FROM SCORE TO DIAGNOSIS: SEARCH-INDEPENDENT POST-DIAGNOSIS FOR SYMBOLIC REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression (SR) solvers employ diverse search mechanisms, yet most rely on the same low-information scalar fitness. Such scores provide little guidance on why a skeleton fails or how it should be revised. Recent methods develop new methods to address this bottleneck. Their enhanced feedback mechanisms are tightly coupled to their search processes and are difficult to reuse across diverse SR solvers. Instead, we ask whether a shared feedback mechanism can further improve diverse solvers facing the same bottleneck. To this end, we propose PostDiag, a search-independent feedback mechanism, which can be shared to different SR solvers. PostDiag introduces Patch-wise Feature Profiling (PFP) to capture local behavior through patch-wise features of level, trend, and roughness. Based on PFP, PostDiag quantifies Skeleton Expressiveness Risk by comparing the fitted candidate with the observations, and Skeleton Stability Risk by measuring variability of behavior after bootstrap refitting. The Diagnostic Risk Plane (DRP) combines two risks to determine whether a skeleton should be preserved, repaired, simplified, or rewritten, while the patch-wise discrepancies provide evidence for the intervention. An LLM editor then performs the diagnosis-guided intervention, and the validated revision is returned to the original solver through a lightweight adapter. Experiments across multiple LLM-based and GP-based SR solvers show that PostDiag consistently improves accuracy and accelerates convergence.

1 INTRODUCTION

Symbolic regression (SR) discovers compact, interpretable mathematical expressions from observed data, enabling data-driven scientific discovery across physics, biology, chemistry, and engineering (Wang et al., 2023; Zhang et al., 2025; Karniadakis et al., 2021; Krenn et al., 2022; Udrescu & Tegmark, 2020; Udrescu et al., 2020). Given observations $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, SR searches for an expression $f : \boldsymbol{x} \mapsto y$ that explains the underlying relationship. Modern SR solvers employ diverse search strategies to enable efficient and accurate SR: PySR uses multi-population evolutionary search (Cranmer, 2023); LLM-SR iteratively refines equations using the code-generation and reasoning capabilities of LLMs (Shojaee et al., 2025a); LaSR co-evolves equations with a reusable concept library (Grayeli et al., 2024); and SR-Scientist autonomously integrates data analysis, tool use, and equation refinement (Xia et al., 2025). This diversity is valuable because different solvers offer complementary strengths.

Despite their diverse search strategies, most SR solvers rely on the same impoverished search feedback: a scalar fitness score, such as MSE or NMSE. By collapsing the residual landscape into a single value, these scores obscure error geometry and reveal little about where, why, or how a candidate fails.Indeed, similar scores may correspond to fundamentally distinct failure modes (Li et al.; Xiang et al.). This limited feedback leaves even sophisticated search mechanisms poorly informed, resulting in inefficient and misdirected exploration.

Recent methods RESTART and Deliberate Evolution (DE) design new SR solvers that embed structured diagnostics into their search loops, providing clearer directions for candidate refinement than scalar errors alone (Li & Pan, 2026; Pang et al., 2026). However, their feedback mechanism depend on solver-specific search information, such as search trajectories memory, parent-selection strategy, and operator policy. Thus, feedback mechanisms are tightly coupled to the method's search process and remain an internal solver capability rather than reusable feedback. This coupling creates

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.

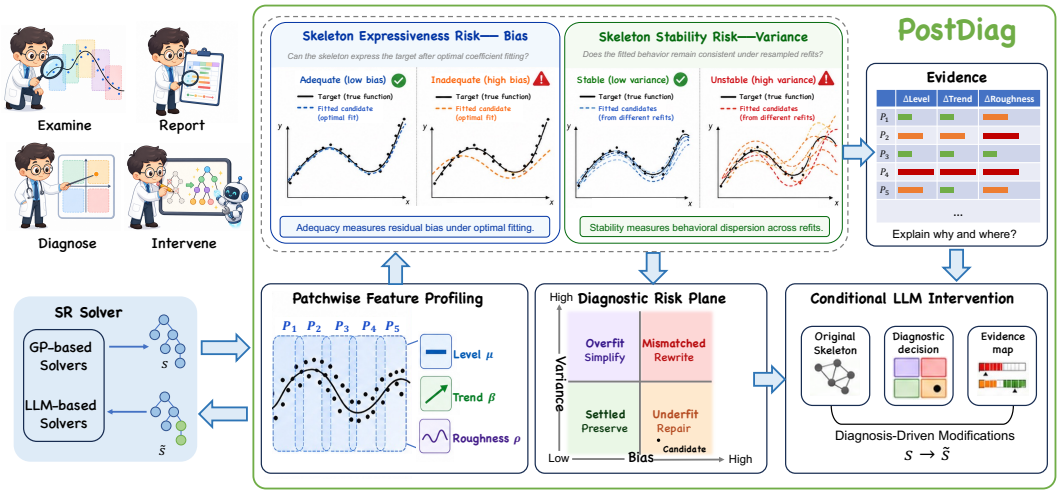


Figure 1: Overview of POSTDIAG, a search-independent post-diagnosis pipeline for symbolic regression. Given a candidate skeleton s produced by LLM-based or GP-based SR solver, POSTDIAG first *examines* its local behavior via patchwise feature profiling, then *reports* the diagnostic report (evidence map), and *diagnoses* the failure type on the diagnostic risk plane, and finally calls a conditional LLM *intervention*, yielding a diagnosis-driven modification $\tilde{s}$ that is returned to the solver.

an undesirable trade-off: obtaining such enhanced feedback mechanism generally requires adopting the accompanying search framework, rather than retaining the distinctive mechanisms of existing solvers, such as LaSR's concept–equation co-evolution, SR-Scientist's autonomous tool-use loop, and PySR's efficient genetic-programming engine.

This motivates a different question: *can richer diagnostic feedback be made reusable across diverse solvers, so that their shared feedback bottleneck can be addressed without sacrificing their distinctive search mechanisms?* Rather than introducing another new solver, we pursue a route orthogonal to solver design: constructing a diagnostic feedback mechanism that can be integrated with different SR solvers. Such a mechanism should be **search-independent**, neither depending on nor altering the solver's underlying search mechanism; **informative**, revealing where and how a candidate fails; and **solver-compatible**, allowing diagnosis-guided revisions to be readily returned to the solver.

To instantiate this view, we propose **PostDiag**, a search-independent post-diagnostic module for SR. Unlike feedback mechanisms embedded in the search loop, PostDiag takes only candidate skeletons and observed data as input, without relying on solver-specific search mechanisms. It can therefore be integrated with solvers that employ different search mechanisms. For each candidate, PostDiag produces two complementary outputs: an evidence map that reveals where and how the candidate fails, and a risk localization that determines the appropriate intervention. PostDiag first introduces Patchwise Feature Profiling (PFP), which represents local candidate behavior using structured fingerprints of *level*, *trend*, and *roughness*. Rather than compressing diagnostic information into a single global error, discrepancies between these local fingerprints form an interpretable evidence map and support the estimation of *Skeleton Expressiveness Risk* and *Skeleton Stability Risk*. The former captures the structural mismatch remaining after coefficient fitting, while stability measures the candidate's sensitivity to data resampling. The Diagnostic Risk Plane (DRP) uses these two quantities to assign the candidate to one of four intervention regions: preserve, repair, simplify, or rewrite. A conditional LLM then performs the corresponding skeleton intervention, and the revised candidate is returned to the original solver through a lightweight adapter. We integrate the PostDiag with different SR solvers spanning LLM-based and GP-based methods, and evaluate them on the LSR-Synth dataset of LLM-SRBench. The results show that PostDiag improves candidate refinement and accelerates convergence without replacing the solvers' original search mechanisms. Ablation studies further validate the contribution of each diagnostic component.

Our contributions come in fourfold: (1) We introduce a solver-independent post-diagnostic perspective for SR that separates feedback design from search design, addressing a shared feedback bottleneck without replacing existing solvers. (2) We propose **PostDiag**, a reusable diagnostic module that takes only a candidate skeleton and observed data as input and returns evidence-supported

diagnoses and candidate revisions through lightweight solver adapters. (3) We develop PFP to extract patch-wise diagnostic evidence and DRP to map skeleton adequacy and stability to targeted interventions. (4) We integrate the PostDiag with seven different LLM-based and GP-based SR solvers, demonstrating broad compatibility and consistent convergence improvements.

2 Related Work

Symbolic regression (SR) spans diverse search and learning paradigms, each contributing distinct inductive biases and capabilities (Chen et al., 2022; Schmidt & Lipson, 2009; Cranmer et al., 2020). Genetic-programming methods evolve expression trees through population-based operators (Koza, 1994; Pawlak & Krawiec, 2018; Moraglio et al., 2012; de Franca & Aldeia, 2021; Stephens, 2024), with PySR providing efficient multi-population search, complexity control, and constant optimization (Cranmer, 2023). Tree-search, reinforcement-learning, and neural methods instead leverage planning, learned generation policies, or pretrained representations (Sun et al., 2023; Petersen et al., 2021; Li et al., 2022; Mežnar et al., 2023; Holt et al., 2023; Biggio et al., 2021; Kamienny et al., 2022; Shojaee et al., 2023; Meidani et al., 2023; Li et al.). More recently, LLM-based methods have incorporated scientific priors and reasoning: LLM-SR combines program proposal with evolutionary experience (Shojaee et al., 2025a), LaSR co-evolves equations and a concept library (Grayeli et al., 2024), and SR-Scientist performs autonomous tool-assisted optimization (Xia et al., 2025). Despite their diverse generation and search mechanisms, these methods share a common feedback bottleneck: candidate behavior is compressed into scalar fitness, which supports ranking but provides little information about failure modes, structural defects, or appropriate interventions.

Deliberate Evolution (DE) (Pang et al., 2026) augments an MSE-driven evolutionary process with data and residual diagnostics, adaptive search operators, and trajectory-level memory, demonstrating that structured feedback can improve symbolic search beyond reliance on scalar errors alone. However, DE addresses this shared feedback bottleneck by introducing a new end-to-end solver, in which diagnosis is tightly coupled with its parent-selection strategy, operator policy, and trajectory memory. Adopting DE therefore entails replacing the original solving framework, making it difficult to retain solver-specific strengths such as LaSR's concept–equation co-evolution, SR-Scientist's agentic closed-loop optimization, or PySR's efficient genetic-programming engine. Its limitation lies in addressing a common challenge across SR methods through a dedicated solver rather than a capability that existing solvers can readily share.

PostDiag takes a different technical route. Instead of replacing existing solvers, it provides a shared diagnostic interface decoupled from their internal search processes. Given only a candidate skeleton produced by the solver and the observation, PostDiag uses Patch-wise Feature Profiling to recover local behavioral information obscured by scalar errors and provide evidence for risk localization and candidate intervention. It can therefore enhance diagnostic feedback across heterogeneous SR solvers while preserving their original search mechanisms and distinctive strengths.

3 A Search-Independent Post-Diagnosis for SR

3.1 Problem Setup

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, we represent a candidate expression as $f_s(\boldsymbol{x}; \boldsymbol{\theta}_s)$. Here, s denotes the symbolic skeleton and $\boldsymbol{\theta}_s$ is its numerical coefficients. PostDiag treats the skeleton s as the object of diagnosis.

We randomly partition $\mathcal{D}$ into a disjoint fit set $\mathcal{D}_{\text{fit}}$ and audit set $\mathcal{D}_{\text{audit}}$. The fit set is used to estimate the numerical coefficients of the candidate skeleton, while the audit set is used to evaluate the fitted candidate and produce its diagnosis. This separation prevents coefficient fitting and candidate diagnosis from using the same observations.

Given a candidate skeleton s and dataset $\mathcal{D}$, PostDiag outputs a structured diagnostic report $\mathcal{G}_s$ and a revised skeleton $\widetilde{s}$:

$$\text{PostDiag}(s, \mathcal{D}) \longrightarrow (\mathcal{G}_s, \widetilde{s})\,.$$

Here, $\mathcal{G}_s = (c_s, \mathcal{E}_s)$ contains the candidate's diagnostic coordinate c_s on the diagnostic risk plane and supporting evidence $\mathcal{E}_s$, while $\widetilde{s}$ is the candidate skeleton revised according to the diagnosis.

PostDiag neither generates initial candidates nor prescribes how a solver conducts its search. It diagnoses candidate skeletons supplied by the solver and returns the diagnostic results and revised skeletons through a unified interface. The same diagnostic core can therefore be integrated with different SR solvers through lightweight adapters.

3.2 PATCH-WISE FEATURE PROFILING (PFP)

To support expressiveness and stability estimation without compressing evaluation into a scalar error, PostDiag uses Patch-wise Feature Profiling to construct structured fingerprints on the observation.

We first normalize each input dimension and denote the normalized coordinate of $\boldsymbol{x}_i$ by $\mathbf{x}_i$. We then construct M local patches $\mathcal{P} = \{P_m \subseteq \mathcal{D}_{\text{audit}}\}_{m=1}^{M}$. Such patches can be constructed in various ways. We adopt an anchor-kNN for its simplicity and robustness in multivariate settings. Specifically, we select spatially well-distributed anchors via farthest point sampling (Gonzalez, 1985) and construct each patch from the k nearest neighbors (Cover & Hart, 1967) of an anchor $\hat{\mathbf{x}}_m$:

$$P_m = \{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{audit}} \mid \mathbf{x}_i \text{ is among the } k \text{ nearest neighbors of } \hat{\mathbf{x}}_m\}.$$

The patches may overlap, reducing sensitivity to neighborhood boundaries, while M and k are chosen to cover all audit samples.

Given a patch P_m, PFP summarizes the local behavior of f using three complementary features.

Level. The level is the mean function value within the patch:

$$\mu_m(f) = \frac{1}{k} \sum_{i \in P_m} f(\boldsymbol{x}_i). \tag{1}$$

Equivalently, $\mu_m(f)$ is the best constant approximation of P_m in the least-squares sense, i.e., $\mu_m(f) = \arg\min_{c \in \mathbb{R}} \sum_{i \in P_m} (f(\boldsymbol{x}_i) - c)^2$. Thus, $\mu_m(f)$ summarizes the zeroth-order behavior of f on the patch P_m.

Trend. The trend captures the direction and magnitude in which P_m varies across the normalized inputs. It is defined as the least-squares slope vector of P_m on each direction:

$$\boldsymbol{\beta}_m(f) = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \sum_{\mathbf{x}_i \in P_m} \left[f(\boldsymbol{x}_i) - \mu_m(f) - \boldsymbol{\beta}^\top (\mathbf{x}_i - \bar{\mathbf{x}}_m) \right]^2,$$

where $\bar{\mathbf{x}}_m = \frac{1}{k} \sum_{\mathbf{x}_i \in P_m} \mathbf{x}_i$ is the normalized patch centroid. $\boldsymbol{\beta}_m(f)$ summarizes the first-order behavior of f on the patch P_m. We retain $\boldsymbol{\beta}_m(f)$ as a d-dimensional vector to capture trends in multivariate symbolic regression. Estimating local slopes via least squares, rather than endpoint differences, further improves robustness to endpoint anomalies and observation noise.

Roughness. Roughness captures the higher-order local behavior that cannot be explained by a constant level or a linear trend. For each $\boldsymbol{x}_i \in P_m$, we can define the local affine approximation use Level and Trend as

$$\hat{f}_m(\boldsymbol{x}_i) = \mu_m(f) + \boldsymbol{\beta}_m(f)^\top (\mathbf{x}_i - \bar{\mathbf{x}}_m).$$

We therefore denote the residual between f and $\hat{f}$ on P_m as an additional signal:

$$\rho_m(f) \;=\; \sqrt{\frac{1}{k} \sum_{i \in P_m} \left(f(\boldsymbol{x}_i) - \hat{f}_m(\boldsymbol{x}_i) \right)^2}.$$

Accordingly, if f is affine within P_m, its local behavior is fully explained by the Level and Trend terms, yielding $\rho_m(f) = 0$. Local nonlinearity or curvature increases Roughness $\rho_m(f)$.

The local fingerprint of f on P_m is

$$\Phi_m(f) = \left(\mu_m(f), \boldsymbol{\beta}_m(f), \rho_m(f)\right),$$

and its complete fingerprint over the $\mathcal{D}_{\text{audit}}$ is

$$\Phi(f) = \{\Phi_m(f)\}_{m=1}^{M}.$$

By profiling different behaviors over the same patch partition, PFP provides a common basis for structured comparison. Each comparison is tied to a specific input-space patch and further decomposed into level, trend, and roughness features. It therefore reveals not only where behaviors differ, but also how they differ, while reducing reliance on individual pointwise residuals. In the next subsection, we use PFP as a comparison tool to assess skeleton adequacy and stability. Together, these comparisons yield the diagnostic coordinate c_s and its supporting evidence map $\mathcal{E}_s$.

3.3 Diagnostic Risk Plane

PostDiag is designed to diagnose a candidate without depending on or altering the solver's search mechanism. It therefore cannot rely on solver-specific signals such as search trajectories, visitation statistics, population states, or operator histories, and must infer the appropriate intervention from only the candidate skeleton and observed data. To meet this constraint, we introduce the Diagnostic Risk Plane (DRP), which characterizes each skeleton using two behavior-based risks: *Skeleton Expressiveness Risk* and *Skeleton Stability Risk*. The former asks whether the skeleton can represent the observed behavior after coefficient fitting, whereas the latter asks whether the skeleton exhibits consistent behavior when its coefficients are re-estimated from different subset of the fitting data.

Skeleton expressiveness risk. We quantify this risk by a fingerprint-based bias score $\mathrm{Bias}(s)$. Specifically, let $\tilde{\boldsymbol{\theta}}_s$ denote the coefficients of skeleton s fitted on $\mathcal{D}_{\mathrm{fit}}$, and let $\tilde{f}_s(\boldsymbol{x}) = f_s(\boldsymbol{x}; \tilde{\boldsymbol{\theta}}_s)$. We evaluate $\tilde{f}_s$ on $\mathcal{D}_{\mathrm{audit}}$ and compare its fingerprint with that of the observed responses $\boldsymbol{y}_{\mathrm{audit}}$ over the same patches. For each patch P_m, we compute

$\Delta\mu_m = |\mu_m(\tilde{f}_s) - \mu_m(\boldsymbol{y}_{\mathrm{audit}})|, \Delta\boldsymbol{\beta}_m = \|\boldsymbol{\beta}_m(\tilde{f}_s) - \boldsymbol{\beta}_m(\boldsymbol{y}_{\mathrm{audit}})\|_2, \Delta\rho_m = |\rho_m(\tilde{f}_s) - \rho_m(\boldsymbol{y}_{\mathrm{audit}})|.$

The local expressiveness risk is

$$\mathrm{Bias}_m(s) = \omega_\mu \Delta\mu_m + \omega_\beta \Delta\boldsymbol{\beta}_m + \omega_\rho \Delta\rho_m,$$

and the overall risk is

$$\mathrm{Bias}(s) = \frac{1}{M} \sum_{m=1}^{M} \mathrm{Bias}_m(s).$$

A low $\mathrm{Bias}(s)$ indicates that the fitted skeleton reproduces the observed behavior across patches, whereas a high value indicates insufficient expressiveness. The nonnegative weights ω_μ, ω_β, and ω_ρ balance the three features. In practice, we assign a smaller weight to roughness because it is more sensitive to local fluctuations in finite patches.

Skeleton stability risk. Expressiveness alone cannot reveal whether the fitted behavior is reproducible or depends on the particular observations used for coefficient fitting. We characterize this dependence as skeleton stability risk. A stable skeleton should induce similar behavior when refitted on different resamples of the same observations, whereas a fragile skeleton relies on sample-specific coefficient configurations and therefore varies substantially across refits. We therefore quantify skeleton stability risk by the dispersion score $\mathrm{Var}(s)$ of the refitted function under different subset of the $\mathcal{D}_{\mathrm{fit}}$. Specifically, we generate B fitting subsets $\{\mathcal{D}_b\}_{b=1}^{B}$ by bootstrap resampling (Efron, 1992) of the $\mathcal{D}_{fit}$ and refit the coefficients of the same skeleton s on each subset. This yields

$$f_s^{(b)}(\boldsymbol{x}) = f_s(\boldsymbol{x}; \boldsymbol{\theta}_s^{(b)}), \qquad b = 1, \ldots, B$$

. All refitted functions are evaluated on the same audit patches. For each patch P_m, we measure the dispersion of the fingerprints produced by the $\{f_s^{(b)}(\boldsymbol{x})\}_{b=1}^{B}$:

$$\Sigma_{\mu,m} = \mathrm{std}_b[\mu_m(f_s^{(b)})], \Sigma_{\rho,m} = \mathrm{std}_b[\rho_m(f_s^{(b)})], \Sigma_{\beta,m} = \frac{1}{B} \sum_{b=1}^{B} \|\boldsymbol{\beta}_m(f_s^{(b)}) - \overline{\boldsymbol{\beta}}_m\|_2,$$

where $\overline{\boldsymbol{\beta}}_m = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\beta}_m(f_s^{(b)})$. The local stability risks of patch P_m are

$$\mathrm{Var}_m(s) = \omega_\mu \Sigma_{\mu,m} + \omega_\beta \Sigma_{\beta,m} + \omega_\rho \Sigma_{\rho,m},$$

and the overall stability risks of skeleton s can be defined as

$$\mathrm{Var}(s) = \frac{1}{M} \sum_{m=1}^{M} \mathrm{Var}_m(s).$$

A larger $\mathrm{Var}(s)$ indicates greater variation in the fitted behavior across resamples and thus higher skeleton stability risk.

Risk localization and diagnostic evidence. Each candidate s is positioned in the Diagnostic Risk Plane (DRP) by the coordinate $c_s = \big(\mathrm{Bias}(f_s), \mathrm{Var}(f_s)\big)$. Meanwhile, the patch-level quantities are retained as the evidence map

$$\mathcal{E}_s = \{(\Delta\mu_m, \Delta\boldsymbol{\beta}_m, \Delta\rho_m, \mathrm{Var}_m(s))\}_{m=1}^{M}.$$

Thus, c_s provides a compact risk localization for selecting an intervention, while $\mathcal{E}_s$ identifies the patches and behavioral features responsible for that diagnosis.

Using thresholds τ_B and τ_V, the DRP distinguishes four diagnostic states, each corresponding to a different failure mode and intervention strategy:

- **SETTLED** ($\mathrm{Bias} \le \tau_B, \mathrm{Var} \le \tau_V$): the skeleton is sufficiently expressive and stable. The intervention is therefore preservation-oriented: the skeleton is either **preserved** or only minimally refined according to the evidence map.
- **UNDERFIT** ($\mathrm{Bias} > \tau_B, \mathrm{Var} \le \tau_V$): the skeleton exhibits stable but systematic behavioral discrepancies, indicating missing or incorrect local structure. The candidate should be **repaired** by adding or replacing local symbolic components guided by the evidence map.
- **OVERFIT** ($\mathrm{Bias} \le \tau_B, \mathrm{Var} > \tau_V$): the skeleton matches the observations but is unstable under resampling, suggesting redundant or poorly identifiable structure. The candidate should be **simplified**.
- **MISMATCHED** ($\mathrm{Bias} > \tau_B, \mathrm{Var} > \tau_V$): the skeleton is neither expressive nor stable, indicating a more fundamental structural mismatch. The candidate should be **rewritten**.

The diagnostic coordinate determines *what* intervention is appropriate, while the evidence map indicates *where and how* the candidate fails. Both rely only on the candidate skeleton and observed data,enabling targeted diagnosis independent of the solver's internal search process.

3.4 DIAGNOSIS-GUIDED INTERVENTION AND SOLVER INTEGRATION

To reduce overhead, PostDiag first applies a low-cost NMSE-based gate to determine whether a candidate needs structured diagnosis. Candidates with either very low or very high errors are excluded from the PostDiag pipeline. Accurate candidates require no structural intervention, while poor candidates are left to the base solver's native search mechanism for elimination or regeneration. PostDiag therefore focuses on intermediate candidates, for which more informative feedback is needed to determine an appropriate intervention. NMSE serves only as a coarse screening signal, whereas PFP and DRP provide the evidence and diagnostic decision that guide the intervention.

For each candidate entering the diagnosis pipeline, PostDiag produces $\mathcal{G}_s = (c_s, \mathcal{E}_s)$. The risk region determined by c_s specifies the intervention type, while $\mathcal{E}_s$ identifies the patches and behavioral features that motivate the intervention. An LLM-based post-editor receives the original skeleton s, c_s, and $\mathcal{E}_s$, and proposes a diagnosis-guided revision $\widetilde{s}$. The LLM acts only as a conditional skeleton editor and does not participate in the solver's candidate generation or search control. Notably, a candidate entering the diagnostic process may still be classified as SETTLED. Although its scalar error is not yet low enough for direct acceptance, the DRP indicates that its skeleton is sufficiently expressive and stable. PostDiag still provides the LLM editor with the original skeleton, the preservation-oriented diagnosis, and the evidence map $\mathcal{E}_s$, but instructs it to either retain the skeleton or make only minor edits supported by the evidence.

We apply a lightweight validation before returning the proposed revision. Specifically, we refit the coefficients of $\widetilde{s}$ on $\mathcal{D}_{\mathrm{fit}}$ and evaluate its prediction error on $\mathcal{D}_{\mathrm{audit}}$. The revision is accepted if it reduces the error; otherwise, it is rejected and $\widetilde{s} = s$.

The revised skeleton can be integrated into different solvers via lightweight adapters. We consider two common solver families: GP-based and LLM-based solvers. For GP-based solvers, an accepted revision is inserted into the population as a repaired offspring. For LLM-based solvers, it replaces the original candidate, while $(s, \mathcal{G}_s, \widetilde{s})$ is appended to the prompt as a *diagnostic revision record*. Additional adapter details are provided in Appendix G. PostDiag leaves candidate generation, selection, and exploration to the base solver, deriving diagnoses solely from the candidate skeleton and observed data without accessing solver-specific search states or histories.

Table 1: ID performance of base solvers and their PostDiag-enhanced variants on LSR-Synth. We report accuracy to tolerance, NMSE and runtime (the mean end-to-end wall-clock seconds per task run, including both SR base solver and PostDiag overhead) over all 129 tasks. All metrics are aggregated over three seeds. A_τ abbreviates Acc_τ.

Category	Method / setting	Overall				Phys			MatSci			Chem			Bio		
		Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
LLM-based	Direct LLM	30.5	7.72E-02	2.6	0.8	7.32E-02	3.0	0.8	6.24E-02	2.7	1.3	3.50E-02	2.8	0.9	1.63E-01	1.4	0.0
	Direct LLM+PostDiag	33.7	**3.23E-02**	**4.4**	**2.3**	**5.22E-02**	**5.3**	**3.8**	**4.34E-03**	**4.0**	**2.7**	**8.76E-03**	**3.7**	0.9	**6.03E-02**	**4.2**	**1.4**
	SGA	320.0	1.39E-02	11.9	5.9	9.43E-03	8.3	6.1	8.21E-03	12.0	5.3	4.92E-03	19.4	7.4	4.15E-02	6.9	4.2
	SGA+PostDiag	350.2	**2.40E-03**	**18.6**	**9.8**	**2.66E-03**	**14.4**	**9.8**	**1.87E-04**	**25.3**	**6.7**	**8.65E-04**	**24.1**	**13.0**	**6.52E-03**	**11.1**	**8.3**
	LLM-SR	167.5	1.45E-03	22.0	8.8	3.66E-03	14.4	8.3	3.12E-05	37.3	13.3	1.75E-04	27.8	8.3	8.15E-04	11.1	5.6
	LLM-SR+PostDiag	191.7	**2.12E-04**	**31.0**	**15.2**	**3.48E-04**	**23.5**	**12.1**	**8.94E-06**	**60.0**	**30.7**	**5.84E-05**	**28.7**	**12.0**	**4.02E-04**	**18.1**	**9.7**
	LaSR	559.2	1.06E-02	13.4	7.5	1.73E-02	11.4	6.8	2.48E-03	14.7	6.7	6.90E-03	18.5	11.1	1.22E-02	8.3	4.2
	LaSR+PostDiag	595.4	**8.62E-04**	**21.4**	**12.7**	**1.52E-03**	**20.5**	**11.4**	**4.82E-05**	**29.3**	**14.7**	**6.45E-04**	**22.2**	**14.8**	**8.29E-04**	**13.9**	**9.7**
	SR-Scientist	648.0	1.12E-03	25.6	11.9	2.74E-03	17.4	9.8	4.47E-05	46.7	21.3	1.49E-04	27.8	11.1	7.26E-04	15.3	6.9
	SR-Scientist+PostDiag	702.2	**1.92E-04**	**35.7**	**20.4**	**3.11E-04**	**25.8**	**15.9**	**1.08E-05**	**62.7**	**37.3**	**6.47E-05**	**38.0**	**20.4**	**3.54E-04**	**22.2**	**11.1**
GP-based	PySR	8.8	2.67E-03	23.8	11.4	4.11E-03	21.2	12.1	1.70E-04	44.0	18.7	8.93E-04	20.4	10.2	5.29E-03	12.5	4.2
	PySR+PostDiag	10.5	**1.34E-03**	**28.7**	**13.2**	**1.82E-03**	**24.2**	**14.4**	2.25E-04	**50.7**	17.3	**5.29E-04**	**25.9**	**12.0**	**2.84E-03**	**18.1**	**8.3**
	gplearn	95.5	1.34E-01	0.0	0.0	8.51E-02	0.0	0.0	2.36E-02	0.0	0.0	6.56E-02	0.0	0.0	4.42E-01	0.0	0.0
	gplearn+PostDiag	101.7	**3.07E-02**	**1.0**	0.0	**4.29E-03**	**2.3**	0.0	**6.91E-03**	0.0	0.0	**4.29E-02**	**0.9**	0.0	**8.58E-02**	0.0	0.0

4 EXPERIMENT

4.1 EXPERIMENTAL SETUP

Dataset. Because PostDiag and most primary baselines use LLMs, we evaluate them on the LSR-Synth subset of LLM-SRBench (Shojaee et al., 2025b). Unlike LSR-Transform, which may remain vulnerable to pretraining contamination (Xia et al., 2025), LSR-Synth combines canonical domain terms with newly synthesized terms and filters tasks for solvability, novelty, and scientific plausibility. It contains 129 tasks across physics (44), materials science (25), chemistry (36), and biology (24), each with in-domain and out-of-domain test sets.

Metrics. We evaluate the discovered equations from 3 perspectives: numerical accuracy, accuracy-to-tolerance, and search efficiency. Numerical accuracy is measured by the normalized mean squared error (NMSE) between the predictions of the discovered equation f and the ground truth:

$$\mathrm{NMSE}(f) = \frac{\sum_{i=1}^{N_{\mathrm{test}}} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N_{\mathrm{test}}} (y_i - \bar{y})^2}.$$

where $\hat{y}_i$ is the prediction target value and y_i is the ground-truth value on $\boldsymbol{x}_i$. NMSE normalizes the error by the target variance, making it directly comparable across tasks with different output scales. We further report $\mathrm{Acc}_{0.01}$ following Kamienny et al. (2022); Shojaee et al. (2025a); Xia et al. (2025) to show the task-level solving accuracy at stringency levels:

$$\mathrm{Acc}_\tau(f) = \mathbb{1}\left(\max_{1 \le i \le N_{\mathrm{test}}} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \le \tau \right).$$

We report Acc_τ for $\tau \in \{.01, .001\}$ Xia et al. (2025); tables use A_τ. Search efficiency is measured by the average per-task search time (in seconds), which reflects the additional computational cost introduced by PostDiag.

Baselines. PostDiag is a search-independent diagnostic module rather than a standalone solver; we therefore compare each base solver with its PostDiag-enhanced version. We evaluate five LLM-based solvers: LLM-SR (Shojaee et al., 2025a), LaSR (Grayeli et al., 2024), SR-Scientist (Xia et al., 2025), Direct LLM (Shojaee et al., 2025b), and SGA (Ma et al., 2024); and two GP-based solvers, PySR (Cranmer, 2023) and gplearn (Stephens, 2024). Integration details are in Appendix G.

Configuration of PostDiag. We describe the default configuration of PostDiag used in all experiments. For the data split, each training dataset $\mathcal{D}$ is randomly partitioned into a fitting set $\mathcal{D}_{\mathrm{tr}}$ and an audit set $\mathcal{D}_{\mathrm{aud}}$ with a $75\%/25\%$ ratio. On $\mathcal{D}_{\mathrm{aud}}$, PostDiag in the experiments of LSR-Synth constructs $M = 64$ overlapping anchor-kNN patches using farthest-point anchors, where $k = 64$. We suggest that the number of anchors and the neighborhood size should be scaled to the audit-set size to guarantee that every audit sample belongs to at least one patch. For fingerprint discrepancies, the three feature dimensions are combined with weights $(\omega_\mu, \omega_\beta, \omega_\rho) = (0.4, 0.4, 0.2)$; the

smaller weight on roughness reflects its higher sensitivity to observation noise on small patches. Skeleton stability is estimated from $B = 3$ bootstrap refits, each of size $\lfloor \rho \, |\mathcal{D}_{\text{fit}}| \rfloor$ points drawn with replacement, where $\rho = 0.7$. The Diagnostic Risk Plane thresholds are setted as $\tau_B = 0.25$ and $\tau_V = 0.15$ based on preliminary experiments on a small held-out subset of tasks. We use these thresholds transfers across all four benchmark domains without per-domain re-tuning.

Search budget. To evaluate the effectiveness of PostDiag under limited search resources, we adopt low-budget configurations in the main experiments. For each task, LLM-SR is limited to 100 LLM calls; LaSR runs for 20 outer iterations; SR-Scientist runs for 10 outer iterations, each with at most 25 interaction turns; PySR runs for 65 iterations; and gplearn evolves for 100 generations. Direct LLM generates 5 candidates in Low and 10 in High from a single prompt batch; SGA runs 5 rounds (80 generated candidates) in Low and 20 rounds (320 generated candidates) in High. For each pair of a base solver and its PostDiag-augmented variant, we keep the base search budget, internal hyperparameters, LLM backbone, and random seeds identical. The diagnostic and structural intervention costs introduced by PostDiag are not deducted from the base solver's search budget, but are included in the reported end-to-end runtime. Measured results are aggregated over three random seeds. Table 1 uses Qwen3.5-27B for all LLM-based solvers and for PostDiag interventions on GP solvers; Appendix H reports the Qwen3.5-122B-A10B comparison, symbolic accuracy, budget efficiency, and additional backbones, while Appendix F reports OOD results.

4.2 Results of Different SR Solvers Integrated with PostDiag

Table 1 compares seven SR solvers with and without PostDiag on LSR-Synth. Across LLM-based solvers, model scales, and domains, PostDiag consistently reduces NMSE and improves $\mathrm{Acc}_{0.01}$, indicating that stronger LLM backbones do not eliminate the need for structured diagnostic feedback. The improvements also extend to GP-based solvers, although it cannot fully compensate for the gplearn's limited search capacity. These gains incur additional diagnostic cost from PostDiag, but require no modification to the underlying search mechanisms.

We evaluate all solvers under reduced search budgets to test whether PostDiag improves search efficiency. Larger budgets can mask differences in discovery speed as solvers approach convergence. Despite using far fewer search steps, PostDiag-enhanced solvers match or exceed results reported with much larger budgets. For example, in SR-Scientist (Xia et al., 2025), the authors report $\mathrm{Acc}_{0.01}$ of 30.49 for LLM-SR and 29.46 for their training-free variant with Qwen3-32B, using 1,000 LLM calls and 40 outer iterations. With Qwen3.5-27B, LLM-SR+PostDiag and SR-Scientist+PostDiag reach 31.0 and 35.7 using only 100 calls and 10 iterations, respectively, indicating improved search efficiency rather than gains from larger budgets.

4.3 Diagnostic Fidelity of the PostDiag

To assess whether PostDiag correctly identifies skeleton risks, we manually construct four labeled skeleton–data pair types. Settled pairs use the correct skeleton with varying Gaussian noise; Underfit pairs preserve the main structure and remove or replace one true term; Overfit pairs retain the correct skeleton and add redundant or unstable terms; and Mismatched pairs pair the data with an unrelated skeleton. Without access to these labels, PostDiag estimates Skeleton Bias and Variance via PFP and maps each candidate onto the DRP.

As shown in Figure 2a, the four pair types largely occupy their expected DRP regions. We observe a moderate overlap between underfit and mismatch near the stability boundary. This is expected, since both cases involve structural bias. When this bias is severe, coefficient refitting on resampled subsets may become unstable, making the boundary between an incomplete skeleton and a mismatched skeleton less sharp. Nevertheless, the clear quadrant structure shows that PFP and DRP provide a meaningful risk localization for different skeleton failure modes, yielding feedback with more explicit diagnostic information.

4.4 Ablation Study

To assess the contribution of different diagnostic signals, we compare three ablated variants with full PostDiag. "NMSE-only" provides only scalar candidate error to the LLM intervention module. "NMSE+DRP" infers skeleton adequacy and stability from NMSE-based signals and places the can-

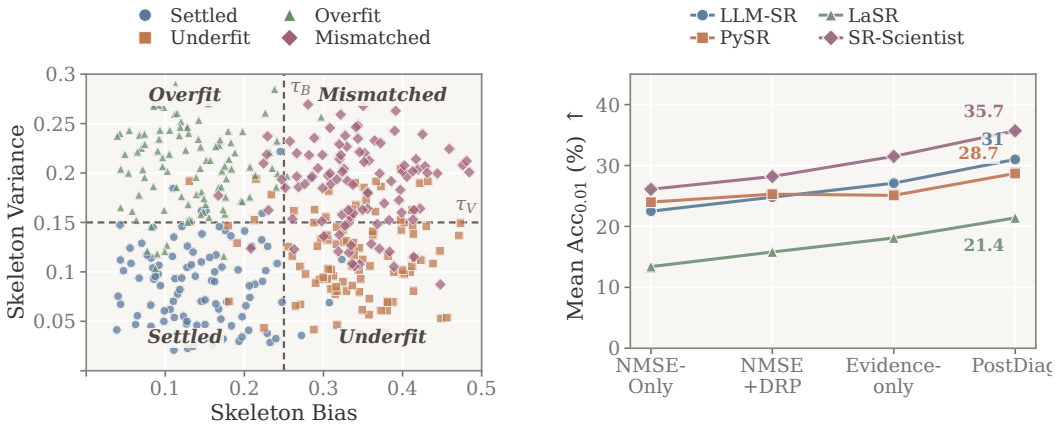


Figure 2: (a) Diagnostic fidelity of PostDiag. PFP-derived bias and variance scores map synthesis skeletons to distinct DRP regions, separating settled, underfit, overfit, and mismatched cases. (b) Component ablation across four base SR solvers. Each point reports pooled $\mathrm{Acc}_{0.01}$ over 129 LSR-Synth tasks and three seeds. Full PostDiag performs best by combining DRP localization with evidence-guided intervention.

didate on the DRP. "Evidence-only" supplies the PFP-based evidence map without DRP localization. "PostDiag" combines the evidence map, DRP localization, and explicit instructions to preserve, repair, simplify, or rewrite the skeleton. We evaluate all variants on LLM-SR, LaSR, SR-Scientist, and PySR. Results in Figure 2b are averaged across the four LSR-Synth domains. Qwen-27B serves as both the backbone for all LLM-based solvers and the intervention model for all four solvers.

As shown in Figure 2b, NMSE-only yields little improvement because the base solvers already rely on scalar fitness, so passing NMSE to the intervention model provides little additional information. NMSE+DRP improves the results, but its gains are limited. Estimating skeleton risk from a scalar NMSE is too coarse to reliably distinguish different failure modes, which can lead to inaccurate DRP localization. Evidence-only produces much larger improvements for LaSR and SR-Scientist, suggesting that these solvers benefit more from detailed information about where and how a candidate fails than from a coarse risk label. For PySR, NMSE+DRP is slightly more effective than Evidence-only, indicating that an explicit intervention operator is useful for guiding its population-based search. Full PostDiag performs best across all solvers, confirming that PFP, the evidence map, and DRP are complementary: PFP localizes risk, the evidence map reveals local defects, and DRP selects the appropriate structural intervention.

5 Conclusion and Limitations

This work addresses a feedback bottleneck shared by diverse SR solvers: scalar fitness can rank candidates but offers little guidance for structural refinement. Rather than developing a new solver, PostDiag separates feedback design from search design. Using only a candidate skeleton and observed data, it converts patch-wise behavior into expressiveness and stability risks, localized evidence, and targeted interventions. Experiments across LLM-based and GP-based solvers show that PostDiag improves accuracy and accelerates convergence without accessing or redesigning their internal search processes. These results establish post-diagnosis as an orthogonal way to strengthen existing SR solvers while preserving their distinctive search mechanisms.

Despite its search-independent diagnostic core, our current implementation is evaluated only on LLM-based and GP-based solvers, which provide natural interfaces for accepting revised candidates. Solvers without such interfaces, such as reinforcement-learning-based methods (Petersen et al., 2021), may require more substantial adaptation. Future work will extend the diagnostic interface to broader search paradigms and improve the efficiency of diagnosis and intervention.

THE USE OF LARGE LANGUAGE MODELS (LLMS)

Beyond the use of LLMs in the method described in this paper, LLMs served only as auxiliary tools for generating some of the cartoon characters in Figure 1 and for polishing and refining the manuscript's language. Their contributions to research conception, experimental design, or manuscript preparation do not warrant authorship or contributor status. All substantive research ideas, methods, and results presented in this work were independently developed or produced by the authors.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

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.

Yuntian Chen, Yingtao Luo, Qiang Liu, Hao Xu, and Dongxiao Zhang. Symbolic genetic algorithm for discovering open-form partial differential equations (sga-pde). *Physical Review Research*, 4 (2):023174, 2022.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

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

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

Fabricio Olivetti de Franca and Guilherme Seidyo Imai Aldeia. Interaction–transformation evolutionary algorithm for symbolic regression. *Evolutionary computation*, 29(3):367–390, 2021.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pp. 569–593. Springer, 1992.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

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.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep generative symbolic regression. *arXiv preprint arXiv:2401.00282*, 2023.

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.

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

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

Mario Krenn, Robert Pollice, Si Yue Guo, Matteo Aldeghi, Alba Cervera-Lierta, Pascal Friederich, Gabriel dos Passos Gomes, Florian Häse, Adrian Jinich, AkshatKumar Nigam, et al. On scientific understanding with artificial intelligence. *Nature Reviews Physics*, 4(12):761–769, 2022.

Qian Li, Yuxiao Hu, Juncheng Liu, and Yuntian Chen. Gensr: Symbolic regression based on equation generative space. In *The Fourteenth International Conference on Learning Representations*.

Wenqiang Li, Weijun Li, Linjun Sun, Min Wu, Lina Yu, Jingyi Liu, Yanjie Li, and Songsong Tian. Transformer-based model for symbolic regression via joint supervised learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Yunlun Li and Sinno Jialin Pan. Robust equation structure learning with adaptive refinement. In *International Conference on Learning Representations*, volume 2026, pp. 70029–70053, 2026.

Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. Llm and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. *arXiv preprint arXiv:2405.09783*, 2024.

Kazem Meidani, Parshin Shojaee, Chandan K Reddy, and Amir Barati Farimani. Snip: Bridging mathematical symbolic and numeric realms with unified pre-training. *arXiv preprint arXiv:2310.02227*, 2023.

Sebastian Mežnar, Sašo Džeroski, and Ljupčo Todorovski. Efficient generator of mathematical expressions for symbolic regression. *Machine Learning*, 112(11), 2023.

Alberto Moraglio, Krzysztof Krawiec, and Colin G Johnson. Geometric semantic genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pp. 21–31. Springer, 2012.

Xinyu Pang, Zhanke Zhou, Xuan Li, Fangrui Lv, Shanshan Wei, Sen Cui, Bo Han, and Changshui Zhang. Deliberate evolution: Agentic reasoning for sample-efficient symbolic regression with llms. *arXiv preprint arXiv:2606.04360*, 2026.

Tomasz P Pawlak and Krzysztof Krawiec. Competent geometric semantic genetic programming for symbolic regression and boolean function synthesis. *Evolutionary computation*, 26(2):177–212, 2018.

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

Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, February 2026. Accessed: 2026-07-22.

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

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

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

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K Reddy. Llm-srbench: A new benchmark for scientific equation discovery with large language models. *arXiv preprint arXiv:2504.10415*, 2025b.

Trevor Stephens. gplearn: Genetic programming in python, with a scikit-learn inspired api, 2024. *URL https://github. com/trevorstephens/gplearn. Accessed*, pp. 05–22, 2024.

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

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

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33:4860–4871, 2020.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

Shijie Xia, Yuhan Sun, and Pengfei Liu. Sr-scientist: Scientific equation discovery with agentic ai. *arXiv preprint arXiv:2510.11661*, 2025.

Chuyang Xiang, Yichen Wei, and Junchi Yan. Mod-sr: Unifying multimodal learning and direct optimization with gradient-guided diffusion model for symbolic regression. In *Forty-third International Conference on Machine Learning*.

Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *Foundations and Trends® in Machine Learning*, 18(4):385–849, 2025.

A Reproducibility Statement

For the reproducibility of our results, we have provided a detailed description of our methods and experimental setups in Sec. 4.1 and Appendix G, C, E. We also confirmed the robustness of our results through the extensive experiments. In addition, to further facilitate the reproduction, we will release our codes and the checkpoints for the trained models.

B Dataset Details

B.1 Task Composition

We conduct all experiments on LSR-Synth, the synthetic subset of LLM-SRBench (Shojaee et al., 2025b). Following the evaluation setting used by SR-Scientist (Xia et al., 2025), the benchmark contains 129 symbolic regression tasks from four scientific domains: 44 in physics, 25 in material science, 36 in chemistry, and 24 in biology.

Representative tasks include damped oscillators, stress–strain relations, chemical kinetics, and population dynamics. Each task provides numerical observations together with the variable names and their scientific meanings. Its ground-truth expression combines established domain-specific terms with novel synthetic terms. Consequently, a solver cannot rely solely on memorized canonical equations and must instead use the observations to search for previously unseen structures. Compared with LSR-Transform, which may be more susceptible to memorization from pretraining, LSR-Synth is therefore better suited to evaluating data-driven equation discovery.

B.2 Data Generation and Splits

The construction of LSR-Synth consists of scientific-problem selection, known- and synthetic-term generation, parameter instantiation, numerical solvability and novelty checks, data generation, and expert validation. For each scientific problem, candidate known and synthetic terms are first generated and combined into complete expressions, whose coefficients are instantiated within scientifically meaningful ranges. For the dynamical systems in chemistry, biology, and physics, data are generated through numerical integration using SciPy's `solve_ivp` function with the RK45 method. The static stress–strain relationships in material science are evaluated directly over predefined input ranges. Candidates that cannot be solved reliably, produce divergent or degenerate behavior, or are excessively similar to existing tasks are filtered out. The retained expressions and visualizations of their generated data are then cross-checked by domain experts.

Each task contains 5,000 observations. Chemistry, biology, and physics tasks are sampled uniformly over the time interval $t \in [0, 60]$, whereas material-science tasks cover strain $\epsilon \in [0, 0.6]$ and temperature $T \in [273, 573]$ K. For time-dependent tasks, the 500 observations with the largest time values form the out-of-distribution (OOD) test set. For material-science tasks, the 500 observations with the largest temperature values form the OOD test set. Of the remaining 4,500 observations, 500 are randomly selected as the in-distribution (ID) test set, and the other 4,000 observations constitute the training set provided to the solver.

B.3 Data-Usage Protocol for PostDiag

All diagnosis and intervention operations performed by PostDiag are restricted to the training split and are strictly isolated from both test sets. Under the default configuration described in the main text, the training data for each task are further divided into a fitting set $\mathcal{D}_{\mathrm{tr}}$ and an audit set $\mathcal{D}_{\mathrm{aud}}$ using a $75\%/25\%$ split. Candidate coefficient fitting and bootstrap refitting use only $\mathcal{D}_{\mathrm{tr}}$, whereas the PFP fingerprints, skeleton adequacy and stability scores, and evidence maps are computed on $\mathcal{D}_{\mathrm{aud}}$. The acceptance check for an intervened candidate also uses only the training split.

The ID and OOD test sets are used solely to evaluate the final expression after search terminates. They are never used for candidate generation, diagnosis, structural intervention, rewrite acceptance, or candidate selection. Although an LLM-based base solver may consume the task and variable descriptions supplied by the benchmark, PostDiag itself receives only a candidate skeleton and the numerical observations from the training split.

C BASE SOLVER CONFIGURATIONS

The low-budget settings used in the main experiments are specified in Section 4.1, while the high-budget reference settings are described in Appendix H.3. To avoid repetition, this section does not restate the limits on LLM calls, outer iterations, or evolutionary generations. Instead, we report the internal configurations that remain fixed across budget levels and between each base solver and its PostDiag-augmented counterpart. Unless otherwise specified, we build on the official implementation of each method and keep its prompt templates, search operators, coefficient-fitting procedure, and candidate-evaluation protocol unchanged. The detailed integration of PostDiag with each solver is provided in Appendix G.

C.1 SEARCH-BUDGET PROTOCOL

The Low and High settings retain each solver's native search unit. Within each base/PostDiag pair, the native budget is identical, while PostDiag overhead is included in the reported end-to-end runtime.

Solver	Low	High	Native budget unit
Direct LLM	5	10	generated candidates
SGA	5 (80)	20 (320)	rounds (generated candidates)
LLM-SR	100	1,000	LLM calls
LaSR	20	40	outer iterations
SR-Scientist	$10 \times \leq 25$	$40 \times \leq 25$	outer iterations $\times$ turns
PySR	65	130	iterations
gplearn	100	200	generations

C.2 LLM-BASED SOLVERS

Direct LLM. We follow the DataBlind Direct Prompting protocol of LLM-SRBench (Shojaee et al., 2025b). The LLM proposes equation skeletons from the scientific context without observing the numerical samples, while the numerical constants in each proposed skeleton are fitted by the benchmark evaluation pipeline. The PostDiag variant retains this direct generator and adds data-driven diagnosis and intervention after candidate evaluation; thus, DataBlind describes the base generator rather than the complete augmented pipeline. Low and High generate 5 and 10 candidates, respectively, from a single prompt batch.

SGA. SGA (Ma et al., 2024) uses a bilevel procedure in which an LLM proposes discrete scientific hypotheses and simulation-based optimization fits their continuous parameters. Its PostDiag variant attaches diagnostic feedback outside this bilevel loop, preserving the underlying hypothesis generation and continuous parameter optimization. Low uses 5 rounds and High uses 20 rounds, with 16 generated candidate attempts per round; the continuous-parameter optimizer is unchanged across budgets.

LLM-SR. LLM-SR (Shojaee et al., 2025a) represents candidate equations as Python programs containing placeholders for numerical constants. An LLM proposes program skeletons conditioned on the task description and high-performing programs retrieved from an experience pool, while the numerical constants are fitted using BFGS. We retain the prompt templates, multi-island experience pool, candidate-ranking procedure, and coefficient-optimization pipeline of the official implementation. Apart from the search budgets specified elsewhere, all configurations are shared across LLM backbones and between the base and PostDiag-augmented variants.

LaSR. LaSR (Grayeli et al., 2024) maintains an LLM-induced library of natural-language concepts on top of a multi-population genetic-programming search. We use 10 populations with 33 individuals per population and run 550 GP cycles within each outer iteration. The maximum expression size is 30, and the concept library contains at most 20 concepts. The weights assigned to LLM-assisted crossover, mutation, and randomization are all set to 0.02. The symbolic operators and nested constraints follow the PySR configuration described below.

SR-Scientist. SR-Scientist (Xia et al., 2025) equips an LLM agent with data-analysis and equation-evaluation tools, allowing it to write and execute code while iteratively improving candidate equations. We retain its official tool interfaces, experience buffer, and candidate-evaluation procedure. Each search iteration uses a target MAPE of 0.1% and terminates early when the error reaches $10^{-4}\%$. The LLM backbone and generation settings are held fixed between each base/PostDiag pair.

C.3 GP-Based Solvers

PySR. PySR (Cranmer, 2023) performs island-based genetic programming using a Julia backend, together with expression-complexity regularization and numerical constant optimization. We use 15 populations with 33 individuals per population and run 550 GP cycles within each iteration. The maximum expression size is 50. The binary operator set is $\{+, -, \times, \div, \mathrm{pow}\}$, and the unary operator set is $\{\exp, \log, \sqrt{\cdot}, \sin, \cos\}$.

gplearn. gplearn (Stephens, 2024) implements single-population tree-based genetic programming. We use a population size of 500 and a tournament size of 20. The function set contains addition, subtraction, multiplication, division, sine, cosine, square root, logarithm, absolute value, negation, and inverse. Other parameters, including the parsimony coefficient and the crossover and mutation probabilities, retain the defaults of the official implementation.

D LLM Backbones

D.1 Model Selection

Our experiments use Qwen3.5-27B and Qwen3.5-122B-A10B as the principal LLM backbones (Qwen Team, 2026). Qwen3.5-27B is reported in the main table, while Qwen3.5-122B-A10B is reported in the appendix. Qwen3.5-27B is a 27B-parameter dense model, whereas Qwen3.5-122B-A10B adopts a mixture-of-experts architecture with 122B total parameters and approximately 10B active parameters. Although both models belong to the same family, they differ in capacity and architecture, allowing us to evaluate whether PostDiag remains effective across different backbone configurations.

We further report supplementary results with Qwen3.5-9B and gpt-oss-20b (Agarwal et al., 2025) in the appendix. Qwen3.5-9B evaluates PostDiag with a smaller model and, together with the two principal Qwen3.5 backbones, provides coverage across different model capacities. The gpt-oss-20b experiments provide complementary evidence from a different model family and help assess whether PostDiag depends on a particular family of LLMs.

D.2 Backbone Consistency

For the five LLM-based base solvers, LLM-SR, LaSR, SR-Scientist, Direct LLM, and SGA, each base method and its PostDiag-augmented counterpart use the same LLM backbone. For example, when a base solver uses Qwen3.5-27B, the corresponding PostDiag structural intervention is also performed by Qwen3.5-27B. This avoids confounding the effect of PostDiag with differences in backbone capability and keeps the primary distinction between each pair to whether PostDiag is applied.

For PySR and gplearn, whose original search procedures do not rely on an LLM, the base search remains unchanged. When PostDiag is added, we use Qwen3.5-27B consistently to perform the diagnosed structural interventions.

D.3 PostDiag Configuration

Each base solver retains its original configuration. Between a base method and its PostDiag-augmented counterpart, we keep the base search budget, internal hyperparameters, and random seeds fixed so that the comparison focuses on the diagnosis and structural intervention introduced by PostDiag.

For each PostDiag structural intervention, we set the sampling temperature to 0.3 and the maximum completion length to 8,192 tokens. Conditioned on the DRP state and Evidence Map, the model returns a structured preserve-or-revise decision, a revised expression skeleton, and the corresponding edit type. The LLM proposes only the structural revision and does not determine the final numerical coefficients. After parsing and validity checking, the coefficients of the resulting expression are refitted. We use the same PostDiag configuration for all base solvers and LLM backbones.

E Compute Resources

Experiments are conducted on a compute cluster equipped with 16 NVIDIA A800 GPUs and 8 NVIDIA H800 GPUs. The LLM backbones are served locally using vLLM. PySR, gplearn, and the diagnostic computation of PostDiag run primarily on CPUs, while GPUs are used mainly for LLM inference. For each base solver, all reported runtime configurations—including Low, Low+PostDiag, and High—use the same GPU type, GPU count, vLLM serving configuration, and server-side concurrency limit. Each method retains its native request-scheduling pattern: LLM-SR issues up to four concurrent requests, whereas the interaction trajectory of SR-Scientist is executed sequentially; the other solvers follow their official implementations.

We report the mean end-to-end wall-clock time per task, measured from the start of the base solver's search until it returns the final expression. The reported time includes base search, candidate evaluation, and coefficient fitting. For PostDiag-augmented variants, it also includes the overhead of diagnostic computation, coefficient refitting, and additional LLM-based structural intervention. All runtime measurements are averaged over three random seeds.

F Out-of-Distribution Performance

LSR-Synth provides an independent out-of-distribution (OOD) test set for each task, with samples extending beyond the training range along a control variable such as time or temperature. The detailed split construction is provided in Appendix B. Following SR-Scientist (Xia et al., 2025), we report OOD $\mathrm{Acc}_{0.01}$. After search terminates, the final expression returned by each method is evaluated directly on the OOD test set, without additional search, coefficient refitting, or candidate selection. The OOD data remain unseen throughout search, diagnosis, and structural intervention.

As shown in Table 2, PostDiag generally improves OOD accuracy across different base solvers and LLM backbones. With Qwen3.5-27B, the overall OOD $\mathrm{Acc}_{0.01}$ of LLM-SR increases from 21.2% to 28.9%, while that of SR-Scientist increases from 24.8% to 33.6%. With Qwen3.5-122B-A10B, the overall OOD accuracy of SR-Scientist increases from 40.6% to 48.3%. PySR also improves from 27.1% to 31.0%, suggesting that the extrapolation benefits of PostDiag are not limited to LLM-based solvers. In contrast, gplearn and its PostDiag-augmented variant both obtain zero OOD accuracy, indicating that diagnosis and structural intervention cannot fully compensate for insufficient search capability in the base solver.

The gains remain substantial under OOD evaluation, indicating that the benefits of PostDiag are largely preserved on out-of-distribution data. In material science, OOD accuracy is generally higher than the corresponding ID accuracy for both the base and PostDiag-augmented methods.

G Details of Integration with Base Solvers

PostDiag is designed to be attached to existing SR solvers through lightweight adapters. We describe the integration recipe for each of the seven representative solvers in this section. The adapters handle representation conversion, diagnostic feedback, and candidate-pool insertion at the solver boundary, while the solver's core candidate generation, selection, and search-update rules remain unchanged.

Direct LLM (Shojaee et al., 2025b) generates equation skeletons from the scientific context without observing numerical samples; numerical constants are fitted by the benchmark pipeline. In our PostDiag, generated skeleton is diagnosed and revised. The accepted revisions enter the selected candidate pool, from which the final result is saved. The base generator remains DataBlind, while the augmented candidate evaluation is explicitly data-driven.

Table 2: Low-budget OOD $\mathrm{Acc}_{0.01}$ on LSR-Synth. Following the SR-Scientist protocol, only $\mathrm{Acc}_{0.01}$ is reported for OOD evaluation. Each LLM-based method is paired with its PostDiag variant under the same backbone. Overall is pooled over all 129 tasks; OOD samples are used only for post-hoc evaluation.

Backbone/category	Method	Overall	Phys	MatSci	Chem	Bio
	Qwen3.5-27B					
LLM-based	Direct LLM (Shojaee et al., 2025b)	1.0	0.8	2.7	0.9	0.0
	Direct LLM (Shojaee et al., 2025b)+PostDiag	**2.3**	**2.3**	2.7	**2.8**	**1.4**
	SGA (Ma et al., 2024)	9.6	6.1	9.3	16.7	5.6
	SGA (Ma et al., 2024)+PostDiag	**16.3**	**12.9**	**21.3**	**21.3**	**9.7**
	LLM-SR	21.2	12.9	41.3	26.9	6.9
	LLM-SR+PostDiag	**28.9**	**20.5**	**61.3**	**27.8**	**12.5**
	LaSR	12.4	10.6	16.0	15.7	6.9
	LaSR+PostDiag	**19.1**	**17.4**	**30.7**	**18.5**	**11.1**
	SR-Scientist	24.8	16.7	50.7	25.0	12.5
	SR-Scientist+PostDiag	**33.6**	**22.7**	**64.0**	**36.1**	**18.1**
	Qwen3.5-122B-A10B					
LLM-based	Direct LLM (Shojaee et al., 2025b)	1.8	2.3	1.3	1.9	1.4
	Direct LLM (Shojaee et al., 2025b)+PostDiag	**3.1**	**3.8**	**2.7**	**2.8**	**2.8**
	SGA (Ma et al., 2024)	11.9	8.3	12.0	18.5	8.3
	SGA (Ma et al., 2024)+PostDiag	**18.3**	**15.2**	**21.3**	**24.1**	**12.5**
	LLM-SR	26.6	18.9	53.3	25.9	13.9
	LLM-SR+PostDiag	**34.1**	**24.2**	**66.7**	**32.4**	**20.8**
	LaSR	14.0	13.6	18.7	15.7	6.9
	LaSR+PostDiag	**21.7**	**20.5**	**34.7**	**20.4**	**12.5**
	SR-Scientist	40.6	27.3	78.7	40.7	25.0
	SR-Scientist+PostDiag	**48.3**	**35.6**	**81.3**	**50.0**	**34.7**
	GP-based methods					
GP-based	PySR	27.1	20.5	68.0	15.7	13.9
	PySR+PostDiag	**31.0**	**22.7**	**74.7**	**20.4**	**16.7**
	gplearn	0.0	0.0	0.0	0.0	0.0
	gplearn+PostDiag	0.0	0.0	0.0	0.0	0.0

SGA (Ma et al., 2024) alternates LLM-based discrete hypothesis generation with simulation-based continuous parameter optimization. The proposed adapter parses each generated equation into SGA's trainable PyTorch module, fits its parameters with the unchanged inner optimizer, and adds valid fitted candidates to the candidate pool. PostDiag diagnoses a fitted candidate, proposes a structural revision, and sends the parsed revision through the same module construction, fitting, and evaluation path before it becomes eligible for the selected pool.

LLM-SR (Shojaee et al., 2025a) treats equation discovery as evolutionary program search, where an LLM proposes candidate skeletons conditioned on in-context examples drawn from a buffer of high-performing programs. PostDiag integrates through this buffer interface: for every newly admitted elite candidate s, PostDiag produces its diagnostic label and evidence map; the rewritten skeleton $\widetilde{s}$ together with a brief textual annotation of the diagnosed failure mode is stored. Subsequent prompts draw both score-ranked and diagnosis-annotated examples, biasing the LLM's next proposals toward the identified failure mode.

LaSR (Grayeli et al., 2024) couples LLMs with genetic programming by periodically abstracting high-performing hypotheses into natural-language concepts, which then guide subsequent symbolic mutation. PostDiag integrates at the concept abstraction stage: when LaSR queries the LLM for concept induction, we additionally supply the diagnostic label and evidence map of the input hypotheses, so that the abstracted concepts carry structural failure-mode information. The resulting diagnosis-aware concepts guide the subsequent GP search toward diagnosed structural fixes, while LaSR's underlying GP kernel remains untouched.

SR-Scientist (Xia et al., 2025) frames equation discovery as an autonomous agent that reads data, writes code to test hypotheses, and iterates on candidates through a tool-use loop that already accepts arbitrary numerical feedback signals. PostDiag integrates as an additional `diagnose(candidate)` tool that the agent can invoke after any candidate evaluation; the tool

Table 3: Low-budget ID results on LSR-Synth for LLM-based solvers using Qwen3.5-122B-A10B. Metrics and aggregation follow Table 1. Each method is paired with its PostDiag variant under the same backbone and three-seed protocol.

Method / setting	Overall				Phys			MatSci			Chem			Bio		
	Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
Direct LLM (Shojaee et al., 2025b)	35.3	4.62E-02	2.8	1.3	3.82E-02	3.0	0.8	5.78E-02	2.7	1.3	2.26E-02	3.7	1.9	8.42E-02	1.4	1.4
Direct LLM (Shojaee et al., 2025b)+PostDiag	39.3	**9.54E-03**	**5.4**	**3.4**	**8.64E-03**	**6.1**	**3.8**	**2.86E-03**	**5.3**	**4.0**	**6.43E-03**	**4.6**	**2.8**	**2.28E-02**	**5.6**	**2.8**
SGA (Ma et al., 2024)	368.0	8.44E-03	14.7	8.8	5.88E-03	12.1	7.6	6.17E-03	13.3	8.0	5.10E-03	22.2	13.0	2.05E-02	9.7	5.6
SGA (Ma et al., 2024)+PostDiag	406.2	**1.44E-03**	**22.7**	**12.7**	**1.52E-03**	**20.5**	**12.9**	**2.06E-04**	**25.3**	**10.7**	**7.32E-04**	**28.7**	**15.7**	**3.66E-03**	**15.3**	**9.7**
LLM-SR	201.1	9.68E-04	27.6	12.7	2.41E-03	21.2	9.8	1.91E-05	50.7	26.7	1.18E-04	27.8	9.3	5.86E-04	15.3	8.3
LLM-SR+PostDiag	231.7	**1.31E-04**	**36.4**	**19.9**	**2.13E-04**	**28.0**	**15.9**	**5.82E-06**	**65.3**	**38.7**	**3.86E-05**	**36.1**	**15.7**	**2.47E-04**	**22.2**	**13.9**
LaSR	716.3	8.71E-03	15.8	9.3	1.47E-02	14.4	7.6	1.92E-03	17.3	9.3	5.60E-03	19.4	13.0	9.45E-03	11.1	6.9
LaSR+PostDiag	762.1	**6.76E-04**	**24.5**	**13.7**	**1.19E-03**	**23.5**	**10.6**	**3.76E-05**	**33.3**	**16.0**	**4.96E-04**	**25.0**	**17.6**	**6.68E-04**	**16.7**	**11.1**
SR-Scientist	833.6	5.96E-04	41.3	24.3	1.43E-03	29.5	16.7	2.03E-05	74.7	50.7	8.12E-05	41.7	21.3	4.37E-04	27.8	15.3
SR-Scientist+PostDiag	902.2	**1.04E-04**	**48.8**	**31.3**	**1.76E-04**	**36.4**	**22.7**	**6.08E-06**	**78.7**	**58.7**	**3.04E-05**	**50.9**	**29.6**	**1.82E-04**	**37.5**	**20.8**

returns the structured diagnostic report and evidence map, which are then included in the agent's next reasoning step. The integration only exposes a new tool endpoint and does not modify SR-Scientist's control flow.

PySR (Cranmer, 2023) performs island-based genetic programming with configurable custom operators and supports external individual injection through its population interface. PostDiag integrates through diagnosis-driven immigration: at the end of each generation, PostDiag diagnoses the best-performing individual. The candidate is revised through diagnosis-guided intervention and injected into the initial population of the next generation as an immigrant, thereby steering the subsequent evolutionary search. Since PySR natively supports the insertion of external individuals, the integration requires only a lightweight adapter that converts Python expressions into expression trees, without modifying its selection, mutation, or crossover operators.

gplearn (Stephens, 2024) implements classical tree-based genetic programming and exposes `init_method` and `warm_start` for warm-starting the population from previously fitted regressors. PostDiag integrates by feeding the rewritten skeletons $\widetilde{s}$ as seeded individuals into the warm-started initial population of the next generation. Since gplearn's tree grammar covers all operators used in PostDiag, the adapter is a thin conversion between our program representation and gplearn's internal tree encoding, and does not alter gplearn's selection or reproduction operators.

Across all seven solvers, PostDiag exposes the same standardized output, $(\mathcal{G}_s, \widetilde{s})$. With only lightweight implementation changes, it equips each solver with post-diagnosis capabilities while leaving the core search algorithm intact. PostDiag therefore represents a complementary technical direction that is orthogonal to the development of new SR algorithms.

H Additional Experimental Results

Unless otherwise specified, we use Qwen3.5-27B as the default experimental backbone, while Qwen3.5-122B-A10B is included to evaluate PostDiag with a stronger model. Since PySR and gplearn do not natively use an LLM, their PostDiag-augmented variants consistently use the default Qwen3.5-27B backbone.

H.1 Low-Budget Results with Qwen3.5-122B-A10B

Table 3 reports the counterpart of the main ID comparison using Qwen3.5-122B-A10B. Each base/-PostDiag pair retains the same search configuration and random seeds, with the LLM backbone shared between the base solver and the PostDiag intervention.

At the Overall level, PostDiag lowers NMSE and improves both accuracy thresholds for all solvers under Qwen3.5-122B-A10B. The stronger backbone therefore continues to benefit from diagnostic feedback, while the final-accuracy gap between Direct LLM and the iterative solvers remains, indicating that the upper of performance still depends on the underlying search procedure.

H.2 SYMBOLIC ACCURACY

We additionally evaluate whether the final expression is symbolically equivalent to the ground-truth expression. Following LLM-SRBench, we remove fitted parameters and constants before judging equivalence so that the metric measures recovery of the functional structure rather than numerical coefficient matching. We then adopt the more conservative two-stage adjudication procedure of SR-Scientist: GPT-OSS-120B evaluates each pair independently ten times, unanimous decisions are retained, and non-unanimous cases are resolved by human evaluators. Failed runs and invalid final expressions are counted as incorrect. Table 4 reports the pooled symbolic accuracy over all tasks and three seeds.

Table 4: Symbolic accuracy (SA) on LSR-Synth. We follow the LLM-SRBench definition and expression preprocessing, together with the SR-Scientist repeated-vote adjudication protocol. Results pool all 129 tasks over three seeds; failed or invalid runs count as incorrect. PostDiag interventions for GP solvers use Qwen3.5-27B. Boldface marks the better value within each base/PostDiag pair.

Base solver	Backbone	Base SA	+PostDiag SA	Δ SA
Direct LLM	Qwen3.5-27B	1.55	1.55	0.00
	Qwen3.5-122B-A10B	2.07	**2.58**	+0.52
SGA	Qwen3.5-27B	3.62	**4.65**	+1.03
	Qwen3.5-122B-A10B	3.88	**5.17**	+1.29
LLM-SR	Qwen3.5-27B	3.88	**5.68**	+1.81
	Qwen3.5-122B-A10B	4.39	**6.46**	+2.07
LaSR	Qwen3.5-27B	1.81	**3.62**	+1.81
	Qwen3.5-122B-A10B	2.58	**4.91**	+2.33
SR-Scientist	Qwen3.5-27B	4.91	**7.24**	+2.33
	Qwen3.5-122B-A10B	6.72	**9.56**	+2.84
PySR	N/A	4.13	**5.68**	+1.55
gplearn	N/A	0.00	0.00	0.00

PostDiag raises pooled SA for most solver–backbone pairs, with gains ranging from 0.52 points for Direct LLM under Qwen3.5-122B-A10B to 2.84 points for SR-Scientist under the same backbone. SGA gains 1.03 and 1.29 percentage points under Qwen3.5-27B and Qwen3.5-122B-A10B, respectively. Direct LLM under Qwen3.5-27B remains at 1.55%, and gplearn remains at 0.00%.

H.3 BUDGET-EFFICIENCY ANALYSIS

We compare PostDiag-augmented solvers against their unaugmented counterparts under the Low and High settings defined in Appendix C.1. The High settings are 10 generated candidates for Direct LLM (Shojaee et al., 2025b); 20 rounds, or 320 generated candidate attempts, for SGA (Ma et al., 2024); 1,000 LLM calls for LLM-SR; 40 outer iterations for LaSR; 40 outer iterations with at most 25 interaction turns each for SR-Scientist; 130 iterations for PySR; and 200 generations for gplearn. Low+PostDiag retains the same native base-solver budget as Low. We report accuracy to tolerance as the primary criterion, arithmetic mean NMSE, and mean end-to-end wall-clock time per task run. Tables 5, 6, and 7 evaluate solution quality at matched native budgets and compare low-budget PostDiag variants with the corresponding high-budget base solvers.

The Low+PostDiag and High comparisons show that diagnosis and additional search are complementary rather than interchangeable. Direct LLM+PostDiag under Qwen3.5-27B exceeds High in Overall accuracy at similar runtime, whereas under Qwen3.5-122B-A10B it has higher accuracy but worse NMSE, exposing a metric tradeoff. For SGA under both backbones, Low+PostDiag approaches the High setting's quality with substantially less end-to-end time, while High retains higher accuracy and lower NMSE. The remaining solvers show similarly method-dependent balances between diagnostic overhead and the benefit of a larger native search budget.

Table 5: Budget-efficiency comparison for LLM-based solvers using Qwen3.5-27B. Each method is paired with its PostDiag variant. Low+PostDiag retains the corresponding Low base-solver budget, while High uses the full-budget reference setting. Boldface marks the best value within each solver block; runtime is reported without ranking.

Method / setting	Overall				Phys			MatSci			Chem			Bio		
	Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
Direct LLM (Shojaee et al., 2025b)	30.5	7.72E-02	2.6	0.8	7.32E-02	3.0	0.8	6.24E-02	2.7	1.3	3.50E-02	2.8	0.9	1.63E-01	1.4	0.0
Direct LLM (Shojaee et al., 2025b)+PostDiag	33.7	**3.23E-02**	**4.4**	**2.3**	**5.22E-02**	**5.3**	**3.8**	**4.34E-03**	4.0	**2.7**	**8.76E-03**	**3.7**	0.9	**6.03E-02**	**4.2**	**1.4**
Direct LLM / High	45.5	5.78E-02	3.4	1.8	5.86E-02	3.0	1.5	3.98E-02	**5.3**	**2.7**	2.80E-02	**3.7**	**1.9**	1.20E-01	1.4	**1.4**
SGA (Ma et al., 2024)	320.0	1.39E-02	11.9	5.9	9.43E-03	8.3	6.1	8.21E-03	12.0	5.3	4.92E-03	19.4	7.4	4.15E-02	6.9	4.2
SGA (Ma et al., 2024)+PostDiag	350.2	2.40E-03	18.6	9.8	2.66E-03	14.4	9.8	**1.87E-04**	**25.3**	6.7	8.65E-04	24.1	**13.0**	6.52E-03	11.1	**8.3**
SGA / High	1250.0	**1.70E-03**	**19.9**	**10.6**	**1.82E-03**	**15.9**	**12.1**	2.27E-04	22.7	**8.0**	**4.33E-04**	**26.9**	**13.0**	**4.89E-03**	**13.9**	6.9
LLM-SR / Low	167.5	1.45E-03	22.0	8.8	3.66E-03	14.4	8.3	3.12E-05	37.3	13.3	1.75E-04	27.8	8.3	8.15E-04	11.1	5.6
LLM-SR+PD / Low	191.7	**2.12E-04**	31.0	**15.2**	**3.48E-04**	23.5	12.1	**8.94E-06**	**60.0**	**30.7**	**5.84E-05**	28.7	**12.0**	**4.02E-04**	18.1	**9.7**
LLM-SR / High	1603.4	3.65E-04	**34.1**	13.2	7.82E-04	**28.0**	**12.9**	1.20E-05	53.3	22.7	7.21E-05	**37.0**	11.1	4.08E-04	**20.8**	6.9
LaSR / Low	559.2	1.06E-02	13.4	7.5	1.73E-02	11.4	6.8	2.48E-03	14.7	6.7	6.90E-03	18.5	11.1	1.22E-02	8.3	4.2
LaSR+PD / Low	595.4	**8.62E-04**	**21.4**	**12.7**	**1.52E-03**	**20.5**	**11.4**	**4.82E-05**	**29.3**	**14.7**	**6.45E-04**	**22.2**	**14.8**	**8.29E-04**	**13.9**	**9.7**
LaSR / High	1118.4	8.13E-03	15.5	8.8	1.36E-02	14.4	7.6	1.73E-03	17.3	9.3	5.10E-03	19.4	12.0	9.30E-03	9.7	5.6
SR-Scientist / Low	648.0	1.12E-03	25.6	11.9	2.74E-03	17.4	9.8	4.47E-05	46.7	21.3	1.49E-04	27.8	11.1	7.26E-04	15.3	6.9
SR-Scientist+PD / Low	702.2	**1.92E-04**	35.7	20.4	**3.11E-04**	25.8	**15.9**	**1.08E-05**	62.7	37.3	6.47E-05	**38.0**	**20.4**	3.54E-04	22.2	**11.1**
SR-Scientist / High	2547.1	2.23E-04	**37.7**	**21.2**	4.47E-04	**26.5**	**15.9**	1.31E-05	**70.7**	**48.0**	**5.03E-05**	**38.0**	16.7	**2.91E-04**	**23.6**	9.7

Table 6: Budget-efficiency comparison for LLM-based solvers using Qwen3.5-122B-A10B. Each method is paired with its PostDiag variant. Low+PostDiag retains the corresponding Low base-solver budget, while High uses the full-budget reference setting. Boldface marks the best value within each solver block; runtime is reported without ranking.

Method / setting	Overall				Phys			MatSci			Chem			Bio		
	Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
Direct LLM (Shojaee et al., 2025b)	35.3	4.62E-02	2.8	1.3	3.82E-02	3.0	0.8	5.78E-02	2.7	1.3	2.26E-02	3.7	1.9	8.42E-02	1.4	1.4
Direct LLM (Shojaee et al., 2025b)+PostDiag	39.3	9.54E-03	**5.4**	**3.4**	8.64E-03	**6.1**	**3.8**	2.86E-03	5.3	**4.0**	6.43E-03	**4.6**	**2.8**	2.28E-02	**5.6**	**2.8**
Direct LLM / High	52.7	**1.55E-03**	3.9	2.1	**3.86E-03**	3.0	2.3	**3.06E-05**	**6.7**	2.7	**1.89E-04**	3.7	1.9	**9.36E-04**	2.8	1.4
SGA (Ma et al., 2024)	368.0	8.44E-03	14.7	8.8	5.88E-03	12.1	7.6	6.17E-03	13.3	8.0	5.10E-03	22.2	13.0	2.05E-02	9.7	5.6
SGA (Ma et al., 2024)+PostDiag	406.2	1.44E-03	22.7	12.7	1.52E-03	20.5	12.9	2.06E-04	25.3	10.7	7.32E-04	28.7	15.7	3.66E-03	15.3	**9.7**
SGA / High	1442.0	**7.52E-04**	**26.9**	**15.0**	**1.43E-03**	**25.0**	**14.4**	**5.63E-05**	**28.0**	**18.7**	**3.89E-04**	**32.4**	**16.7**	**7.79E-04**	**20.8**	**9.7**
LLM-SR / Low	201.1	9.68E-04	27.6	12.7	2.41E-03	21.2	9.8	1.91E-05	50.7	26.7	1.18E-04	27.8	9.3	5.86E-04	15.3	8.3
LLM-SR+PD / Low	231.7	**1.31E-04**	36.4	**19.9**	**2.13E-04**	28.0	**15.9**	5.82E-06	65.3	**38.7**	**3.86E-05**	36.1	**15.7**	**2.47E-04**	22.2	**13.9**
LLM-SR / High	1939.2	2.99E-04	**39.3**	16.8	5.15E-04	**33.3**	14.4	**3.35E-06**	**66.7**	34.7	4.42E-05	**37.0**	12.0	5.93E-04	**25.0**	9.7
LaSR / Low	716.3	8.71E-03	15.8	9.3	1.47E-02	14.4	7.6	1.92E-03	17.3	9.3	5.60E-03	19.4	13.0	9.45E-03	11.1	6.9
LaSR+PD / Low	762.1	**6.76E-04**	**24.5**	**13.7**	**1.19E-03**	**23.5**	**10.6**	**3.76E-05**	**33.3**	**16.0**	**4.96E-04**	**25.0**	**17.6**	**6.68E-04**	**16.7**	**11.1**
LaSR / High	1432.6	2.57E-03	19.1	10.9	4.16E-03	17.4	8.3	6.24E-04	22.7	12.0	3.14E-03	22.2	13.9	8.20E-04	13.9	9.7
SR-Scientist / Low	833.6	5.96E-04	41.3	24.3	1.43E-03	29.5	16.7	2.03E-05	74.7	50.7	8.12E-05	41.7	21.3	4.37E-04	27.8	15.3
SR-Scientist+PD / Low	902.2	1.04E-04	**48.8**	**31.3**	**1.76E-04**	**36.4**	**22.7**	6.08E-06	78.7	58.7	3.04E-05	**50.9**	**29.6**	1.82E-04	**37.5**	**20.8**
SR-Scientist / High	3289.4	**9.21E-05**	48.1	31.0	2.33E-04	34.1	21.2	**2.95E-06**	**86.7**	**68.0**	**4.71E-06**	47.2	25.9	**5.75E-05**	34.7	18.1

Table 7: Budget-efficiency comparison for GP-based solvers. Their native search does not use an LLM; all PostDiag interventions use Qwen3.5-27B. Low+PostDiag retains the corresponding Low base-solver budget, while High uses the full-budget reference setting. Boldface marks the best value within each solver block; runtime is reported without ranking.

Method / setting	Overall				Phys			MatSci			Chem			Bio		
	Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
PySR / Low	8.8	2.67E-03	23.8	11.4	4.11E-03	21.2	12.1	1.70E-04	44.0	18.7	8.93E-04	20.4	10.2	5.29E-03	12.5	4.2
PySR+PD / Low	10.5	**1.34E-03**	**28.7**	13.2	**1.82E-03**	24.2	14.4	2.25E-04	**50.7**	17.3	**5.29E-04**	**25.9**	**12.0**	**2.84E-03**	**18.1**	**8.3**
PySR / High	15.5	1.76E-03	28.4	**13.7**	2.61E-03	**25.8**	**15.2**	**8.12E-05**	49.3	**20.0**	6.18E-04	25.0	**12.0**	3.66E-03	16.7	6.9
gplearn / Low	95.5	1.34E-01	0.0	**0.0**	8.51E-02	0.0	**0.0**	2.36E-02	**0.0**	**0.0**	6.56E-02	0.0	**0.0**	4.42E-01	**0.0**	**0.0**
gplearn+PD / Low	101.7	**3.07E-02**	**1.0**	**0.0**	**4.29E-03**	**2.3**	**0.0**	**6.91E-03**	**0.0**	**0.0**	**4.29E-02**	**0.9**	**0.0**	**8.58E-02**	**0.0**	**0.0**
gplearn / High	189.1	1.12E-01	0.0	**0.0**	6.73E-02	0.0	**0.0**	1.92E-02	**0.0**	**0.0**	5.31E-02	0.0	**0.0**	3.77E-01	**0.0**	**0.0**

H.4 Supplementary Results with Additional LLM Backbones

To examine PostDiag under different backbones, we additionally evaluate Qwen3.5-9B and gpt-oss-20b under the same low-budget protocol. Table 8 reports the ID results for the LLM-based solvers and their PostDiag-augmented variants. All comparisons within a backbone use the same tasks, random seeds, search budgets, coefficient-fitting procedure, and validation protocol.

Table 8: Supplementary low-budget ID results for Qwen3.5-9B and gpt-oss-20b. Accuracy to tolerance is the primary criterion; NMSE is a tail-sensitive complementary aggregate. Time is the mean end-to-end wall-clock seconds per task run and is reported only in Overall. Metrics use the same three-seed aggregation as Table 1.

Category	Method / setting	Overall				Phys			MatSci			Chem			Bio		
		Time (s)	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$	NMSE	$A_{.01}$	$A_{.001}$
Qwen3.5-9B																	
	LLM-SR	129.8	2.36E-03	14.0	5.7	5.32E-03	9.8	3.0	2.38E-05	37.3	17.3	4.96E-04	6.5	2.8	2.14E-03	8.3	2.8
	LLM-SR+PostDiag	146.8	**3.93E-04**	**24.0**	**10.3**	**7.32E-04**	**18.2**	**6.8**	**1.20E-05**	**46.7**	**25.3**	**7.15E-05**	**22.2**	**8.3**	**6.52E-04**	**13.9**	**4.2**
LLM-based	LaSR	382.5	2.16E-02	10.1	6.5	2.24E-02	6.8	4.5	4.15E-03	14.7	9.3	9.40E-03	13.0	9.3	5.64E-02	6.9	2.8
	LaSR+PostDiag	407.9	**3.27E-03**	**18.6**	**10.3**	**6.01E-03**	**13.6**	**6.1**	**6.26E-05**	**36.0**	**18.7**	**8.86E-04**	**18.5**	**12.0**	**5.16E-03**	**9.7**	**6.9**
	SR-Scientist	439.2	2.00E-03	13.7	4.1	4.48E-03	10.6	4.5	6.31E-05	33.3	10.7	4.12E-04	7.4	1.9	1.86E-03	8.3	0.0
	SR-Scientist+PostDiag	477.2	**5.39E-04**	**22.5**	**8.8**	**9.84E-04**	**15.9**	**6.1**	**1.69E-05**	**44.0**	**20.0**	**1.33E-04**	**21.3**	**7.4**	**8.76E-04**	**13.9**	**4.2**
gpt-oss-20b																	
	LLM-SR	244.7	1.31E-03	26.1	9.6	3.18E-03	18.9	4.5	2.62E-05	52.0	33.3	2.03E-04	22.2	3.7	8.91E-04	18.1	2.8
	LLM-SR+PostDiag	260.9	**2.39E-04**	**35.4**	**15.2**	**4.31E-04**	**25.8**	**9.1**	**8.52E-06**	**68.0**	**45.3**	**6.47E-05**	**31.5**	**8.3**	**3.87E-04**	**25.0**	**5.6**
LLM-based	LaSR	395.4	1.21E-02	10.6	5.4	1.92E-02	9.1	5.3	3.18E-03	13.3	5.3	8.20E-03	13.0	7.4	1.43E-02	6.9	2.8
	LaSR+PostDiag	419.6	**9.53E-04**	**16.8**	**9.6**	**1.71E-03**	**14.4**	**8.3**	**5.04E-05**	**18.7**	**10.7**	**6.83E-04**	**19.4**	**11.1**	**9.13E-04**	**15.3**	**8.3**
	SR-Scientist	134.9	8.89E-04	33.3	15.5	2.12E-03	24.2	8.3	3.82E-05	53.3	29.3	1.31E-04	38.0	17.6	6.53E-04	22.2	11.1
	SR-Scientist+PostDiag	159.1	**1.60E-04**	**41.3**	**23.3**	**2.68E-04**	**30.3**	**17.4**	**9.47E-06**	**62.7**	**37.3**	**5.03E-05**	**45.4**	**24.1**	**2.81E-04**	**33.3**	**18.1**

I PostDiag Core Editor Prompts

I.1 System Prompt

```
You are a scientist working on symbolic regression. Use risk
localization in the Diagnostic Risk Plane and the corresponding
diagnostic evidence map to modify the symbolic structure of a candidate
equation. Return a preserved or revised equation skeleton.

The input contains the current candidate, diagnostic coordinates, a
diagnostic state, and local behavioral evidence. The numerical procedure
has already determined the state. Follow its intervention:

- Settled (sufficiently expressive and stable)
  Prefer preserving the current skeleton. Make only small revisions
clearly supported by local evidence; avoid unnecessary terms or
replacing the main structure.

- Underfit (insufficiently expressive but stable)
  Preserve the main functional form. Check for missing terms, incorrect
operators, or unsuitable variable combinations, and make local repairs
targeting the main behavioral discrepancies.

- Overfit (sufficiently expressive but unstable)
  Prefer simplifying the skeleton. Check for redundant terms, repeated
subexpressions, complex nesting, and structures that may depend on
compensating coefficients.

- Mismatched (insufficiently expressive and unstable)
  You may replace the main functional form. Propose a new structure
based on local evidence; unsupported terms or combinations need not be
retained.

Local evidence describes differences in level, trend, and roughness
between the candidate and observations, as well as behavioral changes
after bootstrap refitting of the same skeleton. Focus edits on the
```

System prompt (continued)

```
structural problems indicated by this evidence. Discrepancy magnitudes
do not indicate overprediction or underprediction. Roughness differences
may also arise from local nonlinearity; do not treat them as noise alone.

Represent fitted constants as params[i]. An external numerical procedure
will fit their values after you return the equation. Small integer
exponents may be written as literal constants.

Use only the supplied variables, operators, and functions. Respect the
expression-size and nesting constraints. Return one candidate, without
loops, conditionals, imports, or additional helper functions. If the
evidence does not justify a change, return the original skeleton.

Output only the Python function in the specified format, without
explanation. Subsequent numerical fitting and validation determine
whether the revision is accepted.
```

Figure 3: System prompt for the PostDiag core editor.

I.2 User Prompt Template

```
# Current candidate equation
{candidate_function}

# Diagnostic results
Skeleton Expressiveness Risk (Bias): {bias}
Skeleton Stability Risk (Var): {variance}
Diagnostic state: {diagnostic_state}

# Local behavioral evidence
The following evidence is computed on the same local patches of the
audit set:
- Level discrepancy: the absolute difference between the local means of
candidate predictions and observations.
- Trend discrepancy: the norm of the difference between their local
slope vectors.
- Roughness discrepancy: the absolute difference between their
deviations from their respective local linear approximations.
- Instability: local behavioral variation after bootstrap refitting of
the same skeleton. Larger values indicate greater instability.

Patch ID | Level | Trend | Roughness | Instability
{patch_evidence_rows}

# Expression constraints
Input variables: {variables}
Allowed operators and functions: {allowed_operators}
Fitted parameter slots: {parameter_slots}
Expression-size and nesting limits: {structural_constraints}
Input variable ranges: {input_ranges}

# Output requirements
Use the diagnostic state and local evidence to return one preserved or
revised equation. Represent every fitted constant as params[i] and use
only the supplied parameter slots.

Output a single Python code block with the following function signature.
The function body must contain only one return expression:
'''python
def equation({signature}, params):
    return ...
'''
```

Figure 4: User prompt template for the PostDiag core editor.