RETHINKING FITTING ERROR IN SYMBOLIC REGRESSION THROUGH LOCAL TAYLOR CONSISTENCY: THEORY, EMPIRICAL INVESTIGATION, AND A NEW EVALUATION FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression aims to discover underlying mathematical expressions from data, and most existing methods rely on fitting error to evaluate candidate expressions and guide the search. However, we observe that symbolic regression algorithms often produce structurally diverse but incorrect expressions with low fitting errors. This observation motivates us to investigate whether structurally distinct low-error candidates are prevalent and how they affect fitting-error-driven search. Theoretically, we derive an upper bound on fitting error and show that structurally different candidate expressions can achieve low fitting errors over small sampling domains when they exhibit local Taylor consistency. Furthermore, infinitely many such candidates can be constructed by introducing different higher-order remainder terms, while free constants further increase the attainability of local Taylor consistency. These results suggest that low-error but structurally incorrect candidates may be widespread over small sampling domains. Empirically, we first introduce RDF-SR, a random non-repetitive symbolic regression algorithm, as an unguided-search baseline. For relatively low-complexity expressions, several fitting-error-driven methods achieve lower recovery rates than RDF-SR, suggesting that optimization landscapes induced by fitting error may hinder structural recovery. Our theoretical error bound further indicates that enlarging the sampling domain can reduce low-error spurious solutions. Motivated by this result, we conduct experiments across different sampling domains and show that larger domains improve recovery rates but may weaken fitting-error guidance, making low-error regions harder to reach. Based on these findings, we develop and apply a new evaluation framework comprising new datasets, evaluation metrics, and an unguided-search baseline to systematically assess the search effectiveness, fitting-error guidance, and structural recovery capabilities of symbolic regression algorithms.

1 INTRODUCTION

Symbolic regression (SR) aims to discover mathematical expressions that fit observed data. Given a noise-free dataset $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$, where x_i denotes the input and y_i the corresponding target value, the goal is to recover an expression f such that $y_i = f(x_i)$ for all $i \in 1, \ldots, N$. This problem is NP-hard (Virgolin & Pissis, 2022).

SR can essentially be viewed as a binary decision problem: whether a candidate expression matches the target expression. Relying solely on this criterion, an algorithm cannot distinguish among candidate solutions of different quality or obtain informative search guidance, leaving it with only unguided exploration. Existing SR methods therefore typically evaluate candidate expressions using fitting error and use it to guide the search. Ideally, under fitting-error guidance, the algorithm should progressively converge to the target expression with zero fitting error.

Although this strategy may provide effective search guidance, Table 1 shows that a symbolic regression algorithm often generates structurally different expressions with low fitting errors across

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.

independent runs. This motivates Question 1: Are such low-error candidates with different structures abundant?

To investigate this question, we theoretically analyze both the existence and abundance of such candidates. First, by deriving an upper bound on fitting error, we show that structurally distinct expressions can attain low fitting errors over a finite sampling domain when they exhibit local Taylor consistency. We then analyze the abundance of locally Taylor-consistent candidates. Without free constants, infinitely many such structurally distinct candidates can be constructed by varying higher-order remainder terms. With free constants, candidate structures whose jet-map Jacobians have full row rank can achieve local Taylor consistency in a neighborhood of the target jet through constant adjustment. Under linear weight parameterization, an invertible jet-map matrix extends this attainability to the entire jet space.

In summary, when the sampling domain is small, there may exist a large number of candidate expressions that are structurally different from the target expression but achieve low fitting errors. This leads to Question 2: does transforming the original discrete decision problem into a fitting-error-driven optimization problem facilitate the search, or does it instead make the search more difficult by introducing numerous spurious low-error solutions?

To answer this question, we introduce Random Duplicate-Free Symbolic Regression (RDF-SR), which explores the expression space randomly and without repetition in the absence of search guidance. Comparisons with existing error-driven methods show that, for low-complexity target expressions, RDF-SR achieves higher recovery rates, suggesting that fitting-error guidance may be misled by spurious low-error solutions, thereby making recovery more difficult.

In addition, the fitting-error upper bound suggests that enlarging the sampling domain can reduce the number of spurious low-error solutions. This leads to Question 3: can a larger sampling domain improve recovery rates? Experimental results show that appropriately enlarging the sampling domain may improve recovery rates, but it can also weaken the search guidance provided by fitting error, making low-error regions harder to reach and thereby limiting improvements in recovery.

Based on these results, the difficulties faced by SR vary across sampling domains: under smaller domains, the algorithm needs to identify the correct structure among numerous low-error candidates, whereas under larger domains, it needs to establish a stable search trajectory to reach low-error regions. An effective SR algorithm should address both challenges and outperform RDF-SR under the same search space and search budget.

However, existing evaluation frameworks for SR remain limited: datasets typically use a single sampling domain and lack sets of numerically similar but structurally distinct expressions (Uy et al., 2011; Jin et al., 2019; Trujillo et al., 2016; Keijzer, 2003; Krawiec & Wieloch, 2010); evaluation metrics primarily focus on low-error rate (Biggio et al., 2021; Kamienny et al., 2022) or recovery rate (Bendinelli et al., 2023; Ruan et al., 2026; La Cava et al., 2021); and existing comparisons lack a reference baseline for assessing fundamental search capability.

To this end, we propose and apply a new evaluation framework for SR algorithms, encompassing dataset construction, evaluation metric design, and baseline selection. The dataset is evaluated under both small and large sampling domains and contains 68 expressions divided into 17 groups, with each group comprising expressions with similar fitting errors but structurally distinct forms. In addition to low-error rate and recovery rate, we introduce conditional recovery rate and the within-group standard deviation of recovery rates to measure, respectively, an algorithm's ability to escape spurious low-error solutions and recover the target expression, and the structural balance of its search across different expressions with similar numerical behavior. Finally, RDF-SR and state-of-the-art SR methods serve as baselines in comparative experiments.

The main contributions of this paper are summarized as follows:

1. We theoretically show that structurally distinct expressions can achieve low fitting errors over finite sampling domains through local Taylor consistency, and that infinitely many such candidates can exist, suggesting that small domains may contain numerous spurious low-error solutions.

2. We introduce RDF-SR, and show that fitting-error guidance may make recovery more difficult.

3. We show theoretically and empirically that enlarging the sampling domain can reduce spurious low-error solutions and may improve recovery rates, but may also make low-error regions harder to reach.

4. We propose and apply a new evaluation framework for symbolic regression.

2 Related Work

In recent years, some methods (Taskin et al., 2026; Merler et al., 2024; Shojaee et al., 2025; Saveliev et al., 2026; Grayeli et al., 2024) have leveraged mathematical and scientific knowledge encoded in large language models (LLMs) to guide expression generation, focusing the search on regions covered by existing knowledge. However, prior-free expression-space search remains important for discovering unknown relationships, since the operators and structures of the target expression are generally unknown and may lie outside the training distribution. If an LLM assigns low probability to necessary structures, the search may miss the target expression. Such prior-free SR methods are primarily based on genetic programming (GP), reinforcement learning (RL), and Monte Carlo tree search (MCTS).

GP randomly initializes a population of expressions, evaluates individuals using fitting error as the fitness criterion, and generates successive populations through operators such as crossover and mutation (Koza, 1994; Cranmer, 2023). Subsequent research has mainly focused on improving evolutionary mechanisms (Pawlak et al., 2014; Virgolin et al., 2021), enhancing population diversity, and designing fitness functions (Sambo et al., 2021; Haider et al., 2023). For example, to promote diversity, Chen et al. (2020) introduced a novelty measure based on transformed semantics to alleviate premature convergence, while Bonin et al. (2024) employed a cellular population structure with local-neighborhood interactions to slow population convergence.

RL formulates expression generation as a sequential decision-making process and defines rewards based on fitting error to update the policy (Petersen et al., 2019). To improve global exploration of the expression space, subsequent studies combine RL with GP (Mundhenk et al., 2021; Zhang & Zhou, 2021) or incorporate human feedback through interactive platforms (Crochepierre et al., 2022; Kim et al., 2021), thereby mitigating the limitations of policy-based sampling.

MCTS represents intermediate expression-construction states as nodes in a search tree. Once a complete expression is obtained through node expansion, a reward is assigned based on its fitting error and propagated back through the tree to update node values, thereby guiding subsequent branch expansion(Sun et al., 2022). To improve search efficiency, later methods use neural networks to guide candidate expansion or mutation and incorporate mechanisms such as value learning and subtree reuse to improve branch selection and exploration of expression structures(Li et al., 2025; Shojaee et al., 2023).

Overall, fitting error remains the primary signal for evaluating candidate expressions and guiding search in SR, warranting further investigation of its reliability.

3 Question 1—Existence and Abundance of Structurally Distinct Low-Error Expressions

This section aims to establish the following three theoretical results:

- If a candidate expression g exhibits local Taylor consistency with the target expression f, it can achieve a low fitting error over a finite sampling domain even when the two are structurally distinct.
- Expressions satisfying local Taylor consistency can form an infinite family. Even without free constants, infinitely many such candidates can be constructed by varying the higher-order remainder terms.
- Introducing free constants increases the degrees of freedom of candidate expressions, making it easier to match the low-order Taylor terms of the target expression and thereby increasing the attainability of low-error candidates.

For simplicity of notation, we conduct the analysis on the sampling interval $[-c, c]$ with expansion point $x = 0$. This setting does not affect the generality of our results. For any finite interval $[a, b]$, let $x_0 = \frac{a+b}{2}$, $h = \frac{b-a}{2}$, and $z = x - x_0$; then $x \in [a, b]$ is equivalent to $z \in [-h, h]$. We consider expressions formed by finitely many applications of operators from $\mathcal{O} = \{+, -, \times, \div, \sin, \cos, \exp, \log\}$ to the variable x; when free constants are considered, constant placeholders are introduced for expression construction.. We use the notion of order of contact: if $g^{(k)}(0) = f^{(k)}(0)$ for $k = 0, 1, \ldots, p-1$, then f and g are said to have contact of order at least p at $x = 0$. Equivalently, defining $\Delta(x) = f(x) - g(x)$, we have $\Delta^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$. The complete proofs of all theoretical results in this paper are provided in Appendix A.

3.1 Fitting Error Analysis under Local Taylor Consistency

This subsection aims to characterize an upper bound on the fitting error between two structurally distinct expressions f and g, and to identify the conditions under which this bound becomes small, thereby characterizing when the two expressions can exhibit low fitting error.

Lemma 1 (Boundedness of Higher-Order Derivatives)**.** *Suppose that the candidate expression g and the target expression f are formed by finitely many applications of the given operators and are well-defined and smooth on the sampling interval $I = [-c, c]$. Then, for any fixed $p \in \mathbb{N}$, $\Delta^{(p)}$ is continuous on the compact interval I, and hence there exists a constant $L_p \geq 0$ such that $\left|\Delta^{(p)}(x)\right| \leq L_p$ for all $x \in I$.*

Theorem 1 (Fitting-Error Upper Bound Induced by the Order of Contact)**.** *Let $c > 0$ and $p \in \mathbb{N}^+$. Suppose that $f, g \in C^p([-c, c])$ have contact of order at least p at $x = 0$, i.e., $\Delta^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$. Then, the continuous mean squared error between f and g satisfies $\mathrm{MSE}(f, g) \leq \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1}$. Here, L_p is an upper bound for $|\Delta^{(p)}|$ on $[-c, c]$, as defined in Lemma 1.*

Corollary 1 (Error Convergence)**.** *Suppose $\Delta^{(p)}(x)$ is uniformly bounded independently of p. Then, for any fixed $c > 0$, $\mathrm{MSE}(f, g) \to 0$ as $p \to \infty$. Likewise, for fixed p, if f and g are smooth in a neighborhood of $x = 0$, then $\mathrm{MSE}(f, g) \to 0$ as $c \to 0$.*

3.2 Infinitude of Locally Taylor-Consistent Candidates without Constants

This subsection shows that even without free constants in the search space, infinitely many candidate expressions can remain locally Taylor-consistent with the target expression. Over a small sampling interval, these candidates may achieve low fitting errors, forming a large set of low-error candidates.

For a target expression $f \in C^p([-c, c])$, let $a_j = f^{(j)}(0)$ for $j = 0, 1, \ldots, p-1$. We define the set of candidate expressions having contact of order at least p with f at $x = 0$ as $G_p(f) := \left\{ g \in C^p([-c, c]) \;\middle|\; g^{(j)}(0) = a_j,\ j = 0, 1, \ldots, p-1 \right\}$.

Theorem 2 (Construction of Locally Taylor-Consistent Candidates)**.** *Let $c > 0$ and $p \in \mathbb{N}^+$. For every $g \in G_p(f)$, there exists a unique $q \in Q$ such that $g(x) = \sum_{j=0}^{p-1} \frac{a_j}{j!} x^j + x^p q(x), \quad x \in [-c, c]$, where $Q := \left\{ q \in C^0([-c, c]) \;\middle|\; x^p q(x) \in C^p([-c, c]) \right\}$. Equivalently, $G_p(f)$ and Q are in one-to-one correspondence.*

Theorem 3 (Infinitude of Locally Taylor-Consistent Candidates)**.** *For any $m \in \mathbb{N}^+$, let $q_m(x) = x^m$. Then $q_m \in C^0([-c, c])$ and $x^p q_m(x) = x^{p+m} \in C^p([-c, c])$, so $q_m \in Q$. Since the functions in $\{q_m\}_{m=1}^{\infty}$ are pairwise distinct, Q contains a countably infinite subset. By Theorem 2, $G_p(f)$ is in bijection with Q, and hence $|G_p(f)| = |Q|$. Therefore, $G_p(f)$ is infinite.*

For $f(x) = \sin(x)$, Table 2 presents candidate expressions corresponding to different higher-order remainder terms. All candidates share the same first three Taylor terms as $\sin(x)$ and achieve low fitting errors over a small sampling domain.

3.3 Constants enhance the attainability of local Taylor consistency

This subsection aims to show that, after introducing constant placeholders, adjusting the constants alone may allow a fixed candidate structure to match several low-order Taylor terms of the target expression, thereby making low-error candidates easier to attain. Let a fixed candidate structure be

Table 1: Candidate expressions obtained by DSO for $\exp(-x^2/2)$ across independent runs and their continuous MSEs.

Expression	**MSE** $[-1, 1]$
$[-\log(\cos x) + \cos x]\cos x$	5.80e−5
$\log(\cos^2 x) + 1/\cos x$	4.94e−5
$[\exp(\cos x - 1) - \cos x + 1]\cos x$	1.50e−5
$\exp[-\sin(x^2/2)]$	1.41e−5

Table 2: Locally Taylor-consistent candidates constructed with different higher-order remainder terms for $sin(x)$ and their continuous MSEs.

Higher-Order Remainder $q(x)$	**Candidate Expression** $g(x)$	**MSE** $[-1, 1]$	**MSE** $[-0.5, 0.5]$
$\frac{\frac{1}{2}\sin(2x) - x}{x^3}$	$\frac{1}{2}\sin(2x)$	2.41e−2	5.06e−4
$\frac{1 - \exp(-x) + \frac{x^2}{2} - x}{x^3}$	$1 - \exp(-x) + \frac{x^2}{2}$	1.61e−2	2.48e−4
$\frac{\cos x - 1}{2x^2}$	$\frac{x(1 + \cos x)}{2}$	7.81e−4	1.46e−5
$\frac{\log(1 + x^4)}{x^3}$	$x + \log(1 + x^4)$	6.91e−2	4.76e−4

parameterized as $g(x; \theta)$, where $\theta \in \mathbb{R}^P$. Appendix B provides constructive examples for Theorems 4 and 5 using $f(x) = \sin(x)$ with $p = 3$, based on free-constant adjustment and invertible linear weights.

Theorem 4 (Local Attainability of Taylor Consistency)**.** *Let $p, m \in \mathbb{N}^+$ and $c > 0$. Consider a fixed candidate structure parameterized by $\theta = (\theta_1, \ldots, \theta_m)^\top \in \mathbb{R}^m$, denoted by $g(x; \theta)$, and assume that $g(\cdot; \theta) \in C^p([-c, c])$ for every θ. Define its $(p-1)$-jet map at $x = 0$ by $J_{p-1}(\theta) := \big(g(0;\theta), \partial_x g(0;\theta), \ldots, \partial_x^{p-1} g(0;\theta)\big)^\top \in \mathbb{R}^p$, with Jacobian $DJ_{p-1}(\theta) := \Big[\frac{\partial}{\partial \theta_j}\big(\partial_x^{i-1} g(0;\theta)\big)\Big]_{\substack{i=1,\ldots,p \\ j=1,\ldots,m}} \in \mathbb{R}^{p\times m}$. Suppose that J_{p-1} is continuously differentiable in a neighborhood of θ_0 and that $\operatorname{rank} DJ_{p-1}(\theta_0) = p$. Then $m \geq p$, and there exist a neighborhood $\Theta_0 \subset \mathbb{R}^m$ of θ_0 and a neighborhood $U \subset \mathbb{R}^p$ of $F_0 := J_{p-1}(\theta_0)$ such that $U \subset J_{p-1}(\Theta_0)$. Therefore, for any $f \in C^p([-c, c])$ satisfying $\big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top \in U$, there exists $\theta^* \in \Theta_0$ such that $J_{p-1}(\theta^*) = \big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top$. Hence, $g(x; \theta^*)$ and $f(x)$ have contact of order at least p at $x = 0$.*

Theorem 5 (Global Attainability of Taylor Consistency)**.** *Let $p \in \mathbb{N}^+$ and $c > 0$. Consider a candidate expression family $g(x; \alpha, \gamma) \in C^p([-c, c])$, where $\alpha = (\alpha_1, \ldots, \alpha_p)^\top \in \mathbb{R}^p$ denotes the linear weight parameters and γ the remaining parameters. Suppose $g(x; \alpha, \gamma) = \sum_{j=1}^p \alpha_j \phi_j(x; \gamma)$, with $\phi_j(\cdot; \gamma) \in C^p([-c, c])$. Define the $(p-1)$-jet map at $x = 0$ as $J_{p-1}(\alpha, \gamma) = \big(g(0;\alpha,\gamma), \partial_x g(0;\alpha,\gamma), \ldots, \partial_x^{p-1} g(0;\alpha,\gamma)\big)^\top \in \mathbb{R}^p$. For fixed γ, its Jacobian with respect to α is $M(\gamma) := D_\alpha J_{p-1}(\alpha, \gamma) = \Big[\frac{\partial}{\partial \alpha_j}\big(\partial_x^{i-1} g(0;\alpha,\gamma)\big)\Big]_{\substack{i=1,\ldots,p \\ j=1,\ldots,p}} \in \mathbb{R}^{p\times p}$. Since g is linear in α, $J_{p-1}(\alpha, \gamma) = M(\gamma)\alpha$. If $\det M(\gamma) \neq 0$, then for any $F \in \mathbb{R}^p$, the unique choice $\alpha = M(\gamma)^{-1} F$ satisfies $J_{p-1}(\alpha, \gamma) = F$. Therefore, for fixed γ, the $(p-1)$-jet map is globally attainable over $\mathbb{R}^p$ if and only if $M(\gamma)$ is invertible.*

3.4 Conclusions for Question 1

When the sampling domain is small, structurally distinct expressions may achieve low fitting errors if they exhibit local Taylor consistency. Even without free constants, infinitely many such candidates with diverse structures can be constructed by varying the higher-order remainder terms. With free constants, certain candidate structures can achieve local Taylor consistency solely through constant adjustment. These results suggest that small sampling domains may contain many structurally distinct and diverse candidates with low fitting errors relative to the target expression.

4 QUESTION 2 — EFFECTIVENESS OF FITTING-ERROR GUIDANCE

4.1 EXPERIMENTAL OBJECTIVE

This experiment examines whether, with a sampling-domain length of 2, fitting error can effectively guide the search as an evaluation metric, or instead introduces numerous low-error but structurally incorrect solutions that act as spurious optima and make the search more difficult.

4.2 RANDOM DUPLICATE-FREE SYMBOLIC REGRESSION (RDF-SR)

RDF-SR performs random, duplicate-free search over the valid expression space defined by the operator and complexity constraints. Since its search process is independent of any performance metric, RDF-SR serves as a baseline for assessing whether heuristic criteria or new search strategies genuinely improve recovery performance. Details of RDF-SR are provided in Appendix C.

4.3 EXPERIMENTAL SETUP

We compare RDF-SR with three representative error-driven SR methods: GP-based PySR (Cranmer, 2023), RL-based DSO(Mundhenk et al., 2021), and MCTS-based RSRM (Xu et al., 2024). Experiments are conducted on the Livermore dataset(Mundhenk et al., 2021), using recovery rate and low-error rate as evaluation metrics. Recovery rate is defined as the proportion of output expressions that are equivalent to the target expression and have absolute constant errors below 10^{-5}; low-error rate is defined as the proportion of output expressions with MSE below 10^{-3}. The sampling domain is set to $(-1, 1)$, or $(0, 2)$ for nonnegative inputs. Expression complexity is measured by token count. The search budget is set to 2 million. Algorithm descriptions and parameter settings are provided in Appendix D.

4.4 EXPERIMENTAL RESULTS

Figures 1 and 2 show that, for expressions with complexity ≤ 12, RDF-SR achieves a recovery rate of 89.00%, outperforming all three fitting-error-driven methods; for complexity > 12, its recovery rate drops to 8.25%, below those of the three methods. All fitting-error-driven methods achieve low-error rates above 70%, indicating that they can consistently reach low-error regions but still struggle to exactly recover the target expression.

These results suggest that, for relatively low-complexity targets, fitting-error-based evaluation may hinder recovery by favoring spurious low-error solutions, whereas RDF-SR explores the search space more uniformly and avoids this bias. For target expressions with relatively high complexity, fitting-error guidance becomes beneficial. This may be because RDF-SR cannot sufficiently explore the larger expression space, whereas fitting error provides informative guidance that facilitates the search. Recovery results for each target expression and the corresponding expression forms are provided in Appendix E.

Table 3 summarizes the number of candidate expressions found by RDF-SR across different low-error ranges. For $1e-3 \leq \mathrm{MSE} < 1e-2$, the average candidate count is on the order of 10^4; even for $\mathrm{MSE} < 1e-5$, it remains on the order of 10^3. Per-target-expression candidate counts across low-error ranges are provided in Appendix F.

5 QUESTION 3 — EFFECTS OF SAMPLING DOMAIN ON RECOVERY AND FITTING-ERROR GUIDANCE

5.1 EXPERIMENTAL OBJECTIVE

Theorem 1 shows that enlarging the sampling domain increases the upper bound on fitting error, weakening the low-error advantage induced by local Taylor consistency and thereby reducing the number of structurally incorrect low-error candidates. However, this does not necessarily improve recovery rates, because fewer low-error candidates may leave the algorithm with fewer intermediate solutions for progressive improvement, making the search more difficult. This experiment therefore

Table 3: Statistics of candidate expression counts across low-error ranges in RDF-SR. Statistics are computed from the 50-run average for each target expression.

Statistic	**MSE interval**			
	$[10^{-3}, 10^{-2})$	$[10^{-4}, 10^{-3})$	$[10^{-5}, 10^{-4})$	$[0, 10^{-5})$
Mean	37007.28	7733.09	2342.86	1173.75
Median	22815.75	1948.41	282.34	49.93
Q_1	1747.40	74.25	4.10	1.62
Q_3	53559.75	8893.85	2462.34	1500.33

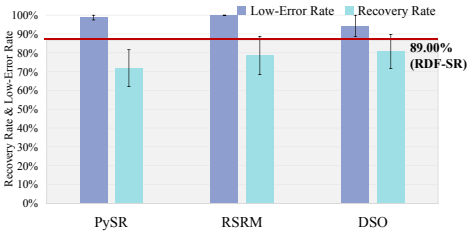


Figure 1: Low-error and recovery rates for expressions with complexity ≤ 12. The red line denotes the RDF-SR baseline. Error bars indicate standard errors.

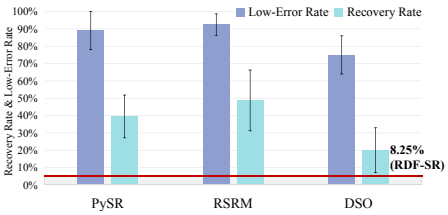


Figure 2: Low-error and recovery rates for expressions with complexity > 12. The red line denotes the RDF-SR baseline. Error bars indicate standard errors

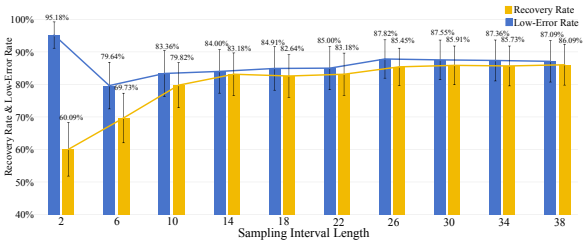


Figure 3: Recovery rate and low-error rate of PySR across different sampling interval lengths. Error bars indicate standard error.

examines two effects of enlarging the sampling domain: whether it improves recovery rates and whether fitting error provides weaker search guidance.

5.2 Experimental Setup

We evaluate the recovery and low-error rates of PySR under different sampling domains on the Livermore dataset. The sampling domain is set to $(-c, c)$ with $c \in \{1, 3, \ldots, 19\}$; for expressions restricted to nonnegative inputs, we use $(0, c)$ with $c \in \{2, 6, \ldots, 38\}$. The sampling spacing is fixed at $(x_{\max} - x_{\min})/N = 0.1$, where N is the number of samples, and therefore increases proportionally with the domain length. To further verify the generality of these findings, we conduct similar experiments with DSO and RSRM; details are provided in Appendix G.

5.3 Experimental Results

The yellow bars in Figure 3 show that PySR's recovery rate generally increases with the sampling-domain length before gradually stabilizing. As the domain length increases from 2 to 30, the recovery rate rises from 60.09% to 85.91%, suggesting that appropriately enlarging the sampling domain may reduce the influence of spurious low-error solutions and thereby improve recovery performance.

The blue bars in Figure 3 show that, with a sampling-domain length of 2, the low-error rate reaches 95.18%, indicating that the algorithm can readily find low-error candidates within a small domain. When the domain length increases to 6, the low-error rate drops to 79.64% and then remains largely stable, indicating that fitting error provides weaker search guidance as the sampling domain expands, making it more difficult for the algorithm to continue finding lower-error candidates. DSO and RSRM show similar trends; see Appendix G for details.

6 A NEW EVALUATION FRAMEWORK FOR SYMBOLIC REGRESSION

The above results show that SR faces different primary difficulties across sampling domains: under larger domains, the algorithm needs to establish a stable search trajectory to reach low-error regions, whereas under smaller domains, it needs to identify and recover the correct structure among numerous low-error candidates. We therefore use low-error rate (LER) to measure the ability to reach low-error regions and conditional recovery rate (C-RR), defined as the proportion of low-error outputs that recover the target expression, to measure structural identification ability. Recovery rate (RR) reflects the combined outcome of the two and should exceed that of RDF-SR under the same search space and search budget to indicate a practical advantage of the search strategy.

Meanwhile, for numerically similar but structurally distinct expressions, an algorithm should not consistently favor particular structures. We therefore use the within-group standard deviation of recovery rates (WSD) to quantify differences in recovery rates across structurally distinct expressions within each subset. Based on these considerations, we adopt the following evaluation procedure:

Step 1: Construct a dataset that satisfies the following conditions: (i) multiple subsets of expressions with similar fitting errors but distinct structures in a small sampling domain, to evaluate recovery difficulty and structural search balance; and (ii) sampling domains of different sizes, to evaluate performance when low-error candidates are abundant or sparse.

Step 2: Use RDF-SR and existing methods as baselines with the same search budget and space.

Step 3: Evaluate SR algorithms: Use RR to assess whether the search strategy provides a practical advantage; LER to evaluate the effectiveness of fitting-error guidance; C-RR to evaluate the algorithm's ability to recover the correct structure among low-error outputs; and WSD to measure the structural balance of the search across numerically similar expressions.

6.1 NEW DATASET

Using the univariate expressions in the Livermore dataset as benchmark expressions, we construct one subset for each benchmark, containing multiple expressions with similar fitting errors but structurally distinct forms. The resulting dataset consists of 17 subsets and 68 expressions in total, all of which are low-error solutions generated by RDF-SR. The sampling domains of each subset follow those of its benchmark expression: $(-1, 1)$ and $(-15, 15)$ when negative inputs are allowed, and $(0, 2)$ and $(0, 30)$ otherwise. The sampling spacing is fixed at 0.1. Expression forms, complexities, and fitting errors relative to their benchmark expressions are provided in Appendix H.

6.2 EXPERIMENTAL SETUP

We evaluate RDF-SR together with RL-based DSO and uDSR (Landajuela et al., 2022), MCTS-based RSRM, TPSR (Shojaee et al., 2023), and iMCTS (Huang et al., 2026), and GP-based PySR, EGGP (de Franca & Kronberger, 2025), and GPlearn[1]. For algorithms that support the corresponding parameter settings, we use the same search space and set the number of searches to 2 million. Detailed algorithm descriptions and parameter settings are provided in Appendix D.

6.3 EXPERIMENTAL RESULTS

Figures 4 and 5 compare recovery rates on the expression subsets with complexity no greater than 12 and greater than 12, respectively. For complexity no greater than 12, under the small sampling domain, all algorithms except uDSR perform below RDF-SR's 67.18%, suggesting that spurious low-error solutions may mislead the search. Under the large sampling domain, most algorithms achieve higher recovery rates, yet only iMCTS slightly outperforms RDF-SR, suggesting that weaker fitting-error guidance may limit performance. RL-based methods do not exhibit the same trend, highlighting that reducing spurious low-error solutions does not necessarily improve recovery. Instead, it may make low-error expressions harder to reach and hinder the formation of a stable optimization trajectory. Such methods may require a larger search space to increase expression diversity and provide

[1] https://gplearn.readthedocs.io/en/stable/

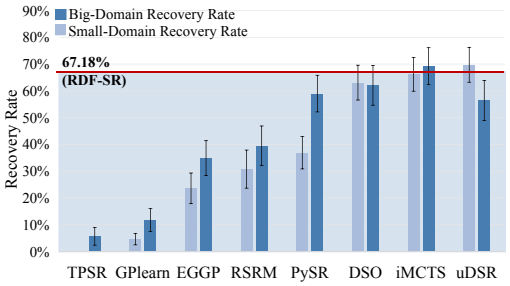


Figure 4: Recovery rates under small and large sampling domains for expressions with complexity ≤ 12.

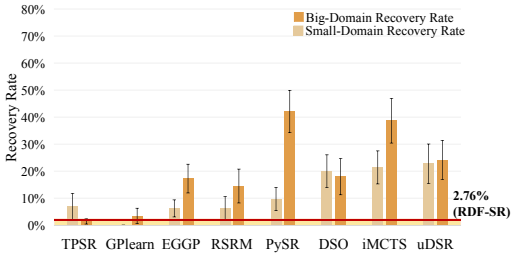


Figure 5: Recovery rates under small and large sampling domains for expressions with complexity > 12.

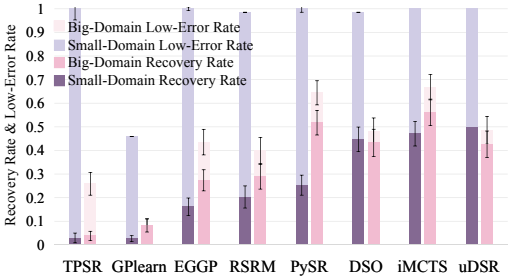


Figure 6: Low-error and recovery rates under different sampling domains on the full dataset, with C-RR given by the dark-to-light bar ratio.

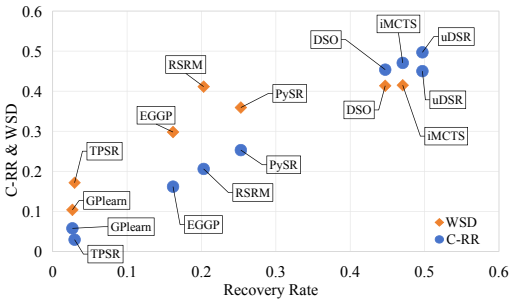


Figure 7: Relationships among RR, C-RR, and WSD across symbolic regression algorithms.

more diverse fitting-error signals. For complexity greater than 12, most methods outperform RDF-SR's 2.76%, indicating that as target complexity increases, RDF-SR becomes less able to reach the target structures, while fitting error still provides useful search guidance.

Figure 6 shows the LER, RR, and C-RR of each algorithm on the full dataset. Under the small sampling domain, all algorithms achieve high LERs but generally low C-RRs, indicating that low-error solutions are easy to find but the correct structure remains difficult to recover. uDSR and iMCTS achieve relatively high C-RRs of 49.71% and 47.06%, respectively. As the sampling domain expands, LER decreases while C-RR increases, suggesting fewer spurious low-error solutions and a higher likelihood that low-error candidates match the target structure.

Figure 7 illustrates the relationships among RR, C-RR, and WSD. uDSR and iMCTS achieve relatively high RR and C-RR, indicating that they are better able to recover the correct structure once they reach low-error regions. Moreover, most algorithms exhibit large WSD values, indicating substantial variation in recovery difficulty even among numerically similar target expressions and suggesting imbalanced exploration of the structural space. Detailed results are provided in Appendix I.

7 Conclusion

Our theoretical analysis shows that local Taylor consistency allows structurally distinct candidates to achieve low fitting errors within a finite sampling domain. Infinitely many such candidates can be constructed by varying higher-order remainder terms, while free constants further increase their attainability. Thus, small sampling domains may contain many spurious low-error solutions. Experiments show that fitting-error guidance may increase search difficulty under smaller domains. Enlarging the sampling domain may improve recovery rates but can also make low-error regions harder to reach, indicating different primary challenges across domains. Based on these findings, we propose an SR evaluation framework that jointly evaluates numerical search and structural identification across sampling domains. Future SR methods should dynamically adjust their search focus according to the data range or current search outcomes.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation and language polishing. We did not use generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. We have reviewed all AI-assisted work and verified that the translated and polished content is fully consistent with the information and meaning intended by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the source code of the proposed Random Duplicate-Free Symbolic Regression (RDF-SR) algorithm in the supplementary material, with its detailed search procedure described in Appendix C. Appendix A specifies the assumptions used in the theoretical analysis and provides complete proofs of the theoretical results. Appendix D details all evaluated symbolic regression methods, including their parameter settings, search budgets, numbers of independent runs, and computing environment. Appendix H provides the expressions in the newly constructed dataset, together with the corresponding sampling domains, numbers of sampling points, and data-generation procedures. Appendix I provides detailed evaluation results for all eight SR algorithms under small and large sampling domains, including LER, RR, C-RR, and WSD. Sections 4, 5, and 6 of the main paper describe the evaluation metrics, experimental settings, and experimental procedures used in the respective studies. Together, these materials provide the information necessary to reproduce our experimental results.

REFERENCES

Tommaso Bendinelli, Luca Biggio, and Pierre-Alexandre Kamienny. Controllable neural symbolic regression. In *International Conference on Machine Learning*, pp. 2063–2077. PMLR, 2023.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International conference on machine learning*, pp. 936–945. Pmlr, 2021.

Lorenzo Bonin, Luigi Rovito, Andrea De Lorenzo, and Luca Manzoni. Cellular geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 25(1):8, 2024.

Qi Chen, Bing Xue, and Mengjie Zhang. Preserving population diversity based on transformed semantics in genetic programming for symbolic regression. *IEEE Transactions on Evolutionary Computation*, 25(3):433–447, 2020.

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

Laure Crochepierre, Lydia Boudjeloud-Assala, and Vincent Barbesant. Interactive reinforcement learning for symbolic regression from multi-format human-preference feedbacks. In *IJCAI*, pp. 5900–5903, 2022.

Fabricio Olivetti de Franca and Gabriel Kronberger. Improving genetic programming for symbolic regression with equality graphs. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 989–998, 2025.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. *Advances in Neural Information Processing Systems*, 37:44678–44709, 2024.

Christian Haider, Fabricio Olivetti de Franca, Bogdan Burlacu, and Gabriel Kronberger. Shape-constrained multi-objective genetic programming for symbolic regression. *Applied Soft Computing*, 132:109855, 2023.

Zhengyao Huang, Daniel Huang, Tiannan Xiao, Dina Ma, Zhenyu Ming, Hao Shi, and Yuanhui Wen. Improving monte carlo tree search for symbolic regression. *Advances in Neural Information Processing Systems*, 38:40819–40856, 2026.

Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892*, 2019.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *European conference on genetic programming*, pp. 70–82. Springer, 2003.

Joanne T Kim, Sookyung Kim, and Brenden K Petersen. An interactive visualization platform for deep symbolic regression. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 5261–5263, 2021.

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

Krzysztof Krawiec and Bartosz Wieloch. Automatic generation and exploitation of related problems in genetic programming. In *IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE, 2010.

William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, 2021(DB1):1, 2021.

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.

Yanjie Li, Weijun Li, Lina Yu, Min Wu, Jingyi Liu, Wenqiang Li, and Meilan Hao. Discovering mathematical formulas from data via gpt-guided monte carlo tree search. *Expert Systems with Applications*, 281:127591, 2025.

Matteo Merler, Katsiaryna Haitsiukevich, Nicola Dainese, and Pekka Marttinen. In-context symbolic regression: Leveraging large language models for function discovery. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 427–444, 2024.

T Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Daniel M Faissol, and Brenden K Petersen. Symbolic regression via neural-guided genetic programming population seeding. *arXiv preprint arXiv:2111.00053*, 2021.

Tomasz P Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(3):326–340, 2014.

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. *arXiv preprint arXiv:1912.04871*, 2019.

Kai Ruan, Yilong Xu, Ze-Feng Gao, Yang Liu, Yike Guo, Ji-Rong Wen, and Hao Sun. Discovering physical laws with parallel symbolic enumeration. *Nature Computational Science*, 6(1):53–66, 2026.

Aliyu Sani Sambo, R Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. Evolving simple and accurate symbolic regression models via asynchronous parallel computing. *Applied Soft Computing*, 104:107198, 2021.

Evgeny S Saveliev, Samuel Holt, Nabeel Seedat, David L Bentley, Jim Weatherall, and Mihaela van der Schaar. Influence-guided symbolic regression: Scientific discovery via llm-driven equation search with granular feedback. *arXiv preprint arXiv:2605.29184*, 2026.

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.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan Reddy. Llm-sr: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*, volume 2025, pp. 16054–16085, 2025.

Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. *arXiv preprint arXiv:2205.13134*, 2022.

Bilge Taskin, Wenxiong Xie, and Teddy Lazebnik. Knowledge integration for physics-informed symbolic regression using pre-trained large language models. *Scientific Reports*, 16(1):1614, 2026.

Leonardo Trujillo, Luis Muñoz, Edgar Galván-López, and Sara Silva. neat genetic programming: Controlling bloat naturally. *Information Sciences*, 333:21–43, 2016.

Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

Marco Virgolin and Solon P Pissis. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter AN Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary computation*, 29(2):211–237, 2021.

Yilong Xu, Yang Liu, and Hao Sun. Reinforcement symbolic regression machine. In *International Conference on Learning Representations*, volume 2024, pp. 54416–54440, 2024.

Hengzhe Zhang and Aimin Zhou. Rl-gep: symbolic regression via gene expression programming and reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

A Detailed Proof

Lemma1 (Boundedness of Higher-Order Derivatives): Suppose that the candidate expression g and the target expression f are formed by finitely many applications of the given operators and are well-defined and smooth on the sampling interval $I = [-c, c]$. Then, for any fixed $p \in \mathbb{N}$, $\Delta^{(p)}$ is continuous on the compact interval I, and hence there exists a constant $L_p \geq 0$ such that $\left|\Delta^{(p)}(x)\right| \leq L_p$ for all $x \in I$.

Proof. Since $f, g \in C^p(I)$, their difference $\Delta := f - g$ also belongs to $C^p(I)$. Therefore, $\Delta^{(p)}$ is continuous on I, and so is $|\Delta^{(p)}|$. Because I is compact, the extreme value theorem guarantees the existence of some $x_0 \in I$ such that

$$L_p := \max_{x \in I} |\Delta^{(p)}(x)| = |\Delta^{(p)}(x_0)| < \infty.$$

Thus, $|\Delta^{(p)}(x)| \leq L_p$ for all $x \in I$. □

Theorem1 (Fitting-Error Upper Bound Induced by the Order of Contact): Let $c > 0$ and $p \in \mathbb{N}^+$. Suppose that $f, g \in C^p([-c, c])$ have contact of order at least p at $x = 0$, i.e., $\Delta^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$. Then, the continuous mean squared error between f and g satisfies $\mathrm{MSE}(f, g) \leq \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1}$. Here, L_p is an upper bound for $|\Delta^{(p)}|$ on $(-c, c)$, as defined in Lemma 1.

Proof. Since $f, g \in C^p([-c, c])$, their difference $\Delta(x) = f(x) - g(x)$ also belongs to $C^p([-c, c])$. Because f and g have contact of order at least p at $x = 0$, we have $\Delta^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$.

For any $x \in [-c, c]$, Taylor's theorem guarantees the existence of some ξ_x between 0 and x such that

$$\Delta(x) = \sum_{k=0}^{p-1} \frac{\Delta^{(k)}(0)}{k!} x^k + \frac{\Delta^{(p)}(\xi_x)}{p!} x^p.$$

Since $\Delta^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$, this reduces to $\Delta(x) = \frac{\Delta^{(p)}(\xi_x)}{p!} x^p$.

By Lemma 1, there exists $L_p \geq 0$ such that $|\Delta^{(p)}(t)| \leq L_p$ for all $t \in [-c, c]$. Hence, $|\Delta(x)| \leq \frac{L_p}{p!}|x|^p$, and therefore $\Delta^2(x) \leq \left(\frac{L_p}{p!}\right)^2 x^{2p}$.

By the definition of the continuous mean squared error,

$$\mathrm{MSE}(f, g) := \frac{1}{2c} \int_{-c}^{c} [f(x) - g(x)]^2, \mathrm{d}x = \frac{1}{2c} \int_{-c}^{c} \Delta^2(x), \mathrm{d}x.$$

Substituting the above bound gives

$$\mathrm{MSE}(f, g) \leq \frac{1}{2c} \left(\frac{L_p}{p!}\right)^2 \int_{-c}^{c} x^{2p}, \mathrm{d}x.$$

Since $\int_{-c}^{c} x^{2p} \, \mathrm{d}x = \frac{2c^{2p+1}}{2p+1}$, we obtain

$$\mathrm{MSE}(f, g) \leq \frac{1}{2c} \left(\frac{L_p}{p!}\right)^2 \frac{2c^{2p+1}}{2p+1} = \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1}.$$

□

Corollary 1.1 (Error Convergence with Increasing Order of Contact): Suppose there exists a constant $L > 0$ independent of the order of contact p such that, for all p under consideration, $\left|\Delta^{(p)}(x)\right| \leq L$, $\forall x \in [-c, c]$. By Theorem 1, $\mathrm{MSE}(f, g) \leq \left(\frac{L}{p!}\right)^2 \frac{c^{2p}}{2p+1}$.

For any fixed $c > 0$,

$$\lim_{p \to \infty} \frac{c^p}{p!} = 0,$$

and hence, by the squeeze theorem,

$$\lim_{p \to \infty} \mathrm{MSE} = 0.$$

Proof. By Theorem 1 and the assumption $\left|\Delta^{(p)}(x)\right| \leq L$, for all p under consideration,

$$\mathrm{MSE}(f, g) \leq \left(\frac{L}{p!}\right)^2 \frac{c^{2p}}{2p+1} = \frac{L^2}{2p+1} \left(\frac{c^p}{p!}\right)^2.$$

Let $a_p := \frac{c^p}{p!}$. Then $\frac{a_{p+1}}{a_p} = \frac{c}{p+1}$. Since $c > 0$ is fixed, $\lim_{p \to \infty} \frac{a_{p+1}}{a_p} = \lim_{p \to \infty} \frac{c}{p+1} = 0$. Hence,

$$\lim_{p \to \infty} a_p = \lim_{p \to \infty} \frac{c^p}{p!} = 0.$$

It follows that

$$\lim_{p \to \infty} \frac{L^2}{2p+1} \left(\frac{c^p}{p!}\right)^2 = 0.$$

Since the continuous mean squared error satisfies $\mathrm{MSE}(f, g) \geq 0$, for all p under consideration,

$$0 \leq \mathrm{MSE}(f, g) \leq \frac{L^2}{2p+1} \left(\frac{c^p}{p!}\right)^2.$$

Therefore, by the squeeze theorem,

$$\lim_{p\to\infty} \mathrm{MSE}(f,g) = 0.$$

□

Corollary 1.2 (Error Convergence as the Sampling Interval Shrinks): Fix the order of contact p. Suppose that f and g are well-defined and smooth on some interval $[-c_0, c_0]$. By Lemma 1, there exists a constant $L_p > 0$ such that $\left|\Delta^{(p)}(x)\right| \le L_p, \forall x \in [-c_0, c_0]$. For any $0 < c < c_0$, the same bound also holds on $[-c, c]$. Since

$$\lim_{c\to 0} \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1} = 0,$$

Theorem 1 therefore implies that

$$\lim_{c\to 0} \mathrm{MSE}(f,g) = 0.$$

Proof. Fix the order of contact p. Since f and g are defined on $[-c_0, c_0]$ and belong to $C^p([-c_0, c_0])$, Lemma 1 guarantees the existence of a constant $L_p \ge 0$ such that $\left|\Delta^{(p)}(x)\right| \le L_p$ for all $x \in [-c_0, c_0]$.

For any $0 < c < c_0$, we have $[-c, c] \subset [-c_0, c_0]$, so $\left|\Delta^{(p)}(x)\right| \le L_p$ continues to hold for all $x \in [-c, c]$. Since f and g have contact of order at least p at $x = 0$, Theorem 1 gives

$$\mathrm{MSE}_{[-c,c]}(f,g) \le \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1}.$$

Here, $\mathrm{MSE}_{[-c,c]}(f,g) := \frac{1}{2c}\int_{-c}^{c}[f(x)-g(x)]^2\,\mathrm{d}x \ge 0$. Since p and L_p are fixed and $\lim_{c\to 0} c^{2p} = 0$,

$$\lim_{c\to 0} \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1} = 0.$$

Hence, for any $0 < c < c_0$,

$$0 \le \mathrm{MSE}_{[-c,c]}(f,g) \le \left(\frac{L_p}{p!}\right)^2 \frac{c^{2p}}{2p+1}.$$

By the squeeze theorem,

$$\lim_{c\to 0} \mathrm{MSE}_{[-c,c]}(f,g) = 0.$$

□

Theorem2 (Construction of Locally Taylor-Consistent Candidates): Let $c > 0$ and $p \in \mathbb{N}^+$. For every $g \in G_p(f)$, there exists a unique $q \in Q$ such that $g(x) = \sum_{j=0}^{p-1} \frac{a_j}{j!}x^j + x^p q(x), \quad x \in [-c, c]$, where $Q := \left\{q \in C^0([-c,c]) \;\middle|\; x^p q(x) \in C^p([-c,c])\right\}$. Equivalently, $G_p(f)$ and Q are in one-to-one correspondence.

Proof. Let $P_{p-1}(x) := \sum_{j=0}^{p-1} \frac{a_j}{j!}x^j$. For any $g \in G_p(f)$, the definition of $G_p(f)$ gives $g^{(j)}(0) = a_j$ for $j = 0, 1, \ldots, p-1$. Hence, $P_{p-1}(x)$ is the $(p-1)$-th order Taylor polynomial of g at $x = 0$.

Define $h(x) := g(x) - P_{p-1}(x)$. Then $h \in C^p([-c, c])$ and $h^{(j)}(0) = 0$ for $j = 0, 1, \ldots, p-1$. By Taylor's theorem with integral remainder, for any $x \in [-c, c]$, $h(x) = \frac{1}{(p-1)!}\int_0^x (x-t)^{p-1} g^{(p)}(t)\,\mathrm{d}t$.

Using the substitution $t = sx$, we obtain

$$h(x) = \frac{x^p}{(p-1)!}\int_0^1 (1-s)^{p-1} g^{(p)}(sx)\,\mathrm{d}s.$$

Define $q(x) := \frac{1}{(p-1)!}\int_0^1 (1-s)^{p-1} g^{(p)}(sx)\,\mathrm{d}s$. Since $g^{(p)}$ is continuous on $[-c,c]$, $q \in C^0([-c,c])$. Therefore,

$$g(x) = P_{p-1}(x) + x^p q(x) = \sum_{j=0}^{p-1} \frac{a_j}{j!} x^j + x^p q(x).$$

Moreover, $x^p q(x) = g(x) - P_{p-1}(x) \in C^p([-c,c])$, so $q \in Q$.

We next establish uniqueness. Suppose $q_1, q_2 \in Q$ and $P_{p-1}(x) + x^p q_1(x) = P_{p-1}(x) + x^p q_2(x)$. For any $x \neq 0$, this implies $q_1(x) = q_2(x)$. Since q_1 and q_2 are continuous on $[-c,c]$, letting $x \to 0$ yields $q_1(0) = q_2(0)$. Hence, $q_1 = q_2$, proving uniqueness.

Conversely, for any $q \in Q$, define $g(x) := P_{p-1}(x) + x^p q(x)$. Since $q \in Q$, $x^p q(x) \in C^p([-c,c])$, and therefore $g \in C^p([-c,c])$.

Let $h(x) := x^p q(x)$. Since q is continuous on the compact interval $[-c,c]$, there exists $M \geq 0$ such that $|q(x)| \leq M$ for all $x \in [-c,c]$. Thus, $|h(x)| \leq M|x|^p$, i.e., $h(x) = O(x^p)$.

Suppose there exists $r \in \{0,1,\ldots,p-1\}$ such that $h^{(r)}(0) \neq 0$, and let r be the smallest such index. Taylor expansion then gives

$$h(x) = \frac{h^{(r)}(0)}{r!} x^r + o(x^r), \qquad x \to 0.$$

However, since $r < p$, we also have $h(x) = O(x^p) = o(x^r)$, a contradiction. Hence, $h^{(j)}(0) = 0$ for $j = 0,1,\ldots,p-1$.

Therefore, for $j = 0,1,\ldots,p-1$, $g^{(j)}(0) = P_{p-1}^{(j)}(0) + h^{(j)}(0) = a_j$, which implies $g \in G_p(f)$.

Thus, the mapping

$$q \longmapsto \sum_{j=0}^{p-1} \frac{a_j}{j!} x^j + x^p q(x)$$

establishes a one-to-one correspondence between Q and $G_p(f)$. □

Theorem3 (Infinitude of Locally Taylor-Consistent Candidates): For any $m \in \mathbb{N}^+$, let $q_m(x) = x^m$. Then $q_m \in C^0([-c,c])$ and $x^p q_m(x) = x^{p+m} \in C^p([-c,c])$, so $q_m \in Q$. Since the functions in $\{q_m\}_{m=1}^{\infty}$ are pairwise distinct, Q contains a countably infinite subset. By Theorem 2, $G_p(f)$ is in bijection with Q, and hence $|G_p(f)| = |Q|$. Therefore, $G_p(f)$ is infinite.

Proof. For any $m \in \mathbb{N}^+$, define $q_m(x) := x^m$. Since q_m is a polynomial, $q_m \in C^0([-c,c])$. Moreover, $x^p q_m(x) = x^{p+m}$ is also a polynomial, so $x^p q_m(x) \in C^p([-c,c])$. By the definition of Q, $q_m \in Q$ for every $m \in \mathbb{N}^+$.

We next show that these functions are pairwise distinct. Let $m, n \in \mathbb{N}^+$ with $m \neq n$. Without loss of generality, suppose $m < n$, and choose $x_0 \in (0, \min\{c,1\})$. Then $q_m(x_0) = x_0^m \neq x_0^n = q_n(x_0)$, and hence $q_m \neq q_n$.

Thus, the family $\{q_m\}_{m=1}^{\infty}$ contains a countably infinite collection of pairwise distinct elements, so Q is infinite.

By Theorem 2, the mapping $T \to G_p(f)$,

$$T(q)(x) = \sum_{j=0}^{p-1} \frac{a_j}{j!} x^j + x^p q(x),$$

is a bijection. In particular, for each $m \in \mathbb{N}^+$, define $g_m(x) := T(q_m)(x) = \sum_{j=0}^{p-1} \frac{a_j}{j!} x^j + x^{p+m}$. Then $g_m \in G_p(f)$.

Since T is injective and the elements of $\{q_m\}_{m=1}^{\infty}$ are pairwise distinct, the elements of $\{g_m\}_{m=1}^{\infty}$ are also pairwise distinct. Hence, $G_p(f)$ contains a countably infinite subset, and therefore $|G_p(f)| = \infty$. □

Theorem4 (Local Attainability of Taylor Consistency): Let $p, m \in \mathbb{N}^+$ and $c > 0$. Consider a fixed candidate structure parameterized by $\theta = (\theta_1, \ldots, \theta_m)^\top \in \mathbb{R}^m$, denoted by $g(x;\theta)$, and assume that $g(\cdot;\theta) \in C^p([-c,c])$ for every θ. Define its $(p-1)$-jet map at $x=0$ by $J_{p-1}(\theta) := \big(g(0;\theta), \partial_x g(0;\theta), \ldots, \partial_x^{p-1} g(0;\theta)\big)^\top \in \mathbb{R}^p$, with Jacobian $DJ_{p-1}(\theta) := \Big[\frac{\partial}{\partial \theta_j}\big(\partial_x^{i-1} g(0;\theta)\big)\Big]_{\substack{i=1,\ldots,p \\ j=1,\ldots,m}} \in \mathbb{R}^{p\times m}$. Suppose that J_{p-1} is continuously differentiable in a neighborhood of θ_0 and that $\operatorname{rank} DJ_{p-1}(\theta_0) = p$. Then $m \geq p$, and there exist a neighborhood $\Theta_0 \subset \mathbb{R}^m$ of θ_0 and a neighborhood $U \subset \mathbb{R}^p$ of $F_0 := J_{p-1}(\theta_0)$ such that $U \subset J_{p-1}(\Theta_0)$. Therefore, for any $f \in C^p([-c,c])$ satisfying $\big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top \in U$, there exists $\theta^* \in \Theta_0$ such that $J_{p-1}(\theta^*) = \big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top$. Hence, $g(x;\theta^*)$ and $f(x)$ have contact of order at least p at $x=0$.

Proof. Let $F(\theta) := J_{p-1}(\theta) = \big(g(0;\theta), \partial_x g(0;\theta), \ldots, \partial_x^{p-1} g(0;\theta)\big)^\top$. By assumption, F is continuously differentiable in a neighborhood of θ_0, with $\operatorname{rank} DF(\theta_0) = p$.

Since $DF(\theta_0) \in \mathbb{R}^{p\times m}$ has rank p, necessarily $p \leq m$. Moreover, it contains an invertible $p \times p$ submatrix. After reordering the parameter components, we may assume without loss of generality that this submatrix consists of the first p columns. Write $\theta = (u, v)$, where $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^{m-p}$, and let $\theta_0 = (u_0, v_0)$. Then $D_u F(u_0, v_0)$ is invertible.

Let $F_0 := F(\theta_0)$ and define $H(u, v, y) := F(u, v) - y$, where $y \in \mathbb{R}^p$. By construction, H is continuously differentiable in a neighborhood of (u_0, v_0, F_0), with $H(u_0, v_0, F_0) = F(u_0, v_0) - F_0 = 0$. Moreover, $D_u H(u_0, v_0, F_0) = D_u F(u_0, v_0)$ is invertible.

By the implicit function theorem, there exist a neighborhood $V \subset \mathbb{R}^{m-p}$ of v_0, a neighborhood $U \subset \mathbb{R}^p$ of F_0, and a continuously differentiable function $\varphi : V \times U \to \mathbb{R}^p$ such that $\varphi(v_0, F_0) = u_0$ and, for every $(v, y) \in V \times U$, $H(\varphi(v, y), v, y) = 0$, equivalently, $F(\varphi(v, y), v) = y$.

Now fix $v = v_0$. For any $y \in U$, define $\theta(y) := (\varphi(v_0, y), v_0)$. After shrinking U if necessary, $\theta(y)$ lies in a neighborhood $\Theta_0 \subset \mathbb{R}^m$ of θ_0, and $F(\theta(y)) = y$. Hence, $U \subset F(\Theta_0) = J_{p-1}(\Theta_0)$.

For any $f \in C^p([-c,c])$ satisfying $\big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top \in U$, define

$$\theta^* := \theta\left(\Big(f(0), f'(0), \ldots, f^{(p-1)}(0)\Big)^\top\right),$$

so that $J_{p-1}(\theta^*) = \big(f(0), f'(0), \ldots, f^{(p-1)}(0)\big)^\top$. It follows that $\partial_x^k g(0;\theta^*) = f^{(k)}(0)$ for $k = 0, 1, \ldots, p-1$, or equivalently, $\big(f - g(\cdot;\theta^*)\big)^{(k)}(0) = 0$ for $k = 0, 1, \ldots, p-1$.

Therefore, $g(x;\theta^*)$ and $f(x)$ have contact of order at least p at $x = 0$. □

Theorem5 (Global Attainability of Taylor Consistency): Let $p \in \mathbb{N}^+$ and $c > 0$. Consider a candidate expression family $g(x;\alpha,\gamma) \in C^p([-c,c])$, where $\alpha = (\alpha_1, \ldots, \alpha_p)^\top \in \mathbb{R}^p$ denotes the linear weight parameters and γ the remaining parameters. Suppose $g(x;\alpha,\gamma) = \sum_{j=1}^p \alpha_j \phi_j(x;\gamma)$, with $\phi_j(\cdot;\gamma) \in C^p([-c,c])$. Define the $(p-1)$-jet map at $x=0$ as $J_{p-1}(\alpha,\gamma) = \big(g(0;\alpha,\gamma), \partial_x g(0;\alpha,\gamma), \ldots, \partial_x^{p-1} g(0;\alpha,\gamma)\big)^\top \in \mathbb{R}^p$. For fixed γ, its Jacobian with respect to α is $M(\gamma) := D_\alpha J_{p-1}(\alpha,\gamma) = \Big[\frac{\partial}{\partial \alpha_j}\big(\partial_x^{i-1} g(0;\alpha,\gamma)\big)\Big]_{\substack{i=1,\ldots,p \\ j=1,\ldots,p}} \in \mathbb{R}^{p\times p}$. Since g is linear in α, $J_{p-1}(\alpha,\gamma) = M(\gamma)\alpha$. If $\det M(\gamma) \neq 0$, then for any $F \in \mathbb{R}^p$, the unique choice $\alpha = M(\gamma)^{-1}F$ satisfies $J_{p-1}(\alpha,\gamma) = F$. Therefore, for fixed γ, the $(p-1)$-jet map is globally attainable over $\mathbb{R}^p$ if and only if $M(\gamma)$ is invertible.

Proof. Fix γ. Since $g(x;\alpha,\gamma) = \sum_{j=1}^p \alpha_j \phi_j(x;\gamma)$ and $\phi_j(\cdot;\gamma) \in C^p([-c,c])$, for each $i = 1, \ldots, p$,

$$\partial_x^{i-1} g(0;\alpha,\gamma) = \sum_{j=1}^{p} \alpha_j \, \partial_x^{i-1} \phi_j(0;\gamma).$$

Hence,

$$J_{p-1}(\alpha,\gamma) = \begin{bmatrix} \sum_{j=1}^{p} \alpha_j \phi_j(0;\gamma) \\ \sum_{j=1}^{p} \alpha_j \partial_x \phi_j(0;\gamma) \\ \vdots \\ \sum_{j=1}^{p} \alpha_j \partial_x^{p-1} \phi_j(0;\gamma) \end{bmatrix}.$$

Define $\widetilde{M}(\gamma) := \left[\partial_x^{i-1}\phi_j(0;\gamma)\right]_{\substack{i=1,\dots,p \\ j=1,\dots,p}} \in \mathbb{R}^{p\times p}$. Then $J_{p-1}(\alpha,\gamma) = \widetilde{M}(\gamma)\alpha$.

By linearity in α, for any $i, j = 1, \dots, p$,

$$\frac{\partial}{\partial \alpha_j}\left(\partial_x^{i-1} g(0;\alpha,\gamma)\right) = \partial_x^{i-1}\phi_j(0;\gamma).$$

Therefore,

$$M(\gamma) = D_\alpha J_{p-1}(\alpha,\gamma) = \left[\partial_x^{i-1}\phi_j(0;\gamma)\right]_{\substack{i=1,\dots,p \\ j=1,\dots,p}} = \widetilde{M}(\gamma).$$

Thus, $J_{p-1}(\alpha,\gamma) = M(\gamma)\alpha$.

For sufficiency, suppose $\det M(\gamma) \neq 0$. Then $M(\gamma)$ is invertible. For any $F \in \mathbb{R}^p$, choose $\alpha = M(\gamma)^{-1}F$. It follows that

$$J_{p-1}(\alpha,\gamma) = M(\gamma)\alpha = M(\gamma)M(\gamma)^{-1}F = F.$$

Thus, for fixed γ, any target jet vector $F \in \mathbb{R}^p$ can be matched exactly by a linear weight vector α.

Moreover, if $\alpha_1, \alpha_2 \in \mathbb{R}^p$ satisfy $J_{p-1}(\alpha_1,\gamma) = J_{p-1}(\alpha_2,\gamma) = F$, then $M(\gamma)(\alpha_1 - \alpha_2) = 0$. Since $M(\gamma)$ is invertible, $\alpha_1 - \alpha_2 = 0$, and hence $\alpha_1 = \alpha_2$. Therefore, the parameter α satisfying $J_{p-1}(\alpha,\gamma) = F$ is unique for each F.

For necessity, suppose that the $(p-1)$-jet is globally attainable for fixed γ. Then, for every $F \in \mathbb{R}^p$, there exists $\alpha \in \mathbb{R}^p$ such that $M(\gamma)\alpha = J_{p-1}(\alpha,\gamma) = F$. Hence, the linear map $\alpha \mapsto M(\gamma)\alpha$ from $\mathbb{R}^p$ to $\mathbb{R}^p$ is surjective. Since $M(\gamma)$ is a $p \times p$ square matrix, this is equivalent to $\operatorname{rank} M(\gamma) = p$, and therefore $\det M(\gamma) \neq 0$. Thus, $M(\gamma)$ is invertible.

Therefore, for fixed γ, the $(p-1)$-jet of the candidate-expression family is globally attainable if and only if $M(\gamma) = D_\alpha J_{p-1}(\alpha,\gamma)$ is invertible. □

B Constructive Examples for Theorems 4 and 5

Table 4 provides several constructive examples for Theorem 4, illustrating the local attainability of Taylor consistency through the adjustment of free constants. Using $f(x) = \sin(x)$ as the target expression with $p = 3$, we consider four structurally distinct candidate expressions and treat their constants as adjustable parameters. The parameters are chosen such that $g(0) = f(0) = 0$, $g'(0) = f'(0) = 1$, and $g''(0) = f''(0) = 0$, so that each candidate has the same second-order jet as $\sin(x)$ at $x = 0$. For each candidate structure, the Jacobian of the jet map with respect to the parameters has full row rank at the selected parameter point, satisfying the local attainability condition of Theorem 4. This demonstrates that, when a candidate structure has sufficient parametric degrees of freedom and its jet map satisfies the full-rank condition, local Taylor consistency can be attained in a neighborhood of the target jet solely by adjusting free constants, without altering the candidate structure. The continuous MSEs reported in the table further show that these structurally distinct candidates can still achieve low fitting errors relative to the target function over a small sampling domain.

Table 4: Constructive examples illustrating the local attainability of Taylor consistency for $f(x) = \sin(x)$ through free-constant adjustment.

Structure	**Parameter choice**	**Candidate expression**	**MSE** $[-1, 1]$	**MSE** $[-0.5, 0.5]$
$a\sin(bx+c) + d\sin(ex) + h$	$a=1,\ b=\frac{1}{2},\ c=0,$ $d=\frac{1}{2},\ e=1,\ h=0$	$\sin(x/2) + \frac{1}{2}\sin x$	$5.06e{-4}$	$8.51e{-6}$
$a\cos(bx+c)\cos(dx) + e$	$a=1,\ b=1,\ c=-\frac{\pi}{2},$ $d=\frac{1}{2},\ e=0$	$\cos(x-\pi/2)\cos(x/2)$	$1.66e{-3}$	$3.24e{-5}$
$a\exp(bx) + c\exp(dx) + e$	$a=-\frac{1}{3},\ b=\frac{3}{2},\ c=3,$ $d=\frac{1}{2},\ e=-\frac{8}{3}$	$-\frac{1}{3}e^{3x/2} + 3e^{x/2} - \frac{8}{3}$	$5.52e{-4}$	$4.66e{-6}$
$a\log\left(\frac{1+bx}{1+cx}\right) + d$	$a=2,\ b=\frac{1}{4},$ $c=-\frac{1}{4},\ d=0$	$2\log\left(\frac{1+x/4}{1-x/4}\right)$	$4.72e{-3}$	$7.73e{-5}$

Table 5: Constructive examples illustrating the global attainability of Taylor consistency for $f(x) = \sin(x)$ through adjustable linear weights.

Basis functions $\boldsymbol{\phi_1, \phi_2, \phi_3}$	**Jacobian** $\boldsymbol{M}$	**Candidate expression**	**MSE** $[-1, 1]$	**MSE** $[-0.5, 0.5]$
$\exp(x),\ \sin(x^2),\ 1$	$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 2 & 0 \end{bmatrix}$	$\exp(x) - \frac{1}{2}\sin(x^2) - 1$	$1.72\text{e}{-2}$	$2.50\text{e}{-4}$
$\exp(-x),\ \cos(2x),\ x^2$	$\begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 1 & -4 & 2 \end{bmatrix}$	$-\exp(-x) + \cos(2x) + \frac{5}{2}x^2$	$5.02\text{e}{-2}$	$4.08\text{e}{-4}$
$\log(1+x^2),\ \exp(x),\ 1$	$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 2 & 1 & 0 \end{bmatrix}$	$\exp(x) - \frac{1}{2}\log(1+x^2) - 1$	$2.07\text{e}{-2}$	$2.78\text{e}{-4}$
$\sin(x^2+x),\ \cos(2x),\ x^2$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 2 & -4 & 2 \end{bmatrix}$	$\sin(x^2+x) - x^2$	$4.83\text{e}{-2}$	$1.39\text{e}{-4}$

Table 5 provides several constructive examples for Theorem 5, illustrating the global attainability of Taylor consistency when the linear weights are adjustable. Using $f(x) = \sin(x)$ as the target expression with $p = 3$, we consider three different sets of three basis functions ϕ_1, ϕ_2, ϕ_3 and construct candidate expressions of the form $g(x) = \sum_{j=1}^{3} \alpha_j \phi_j(x)$. For each set, the corresponding jet-map Jacobian matrix M is invertible. By Theorem 5, any prescribed second-order jet therefore admits a unique linear weight vector $\alpha = M^{-1}F$ that matches it exactly. In the table, we set $F = (f(0), f'(0), f''(0))^\top = (0, 1, 0)^\top$ and solve for the corresponding weights, obtaining structurally distinct candidate expressions that all share the same second-order jet as $\sin(x)$ at $x = 0$. Unlike the local attainability established in Theorem 4, invertibility here guarantees global attainability for any target jet in $\mathbb{R}^3$. The continuous MSE results further show that these structurally distinct candidates can still achieve low fitting errors relative to the target function over a small sampling domain.

C Random Duplicate-Free Symbolic Regression (RDF-SR)

Random Duplicate-Free Symbolic Regression (RDF-SR) represents the space of valid expressions satisfying the operator-set and expression-complexity constraints as a search tree, as illustrated in Figure 8. The root represents an empty expression, while each subsequent node represents a partial expression at an intermediate stage of construction. At each expansion step, RDF-SR selects the first unfilled position in level order, proceeding from top to bottom and left to right, and assigns a valid binary operator, unary operator, variable, or constant placeholder to generate a new partial expression. A node becomes terminal when no unfilled positions remain, corresponding to a complete candidate expression.

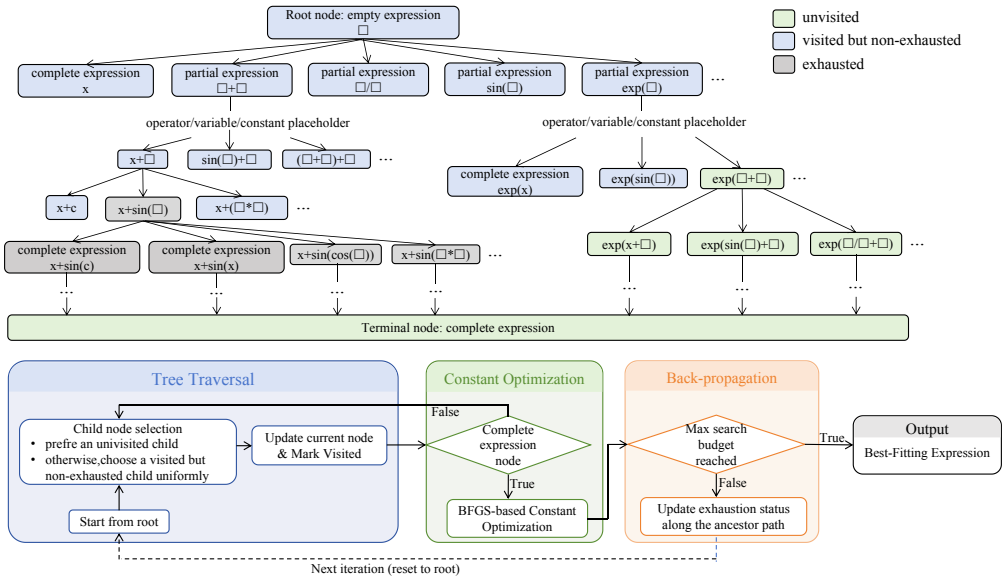


Figure 8: Framework of the RDF-SR algorithm.

During the search, the algorithm first explores an unvisited feasible branch from the current node. Once all feasible branches have been visited, it randomly selects a child branch that is not yet exhausted and continues the expansion.

Once a terminal node produces a complete expression, it is marked as exhausted. A non-terminal node is likewise marked as exhausted once all of its feasible child branches have been exhausted. Because the subtree rooted at an exhausted node contains no expressions that remain to be generated, it can be skipped in subsequent search, ensuring that each expression structure is generated at most once.

After a complete expression structure is generated, the constant placeholders are treated as free parameters and fitted using BFGS to obtain the final expression.

RDF-SR does not use any performance metric to guide the search; instead, it explores the expression space randomly without repetition. It therefore serves as a baseline for assessing whether heuristic criteria such as fitting error, or newly introduced search strategies, genuinely improve target-expression recovery.

D Symbolic Regression Algorithms and Parameter Settings

For algorithms that support these settings, we use the operator set $\{+, -, \times, \div, \sin, \cos, \exp, \log\}$, set the maximum expression complexity to 30 and the maximum tree depth to 6, and use a maximum search budget of 2×10^6. Symbolic regression aims to discover analytical expressions that capture underlying relationships in the data without specifying their functional forms in advance. Because the target expression is unknown before search, structural constraints based on prior knowledge may exclude the true expression from the search space or bias the search toward particular functional forms. We therefore impose no additional structural priors and allow the algorithms to search the full expression space defined by the above constraints. For example, nested trigonometric functions are permitted, as are nested combinations of $\exp$ and $\log$.

For the experiments in Sections 4 and 5, we evaluate representative methods from three symbolic regression paradigms: PySR for genetic programming, DSO for reinforcement learning, and RSRM for Monte Carlo tree search. To isolate the effect of fitting-error guidance itself, these experiments select, where possible, methods without preprocessing or postprocessing mechanisms such as problem decomposition, subexpression replacement, or structural correction, thereby reducing the influence of additional mechanisms on the underlying search process. For each target expression, RDF-

SR and PySR are run independently 50 times, while DSO and RSRM are each run independently 10 times due to runtime considerations.

For the experiments in Section 6, we use PySR, EGGP, and GPlearn as GP-based baselines; DSO and uDSR as RL-based baselines; and RSRM, iMCTS, and TPSR as MCTS-based baselines to evaluate the search performance of existing symbolic regression methods. Each algorithm is run independently five times for every target expression.

Computing infrastructure: All experiments were conducted on a server equipped with an Intel Xeon Platinum 8255C CPU at 2.50 GHz, 32 GB of system memory, four NVIDIA GeForce RTX 4090 GPUs with 24 GB of VRAM each, and four NVIDIA GeForce RTX 3090 GPUs with 24 GB of VRAM each.

The descriptions and parameter settings of each algorithm are as follows:

- PySR (Python Symbolic Regression): PySR employs a multi-population evolutionary framework in which expression trees evolve asynchronously across multiple independent populations. During evolution, candidate expressions are selected via tournament selection, and new individuals are generated through mutation or crossover. Candidate quality is evaluated using prediction error together with an adaptive complexity penalty. PySR further follows an evolution–simplification–optimization cycle, applying algebraic simplification to candidate expressions and BFGS to optimize their real-valued constants. Search diversity is maintained through age regularization, simulated annealing, and inter-population migration. The best expressions at different complexity levels are ultimately retained to form a Pareto front.

  Experimental settings: Candidate expressions are evaluated using the pointwise squared-error loss L2DistLoss(). Computation is performed with 64-bit floating-point precision. The maximum expression complexity is set to 30, the maximum expression depth to 6, and the search budget to 2×10^6. All other parameters are left at their default values.

- EGGP (e-graph Genetic Programming): EGGP augments the conventional genetic programming (GP) framework with e-graphs and equality saturation to track previously visited expressions together with their mathematically equivalent forms, thereby reducing redundant exploration of duplicate or equivalent expressions. New individuals are still generated through tournament selection, subtree crossover, and subtree mutation. However, historical information stored in the e-graph is used to guide these variation operators toward subtree combinations that are more likely to produce previously unseen expressions, improving expression novelty and search-space utilization. Newly generated expressions are inserted into the e-graph, and equivalent forms are merged through one step of equality saturation.

  Experimental settings: The population size is set to 500, the maximum number of iterations to 4000, the maximum expression complexity to 30, and the search budget to 2×10^6. All other parameters are left at their default values.

- GPlearn (Genetic Programming in Python): GPlearn is a conventional genetic-programming-based symbolic regression method that represents candidate expressions as expression trees. It initializes a population randomly, selects parents via tournament selection, and generates new individuals through subtree crossover, subtree mutation, hoist mutation, point mutation, or reproduction. Candidate expressions are evaluated using fitting error as the fitness criterion, while a parsimony coefficient penalizes overly complex expressions to control expression bloat. After multiple generations of evolution, the algorithm returns the expression with the best fitness.

  Experimental settings: The population size is set to 5000, the maximum number of iterations to 400, and the search budget to 2×10^6. All other parameters are left at their default values.

- DSO (Deep Symbolic Optimization): DSO combines neural-guided search with genetic programming. An autoregressive RNN generates prefix expression sequences composed of operators, variables, and constants, which are then used to initialize the genetic-programming population. Genetic programming subsequently evolves the population over multiple generations through selection, crossover, and multiple mutation operators, with candidate fitness defined by the inverse normalized root mean squared error on the training data. After each genetic-programming round, high-fitness expressions accumulated across

generations are combined with RNN-generated samples to update the RNN. The RNN can be trained using standard policy gradient, risk-seeking policy gradient, or priority queue training.

Experimental settings: The minimum and maximum expression complexities are set to 4 and 30, respectively. Priors on the number of constants, inverse operations, trigonometric nesting, Soft length, and Uniform arity are disabled. The search budget is set to 2×10^6, and all other parameters are left at their default values.

- uDSR (Unified Deep Symbolic Regression): uDSR extends Deep Symbolic Regression (DSR) by integrating recursive problem simplification, neural-guided search, large-scale pretraining, genetic programming (GP), and linear models within a unified modular framework. It first uses AI Feynman to recursively decompose the original problem into lower-dimensional subproblems. A pretrained RNN controller then generates candidate expressions and updates its search policy using risk-seeking policy gradient. These candidates are also used to initialize GP for further evolution, with high-performing individuals subsequently fed back to the controller. In addition, a LINEAR token is introduced to efficiently represent and optimize linear substructures, while L-BFGS-B is used to optimize continuous constants. The solutions to the subproblems are finally recombined to obtain the complete expression.

  Experimental settings: The minimum and maximum expression complexities are set to 4 and 30, respectively. Priors on the number of constants, inverse operations, trigonometric nesting, Soft length, and Uniform arity are disabled. The search budget is set to 2×10^6, and all other parameters are left at their default values.

- RSRM (Reinforcement Symbolic Regression Machine): RSRM performs global exploration over the expression-tree space using Monte Carlo tree search (MCTS) and integrates three main modules. The search module expands branches based on simulated rewards and incorporates Double Q-learning to evaluate candidate actions and guide the search toward promising branches, thereby accelerating convergence. The refinement module further improves candidate expressions discovered by MCTS through genetic programming (GP), while BFGS is used to optimize continuous constants. The modulated sub-tree discovery (MSDB) module decomposes high-quality expressions according to additive/subtractive, power, and multiplicative structures, identifies frequently occurring subexpressions, and converts long, complex equations into more manageable subproblems. This substantially reduces the effective search space and problem complexity, while parity detection can additionally be used for structural simplification.

  Experimental settings: The maximum expression complexity is set to 30 and the maximum expression depth to 6. No additional structural priors are imposed. The search budget is set to 2×10^6, and all other parameters are left at their default values.

- iMCTS (Improving Monte Carlo Tree Search for Symbolic Regression): iMCTS extends Monte Carlo tree search (MCTS) with two main improvements. First, it introduces an Extreme Bandit Allocation strategy that selects nodes according to the historical maximum reward of each branch rather than its average reward, improving the search for globally optimal expressions. Second, genetic-programming mutation and crossover are incorporated directly into MCTS as state-jumping actions, enabling the search to move from existing high-quality expressions to new regions of the candidate space. Each node maintains the Top-N highest-reward trajectories, while bidirectional propagation shares information about promising candidates across the search tree, improving both search efficiency and global exploration. For expressions containing free constants, BFGS is used to optimize their values.

  Experimental settings: The maximum expression depth is set to 6 and the search budget to 2×10^6. All other parameters are left at their default values.

- TPSR (Transformer-based Planning for Symbolic Regression): TPSR combines a pretrained Transformer with Monte Carlo tree search (MCTS) to perform lookahead planning during expression generation. MCTS represents each partial expression as a state and the next symbol as an action, while the Transformer's next-token probabilities guide node selection and expansion. Candidate expressions are then completed using beam search, after which rewards are computed from fitting error and expression complexity and propagated back through the search tree. Continuous constants are optimized using BFGS, and a

Table 6: Recovery rates of symbolic regression algorithms on individual target expressions in the Livermore dataset.

Name	Expression	Complexity	RDF-SR	PySR	DSO	RSRM
Livermore-1	$\frac{1}{3} + x + \sin(x^2)$	8	100%	100%	100%	100%
Livermore-2	$\sin(x^2)\cos(x) - 2$	9	94%	62%	40%	100%
Livermore-3	$\sin(x^3)\cos(x^2) - 1$	13	0%	78%	0%	90%
Livermore-4	$\log(x+1) + \log(x^2+1) + \log(x)$	12	56%	26%	80%	30%
Livermore-5	$x^4 - x^3 + x^2 - y$	13	0%	54%	0%	90%
Livermore-6	$4x^4 + 3x^3 + 2x^2 + x$	15	36%	88%	100%	0%
Livermore-7	$\sinh(x)$	9	100%	2%	100%	0%
Livermore-8	$\cosh(x)$	9	100%	0%	90%	0%
Livermore-9	$x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$	25	0%	0%	0%	0%
Livermore-10	$6\sin(x)\cos(y)$	7	100%	100%	100%	100%
Livermore-11	$\frac{x^2y^2}{x+y}$	11	0%	72%	10%	100%
Livermore-12	$\frac{x^5}{y^3}$	15	0%	12%	10%	100%
Livermore-13	$x^{1/3}$	5	100%	96%	100%	100%
Livermore-14	$x^3 + x^2 + x + \sin(x^2) + \sin(x)$	17	0%	24%	0%	10%
Livermore-15	$x^{1/5}$	5	100%	96%	100%	100%
Livermore-16	$x^{2/5}$	5	100%	94%	100%	100%
Livermore-17	$4\sin(x)\cos(y)$	7	100%	100%	100%	100%
Livermore-18	$\sin(x^2)\cos(x) - 5$	9	96%	68%	10%	90%
Livermore-19	$x^5 + x^4 + x^2 + x$	13	30%	60%	50%	100%
Livermore-20	$\exp(-x^2)$	6	100%	98%	100%	100%
Livermore-21	$x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$	21	0%	0%	0%	0%
Livermore-22	$\exp(-0.5x^2)$	6	100%	92%	100%	80%

caching mechanism is employed to avoid redundant computation. In this way, TPSR leverages the prior encoded by the pretrained model while using MCTS to further improve the trade-off between fitting accuracy and expression complexity.

Experimental settings: All parameters are left at their default values.

E Recovery Rates for the Experiments in Section 4

Table 6 reports the target expressions and their complexities in the Livermore dataset, together with the recovery rates achieved by RDF-SR, PySR, DSO, and RSRM for each expression.

F Candidate Counts Across Low-Error Ranges in RDF-SR Search

Table 7 reports the average number of candidate expressions within different low-error ranges for each target expression in the Livermore dataset. RDF-SR is run independently 50 times per target expression with a sampling-domain length of 2, and each run performs 2×10^6 duplicate-free random searches. The results show that, within a small sampling domain, the expression space generally contains a large number of candidates that are structurally distinct from the target expression yet achieve low fitting errors.

G Experimental Results of DSO and RSRM under Different Sampling Domains

To ensure that the conclusions are not specific to a single symbolic regression algorithm, we further conduct cross-algorithm validation experiments with DSO and RSRM to examine how sampling-domain size affects recovery rate and low-error rate.

For the 17 univariate target expressions in the Livermore dataset, each algorithm is run independently 10 times under each sampling domain. For target expressions that admit both positive and negative inputs, the sampling domain is set to $(-c, c)$, $c \in \{1, 5, 9, 13\}$. For target expressions restricted to nonnegative inputs, the sampling domain is set to $(0, c)$, $c \in \{2, 10, 18, 26\}$.

Table 7: Average numbers of candidate expressions in different MSE ranges for each Livermore target expression.

Target expression	MSE range			
	$[10^{-3}, 10^{-2})$	$[10^{-4}, 10^{-3})$	$[10^{-5}, 10^{-4})$	$[0, 10^{-5})$
Livermore-1	32145.43	4169.75	1043.06	255.61
Livermore-2	44485.27	8148.16	1003.82	473.04
Livermore-3	53791.41	1167.71	52.51	4.29
Livermore-4	9595.82	2193.39	338.20	71.47
Livermore-5	30.18	1.77	0.00	0.00
Livermore-6	816.08	20.24	0.63	0.73
Livermore-7	102623.88	27885.84	3239.33	3283.94
Livermore-8	52864.76	14739.08	8266.16	4422.75
Livermore-9	4485.73	265.22	20.20	0.39
Livermore-10	102.96	5.57	1.20	28.39
Livermore-11	834.63	0.39	0.00	0.00
Livermore-12	16.14	0.00	0.00	0.02
Livermore-13	54031.75	8998.84	2338.78	1520.88
Livermore-14	11300.14	1703.43	226.49	23.88
Livermore-15	80393.86	11919.94	3293.69	2071.00
Livermore-16	48303.63	8578.88	2503.53	1438.67
Livermore-17	402.30	11.30	2.23	27.68
Livermore-18	34237.04	5075.84	642.59	370.18
Livermore-19	13486.06	1140.57	183.80	5.53
Livermore-20	110984.14	24852.82	5066.53	4246.80
Livermore-21	5335.39	236.31	9.71	0.57
Livermore-22	153893.47	49013.02	23310.47	7576.78

The results are shown in Figures 9 and 10. At a sampling-domain length of 2, both DSO and RSRM achieve low-error rates close to 100%. As the sampling domain expands, their low-error rates drop rapidly and then gradually stabilize. In contrast, the recovery rate of RSRM increases with the sampling-domain length before reaching a relatively stable level.

However, DSO's recovery rate remains nearly unchanged as the sampling domain expands. One possible explanation is that DSO generates expressions sequentially through a policy network and updates the policy based on the fitting errors of complete expressions. Its search therefore depends strongly on continuously discovering low-error candidates that provide informative reward signals. As the sampling domain expands, such candidates become sparser, making it more difficult to obtain stable and sustained rewards and, consequently, to establish an effective policy-optimization trajectory. This may limit further improvements in recovery rate. In contrast, genetic programming and Monte Carlo tree search can explore alternative candidate structures more directly through local structural modifications or explicit tree search, and may therefore be less sensitive to this effect under the same complexity constraints. Thus, even when algorithms share the same nominal search space and complexity limit, the practical impact of these constraints on DSO's search process may be more pronounced.

To further test this hypothesis, we increase DSO's maximum expression complexity to 64 and rerun the experiments. This supplementary experiment investigates how enlarging the sampling domain affects DSO's recovery rate when a larger expression space enables the exploration of more diverse candidate expressions with a wider range of fitting errors. The sampling domains are set to $(-c, c)$, $c \in \{1, 3, 5, 7, 9\}$. As shown in Figure 11, DSO exhibits the same trend as RSRM and PySR: its recovery rate increases as the sampling domain expands and then gradually stabilizes.

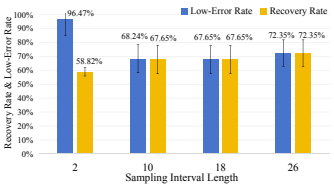


Figure 9: Recovery rate and low-error rate of RSRM across different sampling interval lengths. Error bars indicate standard error.

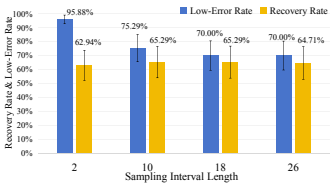


Figure 10: Recovery rate and low-error rate of DSO across different sampling interval lengths. Error bars indicate standard error.

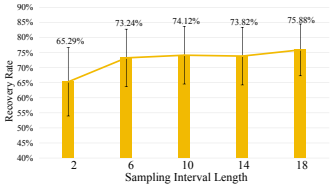


Figure 11: Recovery rate of DSO under different sampling interval lengths in a larger search space. Error bars indicate standard error.

H Detailed Description of the New Dataset

Using the univariate expressions in the Livermore dataset as benchmarks, we construct one subset for each benchmark expression, containing multiple expressions with similar fitting errors but structurally distinct forms. The resulting dataset consists of 17 subsets and 68 expressions in total. All non-benchmark expressions are low-error candidates generated by RDF-SR. For each expression, we consider a small and a large sampling domain: $(-1, 1)$ and $(-15, 15)$ for expressions that admit negative inputs, and $(0, 2)$ and $(0, 30)$ for expressions restricted to nonnegative inputs. Within each sampling domain, x_i is independently sampled from the corresponding uniform distribution, i.e., $x_i \sim \mathcal{U}(x_{\min}, x_{\max})$. The corresponding output is then computed as $y_i = f(x_i)$, yielding the data samples (x_i, y_i) used for symbolic regression. The sampling density is fixed at 0.1 under all settings. Table 8 reports each expression, its complexity, and its fitting error relative to the corresponding benchmark expression; the expression with zero fitting error serves as the benchmark for each subset.

Table 8: Expressions, complexities, and fitting errors in the constructed dataset.

Number	Expression	Complexity	Fitting error
l1-0	$\frac{1}{3} + x + \sin(x^2)$	8	0
l1-1	$\exp(x)\sin\left(x^2 + \frac{1}{3}\right)$	9	3.70×10^{-2}
l1-2	$\cos(0.2x + 1.25)\exp(3x)$	11	6.24×10^{-3}
l1-3	$x + \cos(x) + 0.7\left(2.8x^2 - \frac{1}{\cos(x)}\right)$	15	4.87×10^{-4}
l2-0	$\sin(x^2)\cos(x) - 2$	9	0
l2-1	$\sin\left(\sin(x^2)\cos(x)\right) - 2$	10	5.71×10^{-5}
l2-2	$-x^2 - \cos(2x) - 1$	10	3.15×10^{-4}
l2-3	$\exp\left(\sin(x^2 - 2.7)\right)(\exp(\cos x) - 6)$	13	1.23×10^{-2}
l3-0	$\sin(x^3)\cos(x^2) - 1$	13	0
l3-1	$0.6x - 0.25\sin(x)\exp\left(\cos(2.43x)\right) - 1$	16	3.08×10^{-4}
l3-2	$1.3\sin(2.17x)\left(0.99 - \cos x\right) - 1$	13	7.68×10^{-5}
l3-3	$0.47\sin(1.99x^3) - 1$	12	2.70×10^{-5}
l4-0	$\log(x + 1) + \log(x^2 + 1) + \log(x)$	12	0
l4-1	$\log(x) + x + \exp\left(-\frac{0.88}{x}\right)$	9	1.57×10^{-3}

Continued on next page

Number	Expression	Complexity	Fitting error
l4-2	$-1.27\left(\cos(x+0.32)-\log(x)-1.39\right)$	11	6.06×10^{-3}
l4-3	$-\frac{0.15}{\sin(\sin(x+0.03))}+1.95x+\sin(x)-1.23$	16	4.49×10^{-3}
l6-0	$4x^4+3x^3+2x^2+x$	15	0
l6-1	$(3.58x(x+0.57)-\cos x)\left(x^2+0.24x+0.61\right)+0.61$	20	2.91×10^{-6}
l6-2	$x\left[\exp(x)+x+\left(4.27x^2+2.68x-0.06\right)\sin x\right]$	19	7.06×10^{-3}
l6-3	$4.16x^4+3x^3+x^2+x\sin x+x$	18	1.65×10^{-7}
l7-0	$\frac{\exp(x)-\exp(-x)}{2}$	9	0
l7-1	$\frac{\sin(x)}{\cos(0.84x^2)}$	9	1.12×10^{-3}
l7-2	$x^2+\cos(x)-\exp(-x)$	11	5.94×10^{-7}
l7-3	$x+\frac{1}{3}x^2\left(x-\sin\left(\frac{x}{2}\right)\right)$	14	2.42×10^{-7}
l8-0	$\frac{1}{2}\left(\exp(x)+\exp(-x)\right)$	9	0
l8-1	$-1.72+\exp\left(\exp(0.2x\sin x)\right)$	10	5.89×10^{-5}
l8-2	$\frac{3.74}{\exp(\cos(0.91x))+\cos(x+0.01)}$	12	5.04×10^{-5}
l8-3	$\exp(0.51x\sin x)$	7	5.26×10^{-6}
l9-0	$x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x$	25	0
l9-1	$\frac{0.68x}{\exp(x-1.61)+\sin(0.51-x)}$	13	3.83×10^{-3}
l9-2	$\frac{2.11x}{1-0.53x-0.27x^2}-x$	15	6.62×10^{-3}
l9-3	$\frac{0.75x}{0.79+x\sin(\cos(x+2.81))}$	13	4.61×10^{-3}
l13-0	$\exp\left(\frac{1}{3}\log x\right)$	5	0
l13-1	$0.2x+0.9-\frac{0.15\sin x}{x(x+0.21)}$	16	2.91×10^{-4}
l13-2	$0.34\log\left((x+0.13)(x+15.54)\right)$	10	7.02×10^{-5}
l13-3	$\frac{0.61(x-0.01)}{\log(x+1.18)}+0.22$	12	6.62×10^{-5}
l14-0	$x^3+x^2+x+\sin(x^2)+\sin(x)$	17	0
l14-1	$\exp\left(3-4.22\exp(-x)\right)+0.70x-0.29$	15	5.21×10^{-4}
l14-2	$\frac{x}{\cos((x+1.04)/\exp(0.41x))}$	11	6.11×10^{-4}
l14-3	$\frac{x}{\cos(\exp(-0.30-x)+0.15x-1.78)}$	13	5.96×10^{-4}
l15-0	$\exp\left(\frac{1}{5}\log x\right)$	5	0
l15-1	$0.64+0.13\left[\log(9.11x)+\sin(1.64x)\right]$	11	1.59×10^{-4}
l15-2	$\sin\left[0.19\log(x+0.03)\right]+1$	9	9.18×10^{-5}
l15-3	$\log\left[-0.05+\log(5.82+10.10x)\right]$	7	6.21×10^{-4}
l16-0	$\exp\left(\frac{2}{5}\log x\right)$	5	0
l16-1	$0.45-\sin\left(-0.43x-0.22+\frac{0.08}{x+0.15}\right)$	14	1.84×10^{-5}

Continued on next page

Number	Expression	Complexity	Fitting error
l16-2	$0.73 + \sin\left(0.27x + 0.15\log x\right)$	11	1.55×10^{-4}
l16-3	$2.14\cos\left(-1.45 + 0.21x + 0.44x\exp(-x)\right)$	15	3.26×10^{-4}
l18-0	$\sin(x^2)\cos(x) - 5$	9	0
l18-1	$-4.17 - \exp(-0.21 - x^2)$	8	5.06×10^{-4}
l18-2	$\exp\left(0.44x\sin(2.14x)\right) - 6$	11	7.53×10^{-6}
l18-3	$\frac{4.55}{\sin(\cos x - 2.14)}$	7	2.86×10^{-5}
l19-0	$x^5 + x^4 + x^2 + x$	13	0
l19-1	$\frac{1.06x^2 + 0.91x}{\cos(x + 0.06)}$	12	1.64×10^{-3}
l19-2	$6.24x(x + 1.01)(x + 0.16 - \sin x)$	14	3.90×10^{-5}
l19-3	$-2.03 + \frac{0.82x + 1.82}{\cos x - 0.10}$	12	1.93×10^{-3}
l20-0	$\exp(-x^2)$	6	0
l20-1	$\sin\left(1.57 - 0.67x - 0.53\sin(1.40x)\right)$	13	1.84×10^{-6}
l20-2	$0.50 + 0.50\cos(x + \sin x)$	9	2.53×10^{-7}
l20-3	$\exp(2.12\cos x - 2.12)$	7	2.18×10^{-5}
l21-0	$x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$	21	0
l21-1	$\frac{1.24(x+1)\sin(x)}{\cos(1.32x)}$	13	6.05×10^{-3}
l21-2	$\cos(x - 2.3) + 0.61 + \exp\left(4.94(x - 0.6)\right)$	13	3.56×10^{-3}
l21-3	$\frac{x(x+1.21)}{1.27 + x\cos(x+2.14)}$	14	2.61×10^{-3}
l22-0	$\exp(-0.5x^2)$	6	0
l22-1	$\cos\left(x\left(0.96 - \exp(x - 5.17)\right)\right)$	9	5.73×10^{-5}
l22-2	$\exp\left[(0.29x^2 - 0.58x)\sin(x + 0.05)\right] + \sin(x) - x$	18	9.44×10^{-5}
l22-3	$\exp\left[-\sin(0.51x^2)\right]$	7	3.90×10^{-6}

I Detailed Evaluation Results

Table 9 summarizes four evaluation metrics for eight SR algorithms under small (S) and large (L) sampling domains: low-error rate (LER), recovery rate (RR), conditional recovery rate (C-RR), and within-group standard deviation of recovery rates (WSD). The main difficulty under small domains is identifying the correct structure among numerous low-error candidates, whereas under large domains it is reaching low-error regions through a stable search trajectory. Accordingly, LER and C-RR measure numerical search and structural identification, respectively; RR reflects their combined outcome, while WSD measures the balance of search across different structures. Jointly comparing these metrics helps distinguish whether performance is limited by numerical search, structural identification, or structural search bias, providing a more comprehensive view of algorithm behavior across sampling domains.

Table 9: Evaluation results of different SR algorithms under small and large sampling domains.

	PySR		RSRM		DSO		GPlearn		EGGP		iMCTS		TPSR		uDSR	
	S	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**
LER	100.00%	64.41%	98.53%	40.00%	98.53%	47.94%	45.88%	8.24%	100.00%	43.53%	100.00%	66.76%	100.00%	25.88%	100.00%	48.53%
RR	25.29%	51.76%	20.29%	28.82%	44.71%	43.24%	2.65%	8.24%	16.18%	27.35%	47.06%	56.18%	2.94%	3.82%	49.71%	42.65%
C-RR	25.29%	80.37%	20.60%	72.06%	45.37%	90.18%	5.77%	100.00%	16.18%	62.84%	47.06%	84.14%	2.94%	14.77%	49.71%	87.88%
WSD	35.92%	42.10%	41.14%	45.29%	41.35%	39.53%	10.38%	24.01%	29.80%	37.07%	41.49%	42.91%	17.15%	15.66%	45.01%	42.50%