STRIDE: A Self-Reflective Agent Framework for Reliable Automatic Equation Discovery

Anonymous ACL submission

Abstract

LLM-based equation discovery offers a promising route to recovering symbolic laws from data, but many systems still rely on generation-centered loops that propose candidates, fit parameters, score results, and reuse selected examples. Such loops can misjudge useful skeletons under unreliable fitting, discard near-correct equations that require repair, and accumulate redundant memories that provide limited guidance. We propose **STRIDE**, a self-reflective agent framework that improves reliability by coordinating data-aware generation, mixed-fitting evaluation, critic–executor repair, and diversity-preserving semantic memory. By turning fitted scores and candidate behavior into shared feedback, STRIDE enables equations to be proposed, assessed, refined, and reused within a closed-loop discovery process. Experiments on representative symbolic-regression benchmarks and LSR-Synth suites show that STRIDE improves accuracy, OOD robustness, and structural recovery across multiple LLM backbones, with ablations and analyses confirming the contribution of its core components.

1 Introduction

Equation discovery seeks compact, generalizable, and interpretable mathematical laws directly from data (Lemos et al., 2023; Makke and Chawla, 2024; Dong and Zhong, 2025). Symbolic regression (SR) addresses this goal by searching over both equation structures and numerical parameters (Schmidt and Lipson, 2009). Yet the symbolic search space grows combinatorially, and finite observations often admit spurious fits (Virgolin and Pissis, 2022; La Cava et al., 2021; Matsubara et al., 2024). Reliable equation discovery therefore requires not only low training error, but also compact structure and robustness beyond the observed regime.

LLM-based equation discovery has recently emerged as a promising direction, leveraging LLM priors over mathematical syntax and scientific patterns (Shojaee et al., 2025a; Grayeli et al., 2024; Guo et al., 2025; Zhang et al., 2024). A common paradigm organizes discovery as repeated library-conditioned iterations: sample an expression skeleton from an experience library, ask an LLM to instantiate or modify it into a candidate equation, fit and score the candidate, and write selected cases back to the library for later prompting. This process expands the search beyond hand-crafted templates, but its reliability depends on more than generating plausible equations.

As illustrated in Figure 1, this generation-centered paradigm exposes a deeper coordination problem. A single generation step and a score-based library are expected to propose structures, assess their potential after fitting, decide whether imperfect candidates deserve repair, and preserve useful experience for later iterations. As a result, useful skeletons may be discarded when fitting underestimates their potential; near-correct equations may be abandoned rather than locally repaired; and complex but informative hypotheses may be lost under short-term fitness or length bias. These limitations suggest that reliable LLM-based equation discovery should be organized as a multi-role reflective agent workflow, where generation, evaluation, critique, repair, and memory updating are distinct but coordinated roles.

To address these limitations, we propose **STRIDE**, a **S**elf-reflective agen**T** Framework for **R**el**I**able automatic equation **D**iscov**E**ry. STRIDE

Code is available at: https://anonymous.4open.science/r/STRIDE-1DA8.

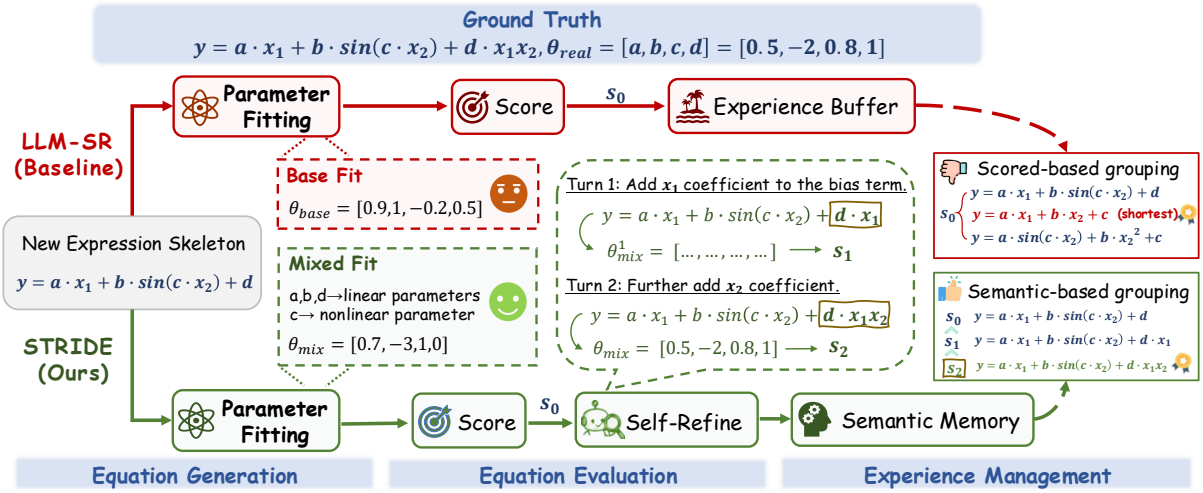


Figure 1: STRIDE versus LLM-SR. The upper part shows a representative LLM-SR iteration that samples a skeleton, generates a candidate, evaluates it, and updates the library. STRIDE adds data-aware sampling, mixed fitting, reflective refinement, and semantic memory to repair promising candidates and discover more reliable equations.

instantiates this idea as a multi-role reflective workflow. **(I)** A generator agent conditions on task context, data hints, and retrieved memory cases to propose candidate skeletons. **(II)** A mixed-fitting evaluator fits free parameters and returns score feedback, giving the agents a more faithful view of skeleton quality. **(III)** A critic agent diagnoses promising but imperfect candidates from fitted-parameter feedback, while an executor agent turns the critic's actions into revised executable equations. **(IV)** A semantic memory module retains diverse high-scoring hypotheses and supplies them as experience for later agent iterations.

The main contributions are as follows:

1. We identify LLM-based equation discovery as a **multi-role self-reflective agent workflow** and introduce **STRIDE**, a coordinated framework for reliable equation discovery.
2. We develop four reliability-oriented mechanisms, namely **data-aware sampling**, **mixed fitting**, **reflective refinement**, and **semantic memory**, to better propose, assess, repair, and retain candidate equations.
3. We provide empirical evidence across benchmarks and LLM backbones, showing that STRIDE improves **reliability** against strong baselines in **ID/OOD** settings, with analyses validating each component.

2 Preliminaries

Symbolic Regression. Given observed data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, symbolic regression seeks an interpretable equation $f(\mathbf{x}; \boldsymbol{\theta})$ that predicts y from $\mathbf{x}$. We formulate this task as a search problem over a symbolic hypothesis space $\mathcal{F}$, where each candidate equation is composed from an operator set $\mathcal{O}$ (e.g., $+, -, \times, \div$, sin, cos, exp and log), input variables $\mathbf{x}$, numerical constants, and free parameters $\boldsymbol{\theta}$. The goal is to identify a symbolic structure that explains the observations accurately while remaining compact, stable, and human-interpretable.

LLM-Based Equation Discovery. Recent LLM-based equation discovery methods, such as LLM-SR (Shojaee et al., 2025a), use LLMs as scientific priors for symbolic regression. The upper part of Figure 1 illustrates one representative iteration rather than the whole system. The step begins by sampling a new expression skeleton from an experience library. The skeleton is combined with the task description, variable information, and operator constraints to prompt an LLM, which instantiates or modifies it into an executable candidate equation or program body. An evaluator then fits the free parameters of this candidate, scores it by prediction error such as normalized mean squared error (NMSE) together with structural preferences, and selectively writes useful cases back to the library.

This paradigm turns equation discovery into a repeated process of library-conditioned sampling, LLM generation, parameter fitting, scoring, and library updating. It provides the functional basis

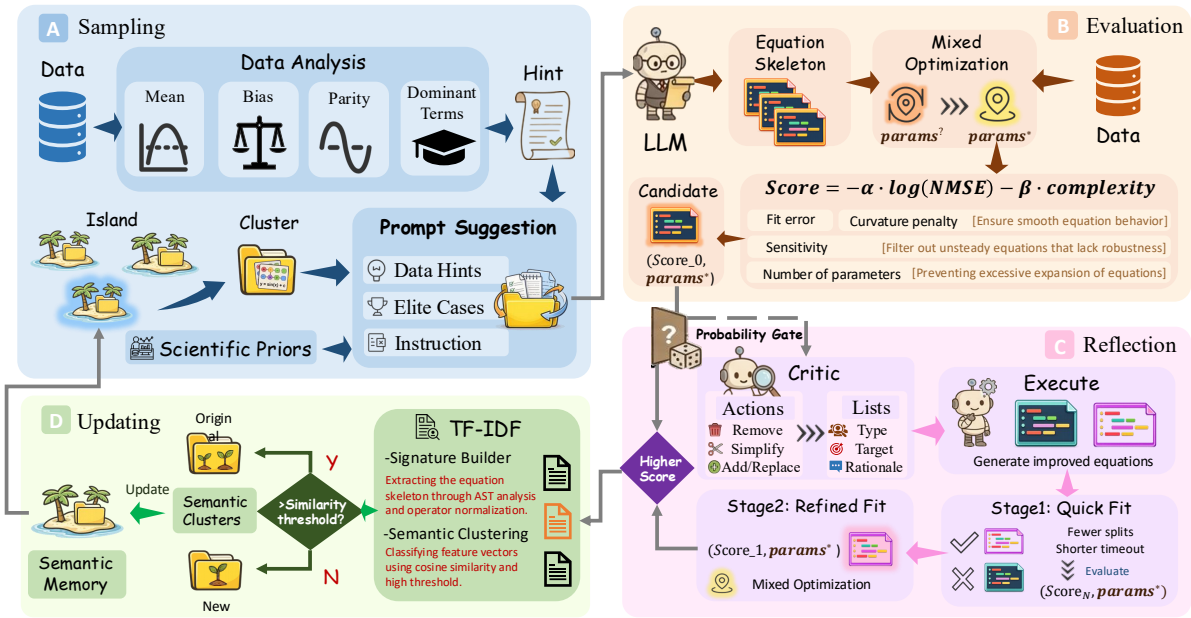


Figure 2: Overall framework of STRIDE: data-aware generation, mixed-fitting evaluation, critic–executor repair, and semantic memory form a closed-loop equation discovery workflow.

for a role-based workflow: a generator proposes candidate equations, an evaluator fits and scores them, and a memory component retains selected experience for future iterations. Its goal is to find a reliable high-scoring equation

$$f^* = \arg\max_{f \in \mathcal{F}} \mathrm{Score}(f, \boldsymbol{\theta}^*; \mathcal{D}), \tag{1}$$

where $\mathcal{F}$ is the candidate equation space defined by the available variables, operators, constants, and expression constraints; f is a candidate symbolic equation; and f^* is the selected best equation under the scoring objective. For each candidate f, $\boldsymbol{\theta}^*$ denotes the parameters fitted on the observed dataset $\mathcal{D}$, and $\mathrm{Score}(f, \boldsymbol{\theta}^*; \mathcal{D})$ measures its quality by combining predictive fidelity with structural preference. The $\arg\max$ operator therefore selects the candidate with the highest score after parameter fitting. This formulation provides the basis for STRIDE, which augments the standard generation, evaluation, and memory loop with critic and executor agents for reflective repair.

3 Method

3.1 Overview

Figures 1 and 2 summarize STRIDE as a reflective workflow that couples data-aware generation, mixed parameter fitting, critic–executor repair, and semantic memory updating. In each iteration, STRIDE proposes parameterized skeletons, fits and scores them, optionally repairs promising candidates, and writes the selected structure back to memory for later sampling.

3.2 Generator Agent: Data-Aware Sampling

Sampling is handled by the generator agent $\mathcal{M}_{\text{gen}}$. STRIDE constructs the sampling prompt from three sources. First, it extracts a data hint h from the training data $\mathcal{D}$ through lightweight data analysis, including mean statistics, bias tendency, parity patterns, and dominant terms from polynomial, interaction, and simple nonlinear feature expansions. Second, it retrieves elite cases $\mathcal{E}$ from the island and cluster structure of semantic memory $\mathcal{B}$. Third, it adds task instructions derived from the task specification $\mathcal{T}$ and variable descriptions.

Let p denote the resulting prompt. Conditioned on p, the generator agent produces a batch of executable equation skeletons

$$\mathcal{C} = \{f_j\}_{j=1}^{b}, \qquad f_j \sim \mathcal{M}_{\text{gen}}(\cdot \mid p). \tag{2}$$

Each skeleton contains free numerical parameters and is passed to the evaluator for fitting and scoring. This stage supplies data-aware structural hypotheses for the subsequent evaluation, repair, and memory-update steps; implementation cases are provided in Appendix B.1. The generator is therefore responsible for structural proposal, while numerical validation is deferred to the evaluator.

3.3 Evaluator: Mixed Parameter Fitting

The evaluator fits each generated skeleton and returns fitted parameters together with a score. Given a candidate equation $f(\mathbf{u};\theta)$, STRIDE parses its skeleton into an abstract syntax tree (AST) (Alfred et al., 2007) and separates the parameters into linear coefficients $\mathbf{w}$ and nonlinear or coupled parameters $\mathbf{q}$ according to their roles in the expression.

The evaluator then performs nested mixed fitting. The outer loop searches over nonlinear parameters $\mathbf{q}$, and the inner loop solves the best linear coefficients $\mathbf{w}^\star(\mathbf{q})$ for each proposed $\mathbf{q}$ and evaluates the resulting normalized error:

$$\begin{aligned} \mathbf{w}^\star(\mathbf{q}) &= \arg\min_{\mathbf{w}} \|f(\mathbf{u};\mathbf{w},\mathbf{q}) - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2, \\ J(\mathbf{q}) &= \frac{1}{N\,\mathrm{Var}(\mathbf{y})}\,\|f(\mathbf{u};\mathbf{w}^\star(\mathbf{q}),\mathbf{q}) - \mathbf{y}\|_2^2\,. \end{aligned} \tag{3}$$

After the outer loop selects $\mathbf{q}^\star$, the evaluator combines it with the corresponding inner-loop solution $\mathbf{w}^\star(\mathbf{q}^\star)$ to form the mixed fit $\theta^\star_{\text{mix}} = (\mathbf{w}^\star(\mathbf{q}^\star), \mathbf{q}^\star)$. This mixed fit is the primary result. STRIDE also runs a generic BFGS optimization (Fletcher, 2013) as a fallback and retains the lower-NMSE solution as the final parameters $\boldsymbol{\theta}^\star$.

The fitted candidate is scored by an accuracy–complexity objective:

$$\begin{aligned} \mathrm{Score}(f,\boldsymbol{\theta}^\star;\mathcal{D}) = W_{\text{fit}}\big[-\log(\mathrm{NMSE}+\epsilon)\big] \\ - W_{\text{comp}}C(f,\boldsymbol{\theta}^\star), \end{aligned} \tag{4}$$

where NMSE measures predictive fidelity and $C(\cdot)$ penalizes structural and behavioral complexity, including effective parameter count, sensitivity, and curvature. The resulting parameters and score are then passed to critic–executor repair and semantic memory updating. This shared feedback lets later decisions depend on fitted behavior rather than the raw generated form. Additional details are provided in Appendix A and B.2.

3.4 Critic–Executor Agents: Reflective Repair

For each evaluated candidate, STRIDE first applies a trigger policy $\mathrm{TriggerCritic}(s)$ to decide whether reflective repair should be activated. The policy selects promising but imperfect candidates according to score positivity and a trigger probability π_c, avoiding reflection budget on clearly poor candidates while retaining stochastic exploration among viable ones. Once triggered, two coordinated agents perform constrained local repair: a critic $\mathcal{M}_{\text{critic}}$ and an executor $\mathcal{M}_{\text{exec}}$. The critic examines the base equation f, fitted parameters, and score feedback, and proposes edits from the action space

$$\mathcal{A} = \{\textsc{Remove}, \textsc{Simplify}, \textsc{Add}\}. \tag{5}$$

The executor then converts these edit instructions into executable refined candidates $\tilde{\mathcal{C}}$.

STRIDE reevaluates the refined candidates in two stages. A lightweight screening pass on a small data subset first retains only a few promising revisions. These finalists are then sent to the full evaluator for mixed fitting and scoring. The best refined candidate is compared with the original equation, and the better one is written back to semantic memory $\mathcal{B}$. Additional implementation details and a case study are provided in Appendix B.3.

3.5 Memory Module: Semantic Updating

After evaluation or reflection, STRIDE inserts the selected equation into a multi-island semantic memory $\mathcal{B}$ (Cranmer, 2023; Romera-Paredes et al., 2024). Before insertion, the equation is canonicalized, encoded with TF-IDF, and compared with existing cases so that structurally related hypotheses are assigned to the same semantic cluster c (Shojaee et al., 2025a). Details of the TF-IDF construction are provided in Appendix A.

Within each cluster c, STRIDE retains only the highest-scoring equation as the elite representative. If a stronger equation enters the cluster, it replaces the current elite. In subsequent iterations, elite cases $\mathcal{E}$ are retrieved from the selected island and fed back into the generator agent $\mathcal{M}_{\text{gen}}$ as in-context exemplars. This keeps memory compact and diverse while feeding evaluated discoveries back into later sampling.

4 Experiments

In this section, we focus on the following questions:

1. Does STRIDE improve equation discovery performance over strong symbolic regression and LLM-based baselines? (Section 4.2)
2. Does STRIDE preserve more reliable structure under distribution shift? (Section 4.2)
3. Which components are most responsible for STRIDE's gains, and what qualitative behaviors do they enable? (Section 4.3)

Table 1: Main results on four symbolic regression benchmarks and four LSR-SYNTH benchmark suites. Acc↑ and NMSE↓ indicate that higher and lower values are better, respectively. The best and second-best results are highlighted in red and blue, respectively.

Method	Oscillator 1			Oscillator 2			E. coli Growth			Stress-Strain		
	$\text{Acc}_{0.1}$↑	$\text{Acc}_{0.001}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{0.001}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{0.001}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{0.001}$↑	NMSE↓
Baselines without LLMs												
GPlearn	10.41	0.12	1.83e-1	17.52	0.36	2.00e-1	0.92	0.00	1.00e0	35.92	0.14	3.68e-1
PySR	97.98	10.29	4.71e-5	99.98	98.53	1.10e-9	10.52	0.08	1.02e-2	83.36	3.74	2.14e-2
DSR	11.19	0.09	9.74e-2	14.98	0.16	1.90e-1	3.52	0.00	6.89e-1	20.53	0.21	3.50e-1
uDSR	12.83	0.11	6.55e-2	16.67	0.15	1.96e-1	2.76	0.00	6.95e-1	7.14	0.00	3.37e-1
GPT-5.1												
LLM-SR	99.99	95.64	1.61e-7	99.49	43.16	1.91e-6	1.24	0.04	8.65e-1	84.19	2.43	1.92e-2
LaSR	94.91	6.92	2.06e-4	32.59	0.27	9.69e-2	2.28	0.04	1.21e-1	75.52	6.31	3.57e-2
SR-LLM	77.87	1.57	2.96e-3	17.48	0.52	2.00e-1	0.92	0.00	8.61e-1	62.83	0.69	7.91e-2
DrSR	99.94	48.44	1.54e-6	84.09	1.12	2.91e-3	1.08	0.04	6.13e-1	77.18	0.69	2.74e-2
STRIDE	100.00	100.00	3.36e-18	100.00	100.00	1.12e-21	10.44	0.16	5.84e-3	88.28	1.25	1.82e-2
Claude-3-Haiku												
LLM-SR	99.77	38.89	2.11e-5	99.99	99.52	6.01e-9	11.64	0.20	1.47e-2	77.05	1.32	2.62e-2
LaSR	96.57	5.37	7.14e-5	94.77	2.94	1.41e-4	2.92	0.00	1.38e-1	80.24	2.57	2.05e-2
SR-LLM	67.23	0.77	7.61e-3	17.48	0.52	2.00e-1	1.24	0.00	3.24e-1	73.86	1.73	4.84e-2
DrSR	99.56	38.76	1.90e-6	75.89	0.75	3.80e-2	1.04	0.00	7.22e-1	66.92	0.49	7.16e-2
STRIDE	100.00	100.00	2.52e-15	100.00	99.99	6.79e-12	14.60	0.20	3.79e-3	85.64	2.43	1.55e-2

Method	CRK			PO			MatSci			BPG		
	$\text{Acc}_{0.1}$↑	$\text{Acc}_{\max,0.1}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{\max,0.1}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{\max,0.1}$↑	NMSE↓	$\text{Acc}_{0.1}$↑	$\text{Acc}_{\max,0.1}$↑	NMSE↓
GPT-5.1												
LLM-SR	90.44	60.00	7.06e-5	75.48	0.00	7.75e-4	99.00	60.00	8.59e-4	81.00	60.00	3.53e-5
SR-Scientist	83.56	40.00	7.76e-5	55.44	0.00	6.36e-3	100.00	100.00	2.14e-3	68.08	20.00	2.71e-4
STRIDE	100.00	100.00	1.83e-9	75.68	0.00	4.17e-4	100.00	100.00	2.14e-3	91.88	80.00	4.16e-7
Claude-3-Haiku												
LLM-SR	91.96	40.00	3.31e-5	65.96	0.00	2.02e-4	100.00	100.00	9.68e-5	74.60	0.00	6.31e-4
SR-Scientist	92.56	80.00	4.75e-5	78.04	0.00	5.28e-4	100.00	100.00	2.13e-3	76.92	20.00	8.17e-5
STRIDE	100.00	100.00	5.35e-9	77.88	0.00	1.72e-3	100.00	100.00	2.18e-3	99.92	80.00	4.82e-9

4.1 Setup

Benchmarks. We evaluate STRIDE on two benchmark groups: four representative LLM-SR tasks (Shojaee et al., 2025a) and four LSR-SYNTH suites from LLM-SRBENCH (Shojaee et al., 2025b); detailed equations are listed in Appendix D. For LSR-SYNTH, we use the top five tasks in each suite according to the difficulty ranking. All benchmarks are evaluated under both in-domain (ID) and out-of-domain (OOD) settings.

Baselines. We compare against traditional symbolic regression baselines, including GPLEARN (Stephens, 2022), PYSR (Cranmer, 2023), DSR (Petersen et al., 2019), and UDSR (Landajuela et al., 2022), as well as recent LLM-based methods: LLM-SR (Shojaee et al., 2025a), LASR (Grayeli et al., 2024), SR-LLM (Guo et al., 2025), SR-SCIENTIST (Xia et al., 2026), and DRSR (Wang et al., 2025). Additional baseline details are deferred to Appendix C.

Metrics. We report both fitting error and tolerance-based accuracy. Specifically, we use NMSE to measure predictive fidelity while normalizing for target scale, together with pointwise tolerance accuracy $\text{ACC}@\tau$ and the stricter split-level accuracy $\text{ACC}_{\max}@\tau$:

$$\text{NMSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\text{Var}(\mathbf{y})},$$

$$\text{ACC}@\tau = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\frac{|\hat{y}_i - y_i|}{|y_i| + \epsilon} \leq \tau\right), \quad (6)$$

$$\text{ACC}_{\max}@\tau = \mathbb{I}\left(\max_{1\leq i\leq N}\frac{|\hat{y}_i - y_i|}{|y_i| + \epsilon} \leq \tau\right).$$

Configuration. We align the search budget across LLM-based methods and evaluate GPT-5.1 (OpenAI, 2025) and CLAUDE-3-HAIKU (Anthropic, 2024) with the same prompt template and decoding temperature. STRIDE generates at most 2000 candidate expressions per dataset, and GPT-5.1 is used as the default backbone unless otherwise specified. Full hyperparameter settings are provided in Appendix C and Appendix A.

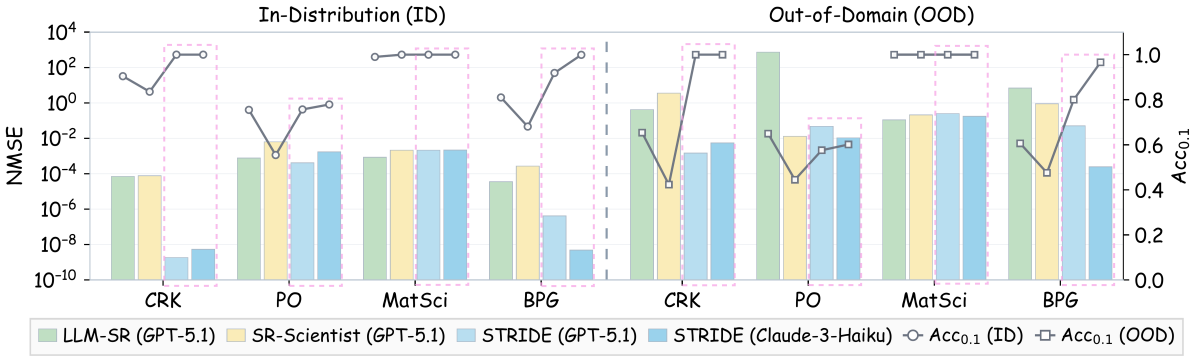


Figure 3: ID and OOD performance on the LSR-Synth suites. Bars report NMSE↓ and line markers report $\mathrm{Acc}_{0.1}$↑ for baselines and STRIDE. Complete OOD results for all benchmarks and LLMs are provided in Appendix B.5.

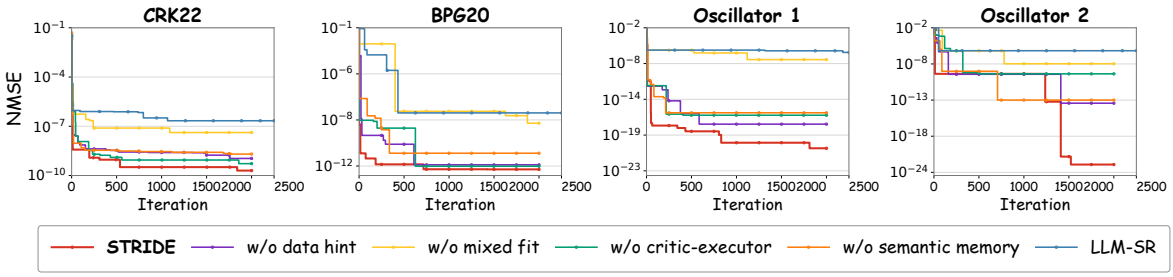


Figure 4: Iteration analysis of STRIDE and its ablated variants. The full system converges more consistently than variants with individual modules removed.

4.2 Main Results

STRIDE Discovers More Reliable Equations Than Strong Baselines. Table 1 answers the first question affirmatively: STRIDE consistently improves equation discovery over strong symbolic regression and LLM-based baselines. Relative to LLM-SR, STRIDE improves average $\mathrm{Acc}_{0.1}$ by 3.45% and 2.95% on the four main benchmarks under GPT-5.1 and Claude-3-Haiku, respectively; on LSR-Synth, the gains further increase to 5.16% and 11.32%. The stricter $\mathrm{Acc}_{\max,0.1}$ results reinforce this conclusion: STRIDE achieves the best or tied-best score in nearly all LSR-Synth domains under both backbones, indicating that its accuracy does not come from fitting only a subset of points but from discovering equations that remain closer to the ground-truth structure as a whole.

STRIDE Preserves More Reliable Structure Under Distribution Shift. Figure 3 answers the second question by showing that STRIDE achieves the highest accuracy across all domains under distribution shift, with consistently stronger OOD $\mathrm{Acc}_{0.1}$ and lower or competitive NMSE. This indicates that STRIDE is not merely fitting the training region more closely; it is more often recovering equations whose structure transfers to unseen regimes, reducing the risk that the improvement comes from overfitting. The effect is especially clear on *CRK* and *BPG*, where STRIDE maintains strong OOD accuracy while competing methods drift after leaving the observed region.

4.3 Ablation Analysis

Each STRIDE Component Contributes to Reliability. Table 2 and Figure 4 show that each module contributes to STRIDE's final performance and search dynamics. Removing data hints, mixed fitting, the critic–executor, or semantic memory degrades at least one key metric on the OOD oscillator tasks, with mixed fitting producing the largest drop in NMSE and strict accuracy. Even with individual modules removed, STRIDE variants generally remain stronger than LLM-SR. This confirms that reliables parameter feedback is central, but not sufficient by itself. The iteration curves further show that the full system converges more consistently than its ablated variants. Overall, the modules contribute complementary gains to the closed-loop search.

Table 2: Ablation results on the OOD oscillator tasks. "w/o data hints" uses only the problem description and exemplar equations during sampling, without data-statistical hints; "w/o mixed fit" uses only BFGS for parameter fitting in the evaluation stage; "w/o critic–executor" removes the reflection stage; and "w/o semantic memory" replaces semantic clustering with score-based experience management.

Method	NMSE↓	$Acc_{0.1}$↑	$Acc_{0.001}$↑
Ours (Oscillator 1)	**5.97e-12**	**100.00**	**100.00**
w/o data hints	5.64e-10	100.00	99.99
w/o mixed fit	4.66e-6	100.00	69.61
w/o critic–executor	4.92e-8	100.00	89.75
w/o semantic memory	9.47e-8	100.00	84.87
LLM-SR (none)	2.85e-4	99.97	41.21
Ours (Oscillator 2)	**4.48e-21**	**100.00**	**100.00**
w/o data hints	4.76e-9	100.00	99.99
w/o mixed fit	1.85e-4	96.01	58.67
w/o critic–executor	1.40e-6	100.00	94.17
w/o semantic memory	4.61e-9	100.00	100.00
LLM-SR (none)	3.05e-3	86.28	19.33

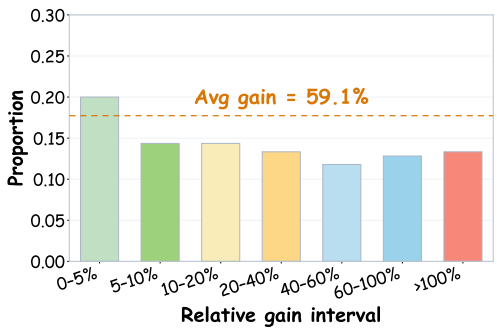


Figure 5: Reflection gains from critic–executor repair. Reflection often improves promising candidates, showing the benefit of targeted local revision.

4.4 Further Analysis

Reflective Repair Converts Feedback Into Local Improvements. Figure 5 shows that critic–executor repair improves many reflected candidates, indicating that reflection acts as targeted repair rather than additional unconstrained generation. This comes from conditioning the critic on fitted parameters, score feedback, and task context, so that edits are grounded in the evaluated behavior of the current equation. Ablations in Table 2 provide supporting evidence: removing the critic–executor reduces final precision on the oscillator tasks. Overall, STRIDE turns evaluator feedback into local symbolic improvements.

Semantic Memory Helps Recover Clear Complex Structures. STRIDE is especially useful when equations contain clear but nontrivial symbolic mechanisms, such as the oscillatory, *CRK*, and *MatSci* cases studied here. Figure 6 provides a memory-level explanation: the inner ring summarizes memory clusters by size, while the outer ring marks how many clusters in each group contain high-score equations. Compared with score-based retention, semantic memory keeps high-score equations in more singleton and small clusters, indicating that useful hypotheses are structurally diverse rather than concentrated in redundant groups. This diversity gives later prompts access to distinct high-quality mechanisms, improving search over clear and interpretable equation structures.

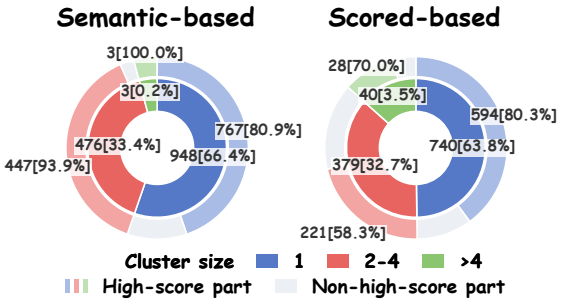


Figure 6: High-score semantic versus score-based memory grouping on *Oscillator 1*. The inner ring shows the distribution of all clusters by size, and the outer ring shows the number and proportion of clusters in each size group that contain high-score equations. High-score candidates are defined as $Acc_{0.1} > 0.9$.

Table 3: Estimated per-problem cost and relative OOD performance of LLM-based methods under our evaluation setting. Costs are approximate; relative metrics are normalized to the lowest method value.

Method	Tokens	Time	Rel. $Acc_{0.1}$ ↑	Rel. NMSE ↓
LLM-SR	∼2.6M	∼6.0h	2.49×	7×
LaSR	∼1.3M	∼2.8h	1.68×	76×
SR-LLM	∼0.7M	∼5.5h	1.00×	6489×
DrSR	∼3.8M	∼7.2h	1.86×	118×
STRIDE	∼3.5M	∼6.7h	**3.26**×	**1**×

STRIDE Balances Cost and Search Efficiency. STRIDE introduces extra evaluator and reflection calls, but uses them to make search more productive. Figure 4 shows that feedback from fitting, repair, and memory helps the system reach strong candidates early, while Figure 10 shows that mixed fitting improves parameter estimation with manageable overhead. As summarized in Table 3, STRIDE achieves substantially stronger OOD performance without requiring markedly higher token or runtime budgets than comparable LLM-based baselines.

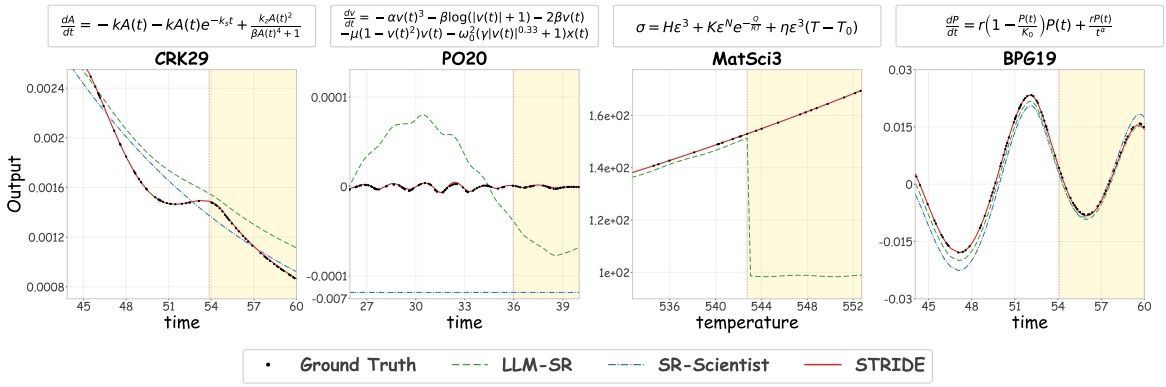


Figure 7: Cross-domain LSR-Synth case analysis. Shaded regions denote OOD intervals. STRIDE more closely tracks the ground truth across CRK19, PO20, MatSci3, and BPG19, indicating more reliable structural recovery.

Case Study. Figure 7 connects memory-level diversity to equation-level behavior. Across CRK19, PO20, MatSci3, and BPG19, STRIDE more closely follows the ground truth in OOD regions, suggesting that it preserves useful structural alternatives rather than only improving local fit. Even for coupled nonlinear components such as $x\exp(x)$, STRIDE better retains the dominant mechanisms that control extrapolation, yielding smoother transitions and smaller post-shift drift. Additional examples are provided in Appendix E.

5 Related Work

Symbolic Regression and LLM-based Equation Discovery. Symbolic regression seeks interpretable equations that explain observed data. Classical methods search over expression trees with genetic programming, multi-island evolution, decomposition, or physics-informed structure search (Koza, 1992; Cranmer, 2023; Udrescu and Tegmark, 2020; Sun et al., 2023). These systems remain strong foundations, but their success still depends on balancing symbolic structure search with reliable parameter fitting, since poor coefficient estimation can make useful structures appear unpromising (Moré, 1978; Nocedal and Wright, 2006; Negishi et al., 2024; Zeng et al., 2025). Recent LLM-based methods use language models as priors over symbolic form, executable programs, and scientific patterns. Some frame discovery as in-context function induction, automatic equation/-model discovery, or library-conditioned program search, where generated candidates are fitted and stored for later prompts (Merler et al., 2024; Du et al., 2024; Li et al., 2024; Shojaee et al., 2025a). Others add concept abstraction, retrieval, data-aware reasoning, or physics-guided agent steps to improve candidate proposal and search direction (Grayeli et al., 2024; Guo et al., 2025; Wang et al., 2025; Xia et al., 2026; Yang et al., 2026). Our work follows this LLM-guided line, but emphasizes fitting-aware feedback, targeted repair, and diversity-preserving memory as coordinated parts of the discovery loop.

Agentic Scientific Discovery. LLM agents have increasingly been used to coordinate tools, memory, and reflection for scientific reasoning and discovery. Tool-augmented, autonomous, and survey work covers scientific problem solving, symbolic solver use, self-improvement, chemistry tool use, and broader research workflows (Ma et al., 2024; Pan et al., 2023; Huang et al., 2023; Bran et al., 2023; Lu et al., 2024; Schmidgall et al., 2025; Zheng et al., 2025). Related agent studies also explore reflection learning, critic guidance, adaptive search, automated paper auditing, and reusable skill memories for long-horizon tasks (Gupta et al., 2024; Xiang et al., 2024; Sun et al., 2026; Tu et al., 2026a,b). These works motivate decomposing complex tasks into tool use, search, critique, and memory; STRIDE grounds this coordination in fitted symbolic candidates for equation discovery.

6 Conclusion

We present **STRIDE**, a self-reflective agent framework for reliable LLM-based equation discovery. STRIDE coordinates data-aware generation, mixed-fitting feedback, critic–executor repair, and semantic memory to move beyond generation-only search. Experiments across multi-domain benchmarks and LLM backbones show consistent gains in accuracy, OOD robustness, and structural recovery, highlighting the value of feedback-driven agent coordination for automatic scientific modeling.

Limitations

We summarize key practical limitations:

- **Parameter-role identification.** STRIDE uses heuristic AST-based analysis to separate linear and nonlinear parameter roles. This may be less stable for deeply coupled expressions, although the BFGS fallback helps maintain robustness.
- **Semantic equivalence.** Semantic memory relies on canonicalization and surface-form similarity. Equivalent equations written in different algebraic forms may therefore lead to imperfect clustering or redundant entries.
- **Real-world deployment.** Our experiments use benchmark datasets with controlled ID/OOD splits. Applying STRIDE to noisier real-world scientific data may require additional uncertainty estimation and domain validation.

Ethics Statement

This work studies automatic equation discovery on benchmark scientific datasets and does not involve human subjects, private data, or personally identifiable information. We do not anticipate direct negative societal impacts. Discovered equations should nevertheless be independently validated before use in scientific or engineering decision-making.

References

V Alfred, S Monica, Sethi Ravi, Ullman Jeffrey D, and 1 others. 2007. *Compilers Principles, Techniques*. Pearson.

Anthropic. 2024. Claude 3 haiku: Our fastest model yet. https://www.anthropic.com/news/claude-3-haiku. Accessed: 2026-04-17.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2023. ChemCrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*.

Mary Ann Branch, Thomas F. Coleman, and Yuying Li. 1999. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23.

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

Jiayuan Dong and Jianye Zhong. 2025. Recent advances in symbolic regression. *ACM Computing Surveys*, 57(11):1–37.

Mengge Du, Yuntian Chen, Zhongzheng Wang, Longfeng Nie, and Dongxiao Zhang. 2024. LLM4ED: Large language models for automatic equation discovery. *arXiv preprint arXiv:2405.07761*.

Roger Fletcher. 2013. *Practical methods of optimization*. John Wiley & Sons.

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

Zelin Guo, Siqi Wang, Yonglin Tian, Jing Yang, Hui Yu, Xiaoxiang Na, Levente Kovács, Li Li, Petros A. Ioannou, and Fei-Yue Wang. 2025. Sr-llm: An incremental symbolic regression framework driven by llm-based retrieval-augmented generation. *Proceedings of the National Academy of Sciences*, 122(52):e2516995122.

Priyanshu Gupta, Shashank Kirtania, Ananya Singha, Sumit Gulwani, Arjun Radhakrishna, Gustavo Soares, and Sherry Shi. 2024. MetaReflection: Learning instructions for language agents using past reflections. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8369–8385, Miami, Florida, USA. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068.

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

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. 2021. Contemporary symbolic regression methods and their relative performance. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. NeurIPS Datasets and Benchmarks.

Mikel Landajuela, Chak Lee, Jiachen Yang, Ruben Glatt, Claudio P. Santiago, Ignacio Aravena, Terrell N. Mundhenk, Garrett Mulcahy, and Brenden K. Petersen. 2022. A unified framework for deep symbolic regression. volume 35, pages 33985–33998.

Pablo Lemos, Niall Jeffrey, Miles Cranmer, Shirley Ho, and Peter Battaglia. 2023. Rediscovering orbital mechanics with machine learning. *Machine Learning: Science and Technology*, 4(4):045002.

Michael Y. Li, Emily B. Fox, and Noah D. Goodman. 2024. Automated statistical model discovery with language models. *arXiv preprint arXiv:2402.17879*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, and Aixin Sun. 2024. SciAgent: Tool-augmented language models for scientific reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15701–15736, Miami, Florida, USA. Association for Computational Linguistics.

Nour Makke and Sanjay Chawla. 2024. Interpretable scientific discovery with symbolic regression: A review. *Artificial Intelligence Review*, 57(1):2.

Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. 2024. Rethinking symbolic regression datasets and benchmarks for scientific discovery. In *Journal of Data-centric Machine Learning Research*.

Matteo Merler, Katsiaryna Haitsiukevich, Nicola Dainese, and Pekka Marttinen. 2024. In-context symbolic regression: Leveraging large language models for function discovery. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 445–462.

Jorge J. Moré. 1978. The levenberg-marquardt algorithm: Implementation and theory. In *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*, pages 105–116.

Masahiro Negishi, Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. 2024. Two-stage coefficient estimation in symbolic regression for scientific discovery. In *Machine Learning and the Physical Sciences Workshop @ NeurIPS 2024*.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization*. Springer, New York, NY.

OpenAI. 2025. Gpt-5.1 model. `https://platform.openai.com/docs/models/gpt-5.1`. OpenAI API documentation, accessed 2026-03-28.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824.

Brenden K. Petersen, Mikel Landajuela, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne T. Kim. 2019. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*.

Juan Ramos and 1 others. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. New Jersey, USA.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, Alhussein Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

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

Hinrich Sch"utze, Christopher D. Manning, and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*, volume 39. Cambridge University Press, Cambridge.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. 2025a. LLM-SR: Scientific equation discovery via programming with large language models. In *The Thirteenth International Conference on Learning Representations*. Oral.

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D. Doan, and Chandan K. Reddy. 2025b. Llm-srbench: A new benchmark for scientific equation discovery with large language models. In *International Conference on Machine Learning*, pages 55325–55359.

Trevor Stephens. 2022. gplearn: Genetic programming in python, with a scikit-learn inspired api. `https://github.com/trevorstephens/gplearn`. GitHub repository, accessed 2026-03-28.

Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. 2023. Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *International Conference on Learning Representations*.

Jingbo Sun, Wenyue Chong, Songjun Tu, Qichao Zhang, Yaocheng Zhang, Jiajun Chai, Xiaohan Wang, Wei Lin, Guojun Yin, and Dongbin Zhao. 2026. AutoSearch: Adaptive search depth for efficient agentic rag via reinforcement learning. *arXiv preprint arXiv:2604.17337*.

Songjun Tu, Yiwen Ma, Jiahao Lin, Qichao Zhang, Xiangyuan Lan, Junfeng Li, Nan Xu, Linjing Li, and Dongbin Zhao. 2026a. PaperAudit-Bench: Benchmarking error detection in research papers for critical automated peer review. *arXiv preprint arXiv:2601.19916*.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. 2026b. Dynamic dual-granularity skill bank for agentic rl. *arXiv preprint arXiv:2603.28716*.

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

Marco Virgolin and Solon P. Pissis. 2022. Symbolic regression is np-hard. *Transactions on Machine Learning Research*.

Runxiang Wang, Boxiao Wang, Kai Li, Yifan Zhang, and Jian Cheng. 2025. Drsr: Llm based scientific equation discovery with dual reasoning from data and experience. *arXiv preprint arXiv:2506.04282*.

Shijie Xia, Yuhan Sun, and Pengfei Liu. 2026. Sr-scientist: Scientific equation discovery with agentic ai. In *International Conference on Learning Representations*.

Yufei Xiang, Yiqun Shen, Yeqin Zhang, and Cam-Tu Nguyen. 2024. Retrospex: Language agent meets offline reinforcement learning critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4650–4666, Miami, Florida, USA. Association for Computational Linguistics.

Jianke Yang, Ohm Venkatachalam, Mohammad Kianezhad, Sharvaree Vadgama, and Rose Yu. 2026. Think like a scientist: Physics-guided llm agent for equation discovery. *arXiv preprint arXiv:2602.12259*.

Zeyu Zeng, Xicheng Peng, Mao Chen, and Sannyuya Liu. 2025. A two-stage symbolic regression method for discovering mathematical formulas. *Applied Soft Computing*, page 114081.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. 2024. A comprehensive survey of scientific large language models and their applications in scientific discovery. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8783–8817, Miami, Florida, USA. Association for Computational Linguistics.

Tianshi Zheng, Zheye Deng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Zihao Wang, and Yangqiu Song. 2025. From automation to autonomy: A survey on large language models in scientific discovery. *arXiv preprint arXiv:2505.13259*.

A Details of Our Inference Framework

Algorithm 1 summarizes the full inference loop of STRIDE, including data-aware sampling, mixed-fitting-based evaluation, critic–executor reflection, and semantic-memory updating.

We run our framework with a maximum budget of 2000 candidate equations per dataset. At each iteration, STRIDE constructs the in-context experience examples from a 10-island semantic memory buffer. Following the multi-island sampling strategy used in LLM-guided program search (Romera-Paredes et al., 2024; Shojaee et al., 2025a), we first randomly select one island from the available islands. Within the selected island, equations are organized into semantic clusters maintained during memory updating. The highest-scoring equation in each cluster is regarded as the elite case, and its score is used as the cluster signature. Clusters with higher scores are sampled with larger probability. Let s_i denote the signature score of cluster c_i. The probability of choosing cluster c_i is

$$P_i = \frac{\exp(s_i/\tau_c)}{\sum_j \exp(s_j/\tau_c)}, \tag{7}$$

where $\tau_c = 0.1$ is the cluster-sampling temperature. After a cluster is selected, STRIDE directly uses the highest-scoring equation in that cluster as the retrieved exemplar, rather than further preferring shorter programs inside the cluster. At the beginning of each iteration, STRIDE builds the Generator Agent's contextual prompt from the task specification and 2 elite exemplars retrieved from the semantic memory; every 25 iterations, it additionally injects a data hint that compactly summarizes the training split, including scale information, symmetry cues, bias-term suggestions, and stable dominant terms selected by sparse regression. The Generator Agent then uses this prompt to generate 4 candidate equation skeletons.

After the generator proposes candidate skeletons, STRIDE enters the evaluation stage, where the mixed-fitting evaluator first fits free parameters and then computes score feedback for candidate assessment. To improve parameter identification and avoid underestimating promising skeletons, we adopt a mixed fitting strategy. AST analysis classifies parameters as nonlinear when they appear inside nonlinear functions, as exponents, or in products with other parameter-dependent terms; the remaining active parameters are treated as linear. Given a nonlinear parameter vector $\mathbf{q}$, we compute

Algorithm 1 STRIDE

Input: LLM $\mathcal{M}$, dataset $\mathcal{D}$, task specification $\mathcal{T}$, iterations N, semantic memory buffer $\mathcal{B}$
Output: best equation f^*, best score s^*

$h \leftarrow$ BuildDataHint$(\mathcal{D})$
$f^*, s^* \leftarrow \varnothing, -\infty$
for $i = 1$ to N **do**
  # A. Sampling
  Sample elite cases $\mathcal{E}$ from memory $\mathcal{B}$
  $p \leftarrow$ GeneratePrompt$(\mathcal{E}, \mathcal{T}, h)$
  $\mathcal{C} \leftarrow \{f_j\}_{j=1}^{b},\ f_j \sim \mathcal{M}_{\text{gen}}(\cdot \mid p)$
  for $f \in \mathcal{C}$ **do**
  # B. Evaluation
    $\theta^* \leftarrow$ MixedOptimize$(f, \mathcal{D})$
    $s \leftarrow \text{Score}(f, \theta^*; \mathcal{D})$
  # C. Reflection
    if TriggerCritic(s) **then**
      $critic \leftarrow \mathcal{M}_{\text{critic}}(\cdot \mid f, \theta^*, \mathcal{D})$
      $\tilde{\mathcal{C}} \leftarrow \mathcal{M}_{\text{exec}}(\cdot \mid f, critic)$
      for $\tilde{f} \in \tilde{\mathcal{C}}$ **do**
        $\tilde{\theta} \leftarrow$ MixedOptimize$(\tilde{f}, \mathcal{D})$
        $\tilde{s} \leftarrow \text{Score}(\tilde{f}, \tilde{\theta}; \mathcal{D})$
      end for
      $(f, s) \leftarrow \text{BestOf}((f, s), \{(\tilde{f}, \tilde{s})\})$
    end if
  # D. Updating
    $c \leftarrow$ cluster ID of f in $\mathcal{B}$ by TF-IDF similarity; see Eqs. (12) and (13).
    $\mathcal{B} \leftarrow$ SemanticInsert$(\mathcal{B}, c, (f, s))$
    if $s > s^*$ **then**
      $f^*, s^* \leftarrow f, s$
    end if
  end for
end for
return f^*, s^*

a bias term $\mathbf{b}(\mathbf{q})$ by setting all linear coefficients to zero. For each linear coefficient $\mathbf{w}_k$, we then construct a unit probe

$$\phi_k(\mathbf{q}) = f(\mathbf{u}; \mathbf{w}_k = 1,\ \mathbf{w}_{j\neq k} = 0,\ \mathbf{q}) - \mathbf{b}(\mathbf{q}). \tag{8}$$

Stacking these probes yields the design matrix $\Phi(\mathbf{q})$. The conditional linear coefficients are solved by ridge least squares with coefficient 10^{-10}:

$$\mathbf{w}^\star(\mathbf{q}) = \arg\min_{\mathbf{w}} \left\|\Phi(\mathbf{q})\mathbf{w} - \big(\mathbf{y} - \mathbf{b}(\mathbf{q})\big)\right\|_2^2 + \lambda\|\mathbf{w}\|_2^2. \tag{9}$$

The nonlinear parameters are optimized by 4-start Powell search over the reduced objective

$$J(\mathbf{q}) = \frac{1}{\mathrm{Var}(\mathbf{y})} \cdot \frac{1}{N} \left\|f\big(\mathbf{u}; \mathbf{w}^\star(\mathbf{q}), \mathbf{q}\big) - \mathbf{y}\right\|_2^2. \tag{10}$$

The resulting mixed solution is compared against a multi-start BFGS fallback, and the lower-NMSE solution is retained as the fitted candidate.

Given the fitted parameters returned by the evaluator, STRIDE computes the score feedback accord-

ing to Equation 4, using $W_{\text{fit}} = 0.7$, $W_{\text{comp}} = 0.3$, and $\epsilon = 10^{-12}$. The complexity term is defined as

$$C = K_{\text{param}} n_{\text{eff}} + K_{\text{sens}} C_{\text{sens}} + K_{\text{curv}} \log(1 + C_{\text{curv}}), \tag{11}$$

where n_{eff} is the number of numerically active parameters, and C_{sens} and C_{curv} penalize excessive sensitivity and curvature. We set $K_{\text{param}} = 1.0$, $K_{\text{sens}} = 0.05$, and $K_{\text{curv}} = 0.01$. The effective parameter count is estimated by perturbing each parameter and checking whether the output changes above a small relative threshold. Sensitivity and curvature are computed on at most 300 training points using first- and second-order finite differences.

The critic–executor reflection stage is activated by $\text{TriggerCritic}(s)$, which triggers reflection when the score feedback is above the threshold 0 and a Bernoulli trial with probability $\pi_c = 0.4$ succeeds. For each selected base candidate, the critic agent $\mathcal{M}_{\text{critic}}$ receives the candidate code, fitted parameters, score feedback, input variables, and task description, and proposes constrained edits from the action space $\{\text{REMOVE}, \text{SIMPLIFY}, \text{ADD}\}$. Each action specifies the target component or proposed replacement, so that the executor agent can make targeted revisions according to the critic's request.

Valid critic actions are used to construct a constrained executor prompt, from which the executor agent $\mathcal{M}_{\text{exec}}$ proposes up to 4 revised equation skeletons. These revisions are first screened with the same mixed-fitting scheme under a reduced budget, using one split, one restart, a 40-second timeout, and no critic calls. The highest-scoring revision, together with the runner-up when its relative score gap is below 0.01, is then evaluated by the full mixed-fitting pipeline, compared against the base equation by final score, and the winner is passed to semantic memory.

In the updating stage, the equation stored in semantic memory comes from either a candidate selected during evaluation or the winning equation after triggered reflection. Before insertion, STRIDE canonicalizes the equation and represents it with term frequency–inverse document frequency (TF-IDF) features over symbolic tokens, including operators, variables, and functions (Salton and Buckley, 1988; Ramos et al., 2003). For a token t in equation f, the weight is defined as

$$\text{tfidf}(t, f) = \text{tf}(t, f) \cdot \log \frac{N}{\text{df}(t)}, \tag{12}$$

where $\text{tf}(t, f)$ denotes the term frequency of t in f, $\text{df}(t)$ is the number of equations containing t, and N is the total number of equations in the buffer. The similarity between two equations f_i and f_j is then computed using cosine similarity (Sch"utze et al., 2008):

$$\text{sim}(f_i, f_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \, \|\mathbf{v}_j\|}, \tag{13}$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are their TF-IDF representations. Equations with cosine similarity above 0.9 are assigned to the same semantic cluster; otherwise, a new cluster is created. This is similar in spirit to diversity-preserving mechanisms in evolutionary symbolic regression (Schmidt and Lipson, 2009; Udrescu and Tegmark, 2020). This update rule preserves a representative elite case for each structural region while reducing redundant memory entries for later generator prompts.

B Extended Experimental Results

B.1 Effect of Data Hints

We first investigate the effect of data-driven hints in guiding equation discovery. The data hint module extracts lightweight statistical and structural signals from the dataset, including distribution statistics, symmetry properties, and dominant functional patterns.

As shown in Figure 8, the extracted hints capture key characteristics of the underlying system. For example, in the oscillator datasets, the hints correctly identify symmetry properties (e.g., odd-/even behavior in x and v) and highlight dominant functional components such as $\sin(x)$, $\cos(x)x$, and interaction terms like xv. These signals provide useful inductive bias for the LLM, effectively narrowing the search space while preserving flexibility.

Importantly, the hints are incorporated as soft guidance rather than hard constraints. This allows the model to balance prior knowledge and exploration, avoiding premature convergence to overly restricted structures. Empirically, we observe that data hints significantly improve early-stage sampling quality and stabilize the search process, leading to faster convergence and better final solutions, as reflected by the iteration curves in Figure 4. These results demonstrate that data-aware guidance is an effective complement to LLM-based equation discovery.

```
[Dataset loaded]
path: data\oscillator1\train.csv
X shape: (10000, 2), y shape: (10000,)
variables: ['x', 'v']
```

Oscillator1

```
[DATA_HINT]
y_mean=0.01433, y_std=0.06605, y_p01=-0.1045, y_p99=0.1104
bias_term_suggested=True
symmetry=x:odd_like (odd_err=0.0711, even_err=2); v:even_like (odd_err=1.98, even_err=0.556)
dominant_terms=cos(x)*x, sin(x), x*v, x**3,
constraint=use_only_signature_vars(x, v, params, np); do_not_invent_external_vars
format=require_return_statement;
output must be executable function body.
```

```
[Dataset loaded]
path: data\oscillator2\train.csv
X shape: (10000, 3), y shape: (10000,)
variables: ['t', 'x', 'v']
```

Oscillator2

```
[DATA_HINT]
y_mean=0.007096, y_std=0.405, y_p01=-0.7034, y_p99=0.5993
bias_term_suggested=False
symmetry=x:odd_like (odd_err=0.251, even_err=1.99); v:odd_like (odd_err=1.8, even_err=1.86)
dominant_terms=sin(t), x, exp(x), sin(x), cos(x)*x
constraint=use_only_signature_vars(t, x, v, params, np); do_not_invent_external_vars
format=require_return_statement; output must be executable function body
```

Figure 8: Examples of data hints extracted from training data. The hints summarize distribution statistics, symmetry patterns, dominant terms, and variable constraints.

B.2 Effect of Mixed Parameter Fitting

We further analyze the role of parameter fitting by comparing different fitting strategies, including Gauss–Newton (GN), Levenberg–Marquardt (LM), trust-region methods (TRF), BFGS, and our mixed fitting scheme (Moré, 1978; Nocedal and Wright, 2006; Branch et al., 1999; Fletcher, 2013).

Conventional nonlinear fitting methods treat candidate parameters as a coupled optimization vector. GN and LM are often stable for least-squares problems but may fail to achieve high precision when the generated skeleton contains strong nonlinear coupling, while TRF and BFGS can still suffer from sensitivity to initialization or ill-conditioned search landscapes. These limitations make parameter fitting unreliable for complex LLM-generated expressions, and a poor fit can cause a useful skeleton to be underestimated.

As illustrated in Figure 9, our mixed fitting method avoids treating all internal parameters as a black box. It first detects parameter roles through AST-based analysis and decomposes them into linear coefficients and nonlinear parameters. Under this decomposition, linear coefficients are fitted efficiently by least squares, while nonlinear parameters are fitted through an outer-loop search. This enables the evaluator to use parameter-type information more directly, improving equation precision when the decomposition is effective.

The quantitative comparison in Figure 10 shows that the mixed fitting strategy achieves a superior balance between accuracy and computational cost. This structure-aware fitting strategy gives reliable parameter estimates across diverse equation forms. Importantly, improved parameter fitting directly reduces the risk of discarding structurally correct equation skeletons due to suboptimal coefficient estimation. This highlights that reliable parameter fitting is essential for preserving promising candidates during the search process. This mixed fitting design is closely related to recent efforts that explicitly separate coefficient estimation from structure discovery (Negishi et al., 2024; Zeng et al., 2025).

Overall, mixed fitting strengthens the evaluator as a search signal: candidates are judged by a better-fitted version of their skeletons, so later memory updates and reflection steps are guided by more faithful evidence.

B.3 Effect of Critic–Executor Reflection

We use a critic–executor reflection stage to improve promising equations through constrained symbolic

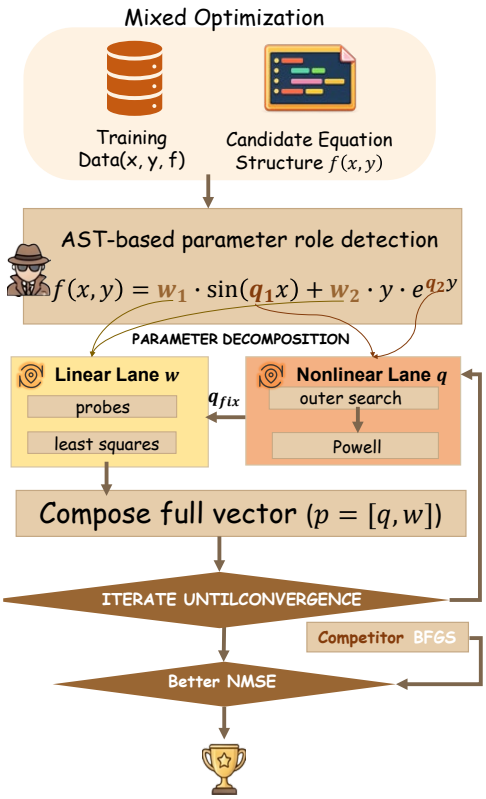


Figure 9: Illustration of the proposed mixed parameter fitting pipeline. Parameters are first decomposed into linear coefficients and nonlinear parameters via AST-based role detection. Linear coefficients are solved with least squares, while nonlinear parameters are fitted by multi-start Powell search over the reduced objective. When the decomposition is unreliable for overly complex expressions, BFGS fitting is used as a fallback.

edits. Given an evaluated candidate with fitted parameters, the critic agent analyzes its structure and fit quality, then proposes local edits from the action space {REMOVE, SIMPLIFY, ADD}.

Figure 11 illustrates the critic–executor reflection pipeline used in the appendix case. After the base expression is fitted and scored, the critic agent diagnoses possible redundant terms, missing components, or unnecessarily complex substructures, and the executor agent instantiates the proposed edits as executable candidates. In the shown example, the initially low-scoring base candidate is repaired after the critic identifies a missing interaction term in the mixed damping component and suggests adding the xv term. The executor incorporates this edit into the revised equation, which substantially improves the fitted score and allows the candidate to be retained as a high-quality memory entry. The restricted action space keeps local repair interpretable and prevents reflection from degenerating into unconstrained regeneration.

B.4 Effect of Semantic Memory

Figure 6 provides a mechanism-level view of why semantic memory improves later sampling. High-score candidates are defined as those with $\text{Acc}_{0.1} > 0.9$. Compared with score-only retention, TF-IDF-based semantic memory distributes high-score equations across more singleton and small clusters, suggesting that strong hypotheses are preserved as structurally diverse representatives rather than collapsed into a few redundant families.

Score-only retention, in contrast, tends to accumulate structurally similar equations and often favors shorter variants. This can fill later prompts with redundant experience while discarding informative terms from more complex but promising candidates. Consistent with the iteration curves in Figure 4, removing semantic memory can make the search converge early to redundant equation patterns in later iterations, rather than continuing to explore structurally diverse alternatives. Semantic memory mitigates this risk by grouping similar hypotheses, keeping the strongest representative in each cluster, and retrieving a more varied set of elite cases for subsequent iterations.

B.5 Extended OOD Analysis

Table 4 reports additional out-of-distribution (OOD) results across both the main benchmarks and LSR-SYNTH suites. Compared to in-distribution performance, OOD results exhibit larger variance across methods, highlighting the importance of structural generalization rather than pure fitting accuracy.

From the upper panel of Table 4, classical methods and several LLM-based baselines suffer significant degradation in NMSE under distribution shift, particularly on *E. coli Growth* and *Stress-Strain*, where complex nonlinear interactions dominate. In contrast, our method maintains consistently low NMSE and high $\text{Acc}_{0.1}$ across all tasks, indicating improved robustness to unseen regimes. The lower panel further shows that this advantage extends to the more diverse LSR-SYNTH benchmarks: while competing methods are often unstable across suites, our approach achieves the best or near-best results in most cases, especially on CRK and BPG.

Overall, these results indicate that STRIDE's multi-role self-reflective workflow does more than improve local fitting quality: reliable mixed fitting, targeted critic–executor repair, and semantic

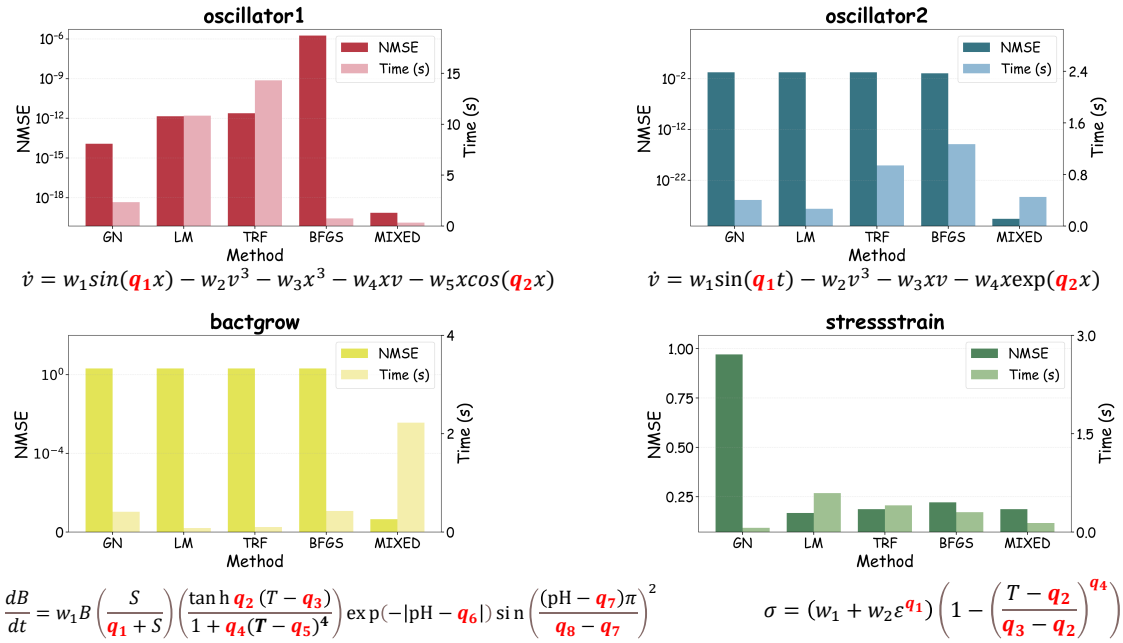


Figure 10: Quantitative comparison of parameter fitting strategies across multiple benchmarks. In the equation legend, red parameters denote nonlinear parameters.

Table 4: OOD results on four symbolic regression benchmarks (top) and four LSR-SYNTH benchmark suites (bottom).Acc↑ and NMSE↓ indicate that higher and lower values are better, respectively.The best and second-best results are highlighted in red and blue, respectively.

Method	Oscillator 1			Oscillator 2			E. coli Growth			Stress-Strain		
	$Acc_{0.1}$↑	$Acc_{0.001}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{0.001}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{0.001}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{0.001}$↑	NMSE↓
Baselines without LLMs												
GPlearn	4.38	0.04	1.49e0	16.13	0.11	1.15e-1	0.76	0.01	1.00e0	18.97	0.27	1.11e0
PySR	29.42	0.31	1.78e-1	99.39	82.10	7.88e-5	3.56	0.04	3.49e1	87.53	0.27	6.40e-2
DSR	3.53	0.04	7.89e-1	5.04	0.01	5.73e-1	2.97	0.04	1.02e0	2.98	0.00	1.15e0
uDSR	3.53	0.04	7.89e-1	18.32	0.13	1.06e-1	1.51	0.01	1.04e0	1.90	0.00	1.52e0
GPT-5.1												
LLM-SR	99.97	41.21	2.85e-4	86.28	19.33	3.05e-3	0.56	0.01	1.20e0	89.02	0.14	8.16e-2
LaSR	22.48	0.21	1.12e-1	2.30	0.01	8.24e-1	1.18	0.01	1.42e1	88.89	0.14	9.86e-2
SR-LLM	46.10	0.21	1.08e-2	16.21	0.12	1.15e-1	0.66	0.01	1.28e0	26.02	0.14	4.68e-1
DrSR	65.33	3.78	3.35e-2	47.36	0.16	4.49e-2	0.51	0.01	2.40e1	90.65	2.98	7.02e-2
STRIDE	100.00	100.00	5.97e-12	100.00	100.00	4.48e-21	4.58	0.05	1.41e-1	93.77	0.41	4.79e-2
Claude-3-Haiku												
LLM-SR	67.43	8.77	3.01e-2	100.00	93.84	1.40e-5	4.75	0.07	1.44e-1	9.62	0.14	3.22e-1
LaSR	28.39	0.39	1.20e-1	76.29	2.96	1.02e-2	1.07	0.01	3.43e0	88.35	0.41	7.91e-2
SR-LLM	4.45	0.04	1.61e+3	16.21	0.12	1.15e-1	0.99	0.02	3.00e0	73.44	0.14	2.75e-1
DrSR	66.81	0.70	3.37e-2	40.82	0.45	4.93e-2	0.59	0.00	4.75e0	30.08	0.00	3.07e-1
STRIDE	100.00	100.00	7.82e-8	100.00	100.00	4.66e-8	9.23	0.07	5.03e-3	92.82	9.08	5.50e-2

Method	CRK			PO			MatSci			BPG		
	$Acc_{0.1}$↑	$Acc_{max,0.1}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{max,0.1}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{max,0.1}$↑	NMSE↓	$Acc_{0.1}$↑	$Acc_{max,0.1}$↑	NMSE↓
GPT-5.1												
LLM-SR	65.40	40.00	4.20e-1	64.96	0.00	7.43e2	100.00	100.00	1.11e-1	60.64	60.00	6.94e0
SR-Scientist	42.32	40.00	3.59e0	44.48	0.00	1.31e-2	100.00	100.00	2.11e-1	47.56	20.00	8.99e-1
STRIDE	100.00	100.00	1.46e-3	57.56	0.00	4.77e-2	100.00	100.00	2.50e-1	80.00	80.00	5.09e-2
Claude-3-Haiku												
LLM-SR	41.56	40.00	1.55e1	48.60	0.00	2.02e5	80.00	80.00	5.12e0	44.84	0.00	2.12e2
SR-Scientist	72.36	20.00	8.39e-1	75.00	0.00	2.99e-2	100.00	100.00	1.84e-1	42.28	20.00	1.38e2
STRIDE	100.00	100.00	5.52e-3	60.12	0.00	1.06e-2	100.00	100.00	1.76e-1	96.60	60.00	2.46e-4

memory coordinate across iterations to preserve symbolic structures that remain meaningful under distribution shift.

C Baseline Implementation Details

We compare our method with both traditional symbolic regression baselines and recent LLM-based equation discovery methods. To ensure a fair comparison, we keep the evaluation data splits fixed across all methods. For LLM-based baselines, we evaluate both GPT-5.1 and CLAUDE-3-HAIKU using the same prompt template and decoding temperature as in the main experiments, unless a comparison is explicitly reported for a single backbone.

gplearn (Stephens, 2022). `gplearn` is a classical genetic-programming SR baseline that evolves symbolic expression trees through selection, crossover, and mutation over a predefined operator set. We use the public `gplearn` package with population size 1000, 2000 generations, and tournament size 20.

PySR (Cranmer, 2023). `PySR` is an advanced SR method that employs asynchronous multi-island GP-based evolution with built-in constant optimization. We use `PySR` as a strong evolutionary baseline and run 2000 search iterations with 50 populations.

DSR and uDSR (Petersen et al., 2019; Landajuela et al., 2022). `DSR` is a neural-guided symbolic regression method that learns a policy for sampling expressions, while `uDSR` extends this line by integrating multiple symbolic regression strategies into a unified modular framework. For `DSR`, we use the original deep symbolic regression framework based on neural sequence generation and policy-gradient optimization, with training batch size 100 and 200000 sampled expressions. For `uDSR`, we keep the same batch size and sample budget, and additionally enable its polynomial, constant, length-prior, and polynomial-optimizer settings.

LLM-SR (Shojaee et al., 2025a). `LLM-SR` is an LLM-guided program-search method that represents equations as executable programs and improves them through iterative proposal, evaluation, and island-based memory updating. We implement `LLM-SR` following its original formulation with `global_max_sample_num`=2500, `samples_per_prompt`=4, and `num_islands`=10.

LaSR (Grayeli et al., 2024). `LaSR` is a concept-library-based LLM-SR method that uses abstract natural-language concepts distilled from high-performing hypotheses to guide subsequent symbolic search. We implement `LaSR` with 20 populations and 100 iterations, using LLM operation probability 0.01, 5 generated equations, 3 generated concepts, and 2 concept-crossover samples.

SR-LLM (Guo et al., 2025). `SR-LLM` is a retrieval-augmented incremental symbolic regression method that reuses prior symbolic experience to guide later equation generation. We retain its core retrieval-and-generation workflow and, following the released setting summary, use batch size 500, 20 epochs, and 30 evolution rounds, corresponding to about 300000 explored samples.

SR-Scientist (Xia et al., 2026). `SR-Scientist` is an agentic equation-discovery baseline that combines tool-assisted data analysis with iterative equation proposal and evaluation. We implement `SR-Scientist` following its original agentic workflow. To keep the comparison computationally comparable, we use the same two-backbone protocol; its main inference parameters are `num_turns`=100, `max_assistant_turns`=25, `mape_threshold`=0.001, and `top_k`=3.

DrSR (Wang et al., 2025). `DrSR` is a dual-reasoning LLM-based SR method that combines data-aware analysis with reflective symbolic search. We preserve its core data-understanding and reflection mechanisms while using the same benchmark setup and two-backbone protocol as the other LLM-based baselines. The main search parameters are 700 iterations, 4 samples per prompt, and 10 islands.

D Dataset Descriptions

To evaluate the effectiveness and generalization ability of our method, we conduct experiments on a collection of benchmark tasks drawn from two primary sources: the datasets introduced in the original LLM-SR work and the LSR-SYNTH benchmark suite. These datasets cover a diverse range of scientific domains, including physics, biology, chemistry, and materials science. Most tasks are associated with known underlying equations that serve as ground-truth targets for symbolic regression. The following paragraphs summarize the key characteristics of these benchmarks, and Table 5 then lists the corresponding equations.

D.1 LLM-SR Benchmarks

The LLM-SR benchmarks consist of four representative symbolic regression tasks designed to evaluate the ability of models to recover nontrivial scientific relationships from data; their equations are listed in Table 5(a). These tasks include two nonlinear oscillator systems, a biological growth process, and a temperature-dependent stress-strain model. The oscillator and growth tasks are defined by explicit governing equations. For *Stress-Strain*, which does not have a single standardized ground-truth law, we report the representative form used to generate and describe the benchmark behavior. The oscillator systems emphasize dynamic interactions between position and velocity, while the growth and material models incorporate nonlinear responses and environmental dependencies. Together, these benchmarks provide a controlled yet diverse testbed for assessing equation discovery performance.

D.2 LSR-Synth Benchmark Suites

We categorize equation difficulty in the LSR-SYNTH benchmark using the ranking script in our benchmark pipeline. The score combines expression tree depth, operator diversity, variable coupling, inverse or non-smooth operators (e.g., $\log$, $\sqrt{\cdot}$, division), domain-related terms, and empirical penalties. We rank equations within each suite by this score and select the top five tasks for evaluation, as listed in Table 5(b), because higher-ranked equations tend to exhibit stronger nonlinear coupling and non-smooth operators, which significantly increases optimization difficulty and leads to lower discovery success rates. The domains are denoted as **CRK**, **BPG**, **PO**, and **MatSci**, following the categorization in the LSR-SYNTH benchmark. Each selected task uses the benchmark-provided training, in-domain test, and out-of-domain test splits.

E Additional Case Study

We use *Oscillator 1* as a qualitative case because it requires recovering a compact nonlinear dynamic law rather than only fitting a smooth response surface. The task contains coupled terms involving position and velocity, so methods that add many weak or redundant terms may still reduce pointwise error locally but fail to preserve the correct phase-space structure. As shown in Figure 12a, STRIDE produces the closest phase-space alignment with the ground-truth acceleration field. This behavior is consistent with the equation-level evidence: the discovered expression retains the main nonlinear interaction patterns of the oscillator, including coupled position-velocity effects and nonlinear restoring terms. In contrast, the competing expressions either miss part of the symbolic structure or compensate with more complex terms that do not align as well with the true dynamics. Thus, the accuracy gain in this case is better explained by improved symbolic identification than by merely accumulating additional polynomial terms.

Figure 12b provides a second oscillator case. Here, STRIDE recovers the dominant forcing, cubic damping, velocity-position coupling, and exponential restoring term, leading to a phase-space trajectory that closely follows the ground truth. LLM-SR captures several relevant components but introduces extra polynomial and absolute-value terms, while PySR recovers partial forcing information but misses important structural interactions. This case again suggests that the improvement comes from identifying the correct symbolic components and their couplings.

Figure 13 shows the stress-strain case under multiple temperatures. STRIDE tracks the ground-truth curves across temperature conditions more consistently than the compared methods, especially in the rapid transition region at small strain and in the high-strain saturation regime. This indicates that the recovered equation captures the temperature-dependent response pattern rather than only fitting one local curve.

Finally, Figure 14 visualizes representative LSR-Synth cases from CRK, PO, MatSci, and BPG. The shaded regions indicate extrapolation intervals. Across these examples, STRIDE generally stays closer to the ground-truth trajectories in both observed and extrapolated regions, whereas competing methods often deviate after the transition into the OOD interval. These cases qualitatively support the OOD results in Appendix B.5: the selected equations are more likely to preserve the underlying functional structure needed for extrapolation.

F Prompt Templates

The prompt design follows the multi-role workflow in Section 3. Figure 15a gives the generator agent task context, variables, data hints, and retrieved elite cases for proposing executable skeletons. Figure 15b separates reflection into critic and executor prompts: the critic agent receives the eval-

Table 5: Dataset equations used in our experiments. The upper panel summarizes the four LLM-SR benchmark tasks, and the lower panel lists the top five hardest LSR-SYNTH equations in each domain according to the proposed difficulty score. The equation identifiers follow the local zero-based benchmark files, while the displayed formulas are matched to the corresponding one-based entries in the LLM-SRBENCH appendix.

(a) Four LLM-SR benchmark tasks.

Dataset	Inputs	Underlying or representative equation
Oscillator 1	x, v	$\dot{v} = F\sin(\omega x) - \alpha v^3 - \beta x^3 - \gamma x v - x\cos(x)$
Oscillator 2	t, x, v	$\dot{v} = F\sin(\omega t) - \alpha v^3 - \beta x v - \delta x \exp(\gamma x)$
Bactgrow	B, S, T, pH	$\frac{dB}{dt} = \mu_{\max} B\left(\frac{S}{K_S+S}\right)\left(\frac{\tanh k(T-x_0)}{1+c(T-x_{\mathrm{decay}})^4}\right)\exp(-\lvert\mathrm{pH}-\mathrm{pH}_{\mathrm{opt}}\rvert)\sin\left(\left(\frac{\mathrm{pH}-\mathrm{pH}_{\min}}{\mathrm{pH}_{\max}-\mathrm{pH}_{\min}}\pi\right)^2\right)$
Stress-Strain	ε, T	$\sigma = (A + B\varepsilon^n)\left(1 - \left(\frac{T-T_r}{T_m-T_r}\right)^m\right)$ (representative form)

(b) Top five hardest LSR-SYNTH equations in each domain.

Domain	Equation ID	Equation
Chemistry	CRK22	$\frac{dA}{dt} = -kA(t)^2 - kA(t)\exp(-k_s t) + \frac{k_z A(t)^2}{\beta A(t)^4 + 1}$
	CRK29	$\frac{dA}{dt} = -kA(t) - kA(t)\exp(-k_s t) + \frac{k_z A(t)^2}{\beta A(t)^4 + 1}$
	CRK3	$\frac{dA}{dt} = -kA(t)^2 - kA(t)\exp(-k_s t) + k_w\cos(\log(A(t)+1))$
	CRK15	$\frac{dA}{dt} = -k\sqrt{A(t)} - kA(t)\exp(-k_s t) + k_p\sin(\log(A(t)+1))$
	CRK19	$\frac{dA}{dt} = -kA(t)^2 - kA(t)\exp(-k_s t) + k_p\sin(\log(A(t)+1))$
Biology	BPG20	$\frac{dP}{dt} = r\left(-1+\frac{P(t)}{\alpha}\right)\left(1-\frac{P(t)}{K_0}\right)P(t) + r\left(1-\frac{P(t)}{K_0}\right)P(t) + \frac{rP(t)}{1+\exp(-\alpha(-\beta+P(t)))}$
	BPG21	$\frac{dP}{dt} = r\left(-1+\frac{P(t)}{\alpha}\right)\left(1-\frac{P(t)}{K_0}\right)P(t) + \frac{rP(t)}{t^\alpha}$
	BPG13	$\frac{dP}{dt} = r\left(-1+\frac{P(t)}{\alpha}\right)\left(1-\frac{P(t)}{K_0}\right)P(t) + rP(t) + \frac{rP(t)}{1+\exp(-\alpha(-\beta+P(t)))}$
	BPG9	$\frac{dP}{dt} = r\left(-1+\frac{P(t)}{\alpha}\right)\left(1-\frac{P(t)}{K_0}\right)P(t) + r\left(1-\frac{P(t)}{K_0}\right)P(t) + r\left(1-\exp(-\gamma P(t))\right)P(t)$
	BPG19	$\frac{dP}{dt} = r\left(1-\frac{P(t)}{K_0}\right)P(t) + \frac{rP(t)}{t^\alpha}$
Physics	PO20	$\frac{dv}{dt} = -\alpha v(t)^3 - \beta\log(\lvert v(t)\rvert+1) - 2\beta v(t) - \mu(1-v(t)^2)v(t) - \omega_0^2(\gamma\lvert v(t)\rvert^{0.33}+1)x(t)$
	PO40	$\frac{dv}{dt} = F_0\sin(t) - \beta\lvert v(t)\rvert^{0.33} - \omega_0^2(\gamma\lvert v(t)\rvert^{0.33}+1)x(t) - \omega_0^2 x(t)^3 - \omega_0^2 x(t)$
	PO12	$\frac{dv}{dt} = F_0\sin(t) - \alpha v(t)^3 - \beta\lvert v(t)\rvert^{0.33} - \omega_0^2(\gamma t+1)x(t) - \omega_0^2 x(t)$
	PO5	$\frac{dv}{dt} = -\beta\sin(v(t)) - 2\beta v(t) - \omega_0^2(\gamma\lvert v(t)\rvert^{0.33}+1)x(t) - \omega_0^2 x(t)^3 - \omega_0^2 x(t)$
	PO14	$\frac{dv}{dt} = F_0\sin(t) - \beta\log(\lvert v(t)\rvert+1) - \beta\sin(v(t)) - 2\beta v(t) - \mu(1-x(t)^2)v(t)$
Material	MatSci12	$\sigma = E_0\epsilon(-\alpha_T(T-T_0)+1) + K\epsilon^N\exp\left(-\frac{Q}{RT}\right) + \eta\epsilon\exp(-(T-T_0)^2)$
	MatSci23	$\sigma = K\epsilon^N\exp\left(-\frac{Q}{RT}\right) - \beta(T-T_0) + \eta(T-T_0)\log(\epsilon+1)$
	MatSci5	$\sigma = E_0\epsilon(-\alpha_T(T-T_0)+1) + K\epsilon^N\exp\left(-\frac{Q}{RT}\right) + \eta\epsilon^M(T-T_0)$
	MatSci1	$\sigma = H\epsilon^3 + K\epsilon^N\exp\left(-\frac{Q}{RT}\right) + \eta\epsilon\sin(T-T_0)$
	MatSci3	$\sigma = H\epsilon^3 + K\epsilon^N\exp\left(-\frac{Q}{RT}\right) + \eta\epsilon^3(T-T_0)$

uated candidate, fitted parameters, score feedback and task description, and outputs constrained edit actions for diagnosing redundant terms, missing components, or over-complex substructures; the executor agentreceives the original candidate and valid critic actions, and turns them into executable revised equations for re-fitting and re-scoring.

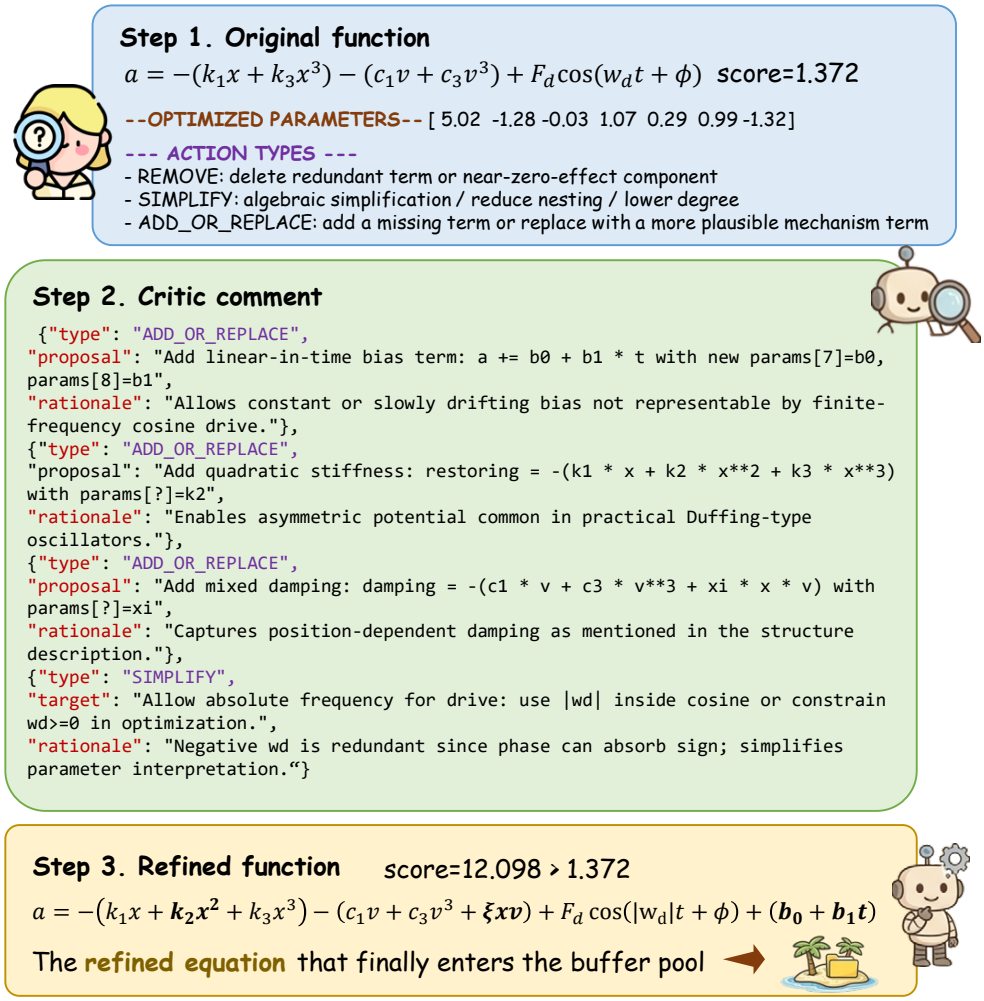
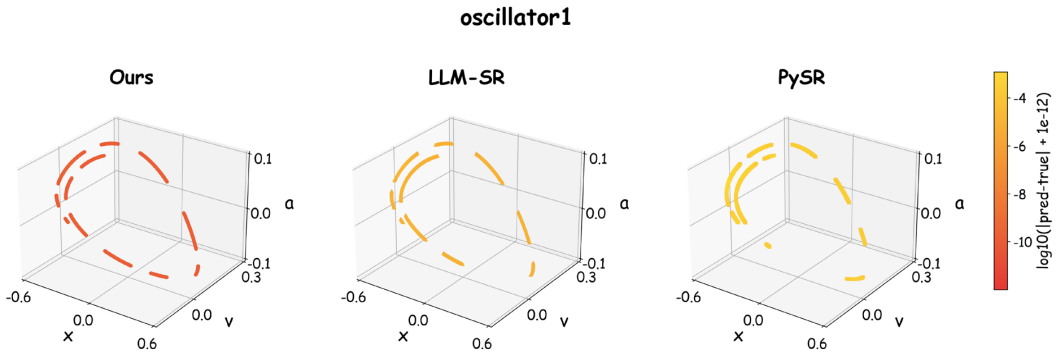
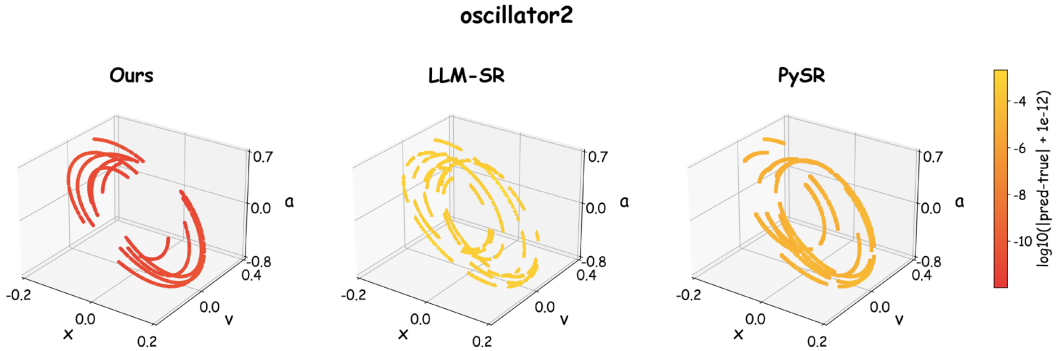


Figure 11: Illustration of critic–executor reflective repair. The critic agent analyzes a fitted candidate equation, proposes constrained symbolic edits, and the executor agent turns valid actions into locally repaired candidates.



Ground Truth	$\dot{v} = 0.8\sin(x) - 0.5v^3 - 0.2x^3 - 0.5xv - x\cos(x)$
OURS	$\dot{v} = 0.560x - 0.23x^3 - 0.76x\cos(1.03x) - 0.5xv - 0.5v^3$
LLMSR	$\dot{v} = 0.82x - 0.17x^3 - 2.0 \times 10^{-5}v - 0.50v^3 + 1.1 \times 10^{-4} - 0.73\sin(1.39x) - 1.1 \times 10^{-4}\cos(0.68x) - 0.50xv$
PySR	$\dot{v} = -0.12\sin\left(\sin\left(\sin\left(\sin\left(\sin\left(x - 0.13(v - 0.02)\cos(x/(-0.39))\right)\right)\right)\right)\right)\right) \cdot \exp(\sin(-(v - 1.00) \cdot 3.69))$

(a) *Oscillator 1.*



Ground Truth	$\dot{v} = 0.3\sin(t) - 0.5v^3 - xv - 5x \cdot \exp(0.5x)$
OURS	$\dot{v} = 0.3\sin(t) - 0.5v^3 - xv - 5x \cdot \exp(0.5x)$
LLMSR	$\dot{v} = -0.0016\cos(t) + 0.30\sin(t) - 5x + 2.51x^2 + 0.6838x^3 - 0.023v + 0.23v\lvert v\rvert + 0.9965xv$
PySR	$\dot{v} = -(0.61x^3 + 2.50 + 5.0001597x) + v[-x + \ln(\cos(v\cos(0.41\sin(v))))] + 0.3\sin(t)$

(b) *Oscillator 2.*

Figure 12: Case studies on the two oscillator benchmarks. We compare phase-space overlays and recovered expressions for **Ours**, **LLM-SR**, and **PySR**. Highlighted terms indicate components aligned with the ground-truth equation.

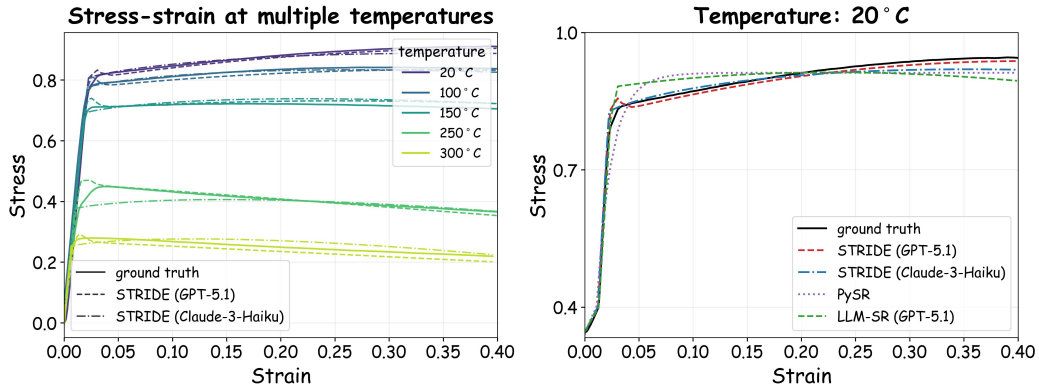


Figure 13: Case study on *Stress-Strain*. The left panel compares predictions across multiple temperatures, and the right panel zooms into the 20°C condition.

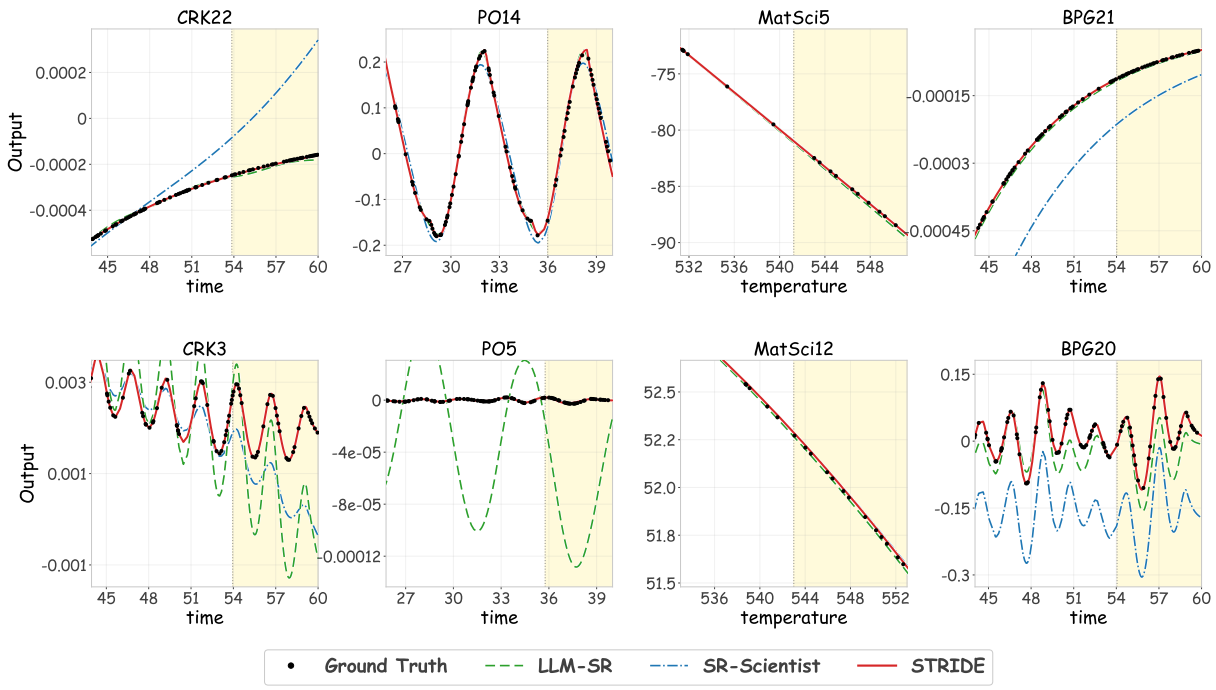


Figure 14: Representative LSR-Synth case studies. The shaded regions denote OOD intervals, and the curves compare ground truth, LLM-SR, SR-Scientist, and STRIDE. The upper row uses GPT-5.1 as the backbone, and the lower row uses Claude-3-Haiku.

Initial input prompt

```
You are a helpful assistant tasked with discovering mathematical function structures for scientific systems. Complete the
'equation' function body below, considering physical meaning and relationships of inputs. Be diverse but parsimonious.
--- Data Hint---
{OPTIONAL_DATA_HINT_TEXT_IF_ENABLED}
--Elite Cases--
{PROGRAM_TEMPLATE_WITH_FUNCTIONS_REPLACED}
"""
Find the mathematical function skeleton that represents acceleration in a damped nonlinear oscillator system with driving force,
given data on position, and velocity.
"""
MAX_NPARAMS = 10
@equation.evolve
def equation_v0(x: np.ndarray, v: np.ndarray, params: np.ndarray) -> np.ndarray:
""" Mathematical function for acceleration in a damped nonlinear oscillator ... """
{ELITE_FUNCTION_BODY_0}
return dv
@equation.evolve
def equation_v1(x: np.ndarray, v: np.ndarray, params: np.ndarray) -> np.ndarray:
"""Improved version of `equation_v0`."""
{ELITE_FUNCTION_BODY_1}
return dv
...
@equation.evolve
def equation_v{K-1}(x: np.ndarray, v: np.ndarray, params: np.ndarray) -> np.ndarray:
"""Improved version of `equation_v{K-2}`."""
{ELITE_FUNCTION_BODY_{K-1}}
return dv
@equation.evolve
def equation_vK(x: np.ndarray, v: np.ndarray, params: np.ndarray) -> np.ndarray:
"""Improved version of `equation_v{K-1}`."""
# BODY IS EMPTY HERE (this is where the model must write)
```

(a) Prompt for the generator agent.

Prompt for critic

```
You are a scientific critic for symbolic regression.
Your ONLY job is to output a JSON object that proposes edit actions to refine the equation.
Do NOT output any markdown, code fences, or explanations outside JSON.
--- SYSTEM DESCRIPTION ---
{problem_description}
--- CANDIDATE CODE ---
{code}
--- OPTIMIZED PARAMETERS (truncated) ---
{param_str}
--- INPUT VARIABLES ---
{input_vars}
--- METRICS ---
{metrics_str if metrics_str else "N/A"}
--- ACTION TYPES ---
- REMOVE: delete redundant term or near-zero-effect component
- SIMPLIFY: algebraic simplification / reduce nesting / lower degree
- ADD_OR_REPLACE: add a missing term or replace with a more plausible mechanism term
--- OUTPUT JSON SCHEMA (exact keys) ---
{
"actions": [
{"type":"REMOVE","target":"...","rationale":"..."},
{"type":"SIMPLIFY","target":"...","rationale":"..."},
{"type":"ADD_OR_REPLACE","proposal":"...","rationale":"..."}
],
"short_comment":"one sentence summary"
}
Rules:
- actions can be empty
- rationale must be specific, feasible, and concise, with fewer than 20 words.
- Output ONLY valid JSON
- Actions and reasons should be concise and to the point, without excessive explanation!
```

Prompt for executor

```
You are generating equation function BODIES for symbolic regression.
HARD REQUIREMENTS (must satisfy):
1) Output ONLY the function body (no def line), indented with exactly 4 spaces.
2) The body MUST be executable Python and MUST contain an explicit `return` statement.
3) The `return` must return a numpy array .
4) Do NOT output partial code. Do NOT stop after defining variables. Always finish with return.
5) Do NOT include markdown fences, explanations, or multiple variants.
6) You may ONLY use variables that appear in the function signature (input variables + `params`)
and `np`.
=== BASE EQUATION (function body) ===
{base_body}
=== CRITIC ACTIONS (JSON) ===
{critic_blob}
```

(b) Prompt for the critic and executor agents.

Figure 15: Prompt templates for the generator agent and critic–executor agents.