# FROM NOISY NEURAL BLACK BOXES TO CONCISE PHYSICAL FORMULAS: STRUCTURE-CONSTRAINED EXACT FORMULA RECOVERY

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

## ABSTRACT

Neural networks can fit noisy physical observations accurately but rarely expose the compact laws governing the underlying system. We present a structure-constrained neural-to-symbolic framework for recovering verifiable formulas from neural black-box surrogates. The method combines bounded search over physically motivated expression families, exactification of continuous coefficients into sparse symbolic constants, and noisy-data cross-fitting with structural and stability checks. This separates denoised structure proposal from discrete identification and validation on the original observations. We evaluate 13 development formulas and 20 held-out formulas over five seeds and Gaussian noise levels from 0% to 50%. On held-out formulas, exact-equivalence recovery decreases from 57.9% without noise to 19.2% at 50% noise; the development rate decreases from 69.2% to 61.5%. Removing exactification reduces recovery to 0% at both 20% and 50% noise, while removing raw-data refit cross-fitting reduces the 50%-noise development rate from 66.7% to 28.3%. Compared with PySR, TPSR, uDSR, Operon, and DSR, our method achieves the highest recovery rate at every nonzero noise level in the evaluated comparison. These results show that numerical agreement alone is insufficient for scientific formula recovery: exact constants, structural validity, and noise-aware validation are essential.

## 1 INTRODUCTION

Scientific machine learning increasingly provides accurate predictors of physical systems from experimental data (Karniadakis et al., 2021; Raissi et al., 2019). When observations are noisy, incomplete, or expensive, neural networks are attractive as flexible black-box surrogates. Their predictive accuracy, however, does not generally expose the compact analytical laws that scientists use to reason about mechanisms, extrapolate beyond observed regimes, or design experiments. Recovering a physical formula from such a surrogate is therefore a distinct scientific objective: the goal is not merely to reproduce values, but to identify a concise and verifiable representation of the latent law (Cranmer et al., 2020).

Symbolic regression offers a natural route from data to interpretable equations, but an unconstrained search can trade structural fidelity for numerical fit. In particular, noisy observations and imperfect neural surrogates can make unnecessary operators, high-degree terms, and arbitrary floating-point constants appear useful. A formula containing $0.4998$ may approximate the same function as one containing $1/2$, yet these expressions differ in symbolic content and scientific interpretability. We consequently distinguish predictive agreement from exact symbolic recovery: a successful method must control expression complexity, recover meaningful supports and constants, and reject candidates that are only supported by a particular noise realization.

We study this problem through a structure-preserving neural-to-symbolic framework. Given a neural black-box surrogate $g_\phi$ trained on noisy observations, we query it at fresh inputs to obtain a smooth discovery target and its input sensitivities. We then search a bounded portfolio of low-degree polynomial and rational forms, elementary transcendental atoms, and selected compositions. The resulting candidates are exactified by pruning unsupported terms and recovering simple constants, then refitted and evaluated on the original noisy observations using held-out folds. This separation lets the



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.

neural surrogate provide denoising and local functional information without allowing an unrestricted symbolic optimizer to absorb the training noise.

The framework has three main components. Structured exact search restricts degree, scale, composition depth, and candidate budgets before optimization. Exactification performs the continuous-to-discrete transition through support pruning, confidence-aware rational recovery, joint numerator–denominator rationalization, and factor-level reconstruction. Noisy-data cross-fitting then tests fixed candidate structures on observations not used for their coefficient fits, while candidate selection combines excess loss above a noise floor, input-Jacobian agreement, numerical validity, and description length.

We evaluate the distinction between development and generalization explicitly. Thirteen formulas are used to design and validate the candidate mechanisms, while 20 formulas are held out from formula and exactification design. Across five random seeds and seven Gaussian-noise levels, the proposed method achieves 45/65 (69.2%) exact recoveries on the development formulas at 0% noise and 40/65 (61.5%) at 50% noise. On the held-out formulas, it achieves 55/95 (57.9%) at 0% noise and 19/99 (19.2%) at 50% noise. The baseline study shows that PySR is competitive without noise, whereas our method has the highest recovery accuracy at every nonzero noise level among the evaluated PySR, TPSR, uDSR, Operon, and DSR runs. The ablations further show that exactification is essential for symbolic recovery at 20% and 50% noise, and that removing raw-data refit cross-fitting reduces the 50%-noise recovery rate from 66.7% to 28.3%.

Our contributions are:

- We formulate recovery from noisy neural black-box surrogates as a constrained scientific model-selection problem requiring predictive agreement, structural validity, and exact symbolic equivalence.
- We introduce a bounded exact-search and exactification pipeline that combines physically motivated candidate families with sparse support recovery, rational constant identification, joint rationalization, and factor-level simplification.
- We incorporate noisy-data refit cross-fitting and confidence-aware selection to test whether fixed symbolic structures remain supported across held-out observations.
- We evaluate cross-formula generalization separately from formula development and compare against representative evolutionary and neural-guided symbolic-regression systems under increasing observation noise.

## 2 Related Work

**Scientific equation discovery and symbolic regression.** Symbolic regression has long been used to discover compact analytical relations directly from data. Evolutionary approaches demonstrated that experimental observations can be distilled into concise natural laws (Schmidt & Lipson, 2009; **?**; Koza, 1994). Sparse identification methods instead construct a prescribed library of candidate functions and select a parsimonious active subset, providing a principled framework for discovering nonlinear dynamical systems and partial differential equations (Brunton et al., 2016; Rudy et al., 2017; Kaiser et al., 2018). Physics-inspired systems such as AI Feynman further exploit algebraic structure and domain-motivated decompositions to improve search efficiency (Udrescu & Tegmark, 2020). These methods motivate our use of bounded expression families and sparse structural selection. Our setting is different: rather than fitting symbolic expressions directly to clean observations, we recover a latent law from a fixed neural black-box surrogate trained on noisy data.

**Neural-to-symbolic physical discovery.** Neural-to-symbolic methods have shown that learned representations can support symbolic extrapolation and interpretable equation learning (Martius & Lampert, 2016; Sahoo et al., 2018; Cranmer et al., 2020). They establish the value of combining neural function approximation with symbolic representations, but typically learn the predictor and symbolic structure jointly or transform learned representations into expressions. In contrast, our neural black-box surrogate is trained first and then treated as a queryable, potentially biased intermediate representation. Symbolic identification, exact coefficient recovery, and noise-aware validation are performed as explicit subsequent stages, allowing their effects to be isolated experimentally.

**Modern symbolic-regression systems.** Recent systems improve symbolic search through population-based evolution, neural proposal mechanisms, or sequence modeling. PySR and Operon represent optimized evolutionary or population-based search strategies (Cranmer, 2023; Burlacu et al., 2020), while DSR, uDSR, and TPSR use neural policies or transformer-based mechanisms to propose symbolic expressions (Petersen et al., 2021; Landajuela et al., 2022; Shojaee et al., 2023). These systems generally balance predictive error against expression complexity and can produce accurate numerical approximations. We use them as representative baselines rather than treating them as interchangeable implementations: their search spaces and proposal mechanisms differ. Our comparison targets the specific setting of recovering formulas from neural black-box models trained at increasing noise levels.

**Exact symbolic recovery versus numerical fitting.** Most symbolic-regression evaluations emphasize validation error, expression complexity, or a Pareto trade-off between the two. Such criteria are useful for predictive modeling but do not guarantee that the recovered expression is algebraically the same as the generating law. For example, a floating-point coefficient can yield nearly indistinguishable values on a finite test set while failing to represent a simple rational constant. Our primary criterion is therefore normalized symbolic equivalence, supplemented by NMSE, validity, and complexity. This distinction motivates exactification: rather than treating rationalization as cosmetic rounding, we combine term pruning, confidence-aware rational projection, joint treatment of rational numerators and denominators, and factor-level reconstruction. It also motivates noisy-data refit cross-fitting, which evaluates fixed structures on held-out observations instead of selecting them solely from surrogate-query or in-sample loss. In this way, our method combines bounded structural search, discrete coefficient recovery, and noise-aware validation in a setting not captured by numerical error alone.

# 3 METHOD

## 3.1 PROBLEM FORMULATION

Let $f^\star : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ denote an unknown physical law and let the observations be

$$y_i = f^\star(\boldsymbol{x}_i) + \epsilon_i, \qquad i = 1, \dots, n, \tag{1}$$

where $\epsilon_i$ denotes measurement noise. A neural black-box surrogate $g_\phi$ is trained on $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. We assume query access to $g_\phi(\boldsymbol{x})$ and, through automatic differentiation, to its input Jacobian $\nabla_{\boldsymbol{x}} g_\phi(\boldsymbol{x})$. The objective is not to approximate $g_\phi$ with an unrestricted symbolic expression, but to recover a concise expression $s \in \mathcal{S}$ that represents the latent law $f^\star$.

This distinction is important because the optimization target available during discovery is the surrogate rather than $f^\star$. A candidate can agree closely with $g_\phi$ while inheriting its approximation bias, and distinct symbolic structures can be nearly indistinguishable on a finite domain. We therefore formulate recovery as constrained model selection,

$$\widehat{s} = \arg \min_{s \in \mathcal{S}_{\text{valid}}} \left\{ \widehat{R}_{\text{cf}}(s) - \widehat{\sigma}_\epsilon^2 \right\}_+ + \lambda\, L(s), \tag{2}$$

where $\widehat{R}_{\text{cf}}$ is a cross-fitted risk on the original noisy observations, $\widehat{\sigma}_\epsilon^2$ is an estimated normalized noise floor, $L(s)$ is a description-length penalty, and $\mathcal{S}_{\text{valid}}$ enforces structural and numerical constraints. The positive part prevents irreducible observation noise from rewarding unnecessarily flexible candidates. The search itself is guided by fresh evaluations of $g_\phi$, whereas the final statistical comparison is grounded in the original observations.

## 3.2 FRAMEWORK OVERVIEW

The method proceeds through four stages: fresh surrogate querying, bounded symbolic search, exactification, and noisy-data cross-fitting. This separation reduces dependence on the neural network's training design and prevents combinatorial search from adapting directly to individual noise realizations. Figure 1 summarizes the workflow.

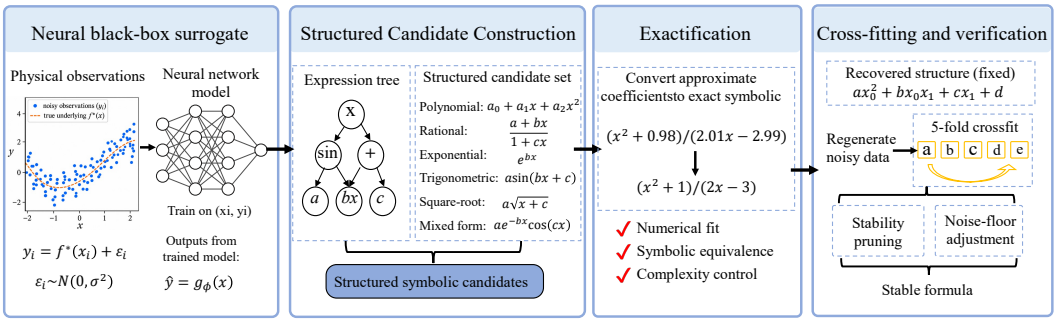


Figure 1: Overall architecture of the proposed structure-constrained neural-to-symbolic formula-recovery framework. Noisy physical observations are first used to train a neural black-box surrogate $g_\phi$. Fresh surrogate queries then guide a bounded collection of polynomial, rational, exponential, trigonometric, square-root, and mixed candidate families. Continuous coefficients are sparsified and projected to simple exact constants during exactification. Finally, each fixed structure is refit by five-fold cross-fitting on regenerated noisy data and is accepted only when it satisfies numerical-fit, symbolic-equivalence, complexity, stability, and noise-floor criteria. The neural surrogate is used for smooth proposal generation, whereas the original observations provide the final statistical verification.

### 3.3 Structure-constrained exact search

**Bounded algebraic families.** For input dimension $d$ and maximum total degree $D$, define the monomial index set

$$\mathcal{A}_D = \left\{ \boldsymbol{a} \in \mathbb{N}_0^d : \|\boldsymbol{a}\|_1 \le D \right\}, \qquad m_{\boldsymbol{a}}(\boldsymbol{x}) = \prod_{r=1}^{d} x_r^{a_r}. \tag{3}$$

The polynomial family is

$$p_{\boldsymbol{c}}(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \mathcal{A}_D} c_{\boldsymbol{a}} m_{\boldsymbol{a}}(\boldsymbol{x}), \tag{4}$$

and the rational family is

$$r_{\boldsymbol{c},\boldsymbol{b}}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{a} \in \mathcal{A}_{D_p}} c_{\boldsymbol{a}} m_{\boldsymbol{a}}(\boldsymbol{x})}{1 + \sum_{\boldsymbol{a} \in \mathcal{A}_{D_q} \setminus \{\boldsymbol{0}\}} b_{\boldsymbol{a}} m_{\boldsymbol{a}}(\boldsymbol{x})}. \tag{5}$$

The denominator intercept is fixed to one to remove the global scale indeterminacy of rational representations. Polynomial coefficients are estimated by linear least squares. Rational coefficients are obtained from the linearized relation $\widetilde{y}\, q(\boldsymbol{x}) = p(\boldsymbol{x})$. We use several relative sparsity thresholds: coefficients below a threshold relative to the largest fitted coefficient are removed, after which the active coefficients are refitted. This produces a regularization path of candidate supports rather than committing to one sparsity level.

**Bounded analytic grammar.** The transcendental search is generated from a finite atom bank rather than an unrestricted expression tree. The bank contains

$$h(\boldsymbol{x}) \in \left\{ x_j,\ \sin(\alpha x_j^k),\ \cos(\alpha x_j^k),\ \exp(\alpha x_j^k),\ \log(1 + x_j^k),\ \sqrt{x_j},\ \sqrt{x_j x_\ell} \right\}, \tag{6}$$

where $k \in \{1, 2\}$ and $\alpha$ belongs to a bounded vocabulary of simple scales. The same grammar is used for every benchmark and does not depend on the ground-truth formula. Candidate templates include a single atom with an affine output, sums of two atoms, selected compatible products, trigonometric phases of the form $\sin(\alpha x + \beta x^2)$, exponentially modulated trigonometric terms, and a linear background augmented by $\sqrt{x_j x_\ell}\ \mathrm{trig}(x_k)$. We also include polynomial-over-analytic forms

$$\frac{p(\boldsymbol{x})}{1 + \gamma h(\boldsymbol{x})} \quad \text{and} \quad \frac{p(\boldsymbol{x})}{\exp(\alpha x_j^k) - 1}, \tag{7}$$

with the latter represented through `expm1` for numerical stability near zero.

For each discrete template, the remaining output coefficients are linear and are solved by least squares. Templates are ranked by normalized fitting error, and only a bounded number of candidates per family is retained. Search terminates early for more expressive families when an algebraic candidate already attains the numerical exact-fit tolerance. Consequently, combinatorial complexity is controlled by degree bounds, finite scale vocabularies, restricted composition rules, and per-template candidate budgets. This is the principal difference from unconstrained recursive symbolic search: complexity is restricted before optimization, not merely penalized after an arbitrarily expressive formula has been generated.

**Candidate validity.** A fitted candidate is discarded if its predictions are non-finite. For rational and analytic-ratio candidates, we additionally require

$$\min_{\boldsymbol{x}\in\mathcal{V}} |q_s(\boldsymbol{x})| \geq \delta_q, \tag{8}$$

where $\mathcal{V}$ is the candidate-evaluation set and $\delta_q > 0$ is a prescribed stability margin. Algebraically identical expressions are canonicalized and deduplicated; when duplicates remain, the lower-error representative is retained.

### 3.4 Exactification as discrete coefficient recovery

Continuous regression may produce approximate coefficients even when the candidate structure is correct. Exactification addresses this continuous-to-discrete gap by pruning unsupported terms, recovering simple constants, and reconstructing compact factors. For rational expressions, numerator and denominator coefficients are treated jointly to preserve their shared scale. Floating-point and exactified candidates are both retained and compared empirically; details of the projection, uncertainty criteria, and factor-level procedure are provided in Appendix C.

### 3.5 Noisy-data cross-fitting and confidence-aware refitting

The surrogate is useful for proposing structures because it smooths observation noise, but it may introduce systematic approximation bias. We therefore return to the original observations after structure generation. For a candidate with fixed discrete structure $s$ and linearly identifiable coefficient vector $\boldsymbol{\theta}$, let

$$\widehat{\boldsymbol{\theta}}^{(-k)} = \arg\min_{\boldsymbol{\theta}} \sum_{i\notin I_k} \left(y_i - s(\boldsymbol{x}_i;\boldsymbol{\theta})\right)^2 \tag{9}$$

be the estimate obtained without fold $I_k$. Its cross-fitted per-observation loss is

$$\ell_i^{\mathrm{cf}}(s) = \frac{\left(y_i - s(\boldsymbol{x}_i;\widehat{\boldsymbol{\theta}}^{(-k)})\right)^2}{\widehat{\mathrm{Var}}(y)}, \qquad i \in I_k, \tag{10}$$

which yields predictions for every observation from coefficients fitted without that observation's fold. This prevents in-sample coefficient adaptation from being mistaken for evidence in favor of a structure.

For each linearized system $\boldsymbol{X}_s\boldsymbol{\theta} \approx \boldsymbol{y}_s$, coefficient uncertainty is estimated by

$$\widehat{\mathrm{Cov}}(\widehat{\boldsymbol{\theta}}) = \widehat{\sigma}_s^2(\boldsymbol{X}_s^\top \boldsymbol{X}_s)^\dagger, \tag{11}$$

where † denotes the pseudoinverse. A coefficient is pruned if its $z$-level interval contains zero. Rationalization is performed only when a bounded-denominator rational lies inside the same interval, linking exactification to statistical uncertainty rather than to decimal proximity alone. After cross-fitted evaluation, coefficients are re-estimated using all observations to construct the final version of that fixed structure.

A direct refit of a rational model through $yq(\boldsymbol{x}) = p(\boldsymbol{x})$ places noisy $y$ inside the design matrix and induces an errors-in-variables bias. Accordingly, once a rational candidate has been exactified, its numerator and denominator shapes are held fixed and cross-fitting estimates only a global output scale. This preserves the discrete structure while allowing calibration to the original observations.

Let $\widehat{R}_{\mathrm{cf}}(s) = n^{-1} \sum_i \ell_i^{\mathrm{cf}}(s)$. We define the noise-corrected empirical risk as

$$\widehat{R}_{\mathrm{excess}}(s) = \max\left\{ \widehat{R}_{\mathrm{cf}}(s) - \widehat{\sigma}_\epsilon^2, 0 \right\}. \tag{12}$$

Subtracting the estimated noise floor discourages the selection rule from rewarding candidates that reduce training error by modeling irreducible noise.

### 3.6 Derivative-aware structural validation

Function values alone may not distinguish nearby structures on a bounded sample. We therefore compare candidate and surrogate input sensitivities on an independent validation set. A robust, dimension-normalized Jacobian discrepancy complements value error, while an effective-variable recall constraint rejects candidates that omit influential inputs. The Jacobian weight is reduced when surrogate uncertainty is high. Full loss definitions, thresholds, and implementation details are given in Appendix E.2.

### 3.7 Description length and paired candidate selection

Operation count alone does not distinguish a simple exact constant from an arbitrary floating-point parameter. We therefore use an MDL-inspired description length (Rissanen, 1978; Grunwald, 2007).

$$L(s) = \mathrm{count_ops}(s) + \sum_{a \in \mathrm{Const}(s)} \ell(a), \tag{13}$$

where arbitrary floating-point atoms receive a fixed large cost, rational constants are charged according to the bit lengths of their numerator and denominator, and integers are charged by integer bit length. The candidate score is

$$S(s) = \widehat{R}_{\mathrm{excess}}(s) + \lambda\, u_\phi L(s), \tag{14}$$

where $u_\phi$ is a surrogate-uncertainty scale. This scaling avoids imposing a complexity preference that is disproportionate to the statistically resolvable error.

Let $s_\star$ be the feasible candidate with minimum score. For every candidate $s$, we compute the standard error of the paired loss differences $\ell_i^{\mathrm{cf}}(s) - \ell_i^{\mathrm{cf}}(s_\star)$. Candidate $s$ enters the confidence set if

$$S(s) \le S(s_\star) + \max\left\{ \eta |S(s_\star)|, z\, \widehat{\mathrm{SE}}\big[\ell^{\mathrm{cf}}(s) - \ell^{\mathrm{cf}}(s_\star)\big] \right\}. \tag{15}$$

The expression with minimum description length is selected from this set, with score used as a tie-breaker. Pairing is essential because candidates are evaluated on the same observations; it removes observation-level variability shared by both formulas and asks whether their predictive difference is statistically resolvable. The final expression is subsequently subjected to random-domain and gradient-based adversarial verification, while exact recovery in the experiments is determined independently through symbolic normalization and equivalence testing.

## 4 Experiments

### 4.1 Experimental setup

**Development and unseen-formula protocols.** We separate formula-development evidence from generalization evidence. The development partition contains 13 analytic formulas spanning polynomial, rational, trigonometric, exponential, damped, and mixed structures. These formulas were used to design and validate the exactification and candidate-structure mechanisms. The evaluation partition contains 20 formulas held out from method development and exactification design. For each partition, a neural black-box surrogate is trained independently from noisy observations for five random seeds and seven relative Gaussian-noise levels, $\{0\%, 5\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$. The unseen-formula experiment therefore evaluates up to $20 \times 5 \times 7 = 700$ formula-recovery trials; the development experiment provides $13 \times 5 \times 7 = 455$ nominal trials. Missing or incomplete runs are retained in the denominator accounting and are reported explicitly.

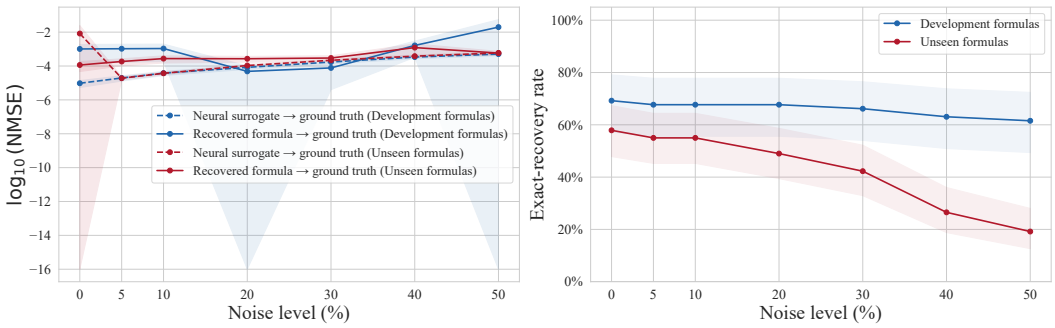


Figure 2: Numerical and symbolic recovery on 13 development formulas and 20 held-out formulas across five random seeds and seven Gaussian-noise levels. The left panel reports the base-10 logarithm of the mean NMSE to the ground-truth function for the neural black-box surrogate and the recovered formula; the right panel reports exact-equivalence recovery rates with 95% Wilson intervals over available formula–seed trials. The held-out formulas were not used to design the candidate grammar or exactification rules.

**Neural black-box training and recovery.** For each formula–noise–seed configuration, we train a four-layer tanh neural black-box surrogate with width 512 on the noisy observations. Training uses up to 20,000 epochs, batches of 2,048, a learning rate of $8 \times 10^{-4}$, warm-up, and validation-based checkpointing. Formula recovery then queries the fitted surrogate on fresh points and applies the same bounded exact-search, exactification, noisy-data refitting, and structural-selection pipeline without using the held-out formula identity. The recovery stage uses 8,192 fresh surrogate-query points, polynomial and rational candidates up to degree three, bounded transcendental templates, five noisy-refit folds, and a minimum denominator magnitude of 0.05.

**Metric and statistical convention.** A recovery is counted as exact only when the normalized recovered expression is symbolically equivalent to the target expression under the shared SymPy-based evaluator. We additionally report normalized mean-squared error (NMSE), completion rate, and expression complexity. For recovery-rate curves, uncertainty bands use 95% Wilson intervals over available formula–seed trials. Incomplete trials are not silently treated as successes; each aggregate is reported with its numerator and denominator. The development and unseen partitions are analyzed separately because the former supports method construction whereas the latter measures cross-formula generalization.

### 4.2 Development versus unseen-formula generalization

Figure 2 summarizes numerical fidelity and exact-equivalence recovery across noise levels. Exact recovery on development formulas decreases from 69.2% without noise to 61.5% at 50% noise; on held-out formulas, it declines from 57.9% to 19.2%. Intermediate noise-level results are shown in the figure. The recovered formulas can remain numerically close even when they are not symbolically equivalent, highlighting the distinction between function approximation and exact identification. The lower held-out performance also reflects the limits of generalization from the bounded candidate grammar.

### 4.3 Ablation study

Figure 3 isolates the mechanisms responsible for robust recovery. The full method obtains 75.0%, 73.3%, and 66.7% exact recovery at 0%, 20%, and 50% noise, respectively. Removing exactification leaves the noise-free rate unchanged at 75.0% but reduces exact recovery to 0% at both 20% and 50% noise. This sharp separation confirms that low-error continuous fits are not sufficient under noisy neural approximation: term pruning and discrete constant recovery are required to cross the boundary from numerical approximation to symbolic equivalence. Removing raw-data refit cross-fitting lowers recovery to 60.0%, 48.3%, and 28.3%, with an increasingly severe deficit at high noise.

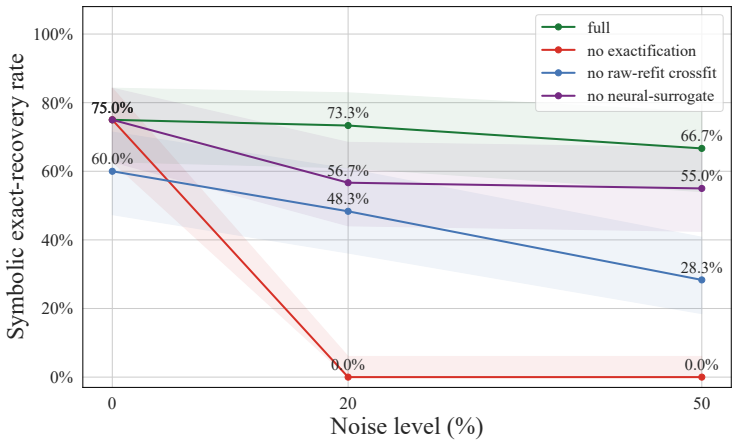


Figure 3: Ablation of exactification, raw-data refit cross-fitting, and the neural black-box surrogate. Curves report symbolic exact-recovery rates over five seeds at 0%, 20%, and 50% noise; annotations give the aggregate rates. Exactification is essential at nonzero noise, while cross-fitting and neural smoothing increasingly improve recovery as the noise level rises.

Cross-fitting therefore acts as a stability mechanism that rejects structures or coefficients tied to a particular noise realization.

Taken together, the ablations support a division of labor among the components: structure-constrained search proposes a finite set of scientific hypotheses, the neural surrogate suppresses observation-level noise during proposal generation, exactification performs the decisive continuous-to-discrete transition, and raw-data cross-fitting tests whether the resulting structure remains supported by the original observations.

### 4.4 Comparison with symbolic-regression baselines

Figure 4 compares our method with PySR (Cranmer, 2023), TPSR (Shojaee et al., 2023), uDSR (Landajuela et al., 2022), Operon (Burlacu et al., 2020), and DSR (Petersen et al., 2021) on recovery from neural black-box models trained at the same seven noise levels. PySR is strongest at 0% noise (61.5% versus 56.3% for ours), but its accuracy falls to 30.8% at 5% noise and to 15.4% at 10–30% noise. In contrast, our method maintains 55.0% at both 5% and 10%, 49.0% at 20%, and 41.7% at 30%. At 40% and 50% noise it still recovers 26.4% and 19.2%, whereas the strongest baseline reaches only 7.7% at either level. Operon exhibits isolated peaks of 38.5% at 5% and 30.8% at 20%, but does not maintain this performance as noise increases. DSR, TPSR, and uDSR similarly decline to zero at 40–50% noise.

The comparison therefore reveals a specific advantage rather than uniform dominance in every regime: under noise-free data, a flexible symbolic-regression baseline can be competitive or superior, but its recovery deteriorates rapidly after the target has been mediated by a neural black box trained on noisy observations. Our method is substantially more robust in this setting because it separates smooth structure proposal, exact coefficient recovery, and held-out noisy-data validation. Across 5–50% noise, it is the best-performing method at every reported level, with the advantage widening in the high-noise regime.

### 4.5 Limitations and reproducibility

The results also delimit the current method. Recovery on unseen formulas decreases from 57.9% without observation noise to 19.2% at 50% noise, and the category analysis shows that this decline is concentrated in formula families that are weakly represented by the bounded transcendental grammar. Thus, the method should not be interpreted as a universal symbolic solver: successful recovery requires that the latent law be sufficiently represented by the candidate portfolio and that the neural black-box surrogate preserve enough functional structure for discrete identification. The 13 devel-

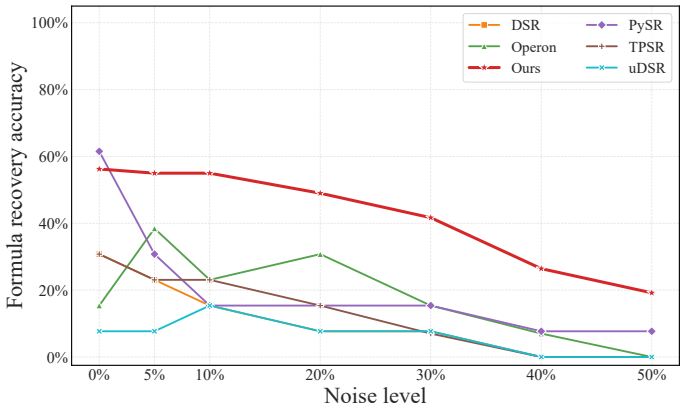


Figure 4: Equivalent-formula recovery accuracy of our method and five symbolic-regression baselines when recovering formulas from neural black-box models trained at seven noise levels. Our method is competitive in the noise-free setting and achieves the highest recovery accuracy at every nonzero noise level.

opment formulas and 20 held-out formulas are kept separate throughout the analysis, all primary experiments use five seeds, and denominators reflect available trials rather than silently imputing missing runs. Per-trial records, plotting tables, and evaluation scripts are retained for auditability; full formula lists, seed-level results, baseline configurations, runtime statistics, and failure cases are provided in the supplementary material.

## 5 Conclusion

Physical observations are typically noisy. Although neural networks can fit such observations accurately, they often fail to reveal the concise laws governing the underlying system. We therefore presented a structure-constrained neural-to-symbolic framework for recovering verifiable formulas from neural black-box surrogates. The framework separates smooth proposal generation from discrete scientific identification by combining fresh surrogate queries, bounded structural search, exactification, and noisy-data cross-fitting with numerical and structural validation.

The full method achieves 75.0%, 73.3%, and 66.7% exact recovery on the development suite at 0%, 20%, and 50% noise, respectively, while held-out recovery decreases from 57.9% to 19.2% across the same noise range. Ablations show that exactification is essential for symbolic recovery under noise and that raw-data refit cross-fitting improves robustness as noise increases. Compared with the evaluated symbolic-regression baselines, the method has the strongest recovery at every nonzero noise level, although it is not uniformly superior in the noise-free setting.

These results also define the method's scope. Recovery requires an adequate bounded grammar, sufficient domain coverage, a surrogate that preserves relevant structure, and distinguishable signal under the available noise level. Low NMSE alone does not establish symbolic equivalence, and the method cannot guarantee recovery of formulas outside its search space. Future work should expand the grammar in a controlled manner, use uncertainty-aware surrogate querying, record complete candidate beams and rejection reasons, and evaluate larger multi-domain benchmarks with stronger out-of-domain tests. Overall, the framework provides a practical route from accurate but opaque neural predictions to compact, testable scientific hypotheses, while making explicit the structural and statistical conditions required for reliable recovery.

## 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.

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon c++ an efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 genetic and evolutionary computation conference companion*, pp. 1562–1570, 2020.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020.

Peter D Grunwald. *The minimum description length principle*. The MIT press, 2007.

Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings. Mathematical, physical, and engineering sciences*, 474(2219):20180335, 2018.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. *Advances in neural information processing systems*, 35:33985–33998, 2022.

Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

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.

Maziar Raissi, Paris Perdikaris, and George 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.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International conference on machine learning*, pp. 4442–4450. Pmlr, 2018.

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

Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan Reddy. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36: 45907–45919, 2023.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science advances*, 6(16):eaay2631, 2020.

## A ATHE USE OF LARGE LANGUAGE MODELS (LLMS)

Large Language Models (LLMs) were used solely as auxiliary tools for language polishing andminor editing. They did not contribute to the research ideation, experimental design, or writing toan extent that would qualify them as contributors. All substantive research ideas, methodologies,and results presented in this paper are entirely the work of the authors.

# B Complete Algorithm and Pseudocode

This appendix gives the complete operational pipeline implemented in the code repository. The implementation separates continuous function approximation from discrete formula identification. A neural black-box surrogate is used to generate a smooth query target, but the candidate structures are drawn from a fixed, truth-agnostic portfolio. After candidate generation, coefficients are sparsified and exactified, fixed structures are refitted on the original noisy observations, and the final candidate is selected using structural and statistical criteria.

## B.1 Overall pipeline

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be the noisy observations and let $g_\phi$ be the fitted neural black-box surrogate. The complete pipeline is:

$$g_\phi \longrightarrow \text{fresh surrogate queries} \longrightarrow \text{bounded structure search} \longrightarrow \text{sparse refit} \longrightarrow \text{rational projection} \longrightarrow \text{factor rationalization} \longrightarrow \text{noisy cross-fitting} \longrightarrow \text{structural validation} \longrightarrow \text{final selection}.$$

```
Algorithm 1: Neural-to-symbolic exact formula recovery
Input: noisy data D, trained surrogate g_phi, input domain X
Output: selected symbolic candidate s_hat

1. Draw fresh search inputs X_s from X and evaluate Y_s = g_phi(X_s).
2. Generate polynomial, rational, and bounded analytic candidates on (X_s, Y_s).
3. For each candidate:
       a. sparse-refit its continuous coefficients;
       b. create an unsnapped candidate;
       c. prune near-zero coefficients and rationalize compatible constants;
       d. for univariate rational candidates, attempt factor rationalization;
       e. discard non-finite or denominator-unstable candidates.
4. Canonicalize candidate expressions and retain the lowest-error duplicate.
5. If original noisy observations are available:
       a. keep each candidate's discrete structure fixed;
       b. split D into K folds;
       c. refit coefficients on K-1 folds and evaluate on the held-out fold;
       d. use confidence intervals to prune and rationalize coefficients;
       e. refit the accepted structure on all observations.
6. Draw validation inputs X_v and evaluate g_phi(X_v) and its Jacobian.
7. Compute value error, robust Jacobian error, effective-variable recall,
   denominator stability, description length, and cross-fitted excess loss.
8. Remove infeasible candidates. Form a paired-confidence set around the
   lowest-score candidate and choose the shortest-description candidate in it.
9. Verify the selected candidate on random and adversarial domain points.
10. Return the selected symbolic expression and all per-candidate diagnostics.
```

The pipeline stops candidate generation for more expressive analytic families when an algebraic candidate reaches the exact-fit tolerance. Candidate search also terminates for a branch when all generated candidates are non-finite, singular, or otherwise invalid. Final selection never treats a missing or invalid candidate as a successful recovery.

## B.2 Structured exact-search pseudocode

```
Algorithm 2: Structured exact search
Input: fit points X_s, target values Y_s, degree bounds D_p,D_q,
       sparsity thresholds T, analytic scale vocabulary A
Output: finite candidate list C

1. C <- empty list.
2. For d = 1,...,D_p:
```

```
       P <- all total-degree monomials with degree at most d, including 1.
       Z <- monomial design matrix P(X_s).
       For tau in T:
           beta <- sparse_refit(Z, Y_s, tau).
           Add beta and its exactified version to C if valid.
3. For p = 1,...,D_q and q = 1,...,D_q:
       P <- numerator monomial basis of degree p.
       Q <- denominator monomial basis of degree q.
       Form [P(X_s), -diag(Y_s) Q_nonconstant(X_s)].
       For tau in T:
           Solve the linearized rational regression.
           Normalize denominator intercept to one.
           Add coefficient-wise and jointly exactified candidates if valid.
4. If no algebraic candidate reaches the exact-fit tolerance:
       Construct the finite atom bank from A.
       Score unary atoms, additive pairs, and allowed products.
       Score nested trigonometric, modulated-trigonometric, and
       linear-root-trigonometric templates.
       Score polynomial-over-exp and polynomial-over-expm1 ratios.
       Keep only the best R candidates from each template group.
5. If factor rationalization is enabled and X_s is univariate:
       For each non-snapped rational candidate, factor numerator and denominator.
       Snap real and conjugate-pair factors and refit one global scale.
       Add the factor candidate if valid.
6. Canonicalize expressions, remove duplicate strings, and sort by
   fit NMSE followed by symbolic complexity.
7. Return C.
```

### B.3 Polynomial candidate generation

For $d$ inputs and maximum total degree $D$, the implementation constructs

$$\mathcal{A}_D = \left\{ \boldsymbol{a} \in \mathbb{N}_0^d : \|\boldsymbol{a}\|_1 \le D \right\}, \qquad m_{\boldsymbol{a}}(\boldsymbol{x}) = \prod_{j=1}^{d} x_j^{a_j}. \tag{16}$$

The constant monomial $m_{\mathbf{0}} = 1$ is always included. The basis is sorted deterministically by total degree and then exponent tuple. For each degree $d_0 \in \{1, \dots, D\}$, the design matrix is

$$Z_{d_0}[i, :] = \left( m_{\boldsymbol{a}}(\boldsymbol{x}_i) \right)_{\boldsymbol{a} \in \mathcal{A}_{d_0}}. \tag{17}$$

The least-squares coefficient vector is first obtained from $\mathrm{lstsq}(Z_{d_0}, \boldsymbol{y})$. For a relative threshold $\tau$, active coefficients satisfy

$$|\hat{c}_j| \ge \tau \max_k |\hat{c}_k|, \tag{18}$$

and the active support is refitted by least squares. The implementation evaluates

$$\tau \in \{10^{-8}, 10^{-4}, 10^{-3}, 10^{-2}\}$$

for each degree, retaining the resulting sparse and non-sparse variants. If all coefficients are initially below threshold, the largest-magnitude coefficient is retained. A candidate is discarded when its prediction is non-finite or its normalized fitting loss is not finite.

### B.4 Rational candidate generation

For numerator degree $D_p$ and denominator degree $D_q$, write

$$p(\boldsymbol{x}) = \boldsymbol{P}(\boldsymbol{x})\boldsymbol{c}, \qquad q(\boldsymbol{x}) = 1 + \boldsymbol{Q}_{-0}(\boldsymbol{x})\boldsymbol{b}, \qquad r(\boldsymbol{x}) = p(\boldsymbol{x})/q(\boldsymbol{x}), \tag{19}$$

where the denominator constant coefficient is excluded from $\boldsymbol{Q}_{-0}$. The relation $y_i q(\boldsymbol{x}_i) = p(\boldsymbol{x}_i)$ gives the linear system

$$[\boldsymbol{P}(\boldsymbol{x}_i), -y_i \boldsymbol{Q}_{-0}(\boldsymbol{x}_i)] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix} \approx y_i. \tag{20}$$

The implementation enumerates $D_p, D_q \in \{1, 2, 3\}$, applies the same sparse-refit thresholds as for polynomials, and then restores the denominator intercept to one. For each solution it retains: (i) the continuous candidate, (ii) independently snapped numerator and denominator coefficients, and (iii) a jointly snapped candidate using a tolerance multiplier of five and maximum denominator twelve. The ordinary coefficient snap uses tolerance $5 \times 10^{-3}$ and maximum denominator $100$.

A candidate is rejected when any prediction is non-finite or when its minimum fitted-domain denominator magnitude is below $10^{-6}$. During final validation, the stricter configured denominator margin is $\delta_q = 0.05$. This distinction prevents numerical singularities during candidate construction while enforcing a more conservative stability condition during selection.

### B.5 Transcendental template generation

For each input coordinate $x_j$, the atom bank includes identity atoms, $\sin(\alpha x_j^k)$, $\cos(\alpha x_j^k)$, and $\exp(\alpha x_j^k)$ for $k \in \{1, 2\}$. It also includes domain-safe $\log(1 + x_j^k)$ atoms, $\sqrt{x_j}$, and pairwise $\sqrt{x_j x_\ell}$ atoms. The universal scale vocabulary in the paper experiments is

$$\{-2.0, -1.5, -1.2, -1.0, -0.8, -0.7, -0.5, -0.4, -0.3, 0.3, 0.4, 0.5, 0.7, 0.8, 1.0, 1.2, 1.5, 2.0\}.$$

The implementation removes sign-symmetric trigonometric duplicates: the negative sign of a sine argument can be absorbed into its outer coefficient, and cosine is even. Exponential atoms retain both signs.

The permitted template groups are:

1. one unary atom with an intercept and one outer coefficient;
2. additive pairs within separate operator banks;
3. products of distinct atoms when their operators are both trigonometric, exponential–trigonometric, or square-root–trigonometric;
4. sums involving a simple trigonometric atom and a quadratic-phase trigonometric atom $\sin(\alpha x + \beta x^2)$ or $\cos(\alpha x + \beta x^2)$;
5. exponential-modulated shifted trigonometric products;
6. linear background plus $\sqrt{x_i x_j}\ \mathrm{trig}(x_k)$;
7. polynomial numerators over $1 + \gamma \exp(\alpha x_j^k)$;
8. polynomial numerators over $\mathrm{expm1}(\alpha x_j^k)$.

For each fixed template, outer coefficients are fitted by least squares or a closed-form equivalent. At most twelve candidates per template group are retained. The search does not permit arbitrary recursive nesting, arbitrary operator products, unbounded inner scales, or polynomial degree beyond the configured bound. The stable $\mathrm{expm1}$ representation is used to avoid cancellation near zero.

## C Exactification Details

### C.1 Coefficient pruning and sparse refitting

The initial sparse-refit stage uses relative support thresholds rather than a single fixed threshold. A coefficient is active when its magnitude is at least $\tau$ times the largest coefficient magnitude in the current solution. The active columns are then refitted without penalty, and inactive coefficients are set to zero. This generates multiple candidate supports for the same expression family. During noisy-data refitting, uncertainty-aware pruning is applied only to parameters marked as structurally pruneable. A coefficient is set to zero when its confidence interval contains zero.

### C.2 Rational constant recovery

For a fitted coefficient $\hat{c}$, the implementation first defines

$$a(\hat{c}) = \tau \max(1, |\hat{c}|). \tag{21}$$

If $|\hat{c}| \leq a(\hat{c})$, it is mapped to zero. Otherwise, denominators $q = 1, \ldots, Q$ are scanned in increasing order, with numerator $p = \mathrm{round}(q\hat{c})$. The first $p/q$ satisfying

$$\left| \hat{c} - \frac{p}{q} \right| \leq a(\hat{c}) \tag{22}$$

is selected. Thus the implementation prioritizes the smallest denominator, rather than the closest rational with an arbitrarily large denominator. If no candidate satisfies the tolerance, the original floating-point coefficient is retained. In the main search, $(\tau, Q) = (5 \times 10^{-3}, 100)$; joint and factor-level procedures use their separately specified tolerances below.

### C.3 Confidence-aware coefficient projection

For a linearized fixed-structure regression with design matrix $\boldsymbol{X}$ and solution $\hat{\boldsymbol{\theta}}$, the residual variance and approximate covariance are

$$\hat{\sigma}^2 = \frac{\|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\theta}}\|_2^2}{\max(n - p, 1)}, \qquad \widehat{\mathrm{Cov}}(\hat{\boldsymbol{\theta}}) = \hat{\sigma}^2 (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^\dagger. \tag{23}$$

The standard error is the square root of the diagonal. With confidence multiplier $z$, coefficient $j$ has interval

$$I_j = \left[ \hat{\theta}_j - z\ \mathrm{SE}(\hat{\theta}_j), \hat{\theta}_j + z\ \mathrm{SE}(\hat{\theta}_j) \right]. \tag{24}$$

The noisy-refit implementation uses $z = 4.0$ and maximum denominator 24. If zero belongs to $I_j$ and the parameter is pruneable, the coefficient is set to zero. Otherwise, rationalization is attempted only for a rational $p/q$ with $q \leq 24$ lying inside $I_j$. This prevents a value such as $0.5001$ from being converted to $1/2$ solely because its decimal value is close; the uncertainty interval must also support that discrete hypothesis. The initial exact-search projection is tolerance-based, whereas the noisy-refit projection is confidence-aware.

### C.4 Joint rationalization of rational expressions

A rational function is invariant under common scaling:

$$\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} = \frac{\alpha p(\boldsymbol{x})}{\alpha q(\boldsymbol{x})}. \tag{25}$$

Independent projection of numerator and denominator coefficients can therefore produce an arbitrary representation or destroy their relative scale. The implementation first divides both coefficient vectors by the denominator intercept $b_0$, then concatenates the normalized numerator vector and the nonconstant denominator vector:

$$\hat{\boldsymbol{w}} = \left[ \hat{\boldsymbol{c}}/b_0, \hat{\boldsymbol{b}}_{1:}/b_0 \right]. \tag{26}$$

The same small-denominator projection is applied to $\hat{\boldsymbol{w}}$, after which the denominator intercept is restored to one. Joint rationalization uses tolerance $5 \times 5 \times 10^{-3}$ and maximum denominator 12. The candidate is rebuilt, symbolically simplified, and rejected if the resulting prediction is non-finite or numerically unstable.

### C.5 Factor-level rationalization

For univariate rational candidates, the implementation additionally attempts factor-level recovery. It first trims numerically negligible trailing polynomial coefficients, computes roots with the polynomial convention used by NumPy, and clusters roots using tolerance $0.5$. A real root $r$ is projected to a simple value and represented by $(x - r)$. A complex root is paired with its conjugate and represented by a real quadratic factor

$$(x - z)(x - \bar{z}) = x^2 - 2\,\mathrm{Re}(z)x + |z|^2. \tag{27}$$

The linear coefficient and constant term of each real quadratic are independently projected using factor snap tolerance $0.15$ and maximum denominator 12. Approximately repeated roots are represented by repeated snapped factors. After reconstructing numerator and denominator factors, a provisional ratio $b(x)$ is formed and one global scale is fitted by

$$\hat{\alpha} = \frac{\sum_i b(x_i) y_i}{\sum_i b(x_i)^2}. \tag{28}$$

The scale is itself projected with factor snap tolerance $0.15$. The factor candidate is retained only if the reconstructed ratio is finite, non-degenerate, and has a valid denominator.

For example, a noisy polynomial with roots numerically close to $-1, -1, -1$ is transformed as

$$1.0002x^3 + 3.001x^2 + 2.998x + 0.9997 \longrightarrow \{-1.00, -1.00, -1.00\} \longrightarrow (x+1)^3.$$

The same procedure applies separately to numerator and denominator polynomials, after which the common output scale is refitted.

# D Noisy-Refit Cross-Fitting

## D.1 Fold construction and refitting

The original noisy data are partitioned into $K$ folds, with $K = 5$ in the paper configuration. In the implementation, fold assignments are deterministic modulo the number of folds:

$$I_k = \{i : i \bmod K = k\}. \tag{29}$$

For a fixed candidate structure $s(\boldsymbol{x}; \boldsymbol{\theta})$, the fold-$k$ coefficient estimate is

$$\hat{\boldsymbol{\theta}}^{(-k)} = \arg\min_{\boldsymbol{\theta}} \sum_{i \notin I_k} \left(y_i - s(\boldsymbol{x}_i; \boldsymbol{\theta})\right)^2, \tag{30}$$

using the candidate-specific linearized design system. Held-out losses are

$$\ell_i^{\mathrm{cf}}(s) = \frac{\left(y_i - s(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}^{(-k)})\right)^2}{\widehat{\mathrm{Var}}(y)}, \qquad i \in I_k. \tag{31}$$

After all folds have been evaluated, the same fixed structure is refit using all observations and subjected to the same confidence-aware pruning and rationalization procedure.

## D.2 Noise floor and excess risk

When the training metadata are available, the noise floor is reconstructed from the realized injected noise. If $\sigma^2_{\mathrm{real}}$ is the realized noise mean-square and $\mathrm{Var}(\boldsymbol{y})$ is the observed target variance, the normalized floor is

$$\widehat{\sigma}^2_{\epsilon} = \frac{\sigma^2_{\mathrm{real}}}{\mathrm{Var}(\boldsymbol{y})}. \tag{32}$$

The cross-fitted loss and excess loss are then

$$\widehat{R}_{\mathrm{cf}}(s) = \frac{1}{n} \sum_i \ell_i^{\mathrm{cf}}(s), \qquad \widehat{R}_{\mathrm{excess}}(s) = \max\{\widehat{R}_{\mathrm{cf}}(s) - \widehat{\sigma}^2_{\epsilon}, 0\}. \tag{33}$$

If the relevant training metadata are absent, the refit stage is disabled rather than assigning an unverified noise floor. A zero error on noisy observations is not an appropriate target: even the true law incurs residual measurement error. Subtracting the estimated irreducible component prevents a flexible candidate from being rewarded merely for explaining random noise, while retaining a nonnegative excess-risk score for model comparison.

## D.3 Rational refitting and errors-in-variables

For an unsnapped rational candidate, the linearized system uses

$$y_i q(\boldsymbol{x}_i) = p(\boldsymbol{x}_i), \tag{34}$$

so the observed response appears in the design matrix. Under noisy observations, this is an errors-in-variables configuration and can bias a direct refit of numerator and denominator coefficients. For a candidate that has already been exactified, the implementation therefore holds the snapped numerator and denominator shapes fixed and fits only one scalar output scale. If $b_s(\boldsymbol{x})$ is the exactified base ratio, the refit is

$$\hat{\alpha} = \arg\min_{\alpha} \sum_i (y_i - \alpha b_s(\boldsymbol{x}_i))^2 = \frac{\sum_i b_s(\boldsymbol{x}_i) y_i}{\sum_i b_s(\boldsymbol{x}_i)^2}. \tag{35}$$

This treatment preserves the discrete rational structure while still allowing calibration to the noisy observations.

# E Candidate Selection and Complexity

## E.1 Prediction and robust structural losses

On the validation set $\mathcal{V}$, value NMSE is

$$R_v(s) = \frac{\frac{1}{|\mathcal{V}|}\sum_{\boldsymbol{x}\in\mathcal{V}}(s(\boldsymbol{x}) - g_\phi(\boldsymbol{x}))^2}{\mathrm{Var}_{\mathcal{V}}(g_\phi)}. \tag{36}$$

The implementation also forms per-sample normalized squared residuals and, for Jacobian residuals, uses the Huber loss

$$\rho_\delta(r) = \begin{cases} r^2, & |r| \le \delta, \\ 2\delta|r| - \delta^2, & |r| > \delta. \end{cases} \tag{37}$$

The paper configuration uses $\delta = 1.5$ and trims the largest two percent of per-sample Jacobian losses. Trimming is applied only to the robust validation component; exact recovery is evaluated separately using symbolic equivalence.

## E.2 Jacobian and effective-variable validation

Let $J_s(\boldsymbol{x})$ and $J_g(\boldsymbol{x})$ denote candidate and surrogate Jacobians. The normalized Jacobian residual is

$$r_J(\boldsymbol{x}) = \frac{J_s(\boldsymbol{x}) - J_g(\boldsymbol{x})}{\sqrt{\mathbb{E}_{\mathcal{V}}[J_g(\boldsymbol{x})^2] + \varepsilon}}. \tag{38}$$

The candidate Jacobian loss is the mean Huber loss of this residual after upper-tail trimming. A variable is relevant when its mean absolute surrogate sensitivity is at least $0.01$ times the largest mean sensitivity. It is effectively recovered when the candidate-to-surrogate sensitivity ratio is at least $0.05$. Candidates must recover all relevant variables when structural validation is enabled. The Jacobian weight is $0.1$ and is attenuated by surrogate uncertainty $u$ according to

$$w_J = \frac{0.1}{1 + u/10^{-3}}. \tag{39}$$

This prevents uncertain surrogate derivatives from dominating structural selection.

## E.3 Numerical validity and denominator stability

A candidate is numerically valid only if its predictions and required derivatives are finite. For every rational or analytic-ratio candidate, we compute

$$q_{\min}(s) = \min_{\boldsymbol{x}\in\mathcal{V}} |q_s(\boldsymbol{x})|. \tag{40}$$

The final candidate is feasible only when $q_{\min}(s) \ge 0.05$. During initial construction a weaker $10^{-6}$ fitted-domain check prevents division by nearly zero. Candidates failing either check are removed from the feasible set; if no feasible candidate remains, the implementation falls back to the finite-candidate pool but still records the denominator diagnostic. Random verification samples the benchmark domain independently, whereas adversarial verification searches for high-error or low-denominator regions through gradient-based restarts and optimization steps. Neither verification stage changes the symbolic-equivalence label.

## E.4 Description length

The MDL-inspired description length is

$$L(s) = \mathrm{count_ops}(s) + \sum_{a\in\mathrm{Const}(s)} \ell(a). \tag{41}$$

The operation count is computed from the SymPy expression tree. A floating-point constant receives a fixed penalty of 12. For a rational number $p/q$, the implementation adds the bit lengths of $|p|$ and $q$ (each lower bounded by one bit). An integer receives its absolute-value bit length, also lower bounded by one. Thus $1/2$ is cheaper than an arbitrary long decimal, and a large rational is more

expensive than a small rational even when both count as one symbolic constant. The complexity penalty entering the score is

$$P_{\text{complexity}}(s) = \lambda u_\phi L(s), \qquad \lambda = 0.01, \tag{42}$$

where $u_\phi$ is the surrogate validation NMSE uncertainty floor.

### E.5 Paired confidence-set selection

Let $s_\star$ be the feasible candidate with minimum selection score. For any candidate $s$, the paired loss difference is

$$d_i(s) = \ell_i(s) - \ell_i(s_\star), \tag{43}$$

and its standard error is the empirical standard deviation divided by $\sqrt{n}$. Candidate $s$ is considered statistically near-best when

$$S(s) \le S(s_\star) + \max\left\{0.02|S(s_\star)|, 1.96\ \mathrm{SE}(d(s))\right\}. \tag{44}$$

The final output is the candidate with minimum description length among this confidence set; selection score breaks ties. When paired-confidence selection is disabled, the minimum-score candidate is returned directly. Pairing uses identical observations for every candidate and therefore removes noise variation shared by the compared expressions.

## F Conditional Identifiability and Limitations

### F.1 Conditional rational identifiability

Exactification does not imply that arbitrary real-valued physical constants are rational. It gives a conditional guarantee for constants that belong to a bounded rational vocabulary. Let $c^\star = p^\star/q^\star$ be reduced with $q^\star \le Q$. For two distinct reduced rationals $p/q$ and $p'/q'$ with denominators at most $Q$, the determinant identity gives

$$\left|\frac{p}{q} - \frac{p'}{q'}\right| = \frac{|pq' - p'q|}{qq'} \ge \frac{1}{Q^2}. \tag{45}$$

Consequently, if an estimate satisfies

$$|\hat{c} - c^\star| < \frac{1}{2Q^2}, \tag{46}$$

then $c^\star$ is the unique rational with denominator at most $Q$ in that symmetric neighborhood. This is a conditional identifiability statement: it requires a correct structural support, sufficient coefficient accuracy, and a rational constant with bounded denominator. It does not claim that every physical constant is rational or that a rational candidate outside the search vocabulary can be recovered.

### F.2 Conditions supporting structural recovery

A correct candidate can be retained when the following conditions jointly hold:

1. the latent law belongs to, or is sufficiently well approximated by, the bounded candidate grammar;
2. fresh query points provide a design matrix with adequate rank and domain coverage;
3. the surrogate approximation error is smaller than the separation between the correct structure and competing candidates on the query domain;
4. true simple coefficients lie within the configured projection tolerance or confidence interval;
5. rational denominators remain bounded away from zero on the validation domain;
6. the original noisy sample provides enough observations for cross-fitted coefficient estimates.

If these conditions fail, a low-complexity candidate can still be selected, but exact recovery is not guaranteed. The held-out-formula experiments are intended to measure this boundary rather than to claim universal symbolic identification.

### F.3 Numerical agreement does not imply symbolic equivalence

For a finite or bounded validation distribution, small function error does not imply equality of symbolic expressions:

$$\|s - f^\star\|_{L^2(\mathcal{X})} \text{ small} \quad \not\Rightarrow \quad s \equiv f^\star. \tag{47}$$

Different expressions can be close on a finite domain, omit weakly sensitive variables, or use floating-point constants that approximate simple exact values. They may also extrapolate differently outside the sampled region. Exactification addresses coefficient and support discreteness, Jacobian validation tests local sensitivity structure, denominator checks enforce numerical validity, and symbolic equivalence is evaluated separately from NMSE. This separation is the reason the paper reports both numerical metrics and exact-recovery rates.

## G Complete Baseline Comparison and Protocols

The main text reports aggregate baseline results; this appendix documents the evaluation contract used to make those comparisons auditable. All baseline methods receive the same generated dataset for a given benchmark, seed, and noise level. The dataset contains the input columns $x_0, \ldots, x_{d-1}$ and the noisy target $y$ in `train.csv`; the shared validation file contains the target values used for numerical and symbolic evaluation. Baselines are not given the ground-truth expression, the formula family, the development/unseen label, or any candidate-specific branch. The development formulas are available only for method development, while the frozen formulas are evaluated without benchmark-specific grammar additions or post hoc hyperparameter tuning.

### G.1 Baseline protocol and fairness

For every method, the following aspects are held fixed at the dataset level:

1. the benchmark identity, input domain, observations, and injected Gaussian-noise realization;
2. the input columns and noisy target supplied to the symbolic-regression method;
3. the random seed recorded in the run metadata;
4. the validation points and ground-truth evaluator used after formula generation;
5. the parsing, numerical-evaluation, symbolic-normalization, and exact-recovery rules;
6. the treatment of missing, timed-out, invalid, or unparsable outputs as unsuccessful trials.

The comparison is fair with respect to data, noise realization, evaluation data, and recovery criterion. It is not a claim that all methods have identical internal search algorithms, operator semantics, or computational implementations. PySR is invoked through its Python interface, whereas DSR, uDSR, Operon, and TPSR are invoked through method-specific external adapters in the baseline archive. Their versions and native search parameters must therefore be read from per-run configuration metadata; they are not inferred from recovered expressions.

### G.2 PySR

PySR is run through `PySRRegressor`. The repository adapter configures

$$\texttt{niterations} = 1000, \quad \texttt{populations} = 8, \quad \texttt{population_size} = 1000,$$
$$\texttt{maxsize} = 40, \quad \texttt{maxdepth} = 10, \quad \texttt{timeout_in_seconds} = 600,$$
$$\texttt{parsimony} = 0.0032, \quad \texttt{procs} = 8,$$

with multithreading enabled when more than one thread is configured. Its binary operator library is $\{+, -, \times, /, \}$ and its unary operator library is $\{\sin, \cos, \exp, \log, \sqrt{}\}$. The adapter uses the supplied seed as `random_state`, fits on the raw training arrays, and extracts the selected expression through `model.sympy(0)`. The installed PySR and Julia versions should be reported from the run environment or summary metadata.

Table 1: Baseline configuration contract. Values shown as "runner default" are the defaults implemented by the repository adapter. External-method fields are recorded from method-specific run metadata when available and are not silently imputed.

| Field | Protocol |
|---|---|
| Software/version | PySR version is exported by the PySR model when available; external baseline versions are recorded by their run environment or archive metadata. The repository does not pin a common version for external commands. |
| Operator library | PySR: $+, -, \times, /$, and $\sin, \cos, \exp, \log, \sqrt{}$. External methods use their native configured operator libraries; no operator is added during evaluation. |
| Population/search budget | PySR runner defaults: 8 populations, population size 1000, 1000 iterations, maximum expression size 40, maximum depth 10, and 600-second time limit. External-method budgets are method-specific and reported from their run metadata. |
| Random seed | Passed through the dataset/run configuration and recorded in each summary. The proposed method and unseen-formula evaluation use seeds 1–5; baseline runs use the seed attached to each archived trial. |
| Input normalization | The baseline runner passes raw $x_i$ values from `train.csv`; no hidden standardization is performed by the shared adapter. Any native normalization performed internally by an external baseline must be reported by that baseline configuration. |
| Noise protocol | Each method receives the same noisy `train.csv` generated for the corresponding formula, noise level, and seed. Noise is not regenerated inside the baseline evaluator. |
| Stopping criteria | PySR stops at its configured iteration, timeout, or native convergence condition. External methods stop according to their native iteration, timeout, or convergence configuration, which is retained with the archived run. |
| Output parsing | The recovered expression is read from the method output, whitespace is stripped, and it is parsed with SymPy using symbols $x_0, \ldots, x_{d-1}$. Parse failures and missing output files are invalid trials. |
| Exact recovery | The parsed expression is compared with the ground truth using the shared structural-metrics evaluator after constant normalization and symbolic simplification. Exact recovery requires `exact_recovery=True`; low NMSE alone is insufficient. |

### G.3 DSR AND UDSR

The repository isolates DSR and uDSR behind external-command adapters because their published environments require separate dependency stacks. Each command receives

```
--input <dataset>/train.csv --output <run>/formula.txt
```

and must write one recovered expression to `formula.txt`. The adapter does not modify the method's native operator library, population mechanism, iteration count, normalization, or stopping rule. These fields are therefore taken from the command wrapper and archived run configuration. The shared protocol fixes the dataset, seed, noise realization, output path, parser, validation data, and exact-equivalence evaluator. A failed command, absent output, or unparsable expression is retained as a failed trial rather than removed from the denominator.

For reproducibility, the supplementary release should include for each DSR and uDSR run: repository commit or package version, Python/TensorFlow/PyTorch version, operator set, population or batch size, number of iterations, timeout, native input normalization, stopping condition, and seed. The current baseline runner intentionally does not fabricate these values because these methods are executed through external commands rather than pinned in-repository implementations.

### G.4 OPERON AND TPSR

Operon and TPSR are evaluated under the same dataset-level contract. Their native search engines are not replaced by the PySR settings: each method retains its own population or sequence-generation mechanism, operator library, regularization, and stopping rule. The archive must record

the exact version or commit, operator library, population or batch configuration, iteration or epoch budget, timeout, random seed, and any input normalization performed by the native implementation. Equating all baselines to a common iteration count would not produce comparable computational effort across different search paradigms.

The shared post-processing is identical for all methods. The output expression is read from the method-specific result file, parsed with SymPy, evaluated on the common validation set, and passed to the same structural evaluator. No expression is manually simplified using knowledge of the target formula before evaluation. If a method returns multiple candidates, the archive records the candidate-selection rule used by that method; if it returns one expression, that expression is evaluated directly. Missing, invalid, or unparsable outputs count as failures.

### G.5 Shared parsing and exact-recovery evaluator

For an output expression $\hat{s}$, the evaluator constructs symbols $x_0, \ldots, x_{d-1}$ and parses it with the same local symbol map used for the ground truth. It computes numerical predictions on the shared validation points and the normalized error

$$\mathrm{NMSE}(\hat{s}) = \frac{m^{-1} \sum_{i=1}^{m} (\hat{s}(\boldsymbol{x}_i) - f^\star(\boldsymbol{x}_i))^2}{\mathrm{Var}(f^\star(\boldsymbol{x}_{1:m})) \vee 10^{-12}}. \tag{48}$$

For structural evaluation, constants are normalized within the configured tolerance and the difference between recovered and ground-truth expressions is simplified using algebraic, trigonometric, and rational simplification. Exact recovery is true only when the resulting symbolic difference is zero or SymPy certifies equality. Thus,

$$\mathrm{NMSE}(\hat{s}) \approx 0 \quad \not\Rightarrow \quad \mathrm{exact_recovery}(\hat{s}) = 1. \tag{49}$$

This common evaluator prevents a baseline from receiving credit for a numerically close but structurally different expression.

### G.6 Coverage and interpretation

Baseline archives may contain different numbers of completed trials because external methods have different installation requirements, runtime behavior, and output formats. We report, for every method, completed/requested, parsed/requested, and exact/requested counts. Exact-recovery rates use the denominator specified in each main-text table or figure caption; missing outputs are never silently imputed as successful recoveries, and failed parses are not removed from the recovery denominator. Aggregate comparisons should therefore be interpreted together with trial coverage, seed coverage, computational budget, and version metadata. The comparison supports the claim that the proposed structure-constrained pipeline is more robust in the evaluated noisy-neural-surrogate setting, but it is not a substitute for a future fully matched multi-seed benchmark in which all five baseline environments are pinned and rerun under identical task coverage.

## H Additional Implementation Details

## I Qualitative Recovery Cases

This appendix illustrates how numerical agreement, exactification, cross-fitting, and structural validation interact. The numerical values below are taken from the v3.5 main experimental report. That report stores the surrogate error, final recovered expression, and aggregate diagnostics, but it does not serialize every intermediate candidate generated during a search. We therefore distinguish report-backed quantities from schematic intermediate expressions. An expression labeled "schematic" explains a mechanism and is not presented as a logged candidate from a particular trial.

### I.1 Case 1: Low error but non-exact coefficient

A minimal exactification example is

$$0.4998 \cos(x) \;\longrightarrow\; \frac{1}{2} \cos(x).$$

Table 2: Report-backed exactification example. Intermediate continuous and noisy-refit expressions are not serialized in the main report; entries marked "schematic" describe the corresponding pipeline stage without fabricating a logged candidate.

| Stage | Expression or diagnostic |
|---|---|
| Ground truth | $\sin(x_0) + 0.5\cos(x_1)$ |
| Neural surrogate | NMSE $= 6.60232 \times 10^{-4}$ |
| Initial continuous candidate | Schematic: $\sin(x_0) + 0.4998\cos(x_1)$ |
| Exactified candidate | Schematic: $\sin(x_0) + \cos(x_1)/2$ |
| Noisy-refit candidate | The report does not serialize this intermediate expression; the fixed exactified structure is refitted on noisy data. |
| Final selected formula | $\sin(x_0) + \cos(x_1)/2$ |
| Recovered-to-truth NMSE | $1.51921 \times 10^{-15}$ |
| Symbolic equivalence | True; exact recovery: True |
| Expression complexity | Four operation nodes (sin, cos, division, addition) plus rational-constant description length. |

The first expression may have negligible validation error, but it does not encode the same discrete coefficient as the second expression. Rational projection is allowed only when the candidate rational lies within the coefficient confidence interval. This example is schematic: the decimal $0.4998$ is not claimed to be a stored output of a specific v3.5 trial.

The corresponding report-backed case is the noise-50 development trial for `representative_trigonometric`, seed 1, compatibility mode:

$$f^\star(x_0, x_1) = \sin(x_0) + 0.5\cos(x_1).$$

The surrogate NMSE is $6.60232 \times 10^{-4}$ and the final recovered NMSE is $1.51921 \times 10^{-15}$. The reported recovered expression is

$$\hat{f}(x_0, x_1) = \sin(x_0) + \frac{\cos(x_1)}{2},$$

and the report marks numerical equivalence, symbolic equivalence, and exact recovery as true. The decimal coefficient in the ground truth is normalized to $1/2$ before comparison.

This case shows that exactification is not cosmetic rounding. It converts a continuous coefficient estimate into a discrete, testable scientific hypothesis while preserving the fixed structure during subsequent noisy refitting.

### I.2 Case 2: Complex pseudo-formula versus a concise exact candidate

A flexible search may produce a high-degree or nested expression with low surrogate loss. A representative pseudo-candidate is

$$s_{\text{complex}}(x_0) = \frac{a_0 + a_1 x_0 + a_2 x_0^2 + a_3 \exp(b_1 x_0 + b_2 x_0^2)}{1 + c_1 x_0 + c_2 x_0^2 + c_3 \exp(b_3 x_0)},$$

where the coefficients are unrelated floating-point estimates. Such a candidate can be rejected when its cross-fitted excess loss is worse, its Jacobian disagrees with the surrogate, its denominator approaches zero, its extra coefficients are unsupported by confidence intervals, or its description length is larger without a statistically meaningful loss reduction.

The report-backed concise case is the noise-50 development trial for `representative_exponential`, seed 1, compatibility mode:

$$f^\star(x_0) = 1.5\exp(-0.8x_0) + 0.2, \qquad \hat{f}(x_0) = \frac{2 + 15\exp(-4x_0/5)}{10}.$$

The surrogate NMSE is $1.43503 \times 10^{-4}$ and the recovered-to-truth NMSE is $9.58617 \times 10^{-15}$. Numerical equivalence, symbolic equivalence, and exact recovery are all true. The selected expression

Table 3: Report-backed concise-formula case and the criteria used to reject structurally unsupported alternatives.

| Stage | Expression or diagnostic |
|---|---|
| Ground truth | $1.5\exp(-0.8x_0) + 0.2$ |
| Neural surrogate | NMSE $= 1.43503 \times 10^{-4}$ |
| Complex candidate | High-order/nested diagnostic form shown above; rejected-candidate details are not serialized. |
| Exactified concise candidate | $(2 + 15\exp(-4x_0/5))/10$ |
| Noisy-refit candidate | Fixed exactified structure refitted on noisy observations; intermediate coefficients are not serialized. |
| Final selected formula | $(2 + 15\exp(-4x_0/5))/10$ |
| Recovered-to-truth NMSE | $9.58617 \times 10^{-15}$ |
| Symbolic equivalence | True; exact recovery: True |
| Expression complexity | Four core operation nodes plus rational-constant description length. |

has one exponential, one affine numerator, one division, and rational constants; its core operation count is four, with rational-constant description length added separately.

The selection sequence is therefore

complex continuous candidate $\longrightarrow$ cross-fitted refit and structural validation $\longrightarrow$ rationalized concise candidate $\longrightarrow$ final sele

The individual rejected candidates are not serialized in the main report, so the complex expression above is a diagnostic illustration rather than a claim about one specific rejected row. The report does contain the complementary failure mode: for noise-50 `frozen_trig_02`, the recovered NMSE is $1.26792 \times 10^{-3}$ but symbolic equivalence is false. This demonstrates why low numerical error cannot replace exact symbolic evaluation.

### I.3 Interpretation and reproducibility

These cases separate the roles of the pipeline. Exactification addresses the discrete nature of scientific constants. Cross-fitting and structural validation address the statistical risk that a flexible expression explains one surrogate or one noise realization. Because the current report stores final trial summaries rather than the full candidate beam, a future release should additionally serialize the top-$k$ candidates, pre- and post-exactification coefficients, fold losses, confidence intervals, denominator margins, and selection scores. The aggregate results reported in the main paper are unchanged by this qualitative presentation.

## J Failure Cases and Limitations

This appendix reports failures rather than only successful recoveries.

### J.1 Representative unrecovered unseen formulas

At noise level 50%, the following 20 frozen formulas are representative unrecovered cases in the compatibility run. "Unrecovered" means that the reported exact-recovery indicator is false; it does not necessarily mean that the numerical approximation is useless.

The table is intentionally diagnostic rather than a claim that one mechanism is proven to be the sole cause of each failure. For example, the same formula can fail because a rich candidate family is present but loses during selection, because the neural surrogate has already erased a weak structural signal, or because the finite noisy domain makes two structures nearly indistinguishable.

### J.2 Observed failure modes

The v3.5 report shows several recurring patterns.

Table 4: Representative unrecovered frozen formulas at 50% noise. Categories are diagnostic interpretations based on the ground-truth family and the reported recovered expression; the report does not log a causal error label for each trial.

| Formula | Ground truth | Primary failure interpretation |
|---|---|---|
| `frozen_exp_01` | $\exp(0.6x_0) - 1$ | scale/structure mismatch |
| `frozen_exp_02` | $\exp(-0.4x_0^2) + 0.3x_1$ | mixed exponential–polynomial structure |
| `frozen_exp_04` | $\exp(0.3x_0 + 0.2x_1)$ | multivariate exponential composition |
| `frozen_mix_01` | $\exp(-0.2x_0^2)\cos(1.4x_0)$ | product/composition separation |
| `frozen_mix_03` | $\log(x_0^2 + 1) + \sin(x_1)$ | logarithmic structure confused with trigonometric terms |
| `frozen_mix_04` | $\sqrt{x_0^2 + 1} + \cos(x_1 + x_2)$ | nested multivariate structure |
| `frozen_mix_05` | $\log(x_0 + 1) + 0.5x_1^2$ | finite-domain polynomial approximation |
| `frozen_mix_06` | $\sqrt{x_0 x_1} + \sin(x_0)$ | radical/product identifiability |
| `frozen_mix_07` | $\sin(x_0)/(1 + 0.2x_0^2)$ | rational denominator instability |
| `frozen_mix_08` | $\exp(-x_0^2)\sin(x_1 + x_2)$ | multivariate product and phase |
| `frozen_mix_09` | $\log(x_0^2 + 1)/(1 + x_1^2)$ | ratio of nonlinear atoms |
| `frozen_mix_10` | $\sqrt{x_0^2 + 1}\exp(-0.2x_0)$ | radical–exponential composition |
| `frozen_nested_01` | $\sin(\exp(0.3x_0))$ | nested composition outside effective grammar |
| `frozen_nested_02` | $\exp(\sin(x_0))$ | nested composition outside effective grammar |
| `frozen_nested_03` | $\log(1 + \cos^2 x_0)$ | nested nonlinear atom |
| `frozen_nested_04` | $\sin(x_0^2 + x_1)$ | phase structure not stably selected |
| `frozen_poly_04` | $0.25x_0^4 - 0.4x_1 + 0.9$ | degree-four polynomial beyond preferred degree |
| `frozen_trig_01` | $\sin(1.3x_0) + 0.2$ | frequency ambiguity on finite domain |
| `frozen_rat_02` | $(x_0^2 + 0.5)/(1 + x_1^2)$ | rational denominator/scale instability |
| `frozen_trig_04` | $\sin(x_0 + x_1)$ | multivariate phase versus polynomial surrogate |

**Search grammar mismatch.** Nested cases such as `frozen_nested_01` and `frozen_nested_02` are represented by expressions such as $\sin(\exp(0.3x_0))$ and $\exp(\sin(x_0))$. The bounded template grammar contains selected compositions but does not contain arbitrary recursion. A rational, polynomial, or shallow trigonometric approximation can therefore achieve moderate value error while failing exact symbolic recovery. This is a deliberate limitation of bounded search, not evidence that the unrestricted expression is identifiable.

**Surrogate information loss.** When the neural surrogate error is non-negligible, symbolic recovery cannot recover information absent from the surrogate queries. The surrogate NMSE is a diagnostic upper-level failure signal: if $g_\phi$ has smoothed a high-frequency phase, weak interaction, or sharp denominator feature, an exact symbolic search over $g_\phi$ can only recover a law compatible with the smoothed function. This is distinct from a symbolic optimizer overfitting the original noisy observations.

**Noise-dominated selection.** At 50% noise, several frozen formulas have recovered NMSE in the $10^{-3}$–$10^{-2}$ range while their exact-recovery indicators remain false. For example, the report gives `frozen_nested_04` a recovered NMSE of approximately $1.79733$ at noise 50, whereas `frozen_mix_09` has recovered NMSE approximately $9.93839 \times 10^{-3}$ and remains symbolically incorrect. Noise can change the relative ranking of structures even when the candidate grammar contains an adequate approximation.

**Candidate-selection error.** The final report stores only the selected candidate, not the complete candidate beam, fold losses, and selection scores. Thus, a case in which the correct candidate was generated but later discarded cannot be certified from the current report alone. Such a failure is operationally possible when the correct structure has slightly worse surrogate loss, a less stable denominator, or a larger estimated variance than a competing approximation. We do not label any particular row as a confirmed "correct candidate eliminated" case without the missing beam-level log. This limitation motivates storing top-$k$ candidates and per-stage scores in future runs.

**Denominator instability.** Rational candidates can fit a bounded sample while having a denominator close to zero elsewhere in the validation or adversarial domain. The denominator margin $\min_{x \in \mathcal{V}} |q(x)| \geq \delta_q$ and the finite-value checks remove many such candidates, but the condition cannot guarantee global stability outside the tested domain. A formula may consequently be numerically adequate on the sampled points and unsafe under extrapolation.

**Finite-domain non-identifiability.** Two different expressions can be almost indistinguishable on the finite query domain. For example, a low-order polynomial or rational approximation can mimic an exponential, logarithm, or square-root function over a restricted interval. In such cases, cross-fitting can reject noise-sensitive candidates but cannot manufacture information outside the observed domain. The correct interpretation is observational equivalence on the tested domain, not global formula identification.

### J.3 Exactification can increase numerical error

Exactification is a discrete projection and therefore need not monotonically decrease numerical error. Let $s_{\text{cont}}$ be a continuous candidate and $s_{\text{exact}}$ its rationalized version. In general,

$$\text{NMSE}(s_{\text{exact}}) \not\leq \text{NMSE}(s_{\text{cont}}). \tag{50}$$

The projection can increase value error when the coefficient is only approximately rational, when several coefficients are jointly coupled in a rational function, or when a small coefficient is incorrectly retained or pruned under finite-sample uncertainty. The noisy-refit stage mitigates this effect by refitting the allowed global scale or the fixed supported structure, but it does not guarantee recovery of the original continuous optimum.

A report-backed warning example is the noise-20 frozen trial for `frozen_poly_02`. Its reported recovered formula is $(72x_0^2 + 27x_0x_1 - 54x_1^2 + 100)/90$, with recovered NMSE $7.80095 \times 10^{-5}$ and symbolic equivalence false, whereas the same family is exactly recovered at lower noise. This is consistent with a rational projection or support decision that is numerically close but not the correct discrete hypothesis. Because the report does not contain pre-exactification NMSE, the exact amount of error introduced by the projection cannot be isolated; we therefore do not claim a measured monotonicity violation for that row.

### J.4 Correct candidates rejected by selection

A correct candidate can be rejected by the final selector even when it is present in the search pool if it loses on the finite-sample selection criteria. The possible mechanisms are:

1. the candidate has a slightly larger cross-fitted loss because noisy coefficient refits are unstable;
2. its Jacobian estimate is noisier than that of a simpler approximation;
3. its rational denominator is close to the stability margin;
4. its exact constants increase the estimated confidence-band penalty;

5. the candidate is outside the paired confidence set because the reported fold variance is large.

The current report cannot identify a confirmed instance because it does not save the full candidate pool. This is an important reproducibility limitation: a final false-negative exact-recovery label does not distinguish "true structure was never generated" from "true structure was generated and rejected." Future runs should save the candidate rank, selection score, cross-fitted loss, Jacobian loss, denominator margin, complexity, and rejection reason for every candidate.

### J.5 Applicability conditions and scope

The method is intended for the following regime:

1. The true formula is equal or sufficiently close to a member of the predefined bounded structure grammar.
2. Input variables and the domain provide sufficient coverage for separating the relevant operators, interactions, and phases.
3. Formula complexity is bounded; the method does not search arbitrary degree, arbitrary nesting depth, or an unbounded operator vocabulary.
4. The neural black-box surrogate preserves the structural information needed by the symbolic stage, including relevant variables and local sensitivities.
5. Rational denominators remain away from zero on the intended validation and deployment domain.
6. The noisy sample size is sufficient for fixed-structure refitting and confidence-aware selection.

The method does not guarantee recovery of an arbitrary formula outside the search space. In particular, a failure can be caused by grammar mismatch even when the neural surrogate is accurate and the noise level is low. Conversely, a formula inside the grammar may remain unrecovered when the surrogate error, noise, finite-domain ambiguity, or candidate-selection variance is too large. These limitations are why the paper reports exact recovery separately from value NMSE and evaluates frozen formulas without adding benchmark-specific templates after seeing their results.