subspace-deviation, and sparsity constraints. These temporal updates are part of spectral initialization.

Cyclic coordinate and candidate selection. Affine EDMD fits within the resulting space use ridge values $\{0.001, 0.01, 0.1\}$. A complex coefficient vector c_{eig} with fifteen entries, including the intercept, gives

$$\widehat{\varphi}(o) = [x_{\text{dr}}(o), 1] c_{\text{eig}}, \qquad z_{\text{dr}}^{(0)}(o) = \frac{[\operatorname{Re}\widehat{\varphi}(o), \operatorname{Im}\widehat{\varphi}(o)]}{|\widehat{\varphi}(o)|}. \tag{13}$$

Candidate selection prioritizes observation-recurrence coverage, single-winding returns, winding error, and then prediction residual. Insufficient recurrence evidence falls back to a usable low-residual candidate and records ambiguity. Polar additionally uses Validation observation-motion R^2 under a one-standard-error tie; Hopper uses recurrence geometry. Selection uses observed recurrence and prediction statistics. The chosen projection receives 200 updates. Mean positive progression fixes orientation; when identifiable, the mean reset direction across positive Fit episodes fixes one global origin. Both conventions are fixed across episodes. Fundamental-cycle recovery remains sensitive to the available recurrence evidence.

Driver-conditioned Dependent space. The initialized Driver supplies fixed-whitened Fourier rows $\Psi = [\cos\theta, \sin\theta, \ldots, \cos 3\theta, \sin 3\theta]$. Let D_y contain dictionary increments in quality class y. Equation 11, with input Ψ_y and target D_y, gives A_y^{align}. Class-wise weighted centering handles intercepts; ridge scales with the mean predictor-covariance eigenvalue. Classes are balanced and episodes equally weighted within class. Circularly shifting only Fourier rows by approximately half each episode's transition count gives A_y^{shift}. We subtract these coefficients before forming response energies:

$$\begin{aligned} A_y^{\text{corr}} &= A_y^{\text{align}} - A_y^{\text{shift}}, \\ G_{\text{pos}} &= (A_+^{\text{corr}})^\top A_+^{\text{corr}}, \qquad G_{\text{contrast}} = (A_+^{\text{corr}} - A_-^{\text{corr}})^\top (A_+^{\text{corr}} - A_-^{\text{corr}}). \end{aligned} \tag{14}$$

The first Gram matrix rewards response on positive motion; the second rewards response differences between quality classes. These coefficient energies measure fitted responses; the shift supplies a temporal-misalignment reference.

Let $\text{scale}(A) = A/(\text{tr}(A)/d_{\text{eff}})$, with a zero-trace guard, and $\rho_{\text{h}} = 0.01$. We combine both criteria by

$$\begin{aligned} M_{\text{de}} &= \mathcal{H}_{\rho_{\text{h}}}(G_{\text{pos}}, G_{\text{contrast}}), \\ \mathcal{H}_{\rho_{\text{h}}}(A, B) &= \left[\left(\frac{(\text{scale}(A) + \rho_{\text{h}} I)^{-1} + (\text{scale}(B) + \rho_{\text{h}} I)^{-1}}{2} \right)^{-1} - \rho_{\text{h}} I \right]_+ . \end{aligned} \tag{15}$$

Here $[\cdot]_+$ clips negative eigenvalues. The regularized harmonic aggregate favors directions supported by both criteria; a large response in one cannot simply compensate for a small response in the other as in an arithmetic sum. For commuting matrices this reduces to regularized scalar harmonic means along their common eigenvectors.

For class-centered increment covariances $C_{\Delta,y} = D_y^\top W_y D_y$, pooled centered state rows X, and centered, whitened Driver rows U, set

$$\begin{aligned} C_{x,\text{dr}} &= X^\top W U, \qquad R_{\text{dr}} = \frac{2 C_{x,\text{dr}} C_{x,\text{dr}}^\top}{\text{tr}(C_{x,\text{dr}} C_{x,\text{dr}}^\top)}, \\ N_{\text{de}} &= \text{scale}\left(\frac{C_{\Delta,+} + C_{\Delta,-}}{2} \right) + R_{\text{dr}} + \varepsilon_{\text{sp,de}} I. \end{aligned} \tag{16}$$

The trace-two redundancy term is bypassed near zero trace. It penalizes linear redundancy with Driver's state variation. Fourteen leading directions initialize 200 sparse updates and yield B_{de}. One scalar spectral solve then initializes w_{de} within that space. Final Dependent features are whitened using fixed Fit statistics, with positive within-episode covariance $C_{\text{de},+}$. The final mappings are Equation 2; the resulting bases, preprocessing, and reference covariances remain fixed during dynamics modeling and abductive learning.

Shared effective-coefficient sparsity. Let W_{eff} be projection coefficients expanded in the raw polynomial dictionary and σ_i its fixed feature standard deviations. Excluding affine constants, the role penalty is

$$\Omega_{\text{role}}(W_{\text{eff}}) = \frac{\sum_{i=1}^{d_\phi} \|\sigma_i [W_{\text{eff}}]_{i,:}\|_2}{\sqrt{\sum_{i=1}^{d_\phi} \|\sigma_i [W_{\text{eff}}]_{i,:}\|_2^2}}. \tag{17}$$

Driver uses row-group norms; scalar Dependent reduces to an ℓ_1/ℓ_2 ratio. Spectral initialization ramps its weight to 0.002 over the first quarter of each local optimization budget, with collapse guards.

A.2 One-Step Dynamics Modeling

Predictors and normalization. Equation 3 is implemented in residual form: $f_{D,\text{dr}}(z_{\text{dr}}) = z_{\text{dr}} + \widehat{\Delta z_{\text{dr}}}$ and $f_{D,\text{de}}(z_{\text{dr}}, z_{\text{de}}) = z_{\text{de}} + \widehat{\Delta z_{\text{de}}}$. Hence next-value and increment residuals coincide. Driver fits $[z_{\text{dr}}, 1] \mapsto \Delta z_{\text{dr}}$ with ridge 0.001 on a positive fitting bank, separately from positive query collection blocks. Dependent uses both quality classes, ridge 0.02, and design

$$[z_{\text{de}}, \text{Fourier}_{1:8}(\theta), z_{\text{de}} \, \text{Fourier}_{1:3}(\theta), 1],$$

where $\mathrm{Fourier}_{1:a}$ contains sine/cosine harmonics through order a. A $[z_{\mathrm{de}}, 1]$ fit supplies the comparison predictor in Equation 4. Query MSE averages over transitions and output coordinates, using fixed initialization-derived prediction-error scales. Regression solutions remain differentiable with respect to the active projections. These action-free predictors describe the collected state-transition statistics.

Quality and regularization terms. The class-specific conditional Gaussian response models in Equation 5 fit increment means and variances with variance floor 0.02. Balanced cross-entropy is divided by $\log 2$, making an uninformative prediction's loss one. The complete prediction, gain, and classification weights remain those in Equation 6.

Let $z_{\mathrm{dr}}^{\mathrm{raw}} = x_{\mathrm{dr}} W_{\mathrm{dr}} + b_{\mathrm{dr}}$, and let $\mathrm{Cov}_{n_{\mathrm{q}}}(z_{\mathrm{dr}}) = n_{\mathrm{q}}^{-1} \sum_i (z_{\mathrm{dr},i} - \overline{z_{\mathrm{dr}}})^\top (z_{\mathrm{dr},i} - \overline{z_{\mathrm{dr}}})$ use the positive query batch of size n_{q}. Its eigenvalues are μ_i. References

$$E_{\mathrm{dr}}^{(0)} = \max(\mathbb{E}_{\mathrm{bank},+} \| z_{\mathrm{dr}}^{\mathrm{raw},(0)} \|_2^2, 10^{-6}), \qquad E_{\mathrm{de}}^{(0)} = \max(\mathbb{E}_{\mathrm{bank},+} (\Delta z_{\mathrm{de}}^{(0)})^2, 10^{-6})$$

are computed once on the initial positive bank and detached. Table 4 defines all regularizers; the three geometric terms use positive-query statistics. Scalar $[x]_+ = \max(x, 0)$.

Table 4: Dynamics-modeling regularizers. Multiply each expression by its weight once. $\xi_\ell = \min(1, \ell/200)$ restarts for each role.

Term	Unweighted expression	Weight	Active role / purpose
Circle	$\sum_{i=1}^{2} [0.10 - \mu_i(\mathrm{Cov}_{n_{\mathrm{q}}}(z_{\mathrm{dr}}))]_+^2$	10	Driver spread
Raw power	$(\mathbb{E}_+ \| z_{\mathrm{dr}}^{\mathrm{raw}} \|_2^2 / E_{\mathrm{dr}}^{(0)} - 1)^2$	0.1	Driver power anchor
Increment energy	$[0.25 - \mathbb{E}_+ (\Delta z_{\mathrm{de}})^2 / E_{\mathrm{de}}^{(0)}]_+^2$	10	Dependent variation
Trust	$(w_{\mathrm{de}} - w_{\mathrm{de}}^{(0)})^\top C_{\mathrm{de},+} (w_{\mathrm{de}} - w_{\mathrm{de}}^{(0)})$	0.1	Dependent drift
Sparsity	$\Omega_{\mathrm{role}}(W_{\mathrm{eff},\nu})$	$0.002\xi_\ell$	Active role

$L_{\mathrm{reg,dr}}$ sums Driver's circle, raw-power, and sparsity terms; $L_{\mathrm{reg,de}}$ sums Dependent's energy, trust, and sparsity terms. Raw power is a two-sided anchor; circle and energy are lower-target penalties. They provide soft constraints on factor variation and scale.

Optimization. Dynamics modeling runs 1,000 Driver Adam updates at 3×10^{-4}, then freezes Driver for 1,000 Dependent updates at 0.002. Each fitting-bank or query-batch pool contains 1,024 transitions; gradients are clipped at one. The bases from spectral initialization remain fixed; the update inputs are continuous factors and quality labels.

A.3 Inductive ABL and Continuous Fitting

Update conventions. Each supported program initializes an independent candidate model from the same dynamics-modeling checkpoint and stays fixed within it. Table 5 fixes the evidence and gradient convention used below. Dictionary statistics, bases, covariances, and normalization scales are shared and frozen; final Validation compares complete refined models. Motion-model targets always come from real adjacent observations.

Table 5: Update conventions within one candidate model. Direct labels come from current observations; abduced labels come from the positive-window proof.

Operation	Fixed evidence	Updated quantities
Label abduction	Current probabilities, program, window constraints	Abduced label path
Grounding loss	Detached abduced labels; real increments	Both projections and boundaries
Motion fitting	Direct phases; detached factors and increments	Motion-model parameters ψ_A
Fitting feedback	Direct phases/mask; entire prediction detached	Both projections through real increments

Trainable intervals and label support. Factor normalization in Equation 2 floors the raw Driver norm at 10^{-10} and the Dependent quadratic form at 10^{-12}. Trainable phase origin β_{ph} and logits α_{ph} define circular intervals $I_{\mathrm{dr},k}$ by accumulating widths

$$\omega_{\mathrm{ph},k} = \delta_{\mathrm{ph}} + (2\pi - 5\delta_{\mathrm{ph}})\,\mathrm{softmax}(\alpha_{\mathrm{ph}})_k, \qquad \delta_{\mathrm{ph}} = 0.03(2\pi). \tag{18}$$

Four cuts define five ordered Dependent intervals $I_{\mathrm{de},j}$, with unbounded outer intervals:

$$c_{\mathrm{de},0} = \beta_{\mathrm{de}}, \qquad c_{\mathrm{de},i} = \beta_{\mathrm{de}} + \sum_{u=0}^{i-1}(0.05 + \mathrm{softplus}(\gamma_{\mathrm{de},u})), \quad i = 1, 2, 3. \tag{19}$$

Here β_{de} and γ_{de} are trainable offset/gap parameters. The reported Polar runs begin with five uniform phase intervals at $0, 72, 144, 216, 288$ degrees. Five labels specify the symbolic capacity.

Circular interval distance and ordered interval projection give

$$\begin{aligned} d_{\mathrm{dr},t,k} &= \mathrm{dist}_{\mathrm{circle}}(\theta_t, I_{\mathrm{dr},k})/(2\pi/5), \\ d_{\mathrm{de},t,j} &= \mathrm{clip}(z_{\mathrm{de},t}, I_{\mathrm{de},j}) - z_{\mathrm{de},t}, \qquad p_{\mathrm{g},\nu,t}(l) = \mathrm{softmax}_l[-d^2_{\nu,t,l}/0.25]. \end{aligned} \tag{20}$$

The local index l is k for Driver and j for Dependent. Distances vanish inside intervals. Each factor retains its two most probable labels at every state, including the first; probabilities are renormalized, rounded to four decimals, and rounded zeros excluded. The resulting $\widetilde{p}$ supplies Equation 7.

Logical constraints. The decoded path $\mathbf{q}^{\mathrm{abd}} = (q_t^{\mathrm{abd}})_t$ must satisfy

$$\begin{aligned} k_{t+1}^{\mathrm{abd}} &\in \{k_t^{\mathrm{abd}}, (k_t^{\mathrm{abd}} + 1) \bmod 5\}, \\ j_{t+1}^{\mathrm{abd}} - j_t^{\mathrm{abd}} &\in \{0, h_{k_t^{\mathrm{abd}}}\}, \qquad j_t^{\mathrm{abd}} \in \{0, \ldots, 4\}. \end{aligned} \tag{21}$$

Each clause consumes exactly one observed transition; there are no skips or episode-boundary crossings. The starting phase supplies the direction, including when phase changes. Termination requires both label sequences to match the observed state count, cover all phases, visit at least two Dependent intervals, and contain a strict increase and a strict decrease. Equality between the initial and final states is optional. Programs have one direction clause per phase, both signs present, and the termination clause illustrated in Section 4.3. Factor roles are fixed. Each observation/factor pair has a unique assignment; repeated fact visits reuse it and count its log probability once.

Search, proposals, and failed queries. Prolog abduces labels for positive Fit windows. Each query also checks a sampled negative window using the same program and existing facts, with its label assignments held fixed. A successful negative proof excludes the explanation. Because positive and negative fact identifiers are separate, missing negative facts can make this check vacuous.

At proposal time every permitted program is scored by Equation 7. The backend prior is $P(H) = 6/[(n_{\mathrm{lit}}(H) + n_{\mathrm{term}}(H))\pi]^2$, where the two counts are clause-body literals and distinct substituted terms. Both counts are constant across these templates, so the prior is equal. A search is *certified* when it proves optimality on the retained label support or proves infeasibility. Budget-limited searches, with or without a feasible path, and errors remain separate outcomes.

A proposal window votes only after all program searches are certified and at least one is feasible. It votes for the best score, with fixed program index breaking ties. Candidates are ranked by votes, feasible-window count, then program index. At most three programs with positive votes are retained. During refinement, only certified optimal paths supervise representation updates; infeasibility or interruption supplies no substitute labels. Certification establishes query optimality within the retained label support. Continuous optimization remains nonconvex.

Grounding and direction losses. State/transition averages use valid observations from windows with certified explanations. Squared interval distances give

$$L_{\mathrm{proj}} = \left\langle d^2_{\mathrm{dr},t,k_t^{\mathrm{abd}}} + d^2_{\mathrm{de},t,j_t^{\mathrm{abd}}} \right\rangle_{\mathrm{states}}. \tag{22}$$

For unit Driver coordinates, the signed shortest phase-angle increment is

$$\Delta\theta_t = \mathrm{atan2}\big([z_{\mathrm{dr},t}]_1[z_{\mathrm{dr},t+1}]_2 - [z_{\mathrm{dr},t}]_2[z_{\mathrm{dr},t+1}]_1, z_{\mathrm{dr},t}z_{\mathrm{dr},t+1}^{\top}\big), \qquad \Delta z_{\mathrm{de},t} = z_{\mathrm{de},t+1} - z_{\mathrm{de},t}. \tag{23}$$

Fixed scales $s_{\mathrm{dr}}, s_{\mathrm{de}}$ are the initial positive-data median of $|\Delta\theta|$ and root mean square (RMS) of Δz_{de}, each floored at 10^{-3}. With $\mathrm{barr}(u) = 0.1\,\mathrm{softplus}((0.05-u)/0.1)$,

$$L_{\mathrm{dir}} = \langle \mathrm{barr}(h_{k_t^{\mathrm{abd}}} \Delta z_{\mathrm{de},t}/s_{\mathrm{de}})\rangle, \qquad L_{\mathrm{advance}} = \langle \mathrm{barr}(\Delta\theta_t/s_{\mathrm{dr}})\rangle. \tag{24}$$

Direction supervision uses the abduced starting phase across both within-phase and cross-phase transitions. Interval projection has zero cost throughout the assigned interval.

Geometry and sparsity terms. Superscript ref denotes the fixed candidate-initialization reference. The safeguards are

$$\begin{aligned}
L_{\mathrm{anchor}} &= \langle \tfrac{1}{2}\|z_{\mathrm{dr}} - z_{\mathrm{dr}}^{\mathrm{ref}}\|_2^2\rangle, \qquad L_{\mathrm{radius}} = \langle (\|z_{\mathrm{dr}}^{\mathrm{raw}}\|_2 - \|z_{\mathrm{dr}}^{\mathrm{raw,ref}}\|_2)^2\rangle, \\
L_{\mathrm{cuts}} &= \tfrac{1}{4}\sum_{i=0}^{3}(c_{\mathrm{de},i} - c_{\mathrm{de},i}^{\mathrm{ref}})^2, \qquad L_{\mathrm{occ}} = \tfrac{1}{5}\sum_{k=0}^{4}[0.04 - \bar{p}_{\mathrm{occ},k}]_+^2, \\
\bar{p}_{\mathrm{occ},k} &= \langle \mathrm{softmax}_k[-d_{\mathrm{dr},t,k}^2/0.01]\rangle, \\
L_{\mathrm{plane}} &= \tfrac{1}{2}\sum_{i=1}^{2}[0.25 - \mu_{\mathrm{plane},i}/\max(\mu_{\mathrm{plane},i}^{\mathrm{ref}}, 10^{-12})]_+^2, \\
L_{\mathrm{sparse}} &= \Omega_{\mathrm{role}}(W_{\mathrm{eff,dr}}) + \Omega_{\mathrm{role}}(W_{\mathrm{eff,de}}).
\end{aligned} \tag{25}$$

Here $\mu_{\mathrm{plane},i}$ are the eigenvalues of $W_{\mathrm{dr}}^\top C_{\mathrm{dr}} W_{\mathrm{dr}}$, with fixed Driver-feature covariance C_{dr}. Sparsity uses Equation 17, with denominator floor 10^{-15}. The complete loss is

$$\begin{aligned}
L_A = L_{\mathrm{proj}} &+ 0.1L_{\mathrm{dir}} + 0.1L_{\mathrm{advance}} + 0.05L_{\mathrm{anchor}} + 0.01L_{\mathrm{radius}} \\
&+ 0.002L_{\mathrm{cuts}} + 10L_{\mathrm{occ}} + L_{\mathrm{plane}} + \lambda_{\mathrm{sparse}}L_{\mathrm{sparse}}.
\end{aligned} \tag{26}$$

$\lambda_{\mathrm{sparse}}$ ramps to 0.002 over the first 200 initial attempts, then restarts over the first 400 feedback attempts.

Motion model and schedule. A 3–32–32–2 multilayer perceptron (MLP) with sigmoid linear unit (SiLU) activations has componentwise output $f_A(\mathbf{z}) = 0.05 + \min\{\mathrm{softplus}(\mathrm{MLP}_{\psi_A}(\mathbf{z})), 8\}$. Final-layer weights initialize at zero and biases at 0.54. Its targets and eligibility masks are Equation 8, and its fit objective is

$$L_{\mathrm{fit}}(\psi_A) = \frac{\sum_{t,\nu} m_{\mathrm{fit},t,\nu}([f_A(\mathrm{sg}(\mathbf{z}_t))]_\nu - \mathrm{sg}(\delta_{t,\nu}))^2}{\sum_{t,\nu} m_{\mathrm{fit},t,\nu}}. \tag{27}$$

Driver uses all real positive transitions; Dependent fitting uses only same-direct-phase transitions. Signed violating targets are retained. Every attempt recomputes direct labels and masks, fixing them during differentiation. Boundaries receive gradients from L_A. Feedback follows Equation 9, updating factors whose revised label probabilities guide subsequent Prolog queries.

Each candidate uses 800 initial attempts with $\lambda_{\mathrm{fit}} = 0$, then 1,600 feedback attempts with $\lambda_{\mathrm{fit}}(\ell) = 0.01\min(1, \ell/300)$, counting from the feedback start. An attempt samples three positive and three negative Fit windows and issues three paired queries, each with at most 100 real transitions. The representation snapshot is held fixed until queries return. All positive windows fit the motion model; only certified paths contribute to the representation update. If none exists, that update is skipped. Candidates share initial motion-model seed and window order, then train independently. Adam rates are $0.0003/0.002/0.001/0.002$ for Driver/Dependent/boundaries/motion model; optimizer state continues across the two phases. Computation uses float64 and gradient clipping at one. Search permits three seconds per query and up to 50 concurrent queries. Across the fifteen full-method runs, 29 supported candidate models receive 2,398–2,400 representation updates each, with two skipped updates in total; each receives 2,400 motion-model updates. Detailed counts are in data/polar-full-update-counts.csv.

Acceptance and complete-model selection. Coarse evaluates direct labels on real transitions: both raw Driver norms exceed 10^{-8}, phase stays or advances by one, and Dependent-label change is in

$\{0, h_{k_t}\}$. This transition-level test uses the directly computed labels. Fit acceptance uses real positive Fit residuals

$$\epsilon_{t,\nu} = \left| \frac{\delta_{t,\nu}}{\widehat{\delta}_{t,\nu}} - 1 \right|, \qquad \kappa_{k,\nu} = Q_{0.95}\{\epsilon_{t,\nu} : k_t = k, m_{\mathrm{fit},t,\nu} = 1\}. \tag{28}$$

Coarse+Fine requires Coarse and every eligible residual below its phase/channel threshold. Driver is eligible on every transition and Dependent only within the same phase. A required test fails when calibration support is missing. Coarse+Fine also requires the continuous directions

$$\Delta\theta_t > 10^{-8}, \qquad h_{k_t}\Delta z_{\mathrm{de},t} > 10^{-8}, \tag{29}$$

including across phase changes. Each candidate's own factors, boundaries, motion model, and Fit-calibrated thresholds define its Validation score

$$S_{\mathrm{val}}(\mathcal{M}_H) = 0.75\mathrm{PA}(\mathcal{M}_H) + 0.25\mathrm{NR}(\mathcal{M}_H). \tag{30}$$

Ties use proposal rank, then program index. The winning model includes preprocessing, bases, scales, and thresholds. Selection uses only Fit-calibrated thresholds and Validation transitions. Table 2 evaluates these selected models under both criteria.

B Controlled Experiment Protocol

B.1 Environments, Splits, and Metrics

Polar3/4/5 have three/four/five physical motion phases but all methods in DynaAbd's internal comparison use five symbolic phases and five Dependent intervals. Episodes have 100 real transitions and the six observation coordinates specified in Appendix A.1. Physical relative-phase-angle boundaries are $0, 120, 240$ degrees for Polar3, with increasing/increasing/decreasing radius, and $0, 90, 180, 270$ degrees for Polar4, with increasing/decreasing/increasing/decreasing radius.

Table 6: Episode partitions per environment. Fit and Validation partition Train; Alignment and Evaluation are separate.

Partition	Positive episodes	Negative episodes	Transitions
Train	700	700	140,000
Fit	560	560	112,000
Validation	140	140	28,000
Alignment	150	150	30,000
Evaluation	150	150	30,000

The five seeds are 2026096501–2026096505. Complete episodes/reset seeds are disjoint across splits. Fit supplies normalization and gradients; Validation supplies candidate selection. Alignment permits restricted post-training coordinate alignment only. Polar3/5 reuse previously analysed Calibration and Development records as Alignment and Evaluation because original policy checkpoints were unavailable. This training/selection round excludes those records; their previous analysis limits interpretation as a blind test.

Native metrics use Equation 10, equal positive/negative weighting, at least three finite pairs, and variance above 10^{-12}. Native phase angles lie in $[0, 2\pi)$ at their original orientation and origin, making the score sensitive to the branch cut. Collapsed/undefined results remain recorded. Means and sample SDs use five seeds; the cross-environment mean first averages the three environments within each seed. Supplementary alignment chooses phase-angle orientation and a 720-point origin grid on Alignment only. External three-dimensional latents may choose a Driver plane and remaining Dependent coordinate; internal methods retain designated slots. These restricted coordinate choices are fixed before Evaluation. Aligned correlations are supplied separately from the native main comparison.

B.2 Baseline Configurations and Comparison Scope

SINDy-AE (Champion et al., 2019) and mechanism sparsity (Lachapelle et al., 2022) use unsupervised training with their own objectives and background assumptions. Their

Table 7: Factor-discovery configurations. Method abbreviations are defined in Table 1. External methods keep their own training objectives.

Method	Configuration
SINDy-AE	Official first-order Lorenz adaptation; latent dimension 3; sigmoid encoder 64/32; degree-3 polynomial library, 20 terms, no sine terms. Derivatives: $(o_{t+1} - o_t)/1$. Reconstruction/latent-derivative/observation-derivative/sparsity weights: $1/0/10^{-4}/10^{-5}$. Threshold 0.1 every 500 epochs; Adam 0.001, batch 1,024; 5,001 epochs plus 1,001 refinement epochs (654,218 updates).
Mechanism sparsity	Official temporal variational autoencoder (VAE); latent dimension 3, one lag; time-sparsity weight 0.5; transition MLP depth 5, width 512; Adam 5×10^{-4}, batch 1,024, 60,000 updates; fixed hyperparameters.
DynaAbd	Spectral initialization: 200 Driver-subspace, 400 temporal, 200 Driver-projection, 200 Dependent-subspace updates plus solves. Dynamics modeling: 1,000 updates per role. Abductive learning: $800 + 1,600$ attempts per supported program.
E+A	Spectral initialization followed directly by abductive learning, including the initialization's temporal refinement.
VQ+D+A	Five vector-quantization codes per role; 3-dimensional pre-quantization factors; 3–64–64–27 SiLU decoder reconstructs the dictionary. Adam 0.002, batch 2,048, 800 updates, clipping 1. Loss: reconstruction, 0.25 commitment, codebook fitting, Driver guard, 0.1 Dependent guard, 0.01 orthogonality, and effective sparsity. Continuous pre-quantization factors enter dynamics modeling and abductive learning.
Random D+A / A	Paired untrained role-structured bases/projections, with identical role dimensions and normalization. The two conditions apply dynamics modeling plus abductive learning, or abductive learning alone, to the random factors.

pinned revisions are, respectively, d50acb28cb97f2af50a3ef4e1b3fdd0d30d3a6f8 and 7fd50ded090a51a7527520f07d933450aa5b872a. In the tested adaptations, SINDy-AE has zero coefficient matrices in 8/15 runs, and the mechanism model's expected graph has approximately $8.98/9$ edges. These observations characterize the tested configurations. Assessing target information elsewhere in their multidimensional latents requires additional readout analysis.

Internal controls share Prolog support, grammar, candidate selection, and abductive-learning attempt budgets. Effective updates and total computational cost vary across conditions. All conditions use fixed-final checkpoints selected independently of physical truth and Evaluation. The seven methods, three environments, and five seeds give 105 runs. Detailed final records are provided as comma-separated values (CSV) in data/polar-final-results.csv; native and aligned scores remain distinct.

B.3 Stage-wise Refinement and Rule Evaluation

Figure 3 reports spectral initialization, dynamics modeling, and the final abductive-learning model. Label abduction and fitting feedback form a single stage.

All rates use real transitions with directly computed labels: PA is accepted positives divided by positives, and NR is rejected negatives divided by negatives. Coarse+Fine applies the same discrete, direction, and fitting checks as Table 2.

Figure **??** uses the full method's five seeds in their fixed order. Polar3/4 ground truth has three/four phases; learned partitions have five. Gray lines are the uniform boundaries before abduction, red lines physical truth, and black lines the final boundaries. Yellow/green sectors show increasing/decreasing Dependent values using each final program. The displayed relative-phase-angle coordinates use the exported endpoints without rotation, reflection, or sign adjustment; the repeated endpoint one turn later is omitted. Different radial lengths distinguish coincident lines. Extra learned boundaries can subdivide a region with unchanged direction, allowing different numbers of physical and symbolic phases.

Degeneracy flags are included in the run CSV.

C SINGLE-ENVIRONMENT RETRAINING PROTOCOL

The in-distribution (ID) retraining experiment uses the default MuJoCo Hopper, HalfCheetah, and Walker2d environments and their original task rewards. Hopper retains MuJoCo 2.00/Gym 0.18. The reported learning budget is 30,000 policy interactions. This section defines how frozen discovered factors supply AE supervision.

C.1 FACTOR MATCHING, RULE RELEARNING, AND SUPERVISION

From learned to predefined factors. For each seed, a frozen discovery representation ranks predefined functions on positive Fit transitions. Hopper and HalfCheetah reuse their previously learned factors; Walker2d runs discovery independently for each of seeds 9000–9009 before freezing its factors. All predefined phases are eligible. Dependent candidates are raw signed thigh/shin/foot angles for Hopper and the six front/hind joint angles for HalfCheetah. Walker2d also matches cyclic Driver candidates and raw joint-angle Dependent candidates. With gen and man denoting generated and predefined factors,

$$\rho_{\mathrm{match,dr}} = \left|\mathbb{E}_{\mathrm{Fit},+} e^{\mathrm{i}(\theta^{\mathrm{gen}} - \theta^{\mathrm{man}})}\right|, \qquad \rho_{\mathrm{match,de}} = \left|\mathrm{Pearson}_{\mathrm{Fit},+}(\Delta z_{\mathrm{de}}^{\mathrm{gen}}, \Delta z_{\mathrm{de}}^{\mathrm{man}})\right|. \tag{31}$$

The phase score uses the generated and predefined phase angles in their native orientations. Top-three distinct candidates per role form nine pairs. Once instantiated, predefined observation-to-factor formulas replace the generated coordinates in the downstream teacher and stay fixed, with positive-Fit normalization. The generated factors guide candidate selection, and the resulting teacher supplies auxiliary supervision to the policy representation. Actions are joint torques; selected joint angles describe motion outcomes.

Relearning the downstream rule. For each pair, Prolog proposes programs and learns the corresponding boundaries for those predefined factors. There are five phase and five Dependent labels, Top-2 support, open 100-transition positive/negative queries, complete phase coverage, and both Dependent directions. Positive and negative windows have separate label assignments. Each program has three seconds/one million nodes; a window votes only after every program is certified. The three highest-vote supported programs are retained.

Only phase origin/widths and ordered Dependent cuts are trained for 800 attempts, three queries each, using Adam 0.001 and clipping 1. The loss is interval projection plus $0.002L_{\mathrm{cuts}} + 10L_{\mathrm{occ}}$ from Appendix A.3. An attempt without feasible labels skips the boundary update. Train/Calibration are separated by complete episodes. Direct-label rule acceptance on real Calibration transitions selects across nine pairs and supported programs by $S_{\mathrm{ID}} = 0.75\mathrm{PA} + 0.25\mathrm{NR}$, using that pair's fixed formulas and learned boundaries.

After selecting the pair, boundaries, and H, all are frozen. A separate one-step network f_{ctrl}, parameterized by ψ_{ctrl}, receives 2,400 positive-Fit updates to predict phase progress and rule-directed Dependent change. It supplies continuous AE targets after candidate selection. Table 8 groups HalfCheetah's Dependent/program choice

Table 8: HalfCheetah selection frequencies across ten seeds. Program signs follow phase order $0, \ldots, 4$.

Dependent	Direction program	Count
Front thigh angle	$(+1, +1, -1, -1, -1)$	5
Rear foot angle	$(+1, +1, -1, -1, -1)$	2
Rear foot angle	$(+1, -1, -1, -1, +1)$	1
Front thigh angle	$(+1, +1, -1, -1, +1)$	1
Front shin angle	$(+1, +1, -1, -1, -1)$	1

C.2 AE SUPERVISION AND GRADIENT HANDLING

Let $c = E(o)$ and $D_{\mathrm{AE}}(c)$ reconstruct the observation. Hopper uses $11 \to 22 \to 11$; HalfCheetah and Walker2d use $17 \to 34 \to 17$. Each encoder and decoder has three 256-unit hidden layers;

Hopper and HalfCheetah use SiLU activations. These expanded latents provide capacity beyond the observation dimensions. The rule objective is

$$L_{\text{AE,ID}} = L_{\text{rec}} + \alpha_{\text{rule}} L_{\text{rule}}, \qquad \alpha_{\text{rule}} = 0.1,$$
$$L_{\text{rule}} = L_{\text{coarse-code}} + 0.5 L_{\text{Fine-numeric}} + 0.1 L_{\text{unit-circle}}. \tag{32}$$

The supervision path is observation → frozen selected factor formulas → fixed grounding/program and f_{ctrl} targets; AE predictions are obtained from its encoding c. Coarse-code supervision preserves phase/Dependent assignments, Fine-numeric supervision preserves continuous effect targets, and the unit-circle term maintains the predicted Driver geometry. These auxiliary outputs guide the encoder, and the policy receives the full latent.

Rule gradients use conflict projection relative to reconstruction gradients. Let g_{rec} and g_{rule} denote the two gradients on shared AE parameters. Conflicting components are projected relative to g_{rec} before combining the updates. Reconstruction-only AE uses L_{rec} with matched initialization, architecture, and ordered minibatches.

Table 9: AE and policy training parameters for Hopper and HalfCheetah, Walker2d. The reported policy-learning window is 0–30k interactions.

Stage	Setting	Value
AE	Optimizer / learning rate	AdamW / 3×10^{-4}
	Batch / updates	1,024 / 10,000
	Weight decay / clipping	10^{-5} / 5
	Checkpoint	Fixed final update
SAC	Actor / critic hidden layers	256×256
	Optimizer / learning rate	Adam / 3×10^{-4}
	Replay / batch / warm-up	100,000 / 256 / 100 steps
	Discount / target coefficient	0.99 / 0.005
	Updates / entropy	One per step / automatic
Evaluation	Checkpoints	0, 5k, 10k, 15k, 20k, 25k, 30k
	Episodes / reset seeds	10 / 10000–10009
	Paired training seeds	Hopper 8000–8009; HalfCheetah 6000–6009

C.3 Policy Evaluation and Statistics

Encoders and supervision models are frozen before randomly initialized SAC policies learn with the original task reward. Raw uses native observations; both AE variants use the same latent dimension within each task. Evaluation restores the training random-number-generator state and runs after the corresponding gradient update. Ten evaluation episodes are averaged within each training seed before computing

$$\text{nAUC} = \frac{1}{B} \sum_{i=0}^{5} \frac{\bar{G}(n_i) + \bar{G}(n_{i+1})}{2} (n_{i+1} - n_i), \qquad B = 30{,}000, \quad n_i = 5{,}000i. \tag{33}$$

The trapezoidal time average retains return units. Reported uncertainty is the sample SD across ten independent training seeds per task; evaluation episodes are repeated measurements within each seed. Results are reported separately for each task. Walker2d uses paired seeds 9000–9009. Its per-seed nAUC values, at the supplied two-decimal precision, are in data/walker2d-retraining-auc.csv. The reported paired AE+rule minus AE difference is 27.00 (95% paired-t CI $[-29.34, 83.34]$, wins in 6/10 seeds); versus Raw it is -15.75 (CI $[-56.47, 24.97]$, wins in 3/10 seeds). Bonferroni intervals for the prespecified family of four AUC comparisons are $[-50.48, 104.47]$ and $[-71.75, 40.25]$, respectively.

Walker2d's held-out reconstruction MSE is 0.00009807 for AE and 0.00125179 for AE+rule, a ratio of 12.76. Its interaction accounting records 900,000 training steps, 394,746 evaluation steps, and 304 additional environment-check steps; offline training adds no interactions.

D SHARED NOTATION

Table 10 collects symbols reused across sections. Intermediate matrices, local indices, and single-use statistics are defined where introduced. Time t indexes transitions; $\nu \in \{\mathrm{dr}, \mathrm{de}\}$ identifies factor/effect roles. A star denotes physical truth, abd an inferred symbolic label, (0) initialization, and a hat a prediction. Phase angles use signed principal differences; $\boldsymbol{\delta}$ collects their normalized increments and the normalized, direction-adjusted Dependent increments.

Table 10: Notation: Data, factors, and grounding. Time indices are omitted when the observation or transition is clear.

Symbol	Type / role	Definition
o	Observation	Continuous observation; o_t is a trajectory sample.
$\mathcal{D}$	Dataset	Quality-labelled episodes.
y	Class label	Trajectory-generating policy quality, in $\{+, -\}$; attached to the complete episode.
z_{dr}	Unit-circle vector	Two-dimensional cyclic Driver factor; $z_{\mathrm{dr},t}$ is its value at time t.
z_{de}	Scalar	Continuous Dependent factor; $z_{\mathrm{de},t}$ is its value at time t.
θ	Angle	Continuous Driver phase angle, $\mathrm{atan2}([z_{\mathrm{dr}}]_2, [z_{\mathrm{dr}}]_1)$, in radians modulo 2π.
F_ϑ	Map	Observation-to-factor map, $F_\vartheta(o) = (z_{\mathrm{dr}}(o), z_{\mathrm{de}}(o))$; distinct from the fitted motion model f_A.
$\boldsymbol{\beta}$	Parameters	Learned phase origin/widths and ordered Dependent cuts; distinct from projection bias b_{dr}.
q	Symbol pair	Direct label pair $q_t = (k_t, j_t)$ computed from observation o_t.
q^{abd}	Symbol pair	$q_t^{\mathrm{abd}} = (k_t^{\mathrm{abd}}, j_t^{\mathrm{abd}})$ from a completed, optimal feasible search; distinct from direct pair q_t.
H	Program	Shared recursive clauses specified by the direction sequence $(h_0, \ldots, h_4)$, with one termination clause.
h	Sign	$h_k \in \{-1, +1\}$ is the Dependent direction in phase k.
$\mathcal{M}$	Model tuple	DynaAbd output $(F_\vartheta, \boldsymbol{\beta}, H, f_A)$. $\mathcal{M}_H$ denotes the independently refined complete candidate for fixed program H.
ϕ	Feature map	Normalized linear and unique quadratic observation features.
B_{dr}	Matrix	Fixed $d_\phi \times 14$ Driver candidate basis.
B_{de}	Matrix	Fixed $d_\phi \times 14$ Dependent candidate basis.

Table 10 (continued): Objectives, evaluation, and control supervision.

Symbol	Type / role	Definition
f_D	Predictor	Ridge next-value predictor; role subscripts identify Driver, conditioned Dependent, or Dependent-only (de,0).
L_A	Scalar	Grounding objective for abduced intervals, starting-phase direction, forward progress, geometry, and sparsity.
f_A	Map	Continuous motion model with parameters ψ_A: Driver phase-angle progress and rule-directed Dependent change on same-phase transitions.
$\boldsymbol{\delta}$	Vector	Signed, normalized real increments: phase-angle progress and Dependent change multiplied by the direct starting-phase direction.
$\widehat{\boldsymbol{\delta}}$	Vector	Positive two-channel prediction of $\boldsymbol{\delta}$ from current factors.
m_{fit}	Binary mask	Driver output is fitted on all real transitions; Dependent output only when the direct phase label is unchanged.
s_{dr}	Scale	Fixed positive-data median absolute phase-angle increment, floored at 10^{-3}.
s_{de}	Scale	Fixed positive-data RMS Dependent increment, floored at 10^{-3}.
S_{abd}	Log score	Window label log probabilities plus a program log prior. Proposal varies $H, \mathbf{q}^{\mathrm{abd}}$; refinement varies only $\mathbf{q}^{\mathrm{abd}}$ at fixed H. Distinct from validation score S_{val}.
S_{val}	Score	Validation score of the complete refined $\mathcal{M}_H$: $0.75\mathrm{PA} + 0.25\mathrm{NR}$ under Coarse+Fine acceptance (discrete, direction, and residual checks), with the candidate's own grounding and thresholds.
κ	Threshold	Positive-Fit 95th percentile of relative residuals for a starting phase and eligible channel; stored with the selected model.
ρ_{dr}	Metric	Absolute state Pearson between learned and true native phase angles in $[0, 2\pi)$, at the original orientation and origin.
ρ_{de}	Metric	Absolute correlation between learned and true Dependent increments.
PA	Rate	Acceptance rate of real positive transitions.
NR	Rate	Rejection rate of real negative transitions.
E	Map	Pretrained autoencoder encoder, frozen for policy retraining.
c	Vector	Pretrained control representation $E(o)$; task-specific dimension, expanded relative to the observation.
f_{ctrl}	Map	Downstream motion model with parameters ψ_{ctrl}, fitted on frozen factors for AE supervision; distinct from f_A.
nAUC	Metric	Area under the reported learning curve divided by B; measured in return units.