PHDN: A PHYSICS-EMBEDDED DICTIONARY NETWORK FOR DATA-DRIVEN DISCOVERY OF PHYSICAL RELATIONSHIPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering physical relationships from data requires flexible approximation, symbolic interpretability, and incorporation of physical priors. Black-box neural networks offer strong approximation but limited access to physical structure, while interpretable dictionary and symbolic methods remain sensitive to predefined features, operator grammars, or discrete structural choices. We propose the Physics-embedded Dictionary Network (PhDN), a compositional explicit-dictionary architecture that organizes SINDy-style local dictionaries as a multilayer differentiable DAG with reusable intermediate states. Local dictionaries combine symbolic expressions, problem-specific operators, and approximation bases, so symbolic structure can be compiled, refined, and augmented within PhDN. PhDN can exactly represent finite symbolic expressions built from its admissible operators, while increasing polynomial or neural dictionaries provide universal approximation on compact domains. When informative structure is unavailable, SR supplies a skeleton for PhDN compilation; partial physical knowledge can enter as soft structural priors. Additional dictionary bases enable local correction without renewed symbolic search. Experiments on Feynman relationships and two dynamical systems show favorable ID/OOD accuracy. Staged ablations show fixed-support gains mainly reflect parameter refinement, while added bases address structural mismatch. Informative priors improve recovery and rollout robustness under derivative-label noise.

1 INTRODUCTION

Discovering physical relationships from data is a fundamental task in scientific machine learning and physical-system identification. An effective model should combine flexible representation, symbolic interpretability, and a clear interface for physical priors. Interpretable structures help researchers examine learned relationships, while suitable priors can improve predictive accuracy and out-of-distribution (OOD) generalization with limited data.

Black-box neural networks, such as multilayer perceptrons (MLPs) Haykin (1994); Hornik et al. (1989); Rumelhart et al. (1986), offer strong approximation ability and mature gradient-based training, but their hidden units and weights are difficult to interpret as physical operators or symbolic relations. This also makes local physical knowledge difficult to embed directly into their architectures. Physics-informed neural networks (PINNs) Raissi et al. (2019) constrain neural solution functions using residuals of known governing differential equations. They primarily address solution learning and parameter estimation under a known equation structure, rather than discovering the unknown physical relationships considered here.

Kolmogorov–Arnold Networks (KANs) (Liu et al., 2025b) replace fixed activations with learnable univariate edge functions. Learned splines often require sparsification or symbolification for compact equations, while gradient-based training may find initialization-dependent local solutions that fit in-distribution (ID) data without reliable extrapolation. Physics-informed KANs incorporate causal knowledge through topology constraints for dynamical modeling (Panahi et al., 2025). MultKAN introduces multiplication nodes, with `kanpiler` enabling symbolic initialization, refinement, and network expansion (Liu et al., 2025a). However, its demonstrated expansion workflow activates

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.

trainable spline components and recovers compact formulas through a subsequent symbolification step. This motivates representations that can enlarge model capacity while keeping both the inherited symbolic structure and newly added corrections explicit throughout refinement.

Dictionary-based methods provide another route to interpretable learning. Sparse Identification of Nonlinear Dynamics (SINDy) Brunton et al. (2016); He & Li (2026); He et al. (2026); Li et al. (2025) represents governing equations as sparse linear combinations of predefined nonlinear dictionary terms. Its coefficient estimation remains a linear regression problem for a fixed dictionary and support, making it efficient, but its performance strongly depends on whether the dictionary contains suitable relationships. A finite flat dictionary may require many redundant terms, or fail, for deeply nested or specialized compositions. Equation Learner (EQL) Martius & Lampert (2017); Sahoo et al. (2018) embeds prescribed mathematical operators, such as identity mappings, trigonometric functions, products, and divisions, into a neural architecture. It can construct compositional symbolic relations, but the admissible structures remain tied to a predefined operator set and its prescribed layerwise placement, limiting flexible incorporation of problem-specific physical operators or symbolic relations. Likewise, its complete reliance on backpropagation-based gradient optimization gives the learned structures a local character.

Symbolic regression (SR) Koza (1992); Schmidt & Lipson (2009); Cranmer (2023); Udrescu & Tegmark (2020); Petersen et al. (2021); Searson (2015) searches directly over expression structures within a prescribed operator grammar. It better handles unknown nested relationships and returns readable expressions, but practical search is constrained by the chosen operators, expression complexity, and computational budget. Once a skeleton is selected, subsequent continuous optimization is typically restricted to its constants, so structural misspecification generally requires renewed discrete search. The selected model therefore lacks a direct mechanism to continuously expand its explicit internal structure by introducing and jointly refining new intermediate relationships. Such a capability would enable imperfect symbolic structures to be refined without restarting discrete search, while remaining explicit and interpretable.

These methods expose a fundamental representation gap. Flat dictionary models preserve explicit feature semantics, but complex compositions must largely be enumerated in advance. Compositional neural and operator-based models construct intermediate features, but generally lack a flexible explicit-dictionary interface for incorporating physical knowledge and reusing symbolic structure. This motivates the question: *Can an explicit dictionary model itself be made compositional, so that intermediate physical or symbolic relationships can be constructed, reused, and continuously refined without sacrificing interpretability or expressive capacity?*

We propose the Physics-Embedded Dictionary Network (PhDN) as a network framework linking symbolic expressions with interpretable differentiable models, from scalar relationships to multidimensional vector fields. PhDN lifts a flat SINDy-style dictionary into a multilayer DAG of explicit dictionaries with reusable intermediate states. Admissible finite expressions can be compiled exactly into PhDNs, while sparse PhDNs retain symbolic readout. Additional polynomial or neural dictionary bases enlarge the explicit hypothesis space.

PhDN contributes a compositional explicit-dictionary representation and a corresponding three-stage data-driven learning procedure. Finite expressions generated by admissible operators have exact PhDN realizations, while polynomial and fixed neural-ridge constructions establish complementary routes to enhance representability. The training procedure combines SR-based skeleton discovery, including soft incorporation of partial physical knowledge, with fixed-support coefficient refinement and subsequent dictionary-support expansion. This organization enables accepted symbolic structures and additional approximation terms to be jointly refined within one explicit, reusable network representation without renewed symbolic search.

2 Related Work

Representations for equation discovery. Scientific discovery benefits from models whose internal computations retain mathematical meaning (Lipton, 2018; Murdoch et al., 2019; Rudin, 2019). SINDy learns sparse combinations of explicit dictionary features (Brunton et al., 2016), SR searches expression structures (Koza, 1992; Schmidt & Lipson, 2009; Cranmer, 2023), and EQL composes prescribed operators (Martius & Lampert, 2017; Sahoo et al., 2018). KAN learns univariate edge

functions (Liu et al., 2025b). MultKAN adds multiplication nodes and `kanpiler` for symbolic initialization, followed by parameter refinement and width/depth expansion (Liu et al., 2025a). KANs also support dynamical-model discovery with topology-based physical constraints (Panahi et al., 2025). PhDN instead organizes multivariate explicit dictionaries over reusable states. Its local representation and expansion units are dictionary functions and their coefficients, rather than individually parameterized univariate edges and additional summation or multiplication nodes. Operator choice, local feature richness, and coefficient support are explicit design dimensions of the architecture.

Prior incorporation and structural refinement. Physical knowledge can enter through governing-equation residuals in PINNs (Raissi et al., 2019), candidate dictionaries in SINDy, operator grammars or search initialization in SR, and architectural operators or topology in EQL and KAN (Karpatne et al., 2017; von Rueden et al., 2023). Appendix A compares these representations. PhDN combines structural operators with polynomial or fixed neural-ridge functions in branch-local dictionaries. The selected symbolic model is reproduced exactly at initialization, while zero-initialized augmentation coefficients provide additional trainable support without altering that initial mapping. Our analysis characterizes exact symbolic closure and completeness relative to the admissible operators, together with density under increasing polynomial or neural capacity. The resulting framework links symbolic initialization, coefficient refinement, and explicit support expansion within one representation.

3 Physics-embedded Dictionary Network (PhDN)

3.1 Problem Formulation and Motivation

Considering a general data-driven modeling task $\mathbf{y} = \mathbf{f}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{R}^{n_y}$, the objective is to construct an accurate and interpretable approximation of $\mathbf{f}$ while allowing available physical knowledge to be embedded into the model structure to enhance the training process based on the flexible prior support. To realize this objective, the SINDy framework, which provides a flexible physics-embedded dictionary, becomes a good starting point. SINDy represents the target relationship as a sparse linear combination of predefined candidate functions (Brunton et al., 2016):

$$\mathbf{y} = \mathbf{\Xi}^\top \mathbf{\Theta}(\mathbf{x}), \qquad \mathbf{\Theta}(\mathbf{x}) = \begin{bmatrix} \varphi_1(\mathbf{x}) & \cdots & \varphi_d(\mathbf{x}) \end{bmatrix}^\top, \tag{1}$$

where $\mathbf{\Theta} : \mathbb{R}^{n_x} \to \mathbb{R}^d$ is the candidate dictionary and $\mathbf{\Xi} \in \mathbb{R}^{d \times n_y}$ is a sparse coefficient matrix. Following the unified notation in Appendix A, we equivalently write

$$\mathbf{y} = \mathbf{W}\mathbf{\Phi}(\mathbf{x}), \qquad \mathbf{W} = \mathbf{\Xi}^\top, \quad \mathbf{\Phi}(\mathbf{x}) = \mathbf{\Theta}(\mathbf{x}). \tag{2}$$

While SINDy offers good interpretability and efficient linear coefficient estimation for a fixed candidate dictionary and support, its flat representation places a substantial burden on the predefined functions φ. Accurate recovery therefore depends strongly on whether suitable relationships are present or can be represented economically by the chosen finite dictionary. Practically, generic approximation bases, such as polynomial monomials or other nonlinear features, may be embedded into $\mathbf{\Phi}$ to improve representability. However, a finite flat dictionary still cannot guarantee a good fitting performance, while relying entirely on generic bases also weakens the recovery of compact physical structure. This challenge motivates the development of the PhDN.

3.2 Architecture of PhDN

PhDN extends this single-layer dictionary model into a multilayer, all-previous-layer-connected directed acyclic graph (DAG). Let $\mathbf{h}^{(0)} = \mathbf{x}, \mathbf{h}^{(L)} = \mathbf{y}, \mathbf{h}^{(\ell)} \in \mathbb{R}^{n_\ell}$. A branch from a preceding layer k to layer $\ell + 1$ retains the SINDy form:

$$\mathbf{b}_k^{(\ell)} = \mathbf{W}_k^{(\ell)} \mathbf{\Phi}_k^{(\ell)} \left(\mathbf{h}^{(k)} \right), \qquad 0 \le k \le \ell, \tag{3}$$

where $\mathbf{\Phi}_k^{(\ell)} : \mathbb{R}^{n_k} \to \mathbb{R}^{d_{k,\ell}}$ is the dictionary associated with this branch, and $\mathbf{W}_k^{(\ell)} \in \mathbb{R}^{n_{\ell+1} \times d_{k,\ell}}$ contains its learnable linear-combination coefficients. The dictionary may contain generic basis functions, physical operators, or composite functions compiled from symbolic expressions.

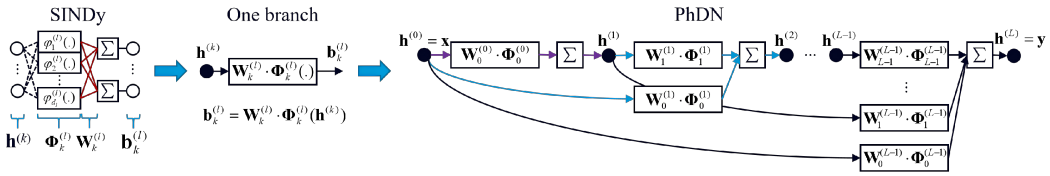


Figure 1: PhDN complete topology.

Because layer $\ell + 1$ receives dictionary features from all preceding layers, its output is $\mathbf{h}^{(\ell+1)} = \sum_{k=0}^{\ell} \mathbf{b}_k^{(\ell)}$. Equivalently, define the concatenated coefficient matrix $\mathbf{W}^{(\ell)} = [\mathbf{W}_0^{(\ell)}, \mathbf{W}_1^{(\ell)}, \cdots, \mathbf{W}_\ell^{(\ell)}]$ and the stacked dictionary vector $\overline{\mathbf{\Phi}}^{(\ell)} = [(\mathbf{\Phi}_0^{(\ell)}(\mathbf{h}^{(0)}))^\top, (\mathbf{\Phi}_1^{(\ell)}(\mathbf{h}^{(1)}))^\top, \ldots, (\mathbf{\Phi}_\ell^{(\ell)}(\mathbf{h}^{(\ell)}))^\top]^\top$, the mapping from layer ℓ to layer $\ell + 1$ can then be written compactly as $\mathbf{h}^{(\ell+1)} = \mathbf{W}^{(\ell)}\overline{\mathbf{\Phi}}^{(\ell)}$.

Accordingly, the complete L-layer PhDN is recursively defined by

$$\mathbf{h}^{(0)} = \mathbf{x},\ \mathbf{h}^{(\ell+1)} = \sum_{k=0}^{\ell} \mathbf{W}_k^{(\ell)} \mathbf{\Phi}_k^{(\ell)}\left(\mathbf{h}^{(k)}\right),\ \mathbf{y} = \mathbf{h}^{(L)},\ \ell = 0, \ldots, L-1. \tag{4}$$

A three-layer expansion demo for equation 4 is as

$$\begin{aligned}
\mathbf{h}^{(1)} &= \mathbf{W}_0^{(0)} \mathbf{\Phi}_0^{(0)}(\mathbf{h}^{(0)}), \\
\mathbf{h}^{(2)} &= [\mathbf{W}_0^{(1)}\ \ \mathbf{W}_1^{(1)}] \cdot [(\mathbf{\Phi}_0^{(1)}(\mathbf{h}^{(0)}))^\top\ \ (\mathbf{\Phi}_1^{(1)}(\mathbf{h}^{(1)}))^\top]^\top, \\
\mathbf{h}^{(3)} &= [\mathbf{W}_0^{(2)}\ \ \mathbf{W}_1^{(2)}\ \ \mathbf{W}_2^{(2)}] \cdot [(\mathbf{\Phi}_0^{(2)}(\mathbf{h}^{(0)}))^\top\ \ (\mathbf{\Phi}_1^{(2)}(\mathbf{h}^{(1)}))^\top\ \ (\mathbf{\Phi}_2^{(2)}(\mathbf{h}^{(2)}))^\top]^\top.
\end{aligned} \tag{5}$$

Each branch in equation 4 is therefore a SINDy-style physics-embedded dictionary model, while their all-previous-layer-connected composition constitutes PhDN, which is a directed acyclic graph, shown in Figure 1. When $L = 1$, PhDN reduces to the conventional single dictionary representation, like SINDy. Coincidentally, the abbreviation for "**SIN**gle **D**ictionar**y**" can also be SINDy, which lends the acronym a more specific meaning within the PhDN framework while simultaneously paying homage to its origins. With multiple layers, intermediate states can be reused downstream. For multi-argument operations spanning different depths, identity bases carry earlier operands forward until jointly available in a state, where a local dictionary can apply the operator.

3.3 Training PhDN

PhDN training compiles a symbolic structure into a differentiable PhDN DAG and refines its coefficients through backpropagation-assisted bounded nonlinear least squares with analytic Jacobians. In the general mode used here, symbolic regression provides the skeleton when informative prior structure is unavailable.

Stage I: symbolic-skeleton discovery. A SINDy model first bypasses outputs already represented accurately by the flat dictionary. The unresolved outputs are then searched jointly by native multi-output PySR over multiple restarts, while candidate archives and scores remain output-specific. For each output, validation-qualified expressions are ranked by a composite score combining validation quality with robust multi-scale knee evidence when two-sided frontier support exists and otherwise using validation alone (Appendix B.1.1). The selected candidate defines the core expression g_j^{core}.

Stage II: expression-to-PhDN compilation and fixed-support refinement. The selected cores are recursively decomposed into fine-grained operators and shared intermediate quantities and compiled into one multilayer PhDN DAG. Each active branch also receives a selectable low-order polynomial or fixed neural-ridge augmentation initialized with zero contribution. After exact reproduction of the Stage-I predictions is verified, only the compiled seed support is refined by validation-controlled bounded NLS using a BP-computed analytic Jacobian. All additional admissible coefficients remain fixed at zero.

Stage III: augmented refinement and pruning. The complete admissible support is released from the Stage-II solution, with all newly released coefficients still initialized at zero. The full PhDN is jointly refined under validation control. Low-contribution terms are then pruned, and each proposed support is briefly reoptimized and retained only when it satisfies the prescribed validation tolerance. Detailed scoring, compilation, optimization, and pruning rules are given in Appendix B.

3.4 Symbolic and Universal Representability of PhDN

We characterize PhDN from two complementary perspectives: exact symbolic representability and generic function approximation.

Symbolic representability. Let Ω denote the admissible primitive operators, $\mathfrak{E}(\Omega)$ the class of finite expressions generated from inputs, real constants, finite linear combinations, and operators in Ω, and $\mathfrak{F}_{\mathrm{PhDN}}(\Omega)$ the class represented by finite PhDNs. A PhDN layer performs exactly the two operations defining this symbolic class: dictionary evaluation applies operators in Ω, and branch coefficients form finite linear combinations. Hence every finite PhDN remains in $\mathfrak{E}(\Omega)$. Conversely, every finite symbolic expression can be recursively decomposed into finitely many primitive operations and intermediate subexpressions. Assigning these intermediate quantities to hidden state coordinates and realizing the required primitive operations through local dictionary branches then yields an exact finite PhDN representation. Therefore,

$$\mathfrak{F}_{\mathrm{PhDN}}(\Omega) = \mathfrak{E}(\Omega). \tag{6}$$

The formal two-direction proof is given in Appendix C.1.

Universal approximation. For compact $K \subset \mathbb{R}^{n_x}$, density means that every continuous function can be approximated arbitrarily well. For polynomial bases, Stone–Weierstrass gives density of finite-degree polynomial classes. A complete degree-p SINDy branch represents the corresponding polynomial class exactly, so increasing p transfers this density to PhDN. Even with a fixed quadratic dictionary, sufficient finite depth and width can generate every finite-degree polynomial, providing a second route without increasing polynomial order.

For continuous nonpolynomial neural bases, classical universal approximation establishes density of freely parameterized ridge networks. Alternatively, a countably infinite target-independent ridge-parameter bank dense in $\mathbb{R}^{n_x+1}$ can be fixed. The union of the spans of finite truncations is dense with only outer coefficients trained. This result concerns increasing truncations, not any fixed finite data-dependent augmentation used experimentally. Every truncation is exactly representable by a two-layer PhDN, and because their union is dense, the PhDN class is dense. Formal results and proofs appear in Appendices C.2 and C.3.

4 Numerical Experiments

We evaluate PhDN from three complementary perspectives: constructive symbolic representability, general physical-relationship learning without case-specific structural priors, and dynamical-system vector-field identification under varying prior strengths.

4.1 Symbolic Representability

We examine three properties of a finite PhDN: exact symbolic representation, direct readout of intermediate expressions, and fixed-support learnability on a nontrivial multi-output relationship:

$$\begin{cases} y_1 = 3\cos(x_2) + 0.5x_1^2 + \sqrt{x_3}, \\ y_2 = \log\left(0.25\exp(x_1x_2) + \dfrac{1}{x_1^2 + x_2 + 1}\right) + \sin(x_2), \\ y_3 = \log\left(0.25\exp(x_1x_2) + \dfrac{1}{x_1^2 + x_3 + 2x_2x_3}\right) + \dfrac{3x_1}{8x_3}. \end{cases} \tag{7}$$

The map combines polynomial, multiplicative, reciprocal, square-root, exponential, logarithmic, and trigonometric operators. Data domains and optimization settings are given in Appendix D.

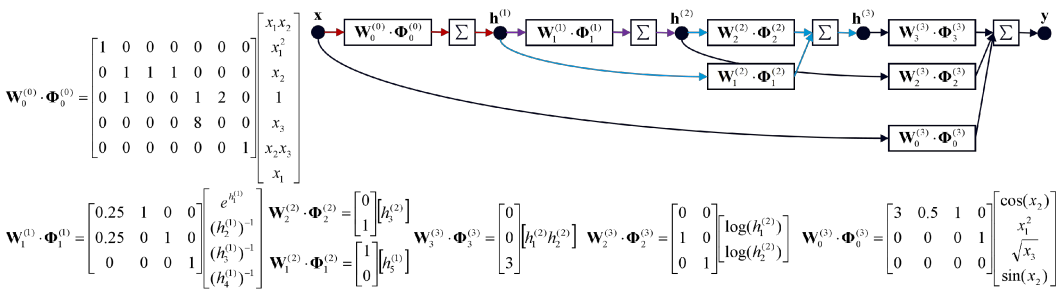


Figure 2: Compiled PhDN for equation 7; retained state coordinates admit direct symbolic readout. Unused augmentation bases and their zero coefficient columns are omitted.

Constructive representability and symbolic readout. Figure 2 shows the compiled PhDN and its direct symbolic readout. For example, the first hidden state stores x_1x_2, $x_1^2 + x_2 + 1$, $x_1^2 + x_3 + 2x_2x_3$, $8x_3$, and x_1, which are reused by downstream branches. Thus, intermediate states, dictionary functions, and coefficients remain directly traceable to the symbolic computation. The network has seven active branches, three skip branches, and 23 active coefficients and reproduces equation 7 to floating-point precision.

Strong-prior learnability. With the compiled support fixed, 20 initializations obtained by independently perturbing coefficients by $\pm 10\%$ are refined using bounded NLS with BP-computed analytic Jacobians. The selected model achieves RMSE below 3×10^{-14} on the training, validation, ID, and OOD sets, demonstrating near-machine-precision functional recovery when the correct structure is available. Additional compiled cases are reported in Appendix D.

4.2 Physical Relationship Learning

We evaluate PhDN on 27 dimensionless Feynman relationships Liu et al. (2025b) to assess scalar physical-map learning without case-specific structural priors. PhDN is compared with MLP, SINDy, SR†, SR$_{\text{native}}$, EQL-Div, and KAN. SR† denotes the proposed-score Stage-I symbolic model, whereas SR$_{\text{native}}$ uses PySR's native score under matched search settings. Each case uses 1500 training, 500 validation, 500 ID-test, and 500 disjoint OOD samples. Experiments are repeated three times with fixed data splits and varied stochastic seeds.

PhDN follows the procedure in Section 3.3. PhDN-Fixed is an internal fixed-support ablation: augmentation-only coefficients remain inactive while the accepted Stage-I support is refined in the PhDN representation. PySR-selected supports are compiled into a PhDN DAG, whereas the two SINDy-bypass models use the $L = 1$ specialization. Together, SR$_{\text{native}}$ $\rightarrow$SR† $\rightarrow$PhDN-Fixed $\rightarrow$PhDN forms a staged ablation of symbolic selection, fixed-support refinement, and augmentation. Table 1 reports seven representative cases; complete settings and 27-case results are provided in Appendix E.

Table 1: Mean ID/OOD RMSE over three runs on representative dimensionless Feynman relationships.

System	PhDN	PhDN-Fixed	SR†	SR$_{\text{native}}$	MLP	SINDy	EQL-Div	KAN
I.6.2	1.04e-15 / $< 10^{-15}$	1.04e-15 / $< 10^{-15}$	1.88e-8 / 2.01e-9	5.84e-4 / 1.66e-4	3.55e-5 / 2.10e-3	3.15e-2 / 1.22e5	3.60e-3 / 8.66e-3	1.66e-4 / 2.38e-1
I.6.2b	7.19e-9 / 7.25e-9	2.62e-7 / 1.93e-7	5.81e-7 / 3.12e-7	1.80e-7 / 2.66e-8	4.79e-5 / 1.35e-3	8.99e-2 / 7.16e2	3.63e-3 / 5.08e-3	1.36e-3 / 2.01e-1
I.9.18	$< 10^{-15}$ / $< 10^{-15}$	$< 10^{-15}$ / $< 10^{-15}$	4.83e-15 / 1.34e-13	1.03e-7 / 1.23e-6	3.63e-2 / 1.77e0	4.05e-1 / 7.05e0	4.38e-2 / 1.22e0	1.43e-1 / 3.80e0
I.12.11	$< 10^{-15}$ / $< 10^{-15}$	$< 10^{-15}$ / $< 10^{-15}$	3.37e-15 / 7.18e-15	3.37e-15 / 7.18e-15	4.09e-4 / 7.34e-1	$< 10^{-15}$ / $< 10^{-15}$	3.45e-3 / 3.06e-2	2.07e-4 / 2.60e0
I.13.12	$< 10^{-15}$ / $< 10^{-15}$	$< 10^{-15}$ / $< 10^{-15}$	3.63e-15 / 2.43e-15	3.63e-15 / 2.43e-15	1.76e-3 / 4.54e-2	$< 10^{-15}$ / $< 10^{-15}$	2.97e-9 / 1.40e-9	1.17e-2 / 3.69e0
I.15.3x	1.94e-5 / 4.11e-2	2.54e-5 / 4.50e-2	3.60e-5 / 4.34e-2	8.70e-4 / 7.68e-2	2.24e-4 / 3.11e-1	1.57e-4 / 1.92e-1	1.40e-2 / 5.43e-1	4.36e-3 / 6.79e0
I.16.6	$< 10^{-15}$ / $< 10^{-15}$	$< 10^{-15}$ / $< 10^{-15}$	$< 10^{-15}$ / $< 10^{-15}$	1.74e-7 / 1.56e-7	6.52e-5 / 1.16e-2	1.38e-2 / 2.49e1	1.49e-3 / 1.36e-3	6.05e-7 / 6.72e-1

†SR is the proposed-score Stage-I model. PhDN-Fixed refines only its accepted support. SR$_{\text{native}}$ uses PySR's native selection score. Values below 10^{-15} are displayed as $< 10^{-15}$ for all methods. Analyses use the raw stored RMSEs.

The symbolic columns isolate the three stages. Excluding the two SINDy-bypass cases, SR† improves both ID and OOD RMSE over SR$_{\text{native}}$ in 13 of 25 cases and is within 1% on both in 10 more, supporting the proposed Stage-I selection strategy. PhDN-Fixed lowers mean ID RMSE relative to SR† in all 27 cases and both metrics in 23, showing that the dominant gain comes from

high-precision refinement of the accepted symbolic structure. Releasing the augmented support then provides complementary flexibility when the Stage-I skeleton is imperfect: full PhDN improves both metrics in four cases, ID alone in two, and is unchanged in 14. Under the 10^{-15} display floor, full PhDN is lowest or tied-lowest on all seven representative ID and OOD cases.

Runtime remains dominated by symbolic search. Across 75 non-bypass runs, the Stage-II refit has median cost 1.67 s (range 0.36–16.38 s) and invokes no PySR search. Six $L = 1$ bypass refits require at most 0.001 s. Complete results and runtimes are reported in the appendix.

4.3 System Identification

We next evaluate PhDN for dynamical-system vector-field identification using

$$\mathcal{E}_{\mathrm{vf}} = \sqrt{\frac{1}{N_{\mathrm{te}} n_x} \sum_{i=1}^{N_{\mathrm{te}}} \sum_{j=1}^{n_x} \left[\widehat{f}_j(\mathbf{x}_i) - f_j(\mathbf{x}_i)\right]^2}, \mathcal{E}_{\mathrm{traj}} = \frac{1}{R} \sum_{r=1}^{R} \sqrt{\frac{1}{M n_x} \sum_{m=1}^{M} \sum_{j=1}^{n_x} \left[\frac{\widehat{x}_j^{(r)}(t_m) - x_j^{(r)}(t_m)}{s_j}\right]^2}. \tag{8}$$

Here, $\mathcal{E}_{\mathrm{vf}}$ is the test vector-field RMSE and $\mathcal{E}_{\mathrm{traj}}$ is the mean normalized trajectory RMSE over R unseen initial conditions, with s_j denoting the fixed scale of state j. Both are evaluated for $N_{\mathrm{tr}} \in \{250, 500, 1000, 2000\}$ using common datasets across methods.

4.3.1 Salient-Pole Generator Identification

We consider a four-state salient-pole single-machine infinite-bus model with flux decay and a linear automatic voltage regulator (AVR), where $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top = [\delta, \Delta\omega, E'_q, E_{\mathrm{fd}}]^\top$. Here, δ is the rotor electrical angle, $\Delta\omega$ is the per-unit speed deviation, E'_q is the transient internal voltage, and E_{fd} is the field voltage. The complete model and parameters are given in Appendix F.1.

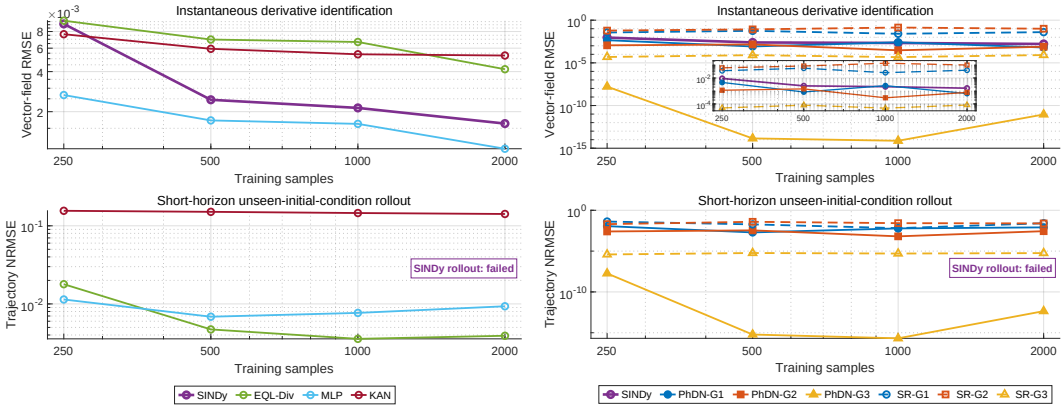
Derivative labels are obtained by directly evaluating the exact vector field. Nested scrambled-Sobol training sets contain 250, 500, 1000, and 2000 samples; fixed validation, ID-test, and disjoint OOD-test sets are generated independently. Dynamic consistency is evaluated from five common OOD initial conditions over 2 s using fixed-step RK4. Further data details are provided in Appendix F.1.

PhDN is compared with its Stage-I symbolic-regression model, the internal PhDN-Fixed ablation, MLP, EQL-Div, KAN, and SINDy. Three variants, PhDN-G_1, PhDN-G_2, and PhDN-G_3, use identical data, grammar, random state, and optimization settings but different shared initial-expression dictionaries. Each dictionary is softly injected into the candidate populations of unresolved outputs and may be modified or discarded during SR rather than assigned to prescribed equations.

The three dictionaries provide progressively more informative physical structures. G_1 supplies a coarse reluctance-related expression and a Euclidean terminal-voltage surrogate; G_2 adds the characteristic $x_3 \sin(x_1)$ and $\sin(2x_1)$ interaction and a trigonometric voltage-magnitude structure; and G_3 further introduces the coupled x_3–$\cos(x_1)$ terminal-voltage topology. Complete definitions are given in Appendix F.2. SR-G_k denotes the corresponding Stage-I model. PhDN-Fixed-G_k compiles the same accepted skeleton and performs only Stage-II fixed-support refinement, with augmentation-only coefficients inactive; full PhDN-G_k subsequently releases the second-order polynomial augmentation. For a matched-prior comparison, SINDy receives the strongest G_3 dictionary as shared grouped dictionary functions.

Figure 3a compares the baseline methods. MLP gives the strongest overall baseline performance, while EQL-Div also provides relatively accurate short-horizon rollouts. KAN shows larger rollout errors under the validation-selected sweep, suggesting that its highly flexible spline representation may generalize less robustly from the finite ID samples to unseen-initial-condition rollouts in this case. Its ID accuracy is also less competitive under the selected configurations. This result does not appear to arise simply from a restricted model-selection budget: KAN is selected over multiple depths and spline-grid resolutions and includes subsequent sparsification, pruning, and recovery, while requiring substantially higher training cost than the other baselines, as shown in Table 15. Further expanding its search space could potentially improve accuracy, but would incur additional computational cost. Although SINDy attains moderate ID vector-field accuracy, its unseen-initial-condition rollouts fail, indicating insufficient dynamic accuracy in the learned flat representation.

Figure 3b shows the prior-level trend, while Table 15 in Appendix F.2 reports the SR-G_k →PhDN-Fixed-G_k →PhDN-G_k ablation. At G_3 and $N_{\mathrm{tr}} = 1000$, vector-field RMSE decreases from 4.72×10^{-5} to 8.33×10^{-10} and then 7.45×10^{-15}, while trajectory NRMSE decreases from 5.03×10^{-6} to 3.73×10^{-11} and 2.07×10^{-16}. Across G_3, fixed-support refinement yields large gains at every sample size, showing that continuous refinement is effective when Stage-I supplies a strong skeleton. Under weaker G_1 and G_2 priors, fixed-support gains are smaller or negligible, whereas augmentation reduces residual errors; full PhDN improves both metrics over PhDN-Fixed at every prior level and sample size. Thus, fixed-support refinement primarily corrects parametric error in an adequate skeleton, while augmentation provides complementary capacity for structural mismatch. The PhDN curves need not vary monotonically with training-set size because Stage I can select different symbolic skeletons, while G3 consistently provides a more informative structural initialization and achieves low errors across sample sizes.



(a) Baseline comparison.

(b) Prior-level and Stage-I SR ablations.

Figure 3: Generator identification results.

4.3.2 Soft-Saturated Lorenz–96 Identification

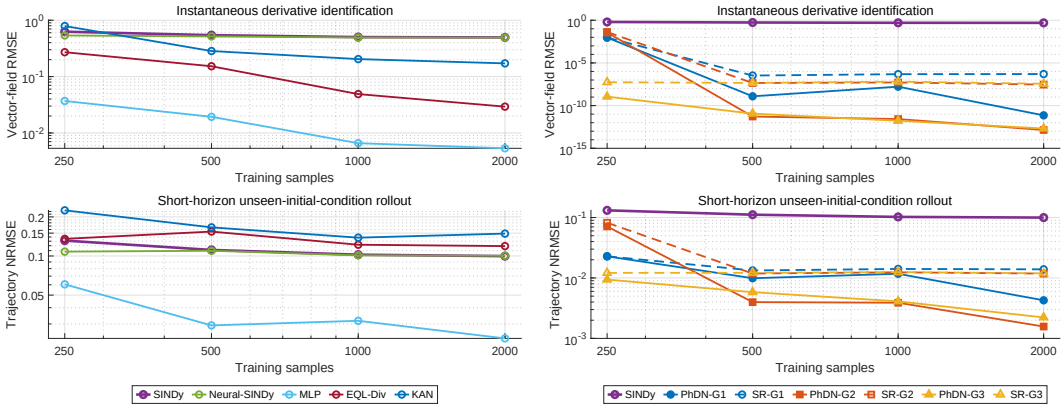
We further consider a higher-dimensional soft-saturated Lorenz–96 system. For $K = 10$ cyclic states,

$$z_i = x_{i-1}(x_{i+1} - x_{i-2}), \qquad \dot{x}_i = \frac{z_i}{\sqrt{1 + (z_i/\kappa)^2}} - x_i + F, \tag{9}$$

where the indices are cyclic, $F = 8$, and $\kappa = 1$. The smooth saturation introduces a nested nonlinearity beyond the standard Lorenz–96 transport law. Complete data domains are given in Appendix F.3. The same nested training sizes $N_{\mathrm{tr}} \in \{250, 500, 1000, 2000\}$ are used, and dynamic consistency is evaluated from five unseen OOD initial conditions over 5 s.

This case also examines PhDN with neural augmentation in a higher-dimensional setting. After Stage-I SR, PhDN-Fixed compiles the accepted skeleton and refines only its support, while full PhDN releases zero-initialized coefficients of $N_{\mathrm{neu}} = 10$ fixed $\tanh$ neural-ridge bases on each active branch, avoiding combinatorial monomial growth. Three prior levels are considered: G_1 provides no initial expression; G_2 provides the ideal unsaturated cyclic transport and damping $x_{i-1}(x_{i+1} - x_{i-2}) - x_i$; and G_3 additionally provides the forcing F. The true saturation structure is unknown at every level. For G_2 and G_3, the complete cyclic expression dictionary is softly shared across the candidate populations of all unresolved outputs rather than assigned to prescribed equations. Further settings are given in Appendix F.4.

Figure 4a compares the baseline methods. MLP gives the strongest overall baseline performance. Neural-SINDy achieves accuracy comparable to polynomial SINDy while reducing the dictionary dimension from 76 to 31, demonstrating the higher representational efficiency of neural candidate functions for mitigating dictionary growth in higher-dimensional systems. EQL-Div and KAN remain less competitive under the validation-selected configurations. Such multi-output vector-field identification with unseen-state rollout generalization was not examined in the original KAN study,



(a) Baseline comparison.

(b) Prior-level and Stage-I SR ablations.

Figure 4: Lorenz–96 identification results.

which focused primarily on scalar function fitting and fixed-image reconstruction. This different task setting may partly explain KAN's less competitive performance here.

Figure 4b shows strong small-sample recovery under the most informative prior, G_3, while larger datasets allow the less informative G_1/G_2 settings to approach G_3. Table 16 in Appendix F.4 reports the matched SR-G_k $\rightarrow$PhDN-Fixed-G_k $\rightarrow$PhDN-G_k ablation. Fixed-support refinement lowers both errors in 11 of 12 configurations. For G_2, $N_{\mathrm{tr}} = 500$, vector-field RMSE decreases $4.47\times10^{-8} \rightarrow 8.34\times10^{-10} \rightarrow 5.27\times10^{-12}$, while trajectory NRMSE decreases $1.18\times10^{-2} \rightarrow 8.75\times10^{-3} \rightarrow 3.98\times10^{-3}$. Full PhDN further lowers both metrics in nine configurations, showing complementary gains from fixed-support refinement and neural-basis expansion.

4.3.3 Robustness to Derivative-Label Noise

We further assess robustness to noisy derivative observations at $N_{\mathrm{tr}} = 1000$ on both dynamical-system cases using a separate three-round fixed-N robustness protocol with the common noise range $\rho \in \{0, 0.1\%, 0.5\%, 1\%\}$. Relative Gaussian noise is added to the training and validation derivative labels, while the held-out ID-test labels remain clean, and each level is repeated over three independent optimization rounds. The generator results show a strong dependence on prior quality. PhDN-G_3 achieves mean clean-test RMSEs of 1.09×10^{-9}, 6.49×10^{-4}, 4.86×10^{-3}, and 3.40×10^{-3} from clean to 1% noise, consistently below the corresponding MLP errors of 1.55×10^{-3}, 3.01×10^{-3}, 1.16×10^{-2}, and 2.60×10^{-2}. The same G_3 ordering persists in rollout, with mean trajectory NRMSE increasing from 1.37×10^{-4} to 8.32×10^{-3}, while MLP increases from 7.87×10^{-3} to 3.20×10^{-2}. Weaker G_1/G_2 priors show larger cross-round variation and isolated rollout failures, indicating that pointwise vector-field accuracy does not guarantee dynamical stability. On the soft-saturated Lorenz–96 system, all three PhDN variants remain substantially more accurate than MLP in vector-field RMSE: at 0.1% their means are 6.40×10^{-5}–8.80×10^{-5} versus 6.71×10^{-3} for MLP, and at 1% all remain below 9.0×10^{-3}. Rollout errors remain lower for stronger G_2/G_3 variants at 1%, but differences narrow with noise. Overall, stronger priors improve both instantaneous and short-horizon dynamical robustness. Complete curves appear in Appendix F.5 and Figure 8.

5 Conclusion

We introduced PhDN, a compositional explicit-dictionary architecture linking symbolic structure with a differentiable DAG of reusable states. PhDN exactly represents finite expressions built from admissible operators, while polynomial and neural dictionaries provide universal approximation under the stated conditions. Experiments show that fixed-support refinement mainly provides high-precision parameter refinement, whereas augmentation supplies complementary capacity for residual structural mismatch. Informative priors improve small-sample recovery and dynamical robustness. The main limitation remains reliance on stochastic SR-based structure discovery, which can dominate runtime.

Author Contributions

Removed for double-blind review.

Acknowledgments

Removed for double-blind review.

Reproducibility Statement

The complete PhDN implementation, experiment drivers, baseline interfaces, and reproduction instructions are provided in the anonymous supplementary code repository accompanying this submission. The repository README documents the required MATLAB and Python environments, main experiment entry points, supplementary data and pretrained-model organization, and procedures for both fast result replay and full retraining. Detailed model definitions, sampling protocols, hyperparameters, and theoretical proofs are additionally provided in the appendices.

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.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023. doi: 10.48550/arXiv.2305.01582.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994. ISBN 978-0-02-352761-6.

Hanyang He and Yan Li. Learning vector fields of differential equations on manifolds with curvature-adaptive geometrically constrained SINDy. *Communications in Nonlinear Science and Numerical Simulation*, 155:109406, 2026. doi: 10.1016/j.cnsns.2025.109406.

Hanyang He, Yan Li, Minghui Zhu, Liang Du, and Daning Huang. Physics-embedded data-driven modeling for model predictive control of battery charging converter. *IEEE Transactions on Industry Applications*, 2026. doi: 10.1109/TIA.2026.3676959. Accepted, in press.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

Anuj Karpatne, Gowtham Atluri, James H. Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017. doi: 10.1109/TKDE.2017.2720168.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: 10.1016/S0893-6080(05)80131-5.

Jiaorui Li, Kaiyuan Li, and Zifei Lin. Sparse identification of nonlinear economic dynamical model. *Physica A: Statistical Mechanics and its Applications*, 675:130861, 2025. doi: 10.1016/j.physa.2025.130861.

Zachary C. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10):36–43, 2018. doi: 10.1145/3233231.

Ziming Liu, Max Tegmark, Pingchuan Ma, Wojciech Matusik, and Yixuan Wang. Kolmogorov-arnold networks meet science. *Phys. Rev. X*, 15:041051, Dec 2025a. doi: 10.1103/4t7t-v19l. URL https://link.aps.org/doi/10.1103/4t7t-v19l.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–Arnold Networks. In *International Conference on Learning Representations*, 2025b.

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. In *5th International Conference on Learning Representations, Workshop Track Proceedings*, 2017.

W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. doi: 10.1073/pnas.1900654116.

Shirin Panahi, Mohammadamin Moradi, Erik M. Bollt, and Ying-Cheng Lai. Data-driven model discovery with kolmogorov-arnold networks. *Phys. Rev. Res.*, 7:023037, Apr 2025. doi: 10.1103/PhysRevResearch.7.023037. URL https://link.aps.org/doi/10.1103/PhysRevResearch.7.023037.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

Subham S. Sahoo, Christoph H. Lampert, and Georg Martius. Learning equations for extrapolation and control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4442–4450. PMLR, 2018.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Dominic P. Searson. GPTIPS 2: An open-source software platform for symbolic data mining. In Amir H. Gandomi, Amir H. Alavi, and Conor Ryan (eds.), *Handbook of Genetic Programming Applications*, pp. 551–573. Springer, Cham, Switzerland, 2015. doi: 10.1007/978-3-319-20883-1_22.

Marshall H. Stone. The generalized Weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948. doi: 10.2307/3029750.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Michal Walczak, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed machine learning: A taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2023. doi: 10.1109/TKDE.2021.3079836.

LARGE LANGUAGE MODEL USAGE

The original PhDN concept, its core architectural idea, research direction, initial mathematical derivations, and initial manuscript draft were developed independently by the authors without large language model assistance. After the initial PhDN framework had been established, large language models were used as general-purpose assistants for literature discovery, mathematical and methodological discussion, proof development, code development, manuscript revision, and language refinement. The authors designed and conducted the research, implemented and executed the experiments, independently verified the mathematical claims, derivations, code-assisted outputs, and numerical results, and reviewed and approved all content in the final manuscript.

A Architectural Comparison of Representative Models

Figure 5 provides a schematic comparison of the representative MLP, EQL-Div, KAN, and SINDy architectures discussed in Section 2. We use φ as a unified notation for a nonlinear operation, which may correspond to an activation, mathematical operator, basis function, or candidate function depending on the model. Black edges denote fixed data flow, red edges or boxes denote learnable coefficients or operations, and dashed edges denote connections determined by the definition of φ.

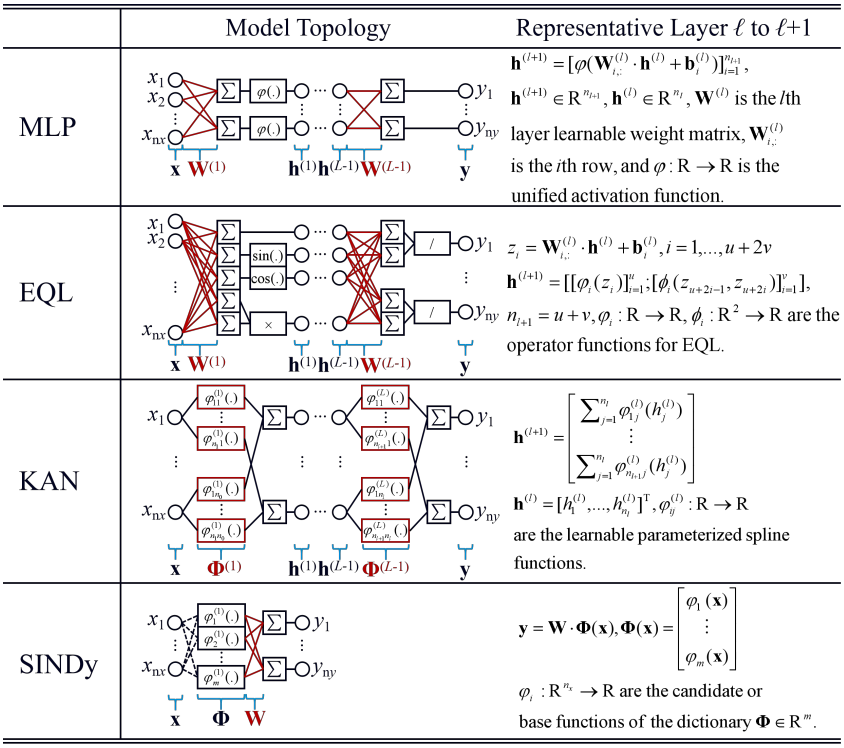


Figure 5: Architectural comparison among representative models.

Relation to compiled KAN workflows. The `kanpiler` workflow compiles formulas into symbolic KANs and supports subsequent refinement and network expansion. PhDN uses a different local representation: each SINDy-style branch combines explicit dictionary functions of a state vector, including multivariate physical operators and approximation features. Once the augmented dictionaries are constructed, the same state topology and dictionary functions are retained throughout subsequent PhDN refinement, while Stages II and III differ in the released coefficient support. For a branch $\mathbf{b} = \mathbf{W}\mathbf{\Phi}(\mathbf{h})$ at fixed input state, $\frac{\partial b_u}{\partial W_{ub}}\Big|_{W_{ub}=0} = \Phi_b(\mathbf{h})$. Thus, zero initialization preserves the compiled mapping without requiring the added dictionary functions themselves to vanish. A nonzero feature response provides a nonzero branch-level coefficient sensitivity even when its initial contribution is zero.

Representability relative to KAN. KAN and MultKAN have established approximation results under their respective architectural assumptions (Liu et al., 2025b;a). Our analysis addresses a complementary question for the explicit-dictionary representation: Proposition C.1 characterizes its exact operator-relative symbolic class, while Theorem C.3 and Corollary C.5 establish increasing-capacity routes to density.

B Detailed PhDN Training Algorithm

This appendix details the three-stage training procedure summarized in Section 3.3.

B.1 Stage I: Symbolic-Skeleton Discovery

Let

$$\mathcal{D}_{\mathrm{tr}} = (\mathbf{X}_{\mathrm{tr}}, \mathbf{Y}_{\mathrm{tr}}), \qquad \mathcal{D}_{\mathrm{val}} = (\mathbf{X}_{\mathrm{val}}, \mathbf{Y}_{\mathrm{val}})$$

denote the training and external-validation datasets, containing N_{tr} and N_{val} samples. For a candidate expression g associated with output j, define its validation loss as

$$L_{\mathrm{val},j}(g) = \frac{1}{N_{\mathrm{val}}} \sum_{i=1}^{N_{\mathrm{val}}} \left[g(\mathbf{x}_i) - y_{ij} \right]^2 .$$

The scale-aware numerical selection floor is

$$\epsilon_j^{\mathrm{sel}} = \max\left\{ \epsilon_{\mathrm{abs}}, \epsilon_{\mathrm{rel}} \max\left(1, \frac{1}{N_{\mathrm{val}}} \sum_{i=1}^{N_{\mathrm{val}}} y_{ij}^2 \right), \epsilon_{\mathrm{tiny}} \right\},$$

where $\epsilon_{\mathrm{tiny}} > 0$ is a numerical safeguard. The effective validation loss used for structural selection is

$$\overline{L}_{\mathrm{val},j}(g) = \max\{ L_{\mathrm{val},j}(g), \epsilon_j^{\mathrm{sel}} \}.$$

Hence, raw validation losses not exceeding $\epsilon_j^{\mathrm{sel}}$ are treated as numerically tied for structural selection.

B.1.1 Symbolic Candidate Score

The score is evaluated separately for each output and SR restart. We fix output j and restart s below and suppress these indices in local neighborhood notation when no ambiguity arises.

The scoring controls are

$$\Gamma_{\mathrm{score}} = \{ \rho_{\mathrm{val}}, w_{\mathrm{val}}, \epsilon_{\mathrm{abs}}, \epsilon_{\mathrm{rel}}, \mathcal{A}_{\mathrm{sc}}, d_{\min}, d_{\max}, \Delta_c, \kappa_{\mathrm{MAD}}, \delta_{\mathrm{loc}}, B, z_{\max}, s_{\mathrm{neu}} \}.$$

Here, $\rho_{\mathrm{val}} > 1$ defines the near-best-validation eligibility threshold, and $w_{\mathrm{val}} \in [0, 1]$ weights validation quality in the final composite score. The parameters ϵ_{abs} and ϵ_{rel} define the absolute and scale-relative numerical selection floors. The set $\mathcal{A}_{\mathrm{sc}}$ contains dimensionless neighborhood-scale multipliers; $d_{\min}$ and $d_{\max}$ bound admissible structural distances between neighboring candidates; and Δ_c is the reference complexity radius. The factor κ_{MAD} scales the median absolute deviation (MAD), δ_{loc} lower-bounds the local dispersion scale, and B clips the resulting knee scores. Finally, $z_{\max}$ bounds the robust standardized score before sigmoid normalization, and s_{neu} is the neutral normalized score assigned to a degenerate finite candidate pool. We use

$$\mathrm{clip}(u, a, b) = \min\{ \max\{u, a\}, b \},$$

and $\mathbb{I}[\cdot]$ denotes the indicator function.

Structural distance. For an expression g, let $\mathrm{sig}(g)$ denote its constant-free structural signature, and let $\mathcal{T}_{\mathrm{ng}}(g)$ contain the unigram, bigram, and trigram tokens extracted from this signature. The structural distance is the Jaccard distance

$$d_{\mathrm{str}}(g_a, g_b) = 1 - \frac{|\mathcal{T}_{\mathrm{ng}}(g_a) \cap \mathcal{T}_{\mathrm{ng}}(g_b)|}{|\mathcal{T}_{\mathrm{ng}}(g_a) \cup \mathcal{T}_{\mathrm{ng}}(g_b)|}.$$

It is used only to form structurally comparable local neighborhoods.

Multi-scale knee score on the validation-loss–complexity frontier.

(1) Validation-loss–complexity coordinates. For candidate g, let $c(g)$ denote the weighted expression-tree complexity reported in the PySR equation table under the configured operator and variable complexity weights, and define

$$z(g) = \log \overline{L}_{\mathrm{val},j}(g).$$

The numerical floor ensures that the logarithm is well defined, while the logarithmic transformation limits the influence of the potentially large dynamic range of validation losses.

(2) Validation screening and scale-dependent frontier neighborhoods. Let $\mathcal{P}_{j,s}$ be the valid, operator-admissible, and semantically distinct candidates in the output-local PySR equation table. Define

$$L^{\star}_{j,s} = \min_{g \in \mathcal{P}_{j,s}} \overline{L}_{\mathrm{val},j}(g), \qquad \mathcal{E}_{j,s} = \left\{ g \in \mathcal{P}_{j,s} : \overline{L}_{\mathrm{val},j}(g) \le \rho_{\mathrm{val}} L^{\star}_{j,s} \right\}.$$

Thus, only candidates with near-best validation accuracy are eligible for selection. Separately, let

$$\mathcal{F}_{j,s} = \mathrm{ValFrontier}(\mathcal{P}_{j,s})$$

denote the validation-loss–complexity frontier formed from the complete valid candidate pool, using one lowest-effective-loss representative at each complexity and retaining only successive validation improvements.

For each $g \in \mathcal{E}_{j,s}$ and $a \in \mathcal{A}_{\mathrm{sc}}$, define the scale-specific complexity radius

$$r_a = \max\{1, a\Delta_c\}.$$

Among $g' \in \mathcal{F}_{j,s}$ satisfying

$$d_{\min} \le d_{\mathrm{str}}(g, g') \le d_{\max}, \qquad |c(g') - c(g)| \le r_a,$$

let $\mathcal{N}_a^-(g)$ and $\mathcal{N}_a^+(g)$ denote the simpler and more-complex neighbors of g. The complexity radius localizes the comparison, $d_{\min}$ excludes near-duplicate structures, and $d_{\max}$ excludes structurally unrelated alternatives. Multiple scales reduce sensitivity to a single neighborhood radius.

(3) Median neighbor representatives. For nonempty neighbor sets, define

$$c_a^{\pm} = \mathrm{median}_{g' \in \mathcal{N}_a^{\pm}(g)}\, c(g'), \qquad z_a^{\pm} = \mathrm{median}_{g' \in \mathcal{N}_a^{\pm}(g)}\, z(g').$$

The points (c_a^-, z_a^-) and (c_a^+, z_a^+) robustly summarize the two sides of g without relying on individual neighboring expressions.

(4) Local knee criterion. When both neighbor sets are nonempty, the chord between their representatives gives the reference log loss at complexity $c(g)$:

$$\widehat{z}_a(g) = z_a^- + \frac{c(g) - c_a^-}{c_a^+ - c_a^-} \left(z_a^+ - z_a^- \right).$$

The residual $\widehat{z}_a(g) - z(g)$ measures the vertical improvement of the candidate relative to this local linear trade-off.

Proposition B.1 (Local chord residual)**.** *Let*

$$\Delta_a^- = c(g) - c_a^-, \qquad \Delta_a^+ = c_a^+ - c(g)$$

denote the complexity gaps to the simpler and more-complex local representatives, respectively. Define the corresponding left and right local secant slopes as

$$m_a^- = \frac{z(g) - z_a^-}{\Delta_a^-}, \qquad m_a^+ = \frac{z_a^+ - z(g)}{\Delta_a^+}.$$

Then

$$\widehat{z}_a(g) - z(g) = \frac{\Delta_a^- \Delta_a^+}{\Delta_a^- + \Delta_a^+} \left(m_a^+ - m_a^- \right).$$

Proof. Since $c_a^+ - c_a^- = \Delta_a^- + \Delta_a^+$ and

$$z_a^+ - z_a^- = \Delta_a^- m_a^- + \Delta_a^+ m_a^+,$$

substitution into the definition of $\widehat{z}_a(g)$ yields

$$\begin{aligned} \widehat{z}_a(g) - z(g) &= -\Delta_a^- m_a^- + \frac{\Delta_a^-}{\Delta_a^- + \Delta_a^+} \left(\Delta_a^- m_a^- + \Delta_a^+ m_a^+ \right) \\ &= \frac{\Delta_a^- \Delta_a^+}{\Delta_a^- + \Delta_a^+} \left(m_a^+ - m_a^- \right). \end{aligned}$$

□

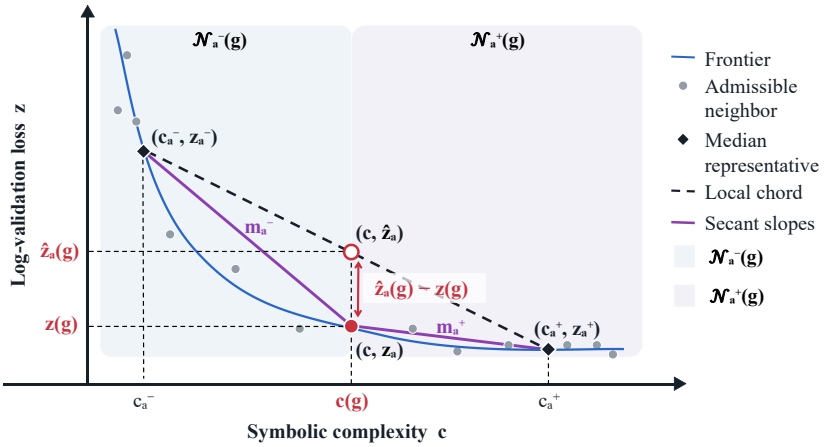


Figure 6: Geometry of the scale-specific frontier-knee criterion. The dashed chord between the median simpler and more-complex representatives gives $\widehat{z}_a(g)$ at complexity $c(g)$; the vertical residual $\widehat{z}_a(g) - z(g)$ is the local chord deviation underlying the knee score.

On a decreasing frontier, a positive residual means $m_a^+ > m_a^-$: the validation loss decreases more steeply before g than after it. Thus, further complexity provides diminishing marginal improvement, and g is locally knee-like. A near-zero residual indicates an approximately linear local trend, whereas a negative residual indicates that more-complex expressions continue to provide comparatively stronger improvement. Figure 6 illustrates this geometry.

(5) MAD-normalized score at each scale. Define the local log-loss collection

$$\mathcal{Z}_a(g) = \{z(g)\} \cup \{z(g') : g' \in \mathcal{N}_a^-(g) \cup \mathcal{N}_a^+(g)\}.$$

For a finite collection $\mathcal{Z}$, let

$$\mathrm{MAD}(\mathcal{Z}) = \mathrm{median}_{u \in \mathcal{Z}} \, |u - \mathrm{median}(\mathcal{Z})| \, .$$

Set

$$\varsigma_a(g) = \max\{\kappa_{\mathrm{MAD}} \, \mathrm{MAD}(\mathcal{Z}_a(g)), \delta_{\mathrm{loc}}\} \, .$$

The scale-specific standardized score is

$$s_a(g) = \mathrm{clip}\left(\frac{\widehat{z}_a(g) - z(g)}{\varsigma_a(g)}, -B, B \right) .$$

Thus, $s_a(g)$ expresses the chord deviation in units of local log-loss variation. In this work, $\kappa_{\mathrm{MAD}} = \frac{1}{\Phi_{\mathrm{N}}^{-1}(0.75)} \approx 1.4826$, where Φ_{N} denotes the standard-normal cumulative distribution function. This places the scaled MAD on the standard-deviation scale under a Gaussian reference distribution; no Gaussian model is assumed for the candidate losses. The floor δ_{loc} and clipping bound B prevent nearly degenerate neighborhoods and numerical extremes from dominating.

(6) Cross-scale consistency and raw score. Because the desired knee is a point beyond which additional complexity has diminishing benefit, positive $s_a(g)$ is favorable. Let

$$\mathcal{A}_g = \{a \in \mathcal{A}_{\mathrm{sc}} : \mathcal{N}_a^-(g) \neq \varnothing, \mathcal{N}_a^+(g) \neq \varnothing\}$$

and define

$$p_+(g) = \frac{1}{|\mathcal{A}_g|} \sum_{a \in \mathcal{A}_g} \mathbb{I}[s_a(g) > 0].$$

Define the median multi-scale evidence

$$m_{\mathrm{sc}}(g) = \mathrm{median}_{a \in \mathcal{A}_g} \, s_a(g).$$

The raw frontier score is

$$S_{\mathrm{front}}^{\mathrm{raw}}(g) = \mathrm{clip}\left[\begin{cases} m_{\mathrm{sc}}(g) \left(\dfrac{1 + p_+(g)}{2} \right), & m_{\mathrm{sc}}(g) > 0, \\ m_{\mathrm{sc}}(g), & m_{\mathrm{sc}}(g) \leq 0, \end{cases} - B, B \right] .$$

Thus, cross-scale persistence strengthens only positive knee evidence; non-positive median evidence is left unattenuated, avoiding sign-dependent weakening of negative evidence. The raw score is invalid when $\mathcal{A}_g = \varnothing$.

Cross-restart consolidation and normalization. For output j, candidates exported by all restarts are grouped by the constant-free signature $\mathrm{sig}(g)$. For each repeated skeleton, the representative with the lowest raw validation loss is retained, while its valid values of $S_{\mathrm{front}}^{\mathrm{raw}}$ are aggregated by their median. Let $\mathcal{C}_j$ denote the resulting cross-restart representative pool and define

$$L_j^\star = \min_{g \in \mathcal{C}_j} \overline{L}_{\mathrm{val},j}(g), \qquad \mathcal{E}_j = \left\{ g \in \mathcal{C}_j : \overline{L}_{\mathrm{val},j}(g) \le \rho_{\mathrm{val}} L_j^\star \right\}.$$

Let

$$\mathcal{S}_j = \{ S_{\mathrm{front}}^{\mathrm{raw}}(g) : g \in \mathcal{E}_j \text{ has a valid two-sided frontier score} \},$$

and set

$$\mu_j^{\mathrm{front}} = \mathrm{median}(\mathcal{S}_j), \qquad \varsigma_j^{\mathrm{front}} = \kappa_{\mathrm{MAD}} \, \mathrm{MAD}(\mathcal{S}_j).$$

When this scale is nondegenerate,

$$\widetilde{S}_{\mathrm{front}}(g) = \mathrm{sigm}\left(\mathrm{clip}\left[\frac{S_{\mathrm{front}}^{\mathrm{raw}}(g) - \mu_j^{\mathrm{front}}}{\varsigma_j^{\mathrm{front}}}, -z_{\max}, z_{\max} \right] \right), \qquad \mathrm{sigm}(u) = \frac{1}{1+e^{-u}}.$$

If the MAD scale is degenerate, the ordinary standard deviation is used as a fallback. A singleton or fully tied finite pool receives the neutral value s_{neu}. If no candidate has a valid two-sided frontier score, the frontier component is zero and ranking is validation-driven. When valid two-sided evidence exists for only part of the pool, candidates without such evidence remain eligible but receive zero frontier contribution rather than a neutral structural reward or hard exclusion.

Stage-I composite score and final selection rule. For each $g \in \mathcal{E}_j$, define

$$S_{\mathrm{val}}(g) = \mathrm{clip}\left[1 - \frac{\log\left(\overline{L}_{\mathrm{val},j}(g)/L_j^\star\right)}{\log \rho_{\mathrm{val}}}, 0, 1 \right].$$

Thus, the validation-best candidate receives $S_{\mathrm{val}} = 1$, whereas a candidate at the eligibility boundary receives $S_{\mathrm{val}} = 0$. The final score uses structural evidence only when it is available:

$$S(g) = \begin{cases} (1 - w_{\mathrm{val}}) \widetilde{S}_{\mathrm{front}}(g) + w_{\mathrm{val}} S_{\mathrm{val}}(g), & \text{valid two-sided frontier evidence}, \\ w_{\mathrm{val}} S_{\mathrm{val}}(g), & \text{otherwise}. \end{cases} \tag{10}$$

Final selection follows two cases. Lexicographic comparison proceeds from left to right, so earlier tuple entries have higher selection priority.

(i) Numerical-floor override. Define

$$\mathcal{C}_j^{\mathrm{floor}} = \left\{ g \in \mathcal{C}_j : L_{\mathrm{val},j}(g) \le \epsilon_j^{\mathrm{sel}} \right\}.$$

If $\mathcal{C}_j^{\mathrm{floor}} \neq \varnothing$, Stage I selects

$$g_j^{\mathrm{core}} = \operatorname*{lex\,argmin}_{g \in \mathcal{C}_j^{\mathrm{floor}}} \left(c(g), L_{\mathrm{val},j}(g) \right).$$

The priority order is (1) lower complexity; (2) lower raw validation loss. Thus, validation-loss differences below the prescribed selection resolution cannot justify a more complex expression.

(ii) Composite-score selection. If $\mathcal{C}_j^{\mathrm{floor}} = \varnothing$, the score in equation 10, including its validation-only boundary fallback, is applied and Stage I selects

$$g_j^{\mathrm{core}} = \operatorname*{lex\,argmax}_{g \in \mathcal{E}_j} \left(S(g), -\overline{L}_{\mathrm{val},j}(g), -c(g), -L_{\mathrm{val},j}(g) \right).$$

The priority order is (1) higher composite score; (2) lower effective validation loss; (3) lower complexity; (4) lower raw validation loss. The resulting expression is retained as the Stage-I core g_j^{core}.

The numerical scoring and selection controls used in the reported experiments are consolidated in Table 2.

Stage-I search controls and pseudocode. Let $\mathcal{G}_{\mathrm{SR}}$ denote the admissible SR grammar, including the prescribed primitive operators and structural constraints, and let $\mathcal{I}_{\mathrm{SR}}$ denote an optional initial-expression library. When $\mathcal{I}_{\mathrm{SR}} \neq \varnothing$, its expressions are injected as soft, unprotected SR initializations with reinjection fraction ρ_{init}; setting $\mathcal{I}_{\mathrm{SR}} = \varnothing$ disables this prior.

The Stage-I controls are

$$\Gamma_{\mathrm{SR}} = \{\Lambda, \tau_{\mathrm{by}}, N_{\mathrm{rst}}, N_{\mathrm{pop}}^{\mathrm{tot}}, N_{\mathrm{ind}}, T_{\mathrm{SR}}, \rho_{\mathrm{init}}, \Gamma_{\mathrm{MP}}, \Gamma_{\mathrm{score}}\},$$

where Λ is the STLSQ-threshold set, τ_{by} the per-output SINDy-bypass threshold, N_{rst} the number of independent PySR restarts, $N_{\mathrm{pop}}^{\mathrm{tot}}$ the total population budget per restart, N_{ind} the number of individuals per population, and T_{SR} the search budget per restart.

Optional machine-precision stopping is controlled by

$$\Gamma_{\mathrm{MP}} = \{\iota_{\mathrm{MP}}, \epsilon_{\mathrm{MP}}^{\mathrm{abs}}, \epsilon_{\mathrm{MP}}^{\mathrm{rel}}, T_{\mathrm{MP}}^{\mathrm{chk}}, T_{\mathrm{MP}}^{\mathrm{min}}, \iota_{\mathrm{MP}}^{\mathrm{rst}}\}.$$

Here, ι_{MP} enables within-restart stopping; $\epsilon_{\mathrm{MP}}^{\mathrm{abs}}$ and $\epsilon_{\mathrm{MP}}^{\mathrm{rel}}$ define absolute and scale-relative MSE thresholds; $T_{\mathrm{MP}}^{\mathrm{chk}}$ and $T_{\mathrm{MP}}^{\mathrm{min}}$ specify the checking interval and minimum search length; and $\iota_{\mathrm{MP}}^{\mathrm{rst}}$ allows subsequent restarts to be skipped after both training and validation criteria are satisfied for all unresolved outputs.

If m outputs remain unresolved, each receives

$$N_{\mathrm{pop}}^{\mathrm{out}} = \left\lfloor \frac{N_{\mathrm{pop}}^{\mathrm{tot}}}{m} \right\rfloor$$

populations; any remainder is unused.

The key method-level controls are summarized in Table 2. Benchmark-specific symbolic search budgets and prior settings remain reported in the corresponding experimental appendices.

The complete Stage-I symbolic-skeleton discovery procedure is summarized in Algorithm 1.

B.2 Stage II: Compilation, Augmentation, and Fixed-Support Refinement

B.2.1 Uniform Augmentation Dictionaries

The augmentation controls are

$$\Gamma_{\mathrm{aug}} = \{\chi, k, N_{\mathrm{neu}}, \varphi, \mathcal{Q}, \mathcal{S}_\alpha, r_{\mathrm{pool}}, \tau_{\mathrm{var}}, \tau_{\mathrm{corr}}, \epsilon_s, s_{\mathrm{aug}}, \iota_{\mathrm{span}}, \iota_{\mathrm{lin}}\},$$

where $\chi \in \{\mathrm{poly}, \mathrm{neu}\}$ selects the augmentation type. The polynomial option uses total degree k. The neural option retains N_{neu} fixed ridge features with activation φ, projection-quantile set $\mathcal{Q}$, scale-factor set $\mathcal{S}_\alpha$, candidate-pool multiplier r_{pool}, variance and correlation thresholds τ_{var} and τ_{corr}, scale floor ϵ_s, and generation seed s_{aug}. The switches ι_{span} and ι_{lin} determine whether the generated directions span the branch-state dimension and whether the original branch coordinates are retained as augmentation terms, respectively. The reported neural augmentation uses $\varphi = \tanh$.

For every active branch from state $\mathbf{h}^{(k)}$ to layer $\ell + 1$, the compiled structural dictionary is supplemented by

$$\mathbf{\Phi}_k^{(\ell)}(\mathbf{h}^{(k)}) = \mathrm{Dedup}_{\mathrm{stable}} \begin{bmatrix} \mathbf{\Phi}_{\mathrm{str},k}^{(\ell)}(\mathbf{h}^{(k)}) \\ \mathbf{a}_{\chi,k}^{(\ell)}(\mathbf{h}^{(k)}; \Gamma_{\mathrm{aug}}) \end{bmatrix}, \qquad \chi \in \{\mathrm{poly}, \mathrm{neu}\}.$$

Stable descriptor deduplication preserves first occurrence, so a structural term is retained when the augmentation contains an exact duplicate.

Polynomial augmentation. For $\mathbf{h} \in \mathbb{R}^d$, the total-degree-$k$ augmentation is

$$\mathbf{a}_k^{\mathrm{poly}}(\mathbf{h}) = \begin{bmatrix} 1 \\ \{\mathbf{h}^{\boldsymbol{\beta}}\}_{1 \leq |\boldsymbol{\beta}| \leq k} \end{bmatrix}, \qquad \mathbf{h}^{\boldsymbol{\beta}} = \prod_{i=1}^{d} h_i^{\beta_i}.$$

It contains $\binom{d+k}{k}$ terms, including the constant. The reported low-dimensional Feynman and generator experiments use $k = 2$.

Algorithm 1: Stage I: protected-output symbolic skeleton discovery

Input: $\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}, \mathbf{\Phi}_{\mathrm{SINDy}}, \mathcal{G}_{\mathrm{SR}}$, optional $\mathcal{I}_{\mathrm{SR}}, \Gamma_{\mathrm{SR}}$

Output: A final single-layer PhDN if all outputs bypass; otherwise $\{g_j^{\mathrm{core}}\}_{j=1}^{n_y}$ and retained compiler-fallback rankings

Unresolved-output index set: $\mathcal{U} \leftarrow \varnothing$; Accumulated SR candidate pool for output j: $\mathcal{C}_j \leftarrow \varnothing$, $j = 1, \ldots, n_y$

`// Protect outputs already represented by the flat dictionary`

for $j \leftarrow 1$ **to** n_y **do**

  $(g_j^{\mathrm{SINDy}}, L_j^{\mathrm{SINDy}}) \leftarrow \mathtt{STLSQSelect}(\mathbf{\Phi}_{\mathrm{SINDy}}, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}, \Lambda, j)$

  if $L_j^{\mathrm{SINDy}} \leq \tau_{\mathrm{by}}$ **then**

    $g_j^{\mathrm{core}} \leftarrow g_j^{\mathrm{SINDy}}$

  else

    $\mathcal{U} \leftarrow \mathcal{U} \cup \{j\}$

if $\mathcal{U} = \varnothing$ **then**

  return *the accepted single-layer dictionary model*

$m \leftarrow |\mathcal{U}|$; $N_{\mathrm{pop}}^{\mathrm{out}} \leftarrow \lfloor N_{\mathrm{pop}}^{\mathrm{tot}}/m \rfloor$

`// Multi-output SR; an empty` $\mathcal{I}_{\mathrm{SR}}$ `disables soft initialization`

for $s \leftarrow 1$ **to** N_{rst} **do**

  $(\{\mathcal{P}_{j,s}\}_{j\in\mathcal{U}}, \zeta_s^{\mathrm{tr}}, \widehat{\mathbf{Y}}_{s,\mathrm{val}}^{\mathrm{HOF}}) \leftarrow$ $\mathtt{NativeMultiOutputPySR}(\mathbf{X}_{\mathrm{tr}}, \mathbf{Y}_{\mathrm{tr},\mathcal{U}}, \mathcal{G}_{\mathrm{SR}}, \mathcal{I}_{\mathrm{SR}}, N_{\mathrm{pop}}^{\mathrm{out}}, N_{\mathrm{ind}}, T_{\mathrm{SR}}, \Gamma_{\mathrm{MP}}, \mathrm{seed}_s)$

  foreach $j \in \mathcal{U}$ **do**

    $\mathcal{E}_{j,s} \leftarrow \mathtt{ScoreRestartPool}(\mathcal{P}_{j,s}, \mathcal{D}_{\mathrm{val}}, j, \Gamma_{\mathrm{score}})$ `// filter candidates, evaluate validation loss, and attach raw frontier scores`

    $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \mathcal{E}_{j,s}$

  if $\iota_{\mathrm{MP}}\iota_{\mathrm{MP}}^{\mathrm{rst}}\zeta_s^{\mathrm{tr}} = 1$ **then**

    if $\mathtt{MachinePrecisionValCheck}(\widehat{\mathbf{Y}}_{s,\mathrm{val}}^{\mathrm{HOF}}, \mathbf{Y}_{\mathrm{val},\mathcal{U}}, \Gamma_{\mathrm{MP}}) = 1$ **then**

      break

`// Cross-restart consolidation and output-wise selection`

foreach $j \in \mathcal{U}$ **do**

  $\mathcal{C}_j \leftarrow \mathtt{MergeBySignature}(\mathcal{C}_j)$ `// best-validation representative and median valid raw frontier score per skeleton`

  Form $\mathcal{E}_j$, normalize the retained frontier scores, and compute $S_{\mathrm{val}}(g)$ and $S(g)$ using the definitions above `// candidates without two-sided frontier evidence use validation-only score`

  $\mathcal{C}_j^{\mathrm{floor}} \leftarrow \{g \in \mathcal{C}_j : L_{\mathrm{val},j}(g) \leq \epsilon_j^{\mathrm{sel}}\}$

  if $\mathcal{C}_j^{\mathrm{floor}} \neq \varnothing$ **then**

    $g_j^{\mathrm{core}} \leftarrow \mathrm{lex\,argmin}_{g\in\mathcal{C}_j^{\mathrm{floor}}} \big(c(g), L_{\mathrm{val},j}(g)\big)$

  else

    Apply the frontier score where valid and the validation-only fallback otherwise

    $g_j^{\mathrm{core}} \leftarrow \mathrm{lex\,argmax}_{g\in\mathcal{E}_j} \big(S(g), -\overline{L}_{\mathrm{val},j}(g), -c(g), -L_{\mathrm{val},j}(g)\big)$

  Retain the ordered compiler-fallback ranking for output j

return $\{g_j^{\mathrm{core}}\}_{j=1}^{n_y}$ *and the retained rankings*

Fixed neural-ridge augmentation. The neural augmentation constructs a compact but diverse set of fixed nonlinear features. For a prescribed budget N_{neu}, the design promotes diversity across projection directions, transition locations, and nonlinear scales while removing nearly constant, duplicate, or strongly correlated responses. Only the outer PhDN coefficients of the retained features are subsequently trained.

For branch-state samples $\{\mathbf{h}_m\}_{m=1}^{N_{\mathrm{tr}}}$, define

$$\widetilde{\mathbf{h}}_m = \mathbf{D}^{-1}(\mathbf{h}_m - \boldsymbol{\mu}), \qquad \mu_i = \frac{1}{N_{\mathrm{tr}}} \sum_{m=1}^{N_{\mathrm{tr}}} h_{m,i},$$

$$D_{ii} = \max\{\mathrm{std}(h_{1,i}, \ldots, h_{N_{\mathrm{tr}},i}), \epsilon_s\}.$$

Here, $\mathbf{D} = \mathrm{diag}(D_{11}, \ldots, D_{dd}) \in \mathbb{R}^{d\times d}$. This standardization prevents coordinates with large numerical scales from dominating feature generation; the structural PhDN states retain their original coordinates.

A candidate pool larger than the final feature budget is first generated so that redundant or numerically uninformative features can be removed while still retaining N_{neu} diverse ridge terms. Let d be the dimension of the current branch state. We set

$$N_{\mathrm{cand}} = \lceil r_{\mathrm{pool}} N_{\mathrm{neu}} \rceil, \qquad N_{\mathrm{dir}} = \max\left\{ \left\lceil \frac{N_{\mathrm{cand}}}{|\mathcal{Q}||\mathcal{S}_\alpha|} \right\rceil, \iota_{\mathrm{span}} d \right\}.$$

Each projection direction is combined with every $q \in \mathcal{Q}$ and $\alpha \in \mathcal{S}_\alpha$, while $\iota_{\mathrm{span}} = 1$ additionally ensures at least d directions.

Using seed s_{aug}, Gaussian matrices are orthogonalized in blocks by QR factorization, sign-normalized, concatenated, and truncated to obtain N_{dir} projection directions $\{\mathbf{u}_p\}$. This promotes directional diversity before response-based redundancy screening.

For a fixed direction $\mathbf{u}_p$, the scalar projections of all training samples are

$$z_{mp} = \mathbf{u}_p^\top \widetilde{\mathbf{h}}_m, \qquad m = 1, \ldots, N_{\mathrm{tr}},$$

with directional scale

$$s_p = \max\{\mathrm{std}(z_{1p}, \ldots, z_{N_{\mathrm{tr}}p}), \epsilon_s\}.$$

Transition locations are placed at representative positions of the projected data distribution:

$$t_{pq} = \mathrm{Quantile}_q(z_{1p}, \ldots, z_{N_{\mathrm{tr}}p}), \qquad q \in \mathcal{Q}.$$

Thus, different quantiles distribute nonlinear transitions across the observed projection range rather than concentrating them around one common location.

The scale parameter α is selected from the prescribed set $\mathcal{S}_\alpha$ rather than optimized internally. Since the projected coordinate is normalized by s_p, smaller α produces a broader transition and larger α a sharper one. The candidate features are

$$\phi_{p,q,\alpha}(\widetilde{\mathbf{h}}) = \varphi\left(\alpha \frac{\mathbf{u}_p^\top \widetilde{\mathbf{h}} - t_{pq}}{s_p} \right), \qquad q \in \mathcal{Q}, \quad \alpha \in \mathcal{S}_\alpha.$$

Hence, $\mathbf{u}_p$, t_{pq}, and α control the projection direction, transition location, and transition scale, respectively. The same affine ridge arguments are converted back to the original branch coordinates, so retained features are stored and evaluated as ordinary explicit dictionary entries.

The candidate pool is then screened for numerical usefulness and redundancy. Features with non-finite responses or empirical standard deviation below τ_{var} are removed, followed by exact symbolic deduplication. A greedy absolute-correlation screen with threshold τ_{corr} preferentially forms a subset with distinct response patterns. If fewer than N_{neu} features remain, the remaining finite unique candidates are restored. Column-pivoted QR is then applied to select the N_{neu} numerically most complementary responses, whose centered response matrix is required to satisfy the prescribed numerical-rank condition.

The resulting augmentation contains a constant, optionally the d linear coordinates when $\iota_{\mathrm{lin}} = 1$, and the N_{neu} retained ridge terms. All normalization statistics and internal ridge parameters remain

fixed after construction. Their outer PhDN coefficients are initialized to zero, remain inactive during Stage II fixed-support refinement, and become trainable when the full augmented support is released in Stage III. The augmentation bank is deliberately constructed from the exact Stage-I compiled states before Stage-II refinement and is then frozen. Consequently, PhDN-Fixed and full PhDN share the same compiled PhDN topology and admissible dictionary; Stage III differs only by releasing the zero-initialized augmentation support rather than regenerating features from the Stage-II-refined states.

This finite data-dependent bank is a practical feature-construction strategy for obtaining a compact and numerically diverse augmentation. No universal approximation claim is made for a fixed value of N_{neu}. The density result in Appendix C.3 instead concerns arbitrarily increasing finite truncations of a separate target-independent dense master bank.

Stage-II controls and pseudocode. For a PhDN DAG $\mathcal{H}$, coefficient collection $\mathbf{W}$, and active support $\mathcal{M}$, let $F_{\mathcal{H},\mathbf{W},\mathcal{M}}$ denote the corresponding forward map and define

$$L_{\mathrm{val}}(\mathcal{H},\mathbf{W},\mathcal{M}) = \frac{1}{N_{\mathrm{val}} n_y} \left\| F_{\mathcal{H},\mathbf{W},\mathcal{M}}(\mathbf{X}_{\mathrm{val}}) - \mathbf{Y}_{\mathrm{val}} \right\|_{\mathrm{F}}^2 .$$

All nonlinear least-squares evaluations use the exact-operator feasibility safeguards of the PhDN specification; invalid or nonfinite evaluations receive a penalty and cannot be retained as validation-best checkpoints.

The fixed-support controls are

$$\Gamma_{\mathrm{II}} = \{\epsilon_{\mathrm{rep}}^{\mathrm{abs}}, \epsilon_{\mathrm{rep}}^{\mathrm{rel}}, T_{\mathrm{II}}, \eta_{\mathrm{II}}\},$$

where $\epsilon_{\mathrm{rep}}^{\mathrm{abs}}$ and $\epsilon_{\mathrm{rep}}^{\mathrm{rel}}$ define the absolute and relative Stage-I reproduction tolerances, T_{II} is the fixed-support nonlinear least-squares budget, and η_{II} is the maximum relative validation increase accepted after refinement. We use $\epsilon_{\mathrm{mach}} > 0$ as a machine-precision scale floor in the reproduction test. The numerical Stage-II controls used in the reported experiments are given in Table 2.

BP-assisted continuous refinement. We use backpropagation (BP) to denote reverse chain-rule differentiation through the PhDN DAG for assembling the analytic residual Jacobian. This Jacobian is supplied to bounded nonlinear least squares (NLS), which performs the actual parameter update. Thus, BP provides the derivative information, while NLS supplies the continuous optimization step. The reported implementation uses MATLAB `lsqnonlin` with finite coefficient bounds and analytic Jacobians. The default coefficient interval is $[-3, 3]$, with StepTolerance 10^{-10} and OptimalityTolerance 10^{-8}. Invalid or nonfinite model evaluations receive a finite penalty and cannot be retained as validation-best checkpoints; unsuccessful refinement therefore falls back to the incoming or best finite validation-accepted checkpoint.

The complete Stage-II compilation, augmentation, and fixed-support refinement procedure is summarized in Algorithm 2.

B.3 Stage III: Full-Support Refinement and Pruning

The Stage-III controls are

$$\Gamma_{\mathrm{III}} = \{T_{\mathrm{III}}, p_{\mathrm{es}}, \eta_{\mathrm{ref}}, N_{\mathrm{pr}}, T_{\mathrm{pr}}, \tau_{\mathrm{abs}}, \tau_{\mathrm{rel}}, \eta_{\mathrm{prune}}\},$$

where T_{III} is the full-support refinement budget, p_{es} is the validation early-stopping patience, and η_{ref} controls acceptance of the refined checkpoint. The pruning stage performs at most N_{pr} passes, each using a support-fixed reoptimization budget T_{pr}. The thresholds τ_{abs} and τ_{rel} determine the contribution-based pruning cutoff, while η_{prune} bounds the accepted relative validation increase. The numerical Stage-III refinement, early-stopping, and pruning controls used in the reported experiments are consolidated in Table 2.

For a coefficient associated with the branch from state k to layer $\ell + 1$, u indexes the destination coordinate and b the corresponding branch-dictionary entry.

The complete Stage-III full-support refinement and pruning procedure is summarized in Algorithm 3.

Algorithm 2: Stage II: expression-to-PhDN compilation and fixed-support refinement

Input: Stage-I core expressions $\{g_j^{\mathrm{core}}\}_{j=1}^{n_y}$, $\mathcal{D}_{\mathrm{tr}}$, $\mathcal{D}_{\mathrm{val}}$, Γ_{aug}, and Γ_{II}

Output: Augmented PhDN DAG $\mathcal{H}$, coefficients $\mathbf{W}^{\mathrm{II}}$, structural seed support $\mathcal{M}_{\mathrm{str}}$, and full admissible support $\mathcal{M}_{\mathrm{full}}$

// Step 1: Parse expressions and identify reusable symbolic quantities

Initialize a canonical subexpression cache $\mathcal{C}_{\mathrm{sub}}$

for $j \leftarrow 1$ **to** n_y **do**

  Parse g_j^{core} into a symbolic expression tree $\mathcal{T}_j$ and rewrite it into the supported fine-grained primitive form; represent addition and subtraction by linear combinations and division by reciprocal followed by multiplication

  Traverse $\mathcal{T}_j$ from leaves to root

  foreach *subexpression node* $v \in \mathcal{T}_j$ **do**

    Compute its canonical key $\mathrm{key}(v)$

    if $\mathrm{key}(v) \in \mathcal{C}_{\mathrm{sub}}$ **then**

      Reuse the previously registered symbolic quantity

    else

      Register v, together with its operand dependencies, as a new symbolic quantity in $\mathcal{C}_{\mathrm{sub}}$

  Associate the root quantity of $\mathcal{T}_j$ with output y_j

// Step 2: Map the shared symbolic dependencies into the structural PhDN

Assign the registered symbolic quantities to PhDN state coordinates according to dependency depth. For a multi-argument primitive whose operands are produced at different depths, propagate the earlier operands through identity bases until all required operands are jointly available in one state; then evaluate the primitive through a local dictionary branch. Represent repeated quantities once and reuse them wherever possible

Construct the resulting structural PhDN topology $\mathcal{H}$ and its active branch set $\mathcal{B}$

foreach *active branch* $(k, \ell) \in \mathcal{B}$ **do**

  Construct the ordered structural dictionary $\mathbf{\Phi}_{\mathrm{str},k}^{(\ell)}$ containing the required identity pass-through terms and primitive operations for that branch

Determine the structural coefficient seed $\mathbf{W}^{\mathrm{str}}$ so that the PhDN states, branch dictionaries, and linear combinations exactly represent $\{g_j^{\mathrm{core}}\}_{j=1}^{n_y}$

// Step 3: Construct and freeze augmentation from the exact Stage-I compiled states

Evaluate the structural PhDN on $\mathbf{X}_{\mathrm{tr}}$ to obtain the branch-state samples required for data-aware augmentation

foreach *active branch* $(k, \ell) \in \mathcal{B}$ **do**

  $\mathbf{a}_{\chi,k}^{(\ell)} \leftarrow \mathtt{BuildAugmentation}(\{\mathbf{h}_i^{(k)}\}_{i=1}^{N_{\mathrm{tr}}}, \Gamma_{\mathrm{aug}})$

  Concatenate $\mathbf{\Phi}_{\mathrm{str},k}^{(\ell)}$ with $\mathbf{a}_{\chi,k}^{(\ell)}$, apply stable exact-descriptor deduplication, and remap the structural seed coefficients so that the compiled symbolic mapping is unchanged

// The resulting augmentation bank remains fixed through Stages II--III; only its outer coefficients are released in Stage III

Set every coefficient introduced only by augmentation to zero and collect the result as $\mathbf{W}^{\mathrm{seed}}$

$\mathcal{M}_{\mathrm{str}} \leftarrow \mathrm{supp}(\mathbf{W}^{\mathrm{seed}})$

$\mathcal{M}_{\mathrm{full}} \leftarrow$ the full admissible coefficient support of the augmented PhDN $\mathcal{H}$

// Step 4: Verify exact reproduction of the Stage-I symbolic model

$\widehat{\mathbf{Y}}_{\mathrm{core}} \leftarrow [g_1^{\mathrm{core}}(\mathbf{X}_{\mathrm{val}}), \ldots, g_{n_y}^{\mathrm{core}}(\mathbf{X}_{\mathrm{val}})]$

$\widehat{\mathbf{Y}}_{\mathrm{seed}} \leftarrow F_{\mathcal{H},\mathbf{W}^{\mathrm{seed}},\mathcal{M}_{\mathrm{str}}}(\mathbf{X}_{\mathrm{val}})$

assert

$$\left\| \widehat{\mathbf{Y}}_{\mathrm{seed}} - \widehat{\mathbf{Y}}_{\mathrm{core}} \right\|_{\infty} \le \epsilon_{\mathrm{rep}}^{\mathrm{abs}} + \epsilon_{\mathrm{rep}}^{\mathrm{rel}} \max\left\{ \|\widehat{\mathbf{Y}}_{\mathrm{core}}\|_{\infty}, \epsilon_{\mathrm{mach}} \right\}.$$

// Step 5: Refine only the compiled structural support

$\widetilde{\mathbf{W}}^{\mathrm{II}} \leftarrow \mathtt{FixedSupportLSQ}(\mathcal{H}, \mathbf{W}^{\mathrm{seed}}, \mathcal{M}_{\mathrm{str}}, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}, T_{\mathrm{II}})$

// BP-computed analytic Jacobian supplied to bounded NLS; coefficients outside $\mathcal{M}_{\mathrm{str}}$ remain zero and pruning is disabled.

if $L_{\mathrm{val}}(\mathcal{H}, \widetilde{\mathbf{W}}^{\mathrm{II}}, \mathcal{M}_{\mathrm{str}}) \le (1 + \eta_{\mathrm{II}}) L_{\mathrm{val}}(\mathcal{H}, \mathbf{W}^{\mathrm{seed}}, \mathcal{M}_{\mathrm{str}})$ **then**

  $\mathbf{W}^{\mathrm{II}} \leftarrow \widetilde{\mathbf{W}}^{\mathrm{II}}$

else

  $\mathbf{W}^{\mathrm{II}} \leftarrow \mathbf{W}^{\mathrm{seed}}$

return $(\mathcal{H}, \mathbf{W}^{\mathrm{II}}, \mathcal{M}_{\mathrm{str}}, \mathcal{M}_{\mathrm{full}})$

Algorithm 3: Stage III: full-support refinement and validation-controlled pruning

Input: $\mathcal{H}$, $\mathbf{W}^{\mathrm{II}}$, $\mathcal{M}_{\mathrm{str}}$, $\mathcal{M}_{\mathrm{full}}$, $\mathcal{D}_{\mathrm{tr}}$, $\mathcal{D}_{\mathrm{val}}$, and Γ_{III}
Output: $\mathbf{W}^{\star}, \mathcal{M}^{\star}$

// Step 1: Release the full augmented support
Set $\mathbf{W}^{(0)}|_{\mathcal{M}_{\mathrm{str}}} = \mathbf{W}^{\mathrm{II}}$ and $\mathbf{W}^{(0)}|_{\mathcal{M}_{\mathrm{full}} \setminus \mathcal{M}_{\mathrm{str}}} = 0$
$L_{\mathrm{seed}} \leftarrow L_{\mathrm{val}}(\mathcal{H}, \mathbf{W}^{(0)}, \mathcal{M}_{\mathrm{full}})$
// Step 2: Jointly refine the full PhDN under validation control
$(\mathbf{W}_{\mathrm{best}}, L_{\mathrm{best}}) \leftarrow \texttt{ValidationControlledLSQ}(\mathcal{H}, \mathbf{W}^{(0)}, \mathcal{M}_{\mathrm{full}}, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}, T_{\mathrm{III}}, p_{\mathrm{es}})$
// BP-assisted bounded NLS with analytic Jacobian; retain the best finite validation checkpoint.
if $L_{\mathrm{best}} \le (1+\eta_{\mathrm{ref}}) L_{\mathrm{seed}}$ **then**
  $(\mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{cur}}) \leftarrow (\mathbf{W}_{\mathrm{best}}, \mathcal{M}_{\mathrm{full}})$
else
  $(\mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{cur}}) \leftarrow (\mathbf{W}^{(0)}, \mathcal{M}_{\mathrm{full}})$
// Step 3: Prune low-contribution terms and reoptimize the retained support
for $\nu \leftarrow 1$ **to** N_{pr} **do**
  foreach $(k,\ell,u,b) \in \mathcal{M}_{\mathrm{cur}}$ **do**
    $\gamma_{k\ell,ub} \leftarrow \dfrac{1}{N_{\mathrm{tr}}} \displaystyle\sum_{i=1}^{N_{\mathrm{tr}}} \left| W_{k,ub}^{(\ell),\mathrm{cur}} \Phi_{k,b}^{(\ell)}(\mathbf{h}_i^{(k)}) \right|$
  $\tau_{\mathrm{prune}} \leftarrow \max\left\{ \tau_{\mathrm{abs}}, \tau_{\mathrm{rel}} \displaystyle\max_{(k,\ell,u,b)\in\mathcal{M}_{\mathrm{cur}}} \gamma_{k\ell,ub} \right\}$
  $\mathcal{M}_{\mathrm{pr}} \leftarrow \{(k,\ell,u,b) \in \mathcal{M}_{\mathrm{cur}} : \gamma_{k\ell,ub} > \tau_{\mathrm{prune}}\}$
  if $\mathcal{M}_{\mathrm{pr}} = \mathcal{M}_{\mathrm{cur}}$ **then**
    break
  $\mathbf{W}_{\mathrm{pr}} \leftarrow \texttt{FixedSupportLSQ}(\mathcal{H}, \mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{pr}}, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}, T_{\mathrm{pr}})$
  if $L_{\mathrm{val}}(\mathcal{H}, \mathbf{W}_{\mathrm{pr}}, \mathcal{M}_{\mathrm{pr}}) \le (1+\eta_{\mathrm{prune}}) L_{\mathrm{val}}(\mathcal{H}, \mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{cur}})$ **then**
    $(\mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{cur}}) \leftarrow (\mathbf{W}_{\mathrm{pr}}, \mathcal{M}_{\mathrm{pr}})$
  else
    break
// Step 4: Return the final validation-accepted sparse PhDN
$(\mathbf{W}^{\star}, \mathcal{M}^{\star}) \leftarrow (\mathbf{W}_{\mathrm{cur}}, \mathcal{M}_{\mathrm{cur}})$
return $(\mathbf{W}^{\star}, \mathcal{M}^{\star})$

B.4 Reported Operational Hyperparameters

Table 2 summarizes the key operational controls used by the reported three-stage experiments. Benchmark-specific operator grammars and data-generation settings remain reported in their corresponding experimental appendices.

Table 2: Key operational hyperparameters used in the reported PhDN experiments. Benchmark-specific search grammars and symbolic-search budgets are reported separately in the corresponding experimental appendices.

Component	Control	Reported value
Stage-I score	Validation/frontier weighting	$\rho_{\mathrm{val}} = 4$, $w_{\mathrm{val}} = 0.30$
Stage-I score	Multi-scale frontier controls	$\mathcal{A}_{\mathrm{sc}} = \{0.50, 0.75, 1.00\}$, $d_{\min} = 0.10$, $d_{\max} = 0.55$, $\Delta_c = 8$
Stage-I score	Robust normalization	$\kappa_{\mathrm{MAD}} = 1.4826$, $\delta_{\mathrm{loc}} = 0.05$, $B = 20$, $z_{\max} = 8$, $s_{\mathrm{neu}} = 0.5$
Stage-I bypass	STLSQ threshold set Λ	$\{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$
Stage-I bypass	Bypass threshold τ_{by}	Set by the corresponding benchmark protocol; the reported value is stated in each experimental appendix
Stage-II compilation	Reproduction tolerances	$\epsilon_{\mathrm{rep}}^{\mathrm{abs}} = 10^{-10}$, $\epsilon_{\mathrm{rep}}^{\mathrm{rel}} = 10^{-6}$
Stage-II refinement	Fixed-support budget T_{II}	At most 500 nonlinear least-squares iterations and 5×10^4 function evaluations
Stage-II refinement	Validation margin η_{II}	0
Continuous refinement	BP/NLS solver	BP-computed analytic Jacobian supplied to bounded MATLAB `lsqnonlin`; default coefficient bounds $[-3, 3]$, StepTolerance 10^{-10}, OptimalityTolerance 10^{-8}
Polynomial augmentation	Degree k	$k = 2$ in the reported polynomial-augmentation experiments
Neural-ridge augmentation	Feature family	$N_{\mathrm{neu}} = 10$ fixed ridge features per active branch, $\varphi = \tanh$
Neural-ridge augmentation	Quantiles and scales	$\mathcal{Q} = \{0.20, 0.40, 0.60, 0.80\}$, $\mathcal{S}_\alpha = \{0.5, 1, 2\}$
Neural-ridge augmentation	Pool and screening controls	$r_{\mathrm{pool}} = 3$, $\epsilon_s = 10^{-10}$, $\tau_{\mathrm{var}} = 10^{-8}$, $\tau_{\mathrm{corr}} = 0.995$
Neural-ridge augmentation	Numerical-rank safeguard	The centered selected response matrix is required to have full attainable column rank; MATLAB's default numerical tolerance in `rank` is used
Stage-III refinement	Full-support budget T_{III}	At most 500 nonlinear least-squares iterations and 5×10^4 function evaluations
Stage-III refinement	Validation patience p_{es}	3
Stage-III refinement	Acceptance margin η_{ref}	10^{-10} relative validation increase
Stage-III pruning	Pruning passes N_{pr}	1
Stage-III pruning	Contribution thresholds $(\tau_{\mathrm{abs}}, \tau_{\mathrm{rel}})$	$(10^{-4}, 0)$
Stage-III pruning	Reoptimization budget T_{pr}	At most 50 iterations and 5000 function evaluations
Stage-III pruning	Validation margin η_{prune}	10^{-4} relative validation increase

C Representability of PhDN

This appendix formalizes the symbolic closure and universal approximation properties stated in Section 3.4.

C.1 Symbolic Representability of PhDN

This subsection characterizes the exact symbolic expression class represented by finite PhDNs. Let $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ denote the input domain and let Ω be the set of admissible primitive symbolic operators. Each $\omega \in \Omega$ has finite arity m_ω, and we consider only expressions for which all operator arguments remain in their valid real-valued domains on $\mathcal{D}$. We assume that constant and identity bases are available when required, branch coefficients may take arbitrary real values, and the PhDN may have sufficient finite width and depth.

Finite symbolic expression class: Let $\mathfrak{E}(\Omega)$ denote the class of finite scalar symbolic expressions generated according to the following rules:

1. real constants and input coordinates $x_1, \ldots, x_{n_x}$ belong to $\mathfrak{E}(\Omega)$;
2. if $e_1, \ldots, e_M \in \mathfrak{E}(\Omega)$ and $a_1, \ldots, a_M \in \mathbb{R}$, then every finite linear combination

$$\sum_{j=1}^{M} a_j e_j$$

belongs to $\mathfrak{E}(\Omega)$;

3. if $\omega \in \Omega$ has arity m_ω and $e_1, \dots, e_{m_\omega} \in \mathfrak{E}(\Omega)$, then
$$\omega(e_1, \dots, e_{m_\omega})$$
belongs to $\mathfrak{E}(\Omega)$ whenever the operation is well defined on $\mathcal{D}$.

Only finitely many applications of these construction rules are allowed. Thus, $\mathfrak{E}(\Omega)$ contains exactly the finite symbolic expressions generated from the inputs, real constants, finite linear combinations, and the primitive operators in Ω.

Every expression in $\mathfrak{E}(\Omega)$ admits a finite symbolic expression tree. Its finite recursive construction directly defines such a tree, whose leaves are input coordinates or real constants and whose internal nodes are primitive-operator applications or finite linear combinations.

Exact symbolic representability of PhDN:

Let
$$\mathfrak{F}_{\mathrm{PhDN}}(\Omega)$$
denote the class of all scalar state coordinates and outputs represented by finite PhDNs whose dictionary operations are drawn from Ω.

A PhDN layer performs precisely the two constructive operations used above. For the architecture
$$\mathbf{h}^{(\ell+1)} = \sum_{k=0}^{\ell} \mathbf{W}_k^{(\ell)} \mathbf{\Phi}_k^{(\ell)}\big(\mathbf{h}^{(k)}\big),$$
each dictionary $\mathbf{\Phi}_k^{(\ell)}$ applies admissible primitive operators to previously available expressions, while the coefficient matrices and incoming-branch summation form finite linear combinations of the resulting features. Finite hidden width allows multiple intermediate expressions to be stored in parallel, and identity bases or all-previous-layer connections allow earlier expressions to be retained and reused.

Proposition C.1 (Exact symbolic expression class of PhDN)**.** *Under the assumptions above,*
$$\mathfrak{F}_{\mathrm{PhDN}}(\Omega) = \mathfrak{E}(\Omega). \tag{11}$$

Proof. We prove the two set inclusions.

Symbolic closure: $\mathfrak{F}_{\mathrm{PhDN}}(\Omega) \subseteq \mathfrak{E}(\Omega)$. The input coordinates belong to $\mathfrak{E}(\Omega)$ by definition. Suppose that all state coordinates available before a given PhDN layer belong to $\mathfrak{E}(\Omega)$. Every dictionary feature in the next layer is obtained by applying an admissible operator in Ω to previously available expressions and therefore remains in $\mathfrak{E}(\Omega)$. Each SINDy-style branch forms a finite linear combination of its dictionary features, and the layerwise summation of the resulting branch outputs yields another finite linear combination. Therefore, every next-layer coordinate remains in $\mathfrak{E}(\Omega)$. Induction over the finite network depth therefore shows that every coordinate and output of a finite PhDN belongs to $\mathfrak{E}(\Omega)$.

Exact symbolic completeness: $\mathfrak{E}(\Omega) \subseteq \mathfrak{F}_{\mathrm{PhDN}}(\Omega)$. Consider any $e \in \mathfrak{E}(\Omega)$. By its finite recursive construction, e admits a finite symbolic expression tree whose leaves are input coordinates or real constants and whose internal nodes are either primitive-operator nodes or finite linear-combination nodes. We construct a finite PhDN recursively from the leaves toward the root of this tree, showing that the subexpression associated with every tree node can be represented exactly as a coordinate of a PhDN state.

Input leaves are already available as coordinates of $\mathbf{h}^{(0)} = \mathbf{x}$. Any real constant leaf can be generated exactly from a constant dictionary basis with the corresponding branch coefficient.

Now suppose that the subexpressions associated with the children of an internal tree node have already been represented as coordinates of previously constructed PhDN states.

If the node is a finite linear combination
$$e_v = \sum_{j=1}^{M} a_j e_j,$$

the already represented subexpressions can be combined by suitable SINDy-style branches with the prescribed coefficients. When the required terms are supplied from different preceding states, the corresponding branch outputs are combined through the layerwise branch summation, yielding e_v exactly as a coordinate of the next state.

If instead the node is a primitive-operator node

$$e_v = \omega(e_1, \ldots, e_{m_\omega}), \qquad \omega \in \Omega,$$

its operands must be jointly available within a single PhDN state before the operator is evaluated. If they are already coordinates of the same state, a SINDy-style branch whose dictionary contains the corresponding operator ω evaluates e_v directly. If the operands are represented in different preceding states, identity features on suitable incoming branches first route them into distinct coordinates of a common state. A subsequent SINDy-style branch can then apply ω to those coordinates and represent e_v exactly.

Applying this construction recursively to the finite expression tree realizes its root, which is the target expression e. Because the tree contains only finitely many nodes, and each construction step introduces only finitely many state coordinates and branches, the resulting PhDN has finite width and depth. Hence

$$e \in \mathfrak{F}_{\mathrm{PhDN}}(\Omega).$$

Since $e \in \mathfrak{E}(\Omega)$ was arbitrary,

$$\mathfrak{E}(\Omega) \subseteq \mathfrak{F}_{\mathrm{PhDN}}(\Omega).$$

Both inclusions hold, proving

$$\mathfrak{F}_{\mathrm{PhDN}}(\Omega) = \mathfrak{E}(\Omega).$$

□

Proposition C.1 establishes exact structural representability rather than density in the space of all continuous functions. It also does not imply that a particular training procedure will recover a desired symbolic representation. Continuous-function density under specific polynomial and neural dictionaries is studied separately in Appendices C.2 and C.3.

C.2 Density of Polynomial-Basis PhDNs

Polynomial bases provide two complementary routes to universal approximation in PhDNs. The first increases the order of a single polynomial dictionary, whereas the second keeps the local dictionary fixed and uses compositional depth to generate higher-order polynomial structure. In function-space terms, universal approximation corresponds to density: every continuous target can be approximated arbitrarily well by functions from the model class. We first establish the density of multivariate polynomial functions and then transfer this result to polynomial SINDy branches and PhDNs.

Density of polynomial function classes:

For a multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n_x}) \in \mathbb{N}_0^{n_x}$, define

$$\mathbf{x}^{\boldsymbol{\alpha}} = \prod_{i=1}^{n_x} x_i^{\alpha_i}, \qquad |\boldsymbol{\alpha}| = \sum_{i=1}^{n_x} \alpha_i.$$

Let

$$\mathcal{P}_p(K, \mathbb{R}^{n_y})$$

denote the class of vector-valued multivariate polynomial functions on $K \subset \mathbb{R}^{n_x}$ whose componentwise total degree does not exceed p, and define

$$\mathcal{P}(K, \mathbb{R}^{n_y}) = \bigcup_{p \geq 0} \mathcal{P}_p(K, \mathbb{R}^{n_y}),$$

the class of all finite-degree vector-valued polynomial functions on K. For $\mathbf{f} : K \to \mathbb{R}^{n_y}$, we use the uniform norm

$$\|\mathbf{f}\|_{\infty,K} = \sup_{\mathbf{x} \in K} \|\mathbf{f}(\mathbf{x})\|_\infty.$$

Theorem C.2 (Density of polynomial function classes)**.** *Let $K \subset \mathbb{R}^{n_x}$ be compact. Then*

$$\overline{\mathcal{P}(K, \mathbb{R}^{n_y})}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}). \tag{12}$$

Equivalently, for every $\mathbf{f} \in C(K, \mathbb{R}^{n_y})$ and every $\epsilon > 0$, there exist a finite degree p and $\mathbf{q} \in \mathcal{P}_p(K, \mathbb{R}^{n_y})$ such that

$$\|\mathbf{f} - \mathbf{q}\|_{\infty,K} < \epsilon.$$

Proof. We first consider scalar-valued functions. Let $\mathcal{P}(K, \mathbb{R})$ denote the restrictions to K of all finite-degree real multivariate polynomials. Since $K \subset \mathbb{R}^{n_x}$ is compact, it is a compact Hausdorff space. Moreover, $\mathcal{P}(K, \mathbb{R})$ is a subalgebra of $C(K, \mathbb{R})$: finite linear combinations and pointwise products of polynomials remain polynomials. It contains all constant functions, and it separates points of K, because for any distinct $\mathbf{x}, \mathbf{z} \in K$ there exists some coordinate i such that $x_i \neq z_i$, while the coordinate polynomial $q_i(\mathbf{u}) = u_i$ satisfies $q_i(\mathbf{x}) \neq q_i(\mathbf{z})$. The real Stone–Weierstrass theorem (Stone, 1948) therefore gives

$$\overline{\mathcal{P}(K, \mathbb{R})}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}).$$

Now let

$$\mathbf{f} = [f_1, \ldots, f_{n_y}]^\top \in C(K, \mathbb{R}^{n_y}).$$

For each component f_j, the scalar result provides a finite-degree polynomial q_j satisfying

$$\sup_{\mathbf{x} \in K} |f_j(\mathbf{x}) - q_j(\mathbf{x})| < \epsilon.$$

Taking

$$p = \max_{1 \leq j \leq n_y} \deg(q_j)$$

and $\mathbf{q} = [q_1, \ldots, q_{n_y}]^\top$ gives $\mathbf{q} \in \mathcal{P}_p(K, \mathbb{R}^{n_y})$ and

$$\|\mathbf{f} - \mathbf{q}\|_{\infty,K} = \sup_{\mathbf{x} \in K} \max_{1 \leq j \leq n_y} |f_j(\mathbf{x}) - q_j(\mathbf{x})| < \epsilon.$$

This proves the vector-valued density statement. □

Density of polynomial SINDy branches:

For a finite polynomial order p, define the complete polynomial dictionary

$$\mathbf{\Phi}_p(\mathbf{x}) = [\mathbf{x}^{\boldsymbol{\alpha}}]_{|\boldsymbol{\alpha}| \leq p},$$

which contains every multivariate monomial of total degree at most p. The function class represented by a polynomial SINDy branch of order p is

$$\mathcal{H}_p^{\text{branch}} = \left\{ \mathbf{g} : K \to \mathbb{R}^{n_y} \;\middle|\; \mathbf{g}(\mathbf{x}) = \mathbf{W}\mathbf{\Phi}_p(\mathbf{x}), \quad \mathbf{W} \in \mathbb{R}^{n_y \times \dim(\mathbf{\Phi}_p)} \right\}.$$

Because the dictionary contains all monomials up to degree p and $\mathbf{W}$ provides arbitrary linear coefficients,

$$\mathcal{H}_p^{\text{branch}} = \mathcal{P}_p(K, \mathbb{R}^{n_y}).$$

Therefore, defining

$$\mathcal{H}^{\text{branch}} = \bigcup_{p \geq 0} \mathcal{H}_p^{\text{branch}},$$

we obtain

$$\mathcal{H}^{\text{branch}} = \mathcal{P}(K, \mathbb{R}^{n_y}).$$

Theorem C.2 consequently gives

$$\overline{\mathcal{H}^{\text{branch}}}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}). \tag{13}$$

Thus, a single polynomial SINDy branch is dense in the continuous vector-valued functions on K when its finite polynomial order is allowed to increase. A fixed finite order p alone yields only a finite-dimensional polynomial space and is generally not dense in $C(K, \mathbb{R}^{n_y})$.

Transfer of density to polynomial-basis PhDNs:

The branch-level density above can be transferred to PhDNs in two complementary ways.

First, if the polynomial dictionary order of a PhDN branch is allowed to increase, the result follows immediately from equation 13. A PhDN contains a single polynomial branch as a special case, with unused branches and connections assigned zero coefficients. Hence, the PhDN function class contains $\mathcal{H}^{\mathrm{branch}}$, which is already dense in $C(K, \mathbb{R}^{n_y})$.

More importantly, density can also be retained while keeping the local polynomial dictionary fixed. Consider the second-order template

$$\mathcal{A}_2(\mathbf{h}) = \{1\} \cup \{h_i\}_{i=1}^{d} \cup \{h_i h_j\}_{1 \le i \le j \le d},$$

which contains constant, identity, and pairwise-product bases but no higher-order polynomial terms. Let

$$\mathcal{H}_{\mathrm{PhDN}}^{L,\mathbf{n}}(\mathcal{A}_2)$$

denote the class of functions $\mathbf{g} : K \to \mathbb{R}^{n_y}$ represented by PhDNs with finite depth L, finite width configuration $\mathbf{n}$, and local dictionary $\mathcal{A}_2$. Define

$$\mathcal{H}_{\mathrm{PhDN}}^{\mathrm{poly}}(\mathcal{A}_2) = \bigcup_{L,\mathbf{n}} \mathcal{H}_{\mathrm{PhDN}}^{L,\mathbf{n}}(\mathcal{A}_2),$$

which collects the functions represented by all finite-depth, finite-width PhDNs using this fixed quadratic dictionary.

Theorem C.3 (Polynomial-basis PhDN density)**.** *Every finite-degree vector-valued polynomial can be represented exactly by a finite PhDN using the fixed dictionary $\mathcal{A}_2$, provided sufficient finite depth and width are available. Consequently,*

$$\overline{\mathcal{H}_{\mathrm{PhDN}}^{\mathrm{poly}}(\mathcal{A}_2)}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}). \tag{14}$$

Proof. For $r \ge 0$, define the complete monomial set

$$\mathcal{M}_r = \{\mathbf{x}^{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathbb{N}_0^{n_x},\ |\boldsymbol{\alpha}| \le r\}.$$

Suppose that all monomials in $\mathcal{M}_r$ are available as coordinates of a PhDN state. For any $\mathbf{x}^{\boldsymbol{\alpha}} \in \mathcal{M}_{2r}$, the nonnegative multi-index $\boldsymbol{\alpha}$ can be decomposed as

$$\boldsymbol{\alpha} = \boldsymbol{\beta} + \boldsymbol{\gamma}, \qquad |\boldsymbol{\beta}| \le r, \qquad |\boldsymbol{\gamma}| \le r.$$

Hence,

$$\mathbf{x}^{\boldsymbol{\alpha}} = \mathbf{x}^{\boldsymbol{\beta}} \mathbf{x}^{\boldsymbol{\gamma}}.$$

Both factors belong to $\mathcal{M}_r$, so their product is available through a quadratic term of $\mathcal{A}_2$. At the same time, the identity terms retain previously constructed monomials. With sufficient finite width, one additional layer can therefore construct and store all monomials in $\mathcal{M}_{2r}$:

$$\mathcal{M}_r \longrightarrow \mathcal{M}_{2r}.$$

The input coordinates and constant basis provide

$$\mathcal{M}_1 = \{1, x_1, \ldots, x_{n_x}\}.$$

Repeated quadratic layers then yield

$$\mathcal{M}_1 \longrightarrow \mathcal{M}_2 \longrightarrow \mathcal{M}_4 \longrightarrow \cdots \longrightarrow \mathcal{M}_{2^q}.$$

For any finite polynomial degree p, choose

$$q = \lceil \log_2 \max\{1, p\} \rceil.$$

Then $p \le 2^q$, and therefore

$$\mathcal{M}_p \subseteq \mathcal{M}_{2^q}.$$

Thus, after finitely many layers and with sufficient finite width, the PhDN contains every monomial required by any function in $\mathcal{P}_p(K, \mathbb{R}^{n_y})$. A final linear combination can therefore represent that polynomial exactly. Since this holds for every finite p,

$$\mathcal{P}(K, \mathbb{R}^{n_y}) \subseteq \mathcal{H}_{\mathrm{PhDN}}^{\mathrm{poly}}(\mathcal{A}_2).$$

By Theorem C.2, $\mathcal{P}(K, \mathbb{R}^{n_y})$ is dense in $C(K, \mathbb{R}^{n_y})$. Moreover, every finite PhDN using only $\mathcal{A}_2$ performs finitely many linear combinations and products, so its output is itself a polynomial and hence belongs to $C(K, \mathbb{R}^{n_y})$. Therefore,

$$\mathcal{P}(K, \mathbb{R}^{n_y}) \subseteq \mathcal{H}^{\mathrm{poly}}_{\mathrm{PhDN}}(\mathcal{A}_2) \subseteq C(K, \mathbb{R}^{n_y}),$$

and taking closures under the uniform norm gives

$$\overline{\mathcal{H}^{\mathrm{poly}}_{\mathrm{PhDN}}(\mathcal{A}_2)}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}).$$

□

The fixed-quadratic construction requires only consecutive-layer multiplication branches, which form a subgraph of the full all-previous-layer-connected PhDN. Thus, the additional PhDN connections are not required for density, although they may improve compositional efficiency through feature reuse. Likewise, higher-order polynomial dictionaries are unnecessary for universality, but may reduce the depth or width required by a finite architecture.

Theorem C.3 is a representational result rather than an optimization guarantee. Although compositional quadratic layers can generate high-degree polynomial structure without explicitly enumerating a high-order flat dictionary, their parameters interact through successive nonlinear compositions. Consequently, fitting such a deep representation may lead to a nonconvex and potentially ill-conditioned optimization problem. Thus, the theorem shows that a suitable finite PhDN representation exists, but does not imply that it can be reliably recovered from arbitrary initialization. This distinction motivates the hybrid training strategy used in this work, where Stage I supplies an explicit symbolic skeleton before continuous PhDN refinement.

C.3 Density of Neural-Basis PhDNs

Neural bases provide another route to universal approximation. The key question is whether density can be retained after fixing the internal ridge parameters in advance, so that only the outer linear coefficients remain trainable. We show that one countably infinite ridge-parameter bank can be fixed independently of the target function, while arbitrarily increasing finite truncations of this same bank remain dense. We then transfer this result directly to neural-basis PhDNs.

Density of fixed neural-ridge banks:

Let $K \subset \mathbb{R}^{n_x}$ be compact and let $\varphi : \mathbb{R} \to \mathbb{R}$ be continuous and nonpolynomial. Consider first the unrestricted scalar ridge-function class

$$\mathcal{R} = \bigcup_{M \geq 1} \left\{ g : K \to \mathbb{R} \;\middle|\; g(\mathbf{x}) = c_0 + \sum_{m=1}^{M} c_m \varphi(\overline{\mathbf{w}}_m^\top \mathbf{x} + \overline{b}_m),\ c_m \in \mathbb{R},\ (\overline{\mathbf{w}}_m, \overline{b}_m) \in \mathbb{R}^{n_x+1} \right\}.$$

The classical universal-approximation theorem states that finite ridge networks with freely selected internal parameters are dense in $C(K)$:

$$\overline{\mathcal{R}}^{\|\cdot\|_{\infty,K}} = C(K) \tag{15}$$

(Leshno et al., 1993; Hornik et al., 1989).

We next fix the internal ridge parameters once and for all. Since $\mathbb{R}^{n_x+1}$ is separable, it admits a countably infinite dense sequence; for example, an enumeration of $\mathbb{Q}^{n_x+1}$ is sufficient. Let

$$\{(\mathbf{w}_j, b_j)\}_{j=1}^{\infty}$$

be any such fixed dense sequence. For each finite N, define the corresponding truncation of the fixed ridge dictionary as

$$\mathbf{r}_N(\mathbf{x}) = \begin{bmatrix} 1 & \varphi(\mathbf{w}_1^\top \mathbf{x} + b_1) & \cdots & \varphi(\mathbf{w}_N^\top \mathbf{x} + b_N) \end{bmatrix}^\top,$$

and let

$$\mathcal{H}^{\mathrm{ridge}}_N = \left\{ \mathbf{g} : K \to \mathbb{R}^{n_y} \;\middle|\; \mathbf{g}(\mathbf{x}) = \mathbf{W}\mathbf{r}_N(\mathbf{x}), \quad \mathbf{W} \in \mathbb{R}^{n_y \times (N+1)} \right\}$$

denote the vector-valued function class generated by the first N fixed features. Thus, the ridge parameters are fixed and only the outer coefficients $\mathbf{W}$ are trainable. Define

$$\mathcal{H}^{\mathrm{ridge}} = \bigcup_{N \geq 1} \mathcal{H}_N^{\mathrm{ridge}},$$

which collects all functions represented by finite truncations of this same fixed infinite bank.

Theorem C.4 (Density of finite truncations of a fixed neural-ridge bank)**.** *For the countably infinite fixed dense parameter sequence above,*

$$\overline{\mathcal{H}^{\mathrm{ridge}}}^{\,\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}). \tag{16}$$

Equivalently, for every $\mathbf{f} \in C(K, \mathbb{R}^{n_y})$ and every $\epsilon > 0$, there exist a finite, target- and tolerance-dependent truncation N and an outer coefficient matrix $\mathbf{W}$ such that

$$\|\mathbf{f} - \mathbf{W}\mathbf{r}_N\|_{\infty,K} < \epsilon.$$

Proof. Consider first scalar outputs and define

$$T_{\boldsymbol{\theta}}(\mathbf{x}) = \varphi(\mathbf{w}^\top \mathbf{x} + b), \qquad \boldsymbol{\theta} = (\mathbf{w}, b).$$

Joint continuity of the affine map and φ, together with compactness of K, implies that $\boldsymbol{\theta} \mapsto T_{\boldsymbol{\theta}}$ is continuous from the parameter space into $C(K)$ under the uniform norm. Therefore, because $\{(\mathbf{w}_j, b_j)\}_{j=1}^{\infty}$ is dense in $\mathbb{R}^{n_x+1}$, every freely parameterized ridge function can be approximated arbitrarily well on K by a ridge function from this same fixed bank.

Let $\mathcal{H}_{\mathrm{scalar}}^{\mathrm{ridge}}$ denote the scalar-output version of $\mathcal{H}^{\mathrm{ridge}}$. Every $g \in \mathcal{R}$ is a finite linear combination of finitely many such ridge functions. Hence, each g can be approximated arbitrarily well by a finite linear combination of fixed-bank features. Any finite collection of these features is contained in the first N entries for some finite N. Consequently,

$$\mathcal{R} \subseteq \overline{\mathcal{H}_{\mathrm{scalar}}^{\mathrm{ridge}}}^{\,\|\cdot\|_{\infty,K}}.$$

By the classical universal-approximation result equation 15,

$$\overline{\mathcal{R}}^{\,\|\cdot\|_{\infty,K}} = C(K).$$

Since $\mathcal{H}_{\mathrm{scalar}}^{\mathrm{ridge}} \subseteq C(K)$, the preceding inclusion gives

$$\overline{\mathcal{H}_{\mathrm{scalar}}^{\mathrm{ridge}}}^{\,\|\cdot\|_{\infty,K}} = C(K).$$

Applying this scalar result independently to each output component and taking the largest finite truncation required among the n_y components yields

$$\overline{\mathcal{H}^{\mathrm{ridge}}}^{\,\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}),$$

which proves the result. □

The theorem concerns one countably infinite master bank whose internal parameters are fixed independently of the target function. It does not assert that any single fixed finite truncation is dense. Rather, for each target and approximation tolerance, some sufficiently large but finite truncation of the same bank is sufficient. Thus, increasing finite feature width, together with trainable outer coefficients, provides the density result.

The fixed dense master bank used in Theorem C.4 is a theoretical existence construction. It should be distinguished from the finite, data-dependent neural augmentation used in the reported experiments. The practical construction in Appendix B.2.1 selects a compact set of ridge features from empirical branch-state statistics to promote numerical conditioning and feature diversity. It is not assumed to be a finite truncation of the dense master bank above, and Theorem C.4 does not assert that any particular fixed finite practical bank is dense.

Transfer of density to neural-basis PhDNs:

Let a PhDN branch use the fixed local dictionary template

$$\mathcal{A}_\varphi(\mathbf{h}) = \{1\} \cup \{h_i\}_{i=1}^d \cup \{\varphi(h_i)\}_{i=1}^d.$$

For a finite width configuration $\mathbf{n}$, let

$$\mathcal{H}_{\mathrm{PhDN}}^{2,\mathbf{n}}(\mathcal{A}_\varphi)$$

denote the class of functions $\mathbf{g} : K \to \mathbb{R}^{n_y}$ representable by two-layer PhDNs with width configuration $\mathbf{n}$ and local dictionary $\mathcal{A}_\varphi$. Define

$$\mathcal{H}_{\mathrm{PhDN}}^{(2)}(\mathcal{A}_\varphi) = \bigcup_{\mathbf{n}} \mathcal{H}_{\mathrm{PhDN}}^{2,\mathbf{n}}(\mathcal{A}_\varphi),$$

the function class represented by all finite-width two-layer PhDNs of this form.

Corollary C.5 (Neural-basis PhDN density)**.** *Let $K \subset \mathbb{R}^{n_x}$ be compact and let $\varphi : \mathbb{R} \to \mathbb{R}$ be continuous and nonpolynomial. Two-layer neural-basis PhDNs with arbitrarily increasing finite width are dense in $C(K, \mathbb{R}^{n_y})$:*

$$\overline{\mathcal{H}_{\mathrm{PhDN}}^{(2)}(\mathcal{A}_\varphi)}^{\|\cdot\|_{\infty,K}} = C(K, \mathbb{R}^{n_y}). \tag{17}$$

Proof. Consider any finite fixed-bank model

$$\mathbf{g}(\mathbf{x}) = \boldsymbol{\beta}_0 + \sum_{j=1}^N \boldsymbol{\beta}_j \varphi(\mathbf{w}_j^\top \mathbf{x} + b_j).$$

Here, $(\mathbf{w}_j, b_j)$ are the prescribed fixed internal parameters from the master sequence of Theorem C.4; they are not trained in this construction.

Using the constant and identity bases in $\mathcal{A}_\varphi$, the first PhDN layer is assigned these prescribed affine parameters to form

$$h_j^{(1)} = \mathbf{w}_j^\top \mathbf{x} + b_j, \qquad j = 1, \ldots, N.$$

The second layer evaluates

$$\varphi(h_j^{(1)}) = \varphi(\mathbf{w}_j^\top \mathbf{x} + b_j)$$

and applies the outer linear coefficients $\boldsymbol{\beta}_0, \ldots, \boldsymbol{\beta}_N$. Thus, every function in any finite $\mathcal{H}_N^{\mathrm{ridge}}$ is represented exactly by a two-layer PhDN of sufficient finite width. Therefore,

$$\mathcal{H}^{\mathrm{ridge}} \subseteq \mathcal{H}_{\mathrm{PhDN}}^{(2)}(\mathcal{A}_\varphi).$$

Since $\mathcal{H}^{\mathrm{ridge}}$ is dense in $C(K, \mathbb{R}^{n_y})$ by Theorem C.4, the PhDN class containing it is also dense. □

Thus, increasing finite neural-basis width already provides a sufficient route to universal approximation, and two PhDN layers are sufficient for this density result. Additional all-previous-layer connections and greater depth are not required for universality, although they may improve compositional efficiency through feature reuse.

D Symbolic-Representability Details and Additional Cases

D.1 Main-Text Case Protocol

For the symbolic map in equation 7, training, validation, in-distribution (ID) testing, and out-of-distribution (OOD) testing use 1500, 500, 500, and 500 independently generated randomized Latin-hypercube samples, respectively. The ID domain is

$$[-1, 1] \times [-0.35, 0.75] \times [0.6, 1.6],$$

and the disjoint OOD domain is

$$([-1.5, -1.1] \cup [1.1, 1.5]) \times ([-0.55, -0.45] \cup [0.85, 1.05]) \times [1.8, 2.4].$$

For the strong-prior learnability test, the compiler-generated structural support is retained. For every active exact coefficient $\theta_j^\star$, each of 20 initializations is sampled as $\theta_{0,j} = \theta_j^\star + 0.1|\theta_j^\star|u_j$, $u_j \sim \mathrm{Uniform}[-1, 1]$. Each initialization is refined using bounded BP-assisted NLS with an analytic Jacobian, at most 50 iterations, and at most 10^5 function evaluations per start. The candidate with the lowest validation error is selected. The base coefficient bounds are $[0, 4]$ and are expanded coefficient-wise only when necessary to contain the prescribed perturbation interval. Its train, validation, ID, and OOD RMSEs are 2.00×10^{-15}, 2.12×10^{-15}, 1.95×10^{-15}, and 2.79×10^{-14}, respectively.

D.2 Additional Symbolic Cases

Two additional cases examine constructive representability and strong-prior coefficient learning under different PhDN structures. They use the same sampling sizes and optimization protocol as the main-text case and pass the prescribed numerical reproduction check.

The first case is a two-input scalar relationship. Define

$$\begin{cases} a = 1 + x_1^2 + x_2^2, & b = 0.5x_1x_2, \\ u = \sqrt{a} + 0.5\sin(x_1), & v = \exp(b) + 0.5\cos(x_2), \\ p = \log(u) + 0.5b, & q = \sin(v) + 0.5a, \end{cases} \tag{18}$$

with

$$y = p + 0.5q + 0.5\cos(b) + 0.5\sin(x_2). \tag{19}$$

Its ID domain is $[-1, 1]^2$, and its disjoint OOD domain is $([-1.6, -1.2] \cup [1.2, 1.6])^2$.

The second case is a three-input, two-output relationship. Let

$$\begin{cases} a = 1 + x_1^2 + x_2^2, & b = 0.5x_1x_2, & c = 1 + x_3^2, \\ u = \sqrt{a} + 0.5\sin(x_1), & v = \exp(b) + 0.5\cos(x_2), & w = c^{-1} + 0.5x_1x_3, \\ p = \log(u) + 0.5b + 0.5\cos(x_3), & q = \sin(v) + 0.5w + 0.5x_2^2, & \end{cases} \tag{20}$$

and

$$\begin{cases} y_1 = p + 0.5q + 0.5\cos(b) + 0.5\sin(x_3), \\ y_2 = q + 0.5p + 0.5u + 0.5\cos(x_1x_2). \end{cases} \tag{21}$$

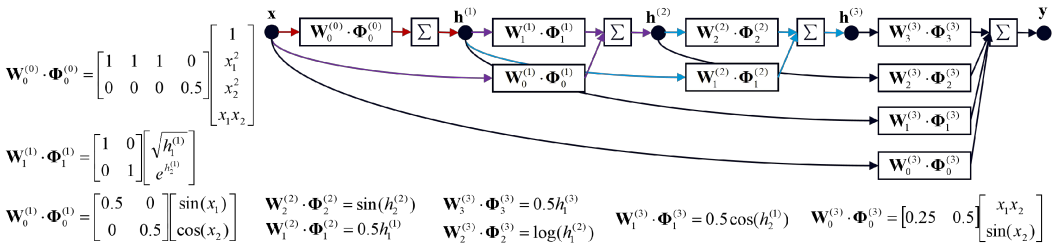
Its ID domain is $[-1, 1]^2 \times [0.2, 1.2]$, and its disjoint OOD domain is $([-1.5, -1.1] \cup [1.1, 1.5])^2 \times [1.4, 2.0]$.

Constructive representability. The scalar map is instantiated as a PhDN with layer dimensions $[2, 2, 2, 2, 1]$, eight active branches, four skip branches, and 16 active coefficients. The multi-output map has layer dimensions $[3, 3, 3, 2, 2]$, ten active branches, six skip branches, and 26 active coefficients. The first case tests middle-layer and direct-output skip connections, whereas the second additionally tests multi-output fan-in and fan-out with shared intermediate quantities. Both PhDNs reproduce their analytical maps to floating-point precision.

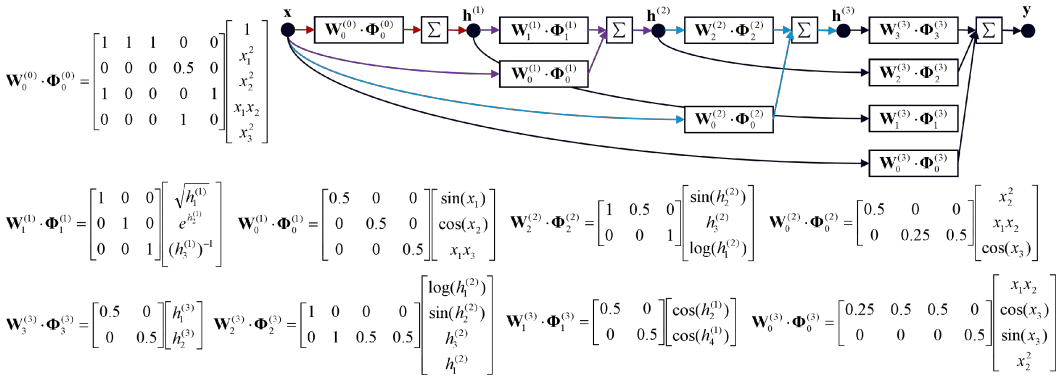
Strong-prior learnability. For each case, the compiler-generated structural support is retained, and the same true-centered 20-start initialization and validation-selected BP-assisted NLS protocol as in the main-text case is applied. The two-input scalar case achieves train/validation/ID/OOD RMSEs of 1.86×10^{-7}, 1.84×10^{-7}, 1.93×10^{-7}, and 1.69×10^{-5}, respectively. The three-input, two-output case achieves 1.79×10^{-8}, 2.00×10^{-8}, 1.80×10^{-8}, and 2.67×10^{-6}, respectively. The higher residual errors of the additional architectures occur under the same local perturbation protocol and are consistent with case-dependent numerical conditioning rather than a change in initialization strategy. Together with the main-text case, these results demonstrate functional recovery across PhDN structures with different dimensions, skip connections, and shared intermediate quantities.

E Complete Feynman Physical-Relationship Benchmark

This appendix provides the benchmark settings, dimensionless expressions, sampling domains, complete accuracy results, and runtimes for the 27 Feynman relationships used in Section 4.2. Within



(a) Two-input scalar case in equation 19.



(b) Three-input, two-output case in equation 21.

Figure 7: Additional compiled PhDN structures.

each run, all methods use the same training, validation, ID-test, and OOD-test samples. PhDN-Fixed is reported as an internal mechanism ablation of the proposed framework: it retains the accepted Stage-I support and refines only its coefficients while keeping augmentation-only support inactive. To avoid over-interpreting floating-point differences, RMSE values below 10^{-15} are displayed as $< 10^{-15}$ for all methods. All numerical comparisons and counts use the raw stored RMSEs unless explicitly described as numerical-floor ties.

E.1 Benchmark Protocol and Method Settings

Table 3 summarizes the principal settings. PhDN is evaluated without any case-specific expression, architecture, or structural mask. PhDN-Fixed reuses the corresponding accepted Stage-I model and data split, so it adds no symbolic search. For PySR-selected cases, the symbolic skeleton is compiled into the explicit PhDN DAG and only the compiled support is refined. For the two SINDy-bypass cases, the accepted flat dictionary is the $L = 1$ PhDN specialization. Under the all-output bypass rule, full PhDN returns this accepted Stage-I model directly, while PhDN-Fixed separately refits its active coefficients on fixed support.

E.2 Benchmark Expressions and Sampling Domains

Tables 4–7 list the dimensionless target maps and their sampling domains. Each case uses 1500 training, 500 validation, and 500 ID-test samples, together with 500 samples from the disjoint OOD domain.

E.3 Complete Comparison Results

Tables 8–11 report the complete comparison. Each entry gives mean ID RMSE/OOD RMSE over three runs. $\mathrm{SR}^{\dagger}$ is the accepted proposed-score Stage-I model. PhDN-Fixed retains exactly that accepted support and refines only its coefficients, whereas full PhDN additionally permits augmented dictionary support during refinement. $\mathrm{SR}_{\mathrm{native}}$ is the independent matched Stage-I ablation selected with PySR's native equation score. Values below 10^{-15} are displayed as $< 10^{-15}$ for all methods; aggregate statements below use the raw stored RMSEs unless a numerical-floor convention is stated.

Table 3: Key settings for the Feynman comparison.

Method	Key settings
PhDN	Stage I first applies the general SINDy bypass and, when necessary, runs PySR with three restarts, 500 iterations per restart, 18 total populations, and population size 64. Candidates are selected by the proposed validation/frontier score and translated into an explicit PhDN representation. Constant–Poly_2 augmentation provides additional admissible support for differentiable BP-assisted NLS refinement and validation-controlled pruning.
PhDN-Fixed	Internal ablation of the proposed fixed-support PhDN refinement mechanism. For PySR-selected cases, the accepted Stage-I skeleton is compiled into the same explicit PhDN representation, every augmentation-only coefficient is fixed at zero, and only coefficients on the compiled support are refined with BP-assisted bounded analytic-Jacobian NLS. STLS and pruning are disabled, so support cannot change. For the two SINDy-bypass cases, the accepted active flat support is treated as an $L = 1$ PhDN and refit by the equivalent exact fixed-support linear least-squares problem. No additional SINDy or PySR search is run.
SR[†]	The accepted Stage-I model before PhDN refinement. For unresolved cases it is the symbolic expression selected by the proposed validation/frontier score; for the two bypass cases it is the accepted SINDy model. The result is collected from the corresponding PhDN Stage-I run.
$\mathrm{SR}_{\mathrm{native}}$	Matched Stage-I ablation using the same SINDy bypass, PySR grammar, three restarts, 500 iterations per restart, total population budget 18, population size 64, data split, and outer-round random-state protocol. The proposed validation/frontier selection rule, including the numerical-floor selector, is replaced by PySR's native equation score. No PhDN refinement or pruning is applied.
MLP	Following the Feynman baseline protocol in Liu et al. (2025b), a validation sweep is performed over width-20 networks with affine depths 2–6 and Tanh, ReLU, or SiLU activations; each candidate is trained using 500-step LBFGS.
SINDy	Poly_2 dictionary with $\{\mathrm{inv}, \sqrt{\cdot}, \exp, \sin, \cos, \log\}$ applied dimensionwise to the inputs and eligible polynomial monomials, including operator–monomial cross terms; the STLSQ threshold is selected by validation.
EQL-Div	Official implementation with a validation sweep over depths $L \in \{2, 3, 4\}$ and sparsity coefficients $\lambda \in \{10^{-5}, 10^{-3}\}$; each function type contains 10 units, and all candidates are trained using Adam.
KAN	Official pyKAN implementation following the numerical Feynman protocol of Liu et al. (2025b); a validation sweep is performed over width 5, depths 2–6, grids $\{3, 5, 10, 20, 50, 100, 200\}$, and sparsity coefficients $\{10^{-2}, 10^{-3}\}$, with 200 LBFGS steps at each grid. Validation-based layerwise early stopping terminates the sweep once deeper architectures cease to yield meaningful improvement.

[†] SR denotes the accepted proposed-score Stage-I model. PhDN-Fixed is an internal PhDN mechanism ablation rather than an external baseline; it isolates fixed-support continuous refinement of the accepted Stage-I representation. Full PhDN additionally permits augmented dictionary support to participate in refinement.

Table 4: Dimensionless Feynman relationships and sampling domains (part 1 of 4).

System	Dimensionless relationship	ID domain	OOD domain
I.6.2	$y = \frac{\exp[-\theta^2/(2\sigma^2)]}{\sqrt{2\pi\sigma^2}}$	$\theta : [-3, 3]$; $\sigma : [0.5, 2.5]$	$\theta : [3.3, 4.8]$; $\sigma : [2.6, 3.1]$
I.6.2b	$y = \frac{\exp[-(\theta-\theta_1)^2/(2\sigma^2)]}{\sqrt{2\pi\sigma^2}}$	$\theta, \theta_1 : [-3, 3]$; $\sigma : [0.5, 2.5]$	$\theta : [3.3, 4.8]$; $\theta_1 : [-4.8, -3.3]$; $\sigma : [2.6, 3.1]$
I.9.18	$y = \frac{a}{(b-1)^2+(c-d)^2+(e-f)^2}$	$a : [0.25, 3]$; $b : [1.3, 4]$; $c, e : [-2, 2]$; $d, f : [-2, 4]$	$a : [0.25, 3]$; $b : [1.1, 1.25]$; $c, e : [1.05, 1.25]$; $d, f : [1.4, 1.55]$
I.12.11	$y = 1 + a \sin\theta$	$a : [-3, 3]$; $\theta : [-\pi, \pi]$	$a : [3.3, 4.8]$; $\theta : [1.1\pi, 1.6\pi]$
I.13.12	$y = a\left(b^{-1} - 1\right)$	$a : [-3, 3]$; $b : [-3, -0.4] \cup [0.4, 3]$	$a, b : [3.3, 4.8]$
I.15.3x	$y = \frac{1-a}{\sqrt{1-b^2}}$	$a : [-3, 3]$; $b : [-0.75, 0.75]$	$a : [3.3, 4.8]$; $b : [-0.9, -0.8] \cup [0.8, 0.9]$
I.16.6	$y = \frac{a+b}{1+ab}$	$a, b : [-0.8, 0.8]$	$a, b : [0.88, 1.28]$

Table 5: Dimensionless Feynman relationships and sampling domains (part 2 of 4).

System	Dimensionless relationship	ID domain	OOD domain
I.18.4	$y = \frac{1+ab}{1+a}$	$a : [-0.75, 4];\ b : [-3, 3]$	$a : [4.2375, 5.425];\ b : [3.3, 4.8]$
I.26.2	$y = \arcsin(n \sin\theta_2)$	$n : [-0.55, 0.55];\ \theta_2 : [-\pi, \pi]$	$n : [0.605, 0.88];\ \theta_2 : [1.1\pi, 1.6\pi]$
I.27.6	$y = \frac{1}{1+ab}$	$a, b : [-0.9, 0.9]$	$a, b : [0.99, 1.44]$
I.29.16	$y = \sqrt{1 + a^2 - 2a\cos(\theta_1 - \theta_2)}$	$a : [0.05, 0.9] \cup [1.1, 3];\ \theta_1, \theta_2 : [-\pi, \pi]$	$a : [3.15, 3.9];\ \theta_1, \theta_2 : [-\pi, \pi]$
I.30.3	$y = \frac{\sin^2(n\theta/2)}{\sin^2(\theta/2)}$	$n : [0.5, 5];\ \theta : [-3, -0.5] \cup [0.5, 3]$	$n : [5.225, 6.35];\ \theta : [3.3, 4.8]$
I.30.5	$y = \arcsin(a/n)$	$a : [-1.2, 1.2];\ n : [1.5, 3]$	$a : [1.32, 1.92];\ n : [3.075, 3.45]$
I.37.4	$y = 1 + a + 2\sqrt{a}\cos\delta$	$a : [0.1, 4];\ \delta : [-\pi, \pi]$	$a : [4.195, 5.17];\ \delta : [1.1\pi, 1.6\pi]$

Table 6: Dimensionless Feynman relationships and sampling domains (part 3 of 4).

System	Dimensionless relationship	ID domain	OOD domain
I.40.1	$y = n_0 \exp(-a)$	$n_0, a : [-3, 3]$	$n_0, a : [3.3, 4.8]$
I.44.4	$y = n \log a$	$n : [-3, 3];\ a : [0.25, 4]$	$n : [3.3, 4.8];\ a : [4.1875, 5.125]$
I.50.26	$y = \cos a + \alpha \cos^2 a$	$a : [-\pi, \pi];\ \alpha : [-3, 3]$	$a : [1.1\pi, 1.6\pi];\ \alpha : [3.3, 4.8]$
II.2.42	$y = (a-1)b$	$a, b : [-4, 4]$	$a, b : [4.4, 6.4]$
II.6.15a	$y = \frac{c}{4\pi}\sqrt{a^2+b^2}$	$a, b : [-4, 4];\ c : [-3, 3]$	$a, b : [4.4, 6.4];\ c : [3.3, 4.8]$
II.11.7	$y = n_0(1 + a\cos\theta)$	$n_0, a : [-3, 3];\ \theta : [-\pi, \pi]$	$n_0, a : [3.3, 4.8];\ \theta : [1.1\pi, 1.6\pi]$
II.11.27	$y = \frac{ab}{1-ab/3}$	$a, b : [-1.4, 1.4]$	$a, b : [1.5, 1.6]$

Table 7: Dimensionless Feynman relationships and sampling domains (part 4 of 4).

System	Dimensionless relationship	ID domain	OOD domain
II.35.18	$y = \frac{a}{\exp(b) + \exp(-b)}$	$a, b : [-3, 3]$	$a, b : [3.3, 4.8]$
II.36.38	$y = a + \alpha b$	$a, \alpha, b : [-4, 4]$	$a, \alpha, b : [4.4, 6.4]$
II.38.3	$y = \frac{a}{b}$	$a : [-4, 4];\ b : [-4, -0.5] \cup [0.5, 4]$	$a, b : [4.4, 6.4]$
III.9.52	$y = a\frac{\sin^2[(b-c)/2]}{[(b-c)/2]^2}$	$a : [-3, 3];\ b : [-3, -1] \cup [1, 3];\ c : [-0.5, 0.5]$	$a : [-3, 3];\ b : [-0.45, -0.15] \cup [0.15, 0.45];\ c : [-0.05, 0.05]$
III.10.19	$y = \sqrt{1 + a^2 + b^2}$	$a, b : [-4, 4]$	$a, b : [4.4, 6.4]$
III.17.37	$y = \beta(1 + \alpha\cos\theta)$	$\alpha, \beta : [-3, 3];\ \theta : [-\pi, \pi]$	$\alpha, \beta : [3.3, 4.8];\ \theta : [1.1\pi, 1.6\pi]$

Table 8: Complete Feynman comparison (part 1 of 4).

System	PhDN	PhDN-Fixed	SR†	SR$_{\text{native}}$	MLP	SINDy	EQL-Div	KAN
I.6.2	1.04e−15/ $< 10^{-15}$	1.04e−15/ $< 10^{-15}$	1.88e−8/2.01e−9	5.84e−4/1.66e−4	3.55e−5/2.10e−3	3.15e−2/1.22e5	3.60e−3/8.66e−3	1.66e−4/2.38e−1
I.6.2b	7.19e−9/7.25e−9	2.62e−7/1.93e−7	5.81e−7/3.12e−7	1.80e−7/2.66e−8	4.79e−5/1.35e−3	8.99e−2/7.16e2	3.63e−3/5.08e−3	1.36e−3/2.01e−1
I.9.18	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	4.83e−15/1.34e−13	1.03e−7/1.23e−6	3.63e−2/1.77e0	4.05e−1/7.05e0	4.38e−2/1.22e0	1.43e−1/3.80e0
I.12.11	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.37e−15/7.18e−15	3.37e−15/7.18e−15	4.09e−4/7.34e−1	$< 10^{-15}$/ $< 10^{-15}$	3.45e−3/3.06e−2	2.07e−4/2.60e0
I.13.12	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.63e−15/2.43e−15	3.63e−15/2.43e−15	1.76e−3/4.54e−2	$< 10^{-15}$/ $< 10^{-15}$	2.97e−9/1.40e−9	1.17e−2/3.69e0
I.15.3x	1.94e−5/4.11e−2	2.54e−5/4.50e−2	3.60e−5/4.34e−2	8.70e−4/7.68e−2	2.24e−4/3.11e−1	1.57e−4/1.92e−1	1.40e−2/5.43e−1	4.36e−3/6.79e0
I.16.6	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	1.74e−7/1.56e−7	6.52e−5/1.16e−2	1.38e−2/2.49e1	1.49e−3/1.36e−3	6.05e−7/6.72e−1

Table 9: Complete Feynman comparison (part 2 of 4).

System	PhDN	PhDN-Fixed	SR†	SR$_{\text{native}}$	MLP	SINDy	EQL-Div	KAN
I.18.4	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	1.97e−15/2.42e−15	1.98e−15/2.41e−15	4.46e−4/1.61e−2	3.82e0/4.67e6	9.06e−3/5.06e−1	1.69e−3/2.66e0
I.26.2	3.08e−7/1.39e−3	5.14e−7/1.53e−3	1.05e−6/1.63e−3	4.45e−5/6.26e−3	1.79e−4/1.12e−1	2.57e−3/2.31e4	1.46e−3/2.46e−1	1.20e−4/6.05e−1
I.27.6	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	1.41e−7/1.14e−8	7.24e−5/3.17e−3	1.24e−1/1.11e2	4.48e−4/1.08e−3	8.63e−4/1.25e0
I.29.16	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.32e−15/3.85e−15	1.55e−7/5.12e−8	2.73e−3/1.43e−2	5.15e−1/7.56e4	3.50e−2/1.32e−1	2.50e−2/7.02e−1
I.30.3	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	8.56e−15/1.09e−14	9.70e−3/1.59e−1	2.65e−3/2.43e0	5.72e−1/7.81e6	4.68e−2/7.29e−1	8.14e−3/4.15e0
I.30.5	2.84e−6/4.60e−5	4.39e−6/3.86e−6	5.56e−6/3.28e−6	1.65e−2/2.58e−2	1.96e−5/5.89e−4	1.63e−3/7.84e0	3.04e−3/7.19e−3	1.27e−5/3.33e−1
I.37.4	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	6.09e−15/1.06e−14	6.09e−15/1.06e−14	1.10e−3/1.08e0	1.15e−1/5.46e1	3.32e−2/2.26e0	9.18e−5/2.56e0

Table 10: Complete Feynman comparison (part 3 of 4).

System	PhDN	PhDN-Fixed	SR†	SR$_{\text{native}}$	MLP	SINDy	EQL-Div	KAN
I.40.1	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.12e−14/ $< 10^{-15}$	3.12e−14/ $< 10^{-15}$	8.51e−4/3.18e−2	5.00e−1/1.74e7	7.73e−2/3.58e0	8.43e−5/1.43e1
I.44.4	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.00e−15/5.11e−15	2.99e−15/5.10e−15	3.09e−4/1.93e−2	$< 10^{-15}$/ $< 10^{-15}$	1.42e−2/3.30e−1	5.76e−5/4.01e0
I.50.26	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	3.49e−15/7.05e−15	3.49e−15/7.05e−15	9.00e−4/1.27e0	1.54e−1/7.22e1	1.64e−2/2.17e0	7.53e−4/2.72e0
II.2.42	1.98e−15/9.54e−15	1.94e−15/1.09e−14	1.98e−15/9.54e−15	1.98e−15/9.54e−15	3.86e−4/3.98e−1	1.94e−15/1.09e−14	4.91e−11/2.46e−10	3.03e−4/1.41e1
II.6.15a	$< 10^{-15}$/ $< 10^{-15}$	$< 10^{-15}$/ $< 10^{-15}$	6.69e−9/3.72e−8	5.91e−8/3.29e−7	7.75e−4/1.31e−1	3.18e−2/1.09e1	9.31e−3/7.57e−1	1.07e−3/7.11e−1
II.11.7	1.33e−15/3.34e−15	$< 10^{-15}$/ $< 10^{-15}$	7.10e−15/3.85e−14	7.10e−15/3.85e−14	7.68e−3/8.27e0	2.33e0/2.16e7	5.24e−3/2.70e−1	7.02e−3/2.42e1
II.11.27	2.82e−15/2.06e−13	2.82e−15/2.06e−13	2.74e−8/2.45e−6	1.07e−7/9.57e−6	7.62e−5/1.99e0	1.55e−1/8.91e0	3.15e−10/9.68e−9	7.28e−4/9.21e0

Table 11: Complete Feynman comparison (part 4 of 4).

System	PhDN	PhDN-Fixed	SR†	SR$_{\text{native}}$	MLP	SINDy	EQL-Div	KAN
II.35.18	1.67e−5/3.13e−1	9.51e−5/2.15e−1	1.66e−4/2.50e−1	9.94e−3/2.77e−1	5.89e−5/6.39e−3	1.10e−2/2.52e5	1.16e−3/6.00e−3	1.65e−5/7.27e−1
II.36.38	1.61e−15/8.26e−15	1.20e−15/6.15e−15	1.61e−15/8.26e−15	1.61e−15/8.26e−15	8.09e−4/4.51e−1	1.20e−15/6.15e−15	6.96e−7/2.76e−6	2.93e−5/2.44e1
II.38.3	$< 10^{-15}/ < 10^{-15}$	$< 10^{-15}/ < 10^{-15}$	2.82e−15/ $< 10^{-15}$	2.82e−15/ $< 10^{-15}$	9.64e−4/1.60e−2	$< 10^{-15}/ < 10^{-15}$	1.22e−3/1.36e−4	2.70e−2/4.71e0
III.9.52	1.03e−5/1.29e−3	1.73e−5/1.95e−3	1.95e−5/1.07e−3	8.58e−4/8.04e−5	1.99e−4/5.10e−4	1.28e−1/2.81e2	1.75e−3/5.33e−3	2.02e−3/4.74e−3
III.10.19	$< 10^{-15}/ < 10^{-15}$	$< 10^{-15}/ < 10^{-15}$	2.64e−15/2.81e−15	1.36e−7/4.59e−8	2.69e−4/4.45e−2	1.61e−1/2.55e1	7.45e−3/8.04e0	3.42e−5/1.92e0
III.17.37	1.40e−15/5.75e−15	$< 10^{-15}$/1.02e−15	7.16e−15/3.88e−14	7.16e−15/3.88e−14	8.87e−3/8.21e0	2.33e0/2.16e7	9.92e−3/3.61e0	7.61e−3/8.03e0

Overall comparison. Under the 10^{-15} numerical-equivalence display convention, full PhDN attains the lowest or tied-lowest mean ID RMSE among the five external baselines in 24 of 27 cases and the lowest or tied-lowest OOD RMSE in 24. Using the raw stored values, the complete PhDN route lowers both mean ID and OOD errors relative to SR† in 22 cases, exactly coincides with the accepted Stage-I endpoint on the two SINDy-bypass cases, and gives mixed ID/OOD changes relative to SR† on I.30.5, II.35.18, and III.9.52. Thus, the end-to-end comparison supports the full framework while also showing that the dominant gain often enters before augmentation.

Fixed-support PhDN refinement. Using the raw stored RMSEs, PhDN-Fixed lowers mean ID error relative to SR† in all 27 cases and lowers both mean ID and OOD errors in 23. The improvement can be substantial without introducing any new support. On I.6.2, mean ID/OOD RMSE decreases from $1.88\times10^{-8}/2.01\times10^{-9}$ to $1.04\times10^{-15}/5.01\times10^{-16}$; on II.11.27 it decreases from $2.74\times10^{-8}/2.45\times10^{-6}$ to $2.82\times10^{-15}/2.06\times10^{-13}$. These results show that the dominant role of PhDN-Fixed on the scalar Feynman tasks is high-precision continuous refinement of the accepted symbolic structure after Stage-I search. The compiled PhDN provides the architecture-consistent parameterization and analytic-Jacobian optimization interface for this refinement, while the broader fixed-support parameterization can supply additional flexibility when the Stage-I skeleton is imperfect. PhDN-Fixed is therefore reported as an internal mechanism ablation of the proposed training pipeline rather than as an external post-processing baseline.

The four cases in which raw OOD RMSE does not decrease from SR† to PhDN-Fixed are I.15.3x, I.30.5, II.2.42, and III.9.52. The II.2.42 difference occurs at the numerical floor: $1.98\times10^{-15}/9.54\times10^{-15}$ for SR† versus $1.94\times10^{-15}/1.09\times10^{-14}$ for PhDN-Fixed. The other SINDy-bypass case, II.36.38, improves to $1.20\times10^{-15}/6.15\times10^{-15}$. These cases confirm that fixed-support refinement remains well defined after a Stage-I SINDy bypass, where the accepted flat model is simply the $L = 1$ PhDN specialization.

Role of augmentation. The ablation shows that augmentation is complementary to, rather than the sole source of, PhDN's improvement. Relative to PhDN-Fixed, Stage III lowers both raw mean ID and OOD RMSE in I.6.2b, I.15.3x, I.26.2, and III.9.52. For I.26.2, the progression SR† →PhDN-Fixed→PhDN is $1.05\times10^{-6}/1.63\times10^{-3}$, $5.14\times10^{-7}/1.53\times10^{-3}$, and $3.08\times10^{-7}/1.39\times10^{-3}$, respectively. On I.6.2b, PhDN-Fixed gives $2.62\times10^{-7}/1.93\times10^{-7}$, whereas full PhDN reaches $7.19\times10^{-9}/7.25\times10^{-9}$. Stage III also lowers ID RMSE on I.30.5 and II.35.18, although their mean OOD errors increase, illustrating that releasing a larger support is not guaranteed to improve every metric. Because Stage III is accepted using the validation set rather than the disjoint OOD test domain, improved validation performance, or even lower ID error, does not guarantee improved extrapolation. Fourteen cases are unchanged from PhDN-Fixed, typically because fixed-support refinement has already reached an adequate or numerical-precision solution. Other raw regressions are confined to approximately machine-precision errors. Augmentation should therefore be interpreted as optional hypothesis-space expansion for residual structural correction rather than as a guaranteed monotonic improvement step.

Stage-I score ablation. Excluding the two cases that pass the SINDy bypass, SR† achieves lower mean ID and OOD RMSE than SR$_{\text{native}}$ in 13 of 25 cases and is within 1% on both metrics in 10 more. SR$_{\text{native}}$ is better on both only for I.6.2b, while III.9.52 is mixed. On I.30.3, SR† achieves ID/OOD RMSEs of $8.56\times10^{-15}/1.09\times10^{-14}$, compared with $9.70\times10^{-3}/1.59\times10^{-1}$ for SR$_{\text{native}}$. These results support the practical utility of the proposed Stage-I selection strategy in the reported experiments.

Time cost. Table 12 reports mean method-level wall time over three runs. PhDN time includes Stage-I processing and, for non-bypass cases, Stage-II compilation and fixed-support refinement

followed by Stage-III augmented refinement and pruning. $SR^{\dagger}$ reports the Stage-I time already incurred within the corresponding PhDN run. PhDN-Fixed is represented by its incremental fixed-support refit cost rather than a separate total that includes the shared Stage-I computation. SR_{native} is retained for the accuracy ablation but is not included in the runtime comparison. Across the 75 non-bypass runs, the incremental PhDN-Fixed refit has median wall time 1.67 s (range 0.36–16.38 s) and invokes no additional PySR search. Across the six replayed $L = 1$ bypass refits, the corresponding cost is at most 0.001 s. For MLP, EQL-Div, and KAN, the reported time includes the complete validation-based hyperparameter sweep.

Table 12: Mean training time in seconds for each Feynman case over three runs.

System	PhDN time (s)	MLP time (s)	SINDy time (s)	$SR^{\dagger}$ time (s)	EQL-Div time (s)	KAN time (s)
I.6.2	402.831	255.535	0.392	399.368	453.968	1722.141
I.6.2b	414.305	257.446	0.965	409.498	450.388	1886.454
I.9.18	377.266	268.530	4.221	372.523	452.774	1335.072
I.12.11	362.098	253.188	0.323	361.708	449.713	1629.053
I.13.12	300.315	251.952	0.313	299.926	444.640	1938.387
I.15.3x	495.055	252.477	0.410	486.173	453.311	1375.668
I.16.6	455.076	252.529	0.415	454.328	451.737	1583.706
I.18.4	325.460	252.637	0.415	323.906	460.974	1533.238
I.26.2	520.612	258.666	0.413	514.365	455.064	1957.520
I.27.6	298.822	239.355	0.383	298.306	451.819	1725.652
I.29.16	484.287	248.230	0.964	481.900	449.849	2050.626
I.30.3	494.470	248.912	0.364	492.331	452.546	1902.089
I.30.5	406.652	249.978	0.364	394.106	454.620	1431.104
I.37.4	454.472	255.261	0.392	452.480	449.756	1493.180
I.40.1	414.263	240.693	0.400	412.727	464.517	1662.056
I.44.4	412.796	267.524	0.312	411.943	467.669	1680.884
I.50.26	450.729	254.992	0.392	448.815	453.321	2105.763
II.2.42	0.161	255.847	0.307	0.161	457.283	1742.402
II.6.15a	381.117	256.707	0.919	379.278	460.319	1586.266
II.11.7	390.026	253.743	0.964	389.401	458.247	2527.432
II.11.27	337.154	257.257	0.401	335.948	462.027	1840.825
II.35.18	568.773	252.842	0.392	559.557	456.908	2082.618
II.36.38	0.227	251.027	0.670	0.227	450.109	1552.433
II.38.3	288.705	245.580	0.315	288.297	455.309	1775.344
III.9.52	452.703	252.239	0.929	447.365	453.341	1513.043
III.10.19	330.523	254.088	0.412	330.000	451.781	1361.350
III.17.37	408.103	250.067	1.005	407.465	451.728	2128.790

Together, these comparisons characterize three stages of the Feynman pipeline: Stage-I symbolic selection, Stage-II fixed-support continuous refinement of the accepted explicit structure, and Stage-III optional refinement with additional dictionary support. The large $SR^{\dagger} \rightarrow$PhDN-Fixed gains emphasize the practical importance of post-search parameter refinement. Stage III provides the complementary PhDN mechanism for expanding the local explicit hypothesis space when residual structural mismatch remains, without requiring a new SR search or implying a monotonic improvement in every case.

F System-Identification Details

Across both dynamical benchmarks, the accepted Stage-I expressions are compiled into the explicit PhDN representation before continuous refinement. Stage III augments this representation with branch-local dictionary terms, using polynomial bases for the generator and fixed neural-ridge functions for Lorenz–96. Validation-controlled pruning then yields a directly inspectable final support whose retained terms and coefficients are available branchwise.

F.1 SALIENT-POLE GENERATOR MODEL AND DATA

Dynamic model. With $x_1 = \delta$, $x_2 = \Delta\omega$, $x_3 = E'_q$, and $x_4 = E_{\mathrm{fd}}$, the salient-pole single-machine infinite-bus model is

$$
\begin{aligned}
\dot{\delta} &= \omega_b \Delta\omega, \\
\dot{\Delta\omega} &= \frac{P_m - P_e - D\Delta\omega}{2H}, P_e = \frac{E'_q V_\infty}{X'_d + X_e}\sin\delta + K_{\mathrm{sal}}\sin(2\delta), K_{\mathrm{sal}} = \frac{V_\infty^2}{2}\left(\frac{1}{X_q + X_e} - \frac{1}{X'_d + X_e}\right), \\
\dot{E}'_q &= \frac{1}{T'_{do}}\left[-\frac{X_d + X_e}{X'_d + X_e}E'_q + \frac{(X_d - X'_d)V_\infty}{X'_d + X_e}\cos\delta + E_{\mathrm{fd}}\right], \\
\dot{E}_{\mathrm{fd}} &= \frac{K_A(V_{\mathrm{ref}} - V_t) - (E_{\mathrm{fd}} - E_{\mathrm{fd},0})}{T_A}, \\
V_t &= \sqrt{V_q^2 + V_d^2}, \qquad V_q = \frac{X_e E'_q + X'_d V_\infty \cos\delta}{X'_d + X_e}, \qquad V_d = \frac{X_q V_\infty \sin\delta}{X_q + X_e},
\end{aligned}
\tag{22}
$$

The operating quantities P_m, $E_{\mathrm{fd},0}$, and V_{ref} are selected such that $[\delta_0, 0, E'_{q,0}, E_{\mathrm{fd},0}]^\top$ is an equilibrium.

Table 13: Generator parameters.

Parameter	Value	Parameter	Value	Parameter	Value
f_{n}	60 Hz	ω_b	120π	H	3.5 s
D	5.0	X_d	1.8	X'_d	0.30
X_q	1.70	X_e	0.40	T'_{do}	8.0 s
V_∞	1.0	K_A	10.0	T_A	0.10 s
δ_0	0.50	$E'_{q,0}$	1.10	P_m	0.352683
$E_{\mathrm{fd},0}$	1.576609	K_{sal}	−0.476190	V_{ref}	1.077035

Sampling and rollout. The sampling domains are summarized in Table 14. A Matoušek–Affine–Owen scrambled-Sobol sequence is used. The nested training sets of sizes $\{250, 500, 1000, 2000\}$ are prefixes of one common master sequence sampled from the central ID region. The validation set contains 500 boundary-shell samples, while the ID and OOD vector-field test sets contain 1000 independently generated samples each.

Table 14: Generator sampling domains.

State	ID domain	OOD test	Rollout IC
δ	$[0.15, 1.05]$	$[1.15, 1.35]$	$[1.18, 1.32]$
$\Delta\omega$	$[-0.012, 0.012]$	$[-0.018, 0.018]$	$[-0.015, 0.015]$
E'_q	$[0.75, 1.45]$	$[0.55, 0.70]$	$[0.58, 0.70]$
E_{fd}	$[0.65, 2.55]$	$[2.70, 3.10]$	$[2.75, 3.00]$

The five rollout initial conditions (ICs) are drawn from the last column of Table 14, with the first fixed to $[1.28, 0.012, 0.62, 2.90]^\top$. Exact and identified systems are integrated over 2 s by classical RK4 with $\Delta t = 0.005$ s. In equation 8, $\mathbf{s}$ is chosen as the coordinatewise maximum of the ID-domain and rollout-domain widths. The trajectory metric is computed separately for each initial condition and then averaged over the five simulations.

F.2 Physical-Prior and Method Settings

Shared initial-expression dictionaries. The complete dictionary of each prior level is periodically injected into the common candidate populations of every unresolved output:

$$\begin{aligned}
\mathcal{I}_{G_1} &= \big\{c_{11} - c_{12}\sin(2x_1),\\
&\qquad\qquad c_{13} - c_{14}x_4 - c_{15}\sqrt{x_3^2 + x_1^2}\big\},\\
\mathcal{I}_{G_2} &= \big\{c_{21} - c_{22}x_3\sin x_1 + c_{23}\sin(2x_1),\\
&\qquad\qquad c_{24} - c_{25}x_4 - c_{26}\sqrt{x_3^2 + \sin^2 x_1}\big\},\\
\mathcal{I}_{G_3} &= \big\{c_{31} - c_{32}x_3\sin x_1 + c_{33}\sin(2x_1),\\
&\qquad\qquad c_{34} - c_{35}x_4 - c_{36}\sqrt{(x_3 + \cos x_1)^2 + \sin^2 x_1}\big\}.
\end{aligned} \tag{23}$$

Here, G_1 provides a coarse reluctance-related expression and a Euclidean voltage-magnitude surrogate. G_2 introduces the characteristic $x_3 \sin x_1$ and $\sin(2x_1)$ electromechanical structure together with a trigonometric voltage-magnitude surrogate, while G_3 further introduces the coupled x_3–$\cos x_1$ terminal-voltage topology. The numerical constants c_{ij} are linear data-regression initializations on the guess structure rather than true machine parameters.

No additional common initial-expression set is used. During SR, each dictionary occupies a 10% reinjection fraction. The expressions and their descendants are unprotected and compete under the same loss, mutation, constant optimization, and model-selection rules as all other candidates.

PhDN, PhDN-Fixed, and SR settings. The three PhDN variants differ only in $\mathcal{I}_{G_k}$. Stage-I uses the shared grammar $\{+, -, \times, \mathrm{square}, \sqrt{\cdot}, \sin, \cos\}$. Additional domain-informed operator-tree constraints are applied to focus the Stage-I search on physically meaningful generator structures, including restricting trigonometric dependence to the rotor-angle state x_1. Three internal SR restarts, 24 total populations of size 100, at most 700 iterations, and maximum expression depth/size $10/30$ are used.

The accepted Stage-I expressions are compiled into a multilayer PhDN whose active branches are augmented with total-degree polynomial terms up to order two. PhDN-Fixed-G_k is the internal Stage-II ablation: only coefficients on the compiled Stage-I support are refined by BP-assisted analytic-Jacobian nonlinear least squares, while every augmentation-only coefficient remains inactive and pruning is disabled. Full PhDN-G_k subsequently releases the augmented support for joint refinement and validation-controlled pruning. SR-G_k denotes the corresponding Stage-I model before compilation and therefore requires no additional SR run.

Baselines. MLP uses width 64, affine depth 2–6, and $\{\tanh, \mathrm{ReLU}, \mathrm{SiLU}\}$ activations. EQL-Div uses depth 2–5 and sparsity coefficients $\{10^{-5}, 10^{-3}\}$. KAN uses width 8, depth 2–6, cubic splines, and grid sizes $\{3, 5, 10, 20, 50, 100, 200\}$, followed by validation-controlled sparsification, pruning, and recovery. All configurations are selected using the fixed validation set.

SINDy uses a common flat total-degree-two polynomial dictionary supplemented with $\{\sqrt{\cdot}, \exp, \sin, \cos, \log\}$ terms and operator cross terms. Trigonometric terms are restricted to x_1, state-dependent division is excluded, and STLSQ thresholds range from 0 to 10^{-3}. For the matched-prior comparison, the full $\mathcal{I}_{G_3}$ library is supplied to SINDy as shared grouped candidate functions, so the same G_3 structural expressions are available within its standard flat-dictionary formulation. No output-specific assignment or true physical coefficient is provided.

MLP, EQL-Div, and KAN use input/output normalization. PhDN remains in the original physical coordinates to preserve exact reproduction of the selected Stage-I expressions, while its residual objective is scaled by the training-target standard deviations. Reported runtime includes the complete validation-selected search or sweep; Stage-I SR and the subsequent PhDN stages are also timed separately.

Each entry in Table 15 reports the vector-field RMSE / mean trajectory NRMSE, while the final column gives total training time over all four sample sizes. For PhDN-Fixed, this total includes the shared Stage-I search.

Table 15: Single-Generator Identification Results

Method	$N = 250$	$N = 500$	$N = 1000$	$N = 2000$	Total time (s)
PhDN-G_1	$4.52\times10^{-3}/1.14\times10^{-2}$	$8.32\times10^{-4}/1.95\times10^{-3}$	$2.55\times10^{-3}/6.02\times10^{-3}$	$6.29\times10^{-4}/8.06\times10^{-3}$	4460
PhDN-G_2	$1.13\times10^{-3}/2.54\times10^{-3}$	$1.44\times10^{-3}/3.47\times10^{-3}$	$2.96\times10^{-4}/6.42\times10^{-4}$	$7.70\times10^{-4}/2.75\times10^{-3}$	4851
PhDN-G_3	$1.64\times10^{-8}/1.85\times10^{-8}$	$1.41\times10^{-14}/6.04\times10^{-16}$	$7.45\times10^{-15}/2.07\times10^{-16}$	$8.99\times10^{-12}/4.41\times10^{-13}$	4403
PhDN-Fixed-G_1	$3.56\times10^{-2}/1.67\times10^{-2}$	$5.64\times10^{-2}/1.87\times10^{-2}$	$2.56\times10^{-2}/6.49\times10^{-3}$	$1.78\times10^{-2}/3.15\times10^{-2}$	4332
PhDN-Fixed-G_2	$5.82\times10^{-2}/2.02\times10^{-2}$	$7.79\times10^{-2}/2.91\times10^{-2}$	$1.37\times10^{-1}/2.64\times10^{-2}$	$9.10\times10^{-2}/2.28\times10^{-2}$	4777
PhDN-Fixed-G_3	$6.61\times10^{-7}/1.65\times10^{-7}$	$2.64\times10^{-8}/7.37\times10^{-10}$	$8.33\times10^{-10}/3.73\times10^{-11}$	$1.55\times10^{-7}/5.10\times10^{-9}$	4401
SR-G_1	$3.61\times10^{-2}/4.15\times10^{-2}$	$5.64\times10^{-2}/1.87\times10^{-2}$	$2.56\times10^{-2}/6.49\times10^{-3}$	$3.97\times10^{-2}/2.56\times10^{-2}$	4262
SR-G_2	$6.04\times10^{-2}/2.02\times10^{-2}$	$8.38\times10^{-2}/3.87\times10^{-2}$	$1.37\times10^{-1}/2.64\times10^{-2}$	$9.96\times10^{-2}/2.46\times10^{-2}$	4757
SR-G_3	$4.89\times10^{-5}/3.90\times10^{-6}$	$7.98\times10^{-5}/5.83\times10^{-6}$	$4.72\times10^{-5}/5.03\times10^{-6}$	$8.22\times10^{-5}/5.69\times10^{-6}$	4384
MLP	$2.67\times10^{-3}/1.14\times10^{-2}$	$1.72\times10^{-3}/6.86\times10^{-3}$	$1.62\times10^{-3}/7.67\times10^{-3}$	$1.05\times10^{-3}/9.33\times10^{-3}$	598
EQL-Div	$9.65\times10^{-3}/1.79\times10^{-2}$	$6.97\times10^{-3}/4.71\times10^{-3}$	$6.68\times10^{-3}/3.57\times10^{-3}$	$4.17\times10^{-3}/3.91\times10^{-3}$	2100
KAN	$7.64\times10^{-3}/1.56\times10^{-1}$	$5.93\times10^{-3}/1.51\times10^{-1}$	$5.40\times10^{-3}/1.46\times10^{-1}$	$5.28\times10^{-3}/1.42\times10^{-1}$	10991
SINDy	$9.13\times10^{-3}/\mathrm{Inf}$	$2.46\times10^{-3}/\mathrm{Inf}$	$2.14\times10^{-3}/\mathrm{Inf}$	$1.63\times10^{-3}/\mathrm{Inf}$	5.47

For SINDy, Inf denotes a failed numerical rollout integration.

Fixed-support and augmentation ablation. The staged comparison clarifies where the PhDN gain enters. With the strongest prior G_3, fixed-support refinement already improves both reported metrics at every training size. At $N_{\mathrm{tr}} = 1000$, for example, SR-G_3 →PhDN-Fixed-G_3 reduces vector-field RMSE from 4.72×10^{-5} to 8.33×10^{-10} and trajectory NRMSE from 5.03×10^{-6} to 3.73×10^{-11}. Releasing augmentation then further reduces them to 7.45×10^{-15} and 2.07×10^{-16}, respectively.

The weaker G_1 and G_2 priors show the complementary role of augmentation. Fixed-support refinement gives smaller gains and can leave the Stage-I endpoint essentially unchanged; for example, at G_2, $N_{\mathrm{tr}} = 1000$, the vector-field RMSE remains approximately 1.37×10^{-1} before augmentation, whereas full PhDN reaches 2.96×10^{-4}. Across all 12 prior/sample configurations, full PhDN improves both reported metrics relative to PhDN-Fixed. Fixed-support refinement is validation-controlled and need not monotonically improve the separate rollout metric: for G_1, $N_{\mathrm{tr}} = 2000$, vector-field RMSE improves while trajectory NRMSE changes from 2.56×10^{-2} to 3.15×10^{-2}. Together, these results indicate that fixed-support refinement is most effective when Stage I provides an adequate structural skeleton, whereas augmentation supplies additional capacity when structural mismatch remains.

F.3 Soft-Saturated Lorenz–96 Model and Data

Dynamic model. We use a $K = 10$ cyclic soft-saturated Lorenz–96 system with $F = 8$ and $\kappa = 1$:

$$z_i = x_{i-1}(x_{i+1} - x_{i-2}), \qquad \dot{x}_i = \frac{z_i}{\sqrt{1 + (z_i/\kappa)^2}} - x_i + F, \quad i = 1, \dots, K, \tag{24}$$

where all indices are interpreted cyclically. The saturation limits the nonlinear transport while retaining the local Lorenz–96 interaction pattern.

Sampling and rollout. The ID domain is $x_i \in [6.5, 9.5]$ for all states. The OOD domain alternates between $x_i \in [5.2, 6.2]$ for odd i and $x_i \in [9.8, 10.8]$ for even i. As in the generator case, a Matoušek–Affine–Owen scrambled-Sobol design uses a nested training master pool in the normalized ID core $r_\infty \le 0.90$, 500 validation samples from $0.92 \le r_\infty \le 1$, and 1000 independent ID-test samples over the full ID box. The nested training sizes are $\{250, 500, 1000, 2000\}$; train, validation, ID-test, and OOD streams use fixed independent seeds 7301, 7401, 7501, and 8501, respectively.

Five rollout initial conditions are drawn from the alternating box

$$x_i(0) \in \begin{cases} [5.7, 6.2], & i \text{ odd}, \\ [9.8, 10.3], & i \text{ even}, \end{cases}$$

with representative point $\mathbf{x}_0 = [6, 10, 6, 10, \dots, 6, 10]^\top$. Reference and identified systems are integrated for 5 s by fixed-step RK4 with $\Delta t = 0.01$ s.

F.4 LORENZ–96 PRIOR AND METHOD SETTINGS

Prior levels. Let $q_i = x_{i-1}(x_{i+1} - x_{i-2}) - x_i$. The three Stage-I prior levels are

$$G_1 : \mathcal{I}_{G_1} = \varnothing, \qquad G_2 : \mathcal{I}_{G_2} = \{q_i\}_{i=1}^{K}, \qquad G_3 : \mathcal{I}_{G_3} = \{q_i + F\}_{i=1}^{K}.$$

Thus, G_2 supplies the ideal unsaturated transport and damping while both the forcing and saturation remain unknown; G_3 additionally supplies the true forcing, leaving only the saturation unknown. For G_2 and G_3, the complete K-expression dictionary is available to the candidate populations of every unresolved output, with no hard output assignment. The dictionaries are soft SR initializations with a 10% reinjection fraction and are not protected during evolution.

PhDN, PhDN-Fixed, and SR settings. Stage-I uses $\{+, -, \times, /, \mathrm{square}, \sqrt{\cdot}\}$ with three internal SR restarts, population size 100, at most 700 iterations, and maximum depth/size $12/40$. The selected expressions are compiled into a shared PhDN DAG. To avoid complete-polynomial dictionary growth in this 10-dimensional problem, each active branch retains constant and linear coordinates and receives $N_{\mathrm{neu}} = 10$ fixed $\tanh$ neural-ridge bases, whose internal ridge parameters remain fixed and whose outer coefficients are initialized at zero.

PhDN-Fixed-G_k is the internal Stage-II ablation: only coefficients on the compiled Stage-I support are refined by BP-assisted analytic-Jacobian nonlinear least squares, while all augmentation-only coefficients remain inactive and pruning is disabled. Full PhDN-G_k subsequently releases the complete augmented support for joint refinement and validation-controlled pruning. SR-G_k denotes the corresponding Stage-I model before compilation and therefore requires no additional symbolic search. The general neural-basis construction is described in Appendix B.2.1.

Baselines. MLP uses width 64, affine depth 2–6, and $\{\tanh, \mathrm{ReLU}, \mathrm{SiLU}\}$ activations. EQL-Div uses depth 2–5 and sparsity coefficients $\{10^{-5}, 10^{-3}\}$. KAN uses width 10, depth 2–6, cubic splines, and grid sizes $\{3, 5, 10, 20, 50\}$; model configurations are selected using the common validation set.

Standard SINDy uses $\{1, x_i, \sqrt{x_i}, x_i x_j\}_{1 \le i \le j \le K}$, giving 76 candidate functions for $K = 10$. Neural-SINDy replaces the complete quadratic block with K fixed neural-ridge bases $\phi_j(\mathbf{x}) = \tanh(\mathbf{w}_j^\top \mathbf{x} + b_j)$, $j = 1, \ldots, K$. The fixed ridge parameters $(\mathbf{w}_j, b_j)$ are constructed using the procedure in Appendix B.2.1, which promotes representative coverage and feature diversity while retaining a compact basis set. Only the outer combination coefficients are subsequently learned. Retaining $\{1, x_i, \sqrt{x_i}\}$ gives a 31-dimensional dictionary. Both methods use common flat dictionaries for all outputs and STLSQ thresholds up to 10^{-3}. Neither receives the completed soft-saturation expression nor the PhDN Stage-I initial-expression dictionaries.

Each entry in Table 16 reports the vector-field RMSE / mean trajectory NRMSE, while the final column gives total training time over all four sample sizes. For PhDN-Fixed, this total includes the shared Stage-I search.

Table 16: Soft-Saturated Lorenz–96 Identification Results

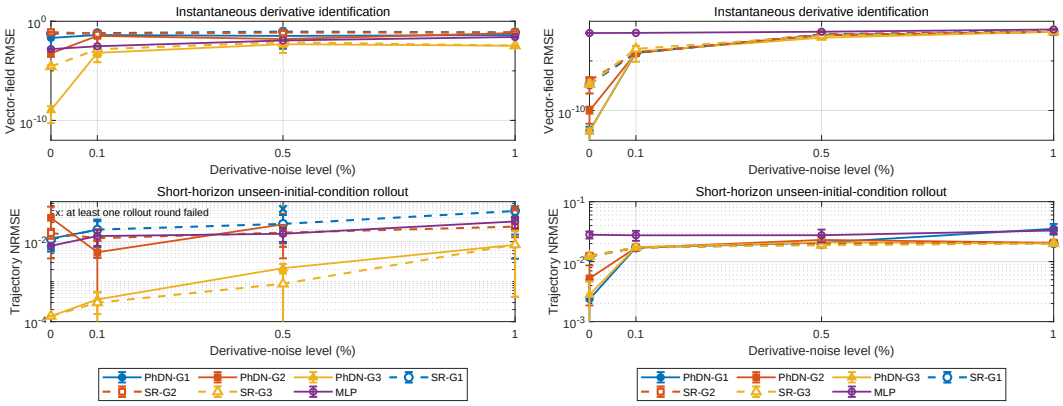
Method	$N = 250$	$N = 500$	$N = 1000$	$N = 2000$	Total time (s)
PhDN-G_1	$9.37\times10^{-3}/2.28\times10^{-2}$	$1.23\times10^{-9}/9.90\times10^{-3}$	$1.65\times10^{-8}/1.17\times10^{-2}$	$7.24\times10^{-12}/4.27\times10^{-3}$	6274
PhDN-G_2	$3.65\times10^{-2}/7.12\times10^{-2}$	$5.27\times10^{-12}/3.98\times10^{-3}$	$2.65\times10^{-12}/3.89\times10^{-3}$	$1.40\times10^{-13}/1.57\times10^{-3}$	8278
PhDN-G_3	$1.12\times10^{-9}/9.40\times10^{-3}$	$1.15\times10^{-11}/5.82\times10^{-3}$	$1.68\times10^{-12}/4.10\times10^{-3}$	$2.08\times10^{-13}/2.22\times10^{-3}$	7706
PhDN-Fixed-G_1	$9.37\times10^{-3}/2.28\times10^{-2}$	$1.63\times10^{-8}/1.17\times10^{-2}$	$1.65\times10^{-8}/1.17\times10^{-2}$	$2.21\times10^{-9}/9.46\times10^{-3}$	5556
PhDN-Fixed-G_2	$3.65\times10^{-2}/7.12\times10^{-2}$	$8.34\times10^{-10}/8.75\times10^{-3}$	$4.32\times10^{-10}/8.91\times10^{-3}$	$1.23\times10^{-10}/7.56\times10^{-3}$	6642
PhDN-Fixed-G_3	$4.72\times10^{-9}/9.71\times10^{-3}$	$8.64\times10^{-10}/9.20\times10^{-3}$	$5.72\times10^{-10}/8.62\times10^{-3}$	$1.31\times10^{-10}/7.37\times10^{-3}$	6778
SR-G_1	$9.37\times10^{-3}/2.28\times10^{-2}$	$3.33\times10^{-7}/1.33\times10^{-2}$	$4.81\times10^{-7}/1.41\times10^{-2}$	$4.99\times10^{-7}/1.39\times10^{-2}$	5133
SR-G_2	$4.20\times10^{-2}/8.20\times10^{-2}$	$4.47\times10^{-8}/1.18\times10^{-2}$	$5.16\times10^{-8}/1.24\times10^{-2}$	$2.89\times10^{-8}/1.18\times10^{-2}$	5707
SR-G_3	$5.48\times10^{-8}/1.21\times10^{-2}$	$4.55\times10^{-8}/1.22\times10^{-2}$	$6.38\times10^{-8}/1.25\times10^{-2}$	$3.64\times10^{-8}/1.19\times10^{-2}$	5731
MLP	$3.69\times10^{-2}/6.05\times10^{-2}$	$1.93\times10^{-2}/2.92\times10^{-2}$	$6.58\times10^{-3}/3.17\times10^{-2}$	$5.34\times10^{-3}/2.32\times10^{-2}$	684
EQL-Div	$2.71\times10^{-1}/1.36\times10^{-1}$	$1.53\times10^{-1}/1.54\times10^{-1}$	$4.89\times10^{-2}/1.22\times10^{-1}$	$2.93\times10^{-2}/1.19\times10^{-1}$	4120
KAN	$7.89\times10^{-1}/2.25\times10^{-1}$	$2.84\times10^{-1}/1.66\times10^{-1}$	$2.05\times10^{-1}/1.38\times10^{-1}$	$1.72\times10^{-1}/1.49\times10^{-1}$	33252
SINDy	$6.26\times10^{-1}/1.31\times10^{-1}$	$5.44\times10^{-1}/1.11\times10^{-1}$	$5.03\times10^{-1}/1.02\times10^{-1}$	$4.96\times10^{-1}/9.97\times10^{-2}$	9.02
Neural-SINDy	$5.37\times10^{-1}/1.08\times10^{-1}$	$5.16\times10^{-1}/1.10\times10^{-1}$	$4.93\times10^{-1}/1.01\times10^{-1}$	$4.92\times10^{-1}/1.00\times10^{-1}$	5.49

Fixed-support and neural-augmentation ablation. The staged comparison isolates the two continuous-refinement mechanisms. Across the 12 prior/sample configurations, PhDN-Fixed lowers both vector-field RMSE and trajectory NRMSE relative to SR-G_k in 11 cases and exactly reproduces the G_1, $N_{\mathrm{tr}} = 250$ endpoint in the remaining case. For example, at G_2, $N_{\mathrm{tr}} = 500$, the progression SR-G_2 →PhDN-Fixed-G_2 →PhDN-G_2 reduces vector-field RMSE from 4.47×10^{-8} to 8.34×10^{-10} and 5.27×10^{-12}, while trajectory NRMSE decreases from 1.18×10^{-2} to 8.75×10^{-3} and 3.98×10^{-3}.

Releasing the neural augmentation subsequently lowers both reported metrics relative to PhDN-Fixed in nine of the 12 configurations. The G_1 cases at $N_{\mathrm{tr}} = 250$ and 1000 remain unchanged, while at G_2, $N_{\mathrm{tr}} = 250$, the raw vector-field and trajectory changes are negligible and mixed. Thus, fixed-support refinement already removes a substantial fraction of the residual error once an adequate symbolic skeleton has been found, whereas neural-basis augmentation provides complementary capacity for further correction rather than a guaranteed monotonic gain.

F.5 Robustness to Derivative-Label Noise

We further evaluate robustness to noisy derivative observations using the generator and soft-saturated Lorenz–96 systems at $N_{\mathrm{tr}} = 1000$ and the common noise levels $\rho \in \{0, 0.1\%, 0.5\%, 1\%\}$. In both cases, the state samples are kept unchanged across the noise sweep, relative Gaussian noise is added only to the training and validation derivative labels, and robustness is evaluated using both clean held-out ID-test derivatives and short-horizon rollouts from unseen initial conditions. Each noise level is evaluated over three independent optimization rounds. The upper panels of Figure 8 report clean vector-field RMSE, while the lower panels report trajectory NRMSE; solid curves denote final PhDN models, dashed curves their corresponding Stage-I SR models, and MLP provides the black-box baseline.



(a) Generator case. (b) Lorenz–96 case.

Figure 8: Robustness to derivative-label noise.

For the generator case, prior quality has a pronounced effect on robustness and cross-round stability. PhDN-G_3 gives mean ID-test RMSEs of 1.09×10^{-9}, 6.49×10^{-4}, 4.86×10^{-3}, and 3.40×10^{-3} at $\rho = 0$, 0.1%, 0.5%, and 1%, respectively, compared with 1.55×10^{-3}, 3.01×10^{-3}, 1.16×10^{-2}, and 2.60×10^{-2} for MLP. The weaker G_1 and G_2 priors are substantially more variable across optimization rounds and are not uniformly superior to MLP. This behavior is consistent with their prior construction: both provide incomplete surrogates for the terminal-voltage magnitude, whereas G_3 additionally supplies the coupled x_3–$\cos(x_1)$ voltage topology. Because these expressions enter Stage I only as soft, unprotected SR initializations, the weaker priors leave the selected symbolic skeleton more sensitive to stochastic structure search.

For instantaneous vector-field error, the final PhDN models nevertheless reduce the Stage-I error substantially in several settings. Averaged across rounds, the SR-G_1 RMSE decreases from 5.60×10^{-2} to 2.11×10^{-2} in the clean case, from 6.29×10^{-2} to 4.13×10^{-2} at 0.1%, from 9.06×10^{-2} to 3.31×10^{-2} at 0.5%, and from 7.49×10^{-2} to 4.12×10^{-2} at 1% after PhDN refinement. For G_2, the corresponding reductions are 7.99×10^{-2} to 5.76×10^{-4}, 5.50×10^{-2} to

3.31×10^{-2}, 7.26×10^{-2} to 1.61×10^{-2}, and 7.47×10^{-2} to 6.49×10^{-2}. For G_3, refinement further reduces the mean Stage-I SR errors from 2.70×10^{-5} to 1.09×10^{-9} in the clean case, from 1.48×10^{-3} to 6.49×10^{-4} at 0.1%, and from 7.71×10^{-3} to 4.86×10^{-3} at 0.5%, while the two models are nearly identical at 1%. The small non-monotonic changes with noise level reflect the stochastic Stage-I structure search rather than an improvement caused by stronger corruption.

The rollout results provide a complementary dynamical view. For PhDN-G_3, mean trajectory NRMSE increases from 1.37×10^{-4}, to 3.58×10^{-4}, 2.17×10^{-3}, and 8.32×10^{-3} over the four noise levels, remaining below the corresponding MLP values of 7.87×10^{-3}, 1.38×10^{-2}, 1.57×10^{-2}, and 3.20×10^{-2}. All PhDN-G_3 rollouts succeed, whereas the weaker priors exhibit isolated instability: PhDN-G_1 succeeds on $14/15$ trajectories at 0.5% noise and PhDN-G_2 on $13/15$ at 1%. The Stage-I SR and final PhDN trajectory errors are also close in several settings, and SR can occasionally be lower, showing that reductions in pointwise vector-field error do not translate monotonically into rollout improvement.

For the soft-saturated Lorenz–96 case ($K = 10$, $F = 8$, $\kappa = 1$), PhDN-G_1, PhDN-G_2, and PhDN-G_3 achieve clean mean ID-test RMSEs of 8.73×10^{-13}, 9.29×10^{-11}, and 7.31×10^{-13}, respectively, while MLP remains at 6.49×10^{-3}. The corresponding Stage-I SR models are already highly accurate, with mean RMSEs of 4.25×10^{-8}, 1.04×10^{-7}, and 4.30×10^{-8}, showing that the symbolic structures are recoverable to high precision before PhDN refinement.

The advantage persists after derivative-label corruption. At $\rho = 0.1\%$, PhDN-G_1, PhDN-G_2, and PhDN-G_3 obtain mean RMSEs of 6.70×10^{-5}, 6.40×10^{-5}, and 8.80×10^{-5}, respectively, compared with 6.71×10^{-3} for MLP. At $\rho = 0.5\%$, the corresponding PhDN errors are 3.87×10^{-3}, 4.20×10^{-3}, and 2.20×10^{-3}, with G_3 giving the best result, whereas MLP increases to 8.81×10^{-3}. At $\rho = 1\%$, the three PhDN variants remain at 8.64×10^{-3}, 8.52×10^{-3}, and 8.98×10^{-3}, respectively, all below the MLP error of 1.49×10^{-2}.

Trajectory rollouts show a similar but less separated trend. With clean derivative labels, PhDN-G_1, PhDN-G_2, and PhDN-G_3 obtain mean trajectory NRMSEs of 2.37×10^{-3}, 5.26×10^{-3}, and 2.81×10^{-3}, respectively, versus 2.81×10^{-2} for MLP. At 0.1% noise, the three PhDN values cluster at 1.70×10^{-2}–1.75×10^{-2} compared with 2.75×10^{-2} for MLP, and at 0.5% they remain at 2.07×10^{-2}–2.30×10^{-2} versus 2.76×10^{-2}. At 1%, PhDN-G_2 and PhDN-G_3 retain lower trajectory NRMSEs of 2.06×10^{-2} and 2.00×10^{-2} than MLP at 3.30×10^{-2}, whereas PhDN-G_1 rises to 3.53×10^{-2}. No PhDN rollout failure occurs in the Lorenz–96 sweep. The Stage-I SR and final PhDN rollout curves become close at the corrupted levels, again indicating that improved instantaneous vector-field accuracy need not produce a proportional trajectory-level gain.

Taken together, the two systems show that stronger structural priors can improve both instantaneous identification and short-horizon dynamical robustness, while the trajectory comparison provides a distinct test of model quality that is not exactly implied by pointwise vector-field RMSE.