LLM-PySR: Large Language Model-Guided Symbolic Regression via Search Control

Anonymous authors
Paper under double-blind review

Abstract

Scientific equation discovery must combine broad domain priors with strict numerical testing. Symbolic regression supplies numerical grounding but faces a combinatorial search space, whereas many language-model systems ask the model to propose or select formulas directly.We systematically test how alternative divisions of decision-making authority between language models and numerical symbolic regression affect equation discovery. We compare role specifications in which the language model acts as equation author, candidate decider or search controller, alongside end-to-end language-model and purely numerical baselines. In the controller setting we propose here, implemented as LLM-PySR, language models specify variables, operators, transformations and search depth; symbolic regression enumerates and fits expressions; and fixed rules with numerical safeguards govern retention. Across 74 AI-Feynman equations and seven complex formula-recovery tasks, search control achieved the strongest observed balance of accuracy, complexity, stability and cost. Notably, on an independent battery dataset, LLM-PySR identified a compact piecewise-linear relation between early voltage-curve displacement and cycle life. Overall, LLM-PySR provides a token-efficient, interpretable, and empirically effective approach for scientific equation discovery, bridging the gap between LLM guidance and classical symbolic regression.

1 Introduction

Inferring compact, mechanistically meaningful equations from data has long been a hallmark of scientific discovery, and is also a central goal of scientific machine learning (Makke & Chawla, 2024). Unlike conventional machine-learning models that primarily optimize predictive accuracy, scientific equation discovery seeks compact analytical relations that can be inspected, tested and connected to mechanism (Brunton et al., 2016). Symbolic regression (SR) is one of the main approaches to this problem: it searches over variables, operators and constants to identify closed-form expressions that explain observed data. This makes SR especially attractive in physics, chemistry, materials science and engineering, where an explicit formula can provide insight beyond numerical prediction (Nair et al., 2025; Angelis et al., 2023; Wang et al., 2024; Vaddireddy et al., 2020; Bartel et al., 2018). However, the same flexibility that makes SR powerful also makes it difficult: the space of possible expressions grows rapidly with the number of variables, operators and allowed expression complexity (de França et al., 2024).

Recent large-language-model (LLM) based SR methods have improved equation discovery by injecting scientific priors, semantic information and code-generation ability into the search process (Shojaee et al., 2025a; Wang et al., 2025; Guo et al., 2025; Merler et al., 2024). Compared with traditional search-based approaches (Schmidt & Lipson, 2009; Ouyang et al., 2018; Cranmer, 2023; Bartlett et al., 2023; Mundhenk et al., 2021; de França, 2018), these approaches provide a hybrid mechanism that couples semantic reasoning with symbolic search. A common strategy places the LLM inside an iterative candidate-generation loop: given data summaries, variable names, scientific context, and feedback from previous candidates, the model proposes equation skeletons that are subsequently fitted and evaluated numerically. This paradigm has shown that LLMs can provide useful high-level priors for SR. However, in such skeleton-generation pipelines, the LLM does more than provide priors: it effectively pre-selects the structural search space by deciding which equation forms are passed to numerical fitting. Although numerical optimization can fit constants after

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.

a skeleton is proposed, the structural search itself remains limited by the LLM's prompt-level interpretation of the data and context. As a result, direct equation generation can be expensive, because many candidate formulas require repeated LLM calls, and fragile, because generated skeletons may reflect memorized priors, semantic associations or hallucinated relationships rather than fine-grained numerical structure (Carlini et al., 2021; Mirzadeh et al., 2025; Ji et al., 2023).

A complementary direction moves language models away from direct equation generation and toward the control of numerical symbolic search. LangLaw (Guan et al., 2026), for example, configures iterative PySR runs through feature selection, tree-depth adjustment, and search-budget allocation. It demonstrates that an LLM can guide a numerical search engine without directly enumerating candidate expressions. However, existing work does not isolate the consequences of assigning LLM authority to different decision points in SR: search-space planning, candidate-structure generation, and candidate retention.

We formulate this question as a decision-authority allocation problem. Within a shared pipeline, we construct three matched configurations: a planning-only controller (search controller), a controller that additionally assigns candidate-structure generation to the LLM (equation author), and a controller that instead assigns candidate-retention authority to the LLM (candidate decider). We evaluate these controlled variants alongside full-stack LLM methods, purely numerical search, and LangLaw as the closest controller baseline. This design separates the benefit of introducing a language model from the consequences of assigning it a particular decision-making role. Our search-controller implementation, LLM-PySR, allows the language model to plan variables, operators, transformations, engineered features, and complexity schedules. PySR retains responsibility for expression enumeration and constant fitting, while deterministic numerical criteria govern candidate retention. Relative to prior controller approaches, our contribution lies in the systematic role comparison, the broader search-space specification exposed to the controller, and the explicit separation between semantic guidance and numerical selection. Together, these results support a practical design principle for LLM-assisted SR: LLMs can contribute semantic priors by shaping where numerical search is performed, while expression construction and final model selection should remain grounded in explicit numerical procedures.

2 Related Work

Classical symbolic-regression methods search for analytical expressions through genetic programming, deterministic enumeration, sparse regression, or neural-guided search (Schmidt & Lipson, 2009; Ouyang et al., 2018; Cranmer, 2023; Bartlett et al., 2023; Mundhenk et al., 2021; de França, 2018). These methods differ in how they represent and explore expressions, but candidate construction and selection are primarily governed by numerical objectives. Their performance depends strongly on the variables, operators, transformations, and complexity constraints defining the accessible hypothesis space.

Recent LLM-based methods introduce semantic and scientific priors into this process. LLM-SR and DrSR represent a predominantly generative paradigm in which the language model proposes or refines symbolic structures that are subsequently fitted and evaluated numerically (Shojaee et al., 2025a; Wang et al., 2025). Other approaches similarly use task descriptions, variable semantics, and feedback from previous candidates to guide equation generation (Guo et al., 2025; Merler et al., 2024). These methods demonstrate that language models can provide useful structural priors, but they place the model inside the candidate-generation loop, making structural coverage and language-model cost dependent on repeated generation.

LangLaw is the most closely related controller-based approach (Guan et al., 2026). It combines an LLM with iterative PySR runs and uses feature semantics and an experience pool to select input features, tree depth, and evolutionary search budgets. LangLaw therefore shares with LLM-PySR the principle that the language model should configure numerical search rather than directly enumerate expressions. Our work differs in research objective and experimental design: we treat role assignment itself as the object of study, compare equation authorship, candidate selection, and search control within a shared framework, and evaluate LangLaw as a controller baseline. LLM-PySR additionally exposes operator, transformation, engineered-feature, and complexity planning to the controller under deterministic sanitization, while numerical criteria retain final selection authority.

3 METHODS

3.1 PROBLEM FORMULATION

Given a dataset $\mathcal{D}$, SR searches for an expression f in a hypothesis class $F(\pi)$ defined by a search specification π, including the available variables, operators, transformations, and complexity constraints. Each candidate consists of a discrete expression structure and numerical constants fitted to the data. We write the search objective as

$$\hat{f} = \arg \min_{f \in F(\pi)} L(f; \mathcal{D}) + \lambda \Omega(f), \tag{1}$$

where Omega measures expression complexity.

We use PySR as the numerical search engine in this work (Cranmer, 2023). PySR enumerates expression structures, fits their constants, and maintains candidates along an accuracy-complexity Pareto front. Its behavior depends strongly on the search specification pi, making the specification itself a natural target for language-model guidance. LLM-PySR therefore assigns high-level search-space planning to the LLM while retaining numerical enumeration and model selection in PySR. A complementary statistical analysis of the pruning-bias–complexity–finite-search trade-off induced by LLM-controlled search spaces is provided in Appendix A.

Algorithm 1 LLM-PySR

Require: Training data $\mathcal{D}_{\mathrm{tr}}$, validation data $\mathcal{D}_{\mathrm{val}}$, context c, base operator set $\mathcal{O}_{\mathrm{base}}$, number of iterations T

Initialize memory $\mathcal{M}_0 \leftarrow \emptyset$ and best formula $f^\star \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $\pi_t^{\mathrm{raw}} \leftarrow \mathrm{Planner}(c, \mathcal{D}_{\mathrm{tr}}, \mathcal{M}_{t-1}, \mathcal{O}_{\mathrm{base}})$
  $\pi_t \leftarrow \mathrm{Sanitize}(\pi_t^{\mathrm{raw}}, \mathcal{M}_{t-1})$
  Construct engineered features and transformed datasets using π_t
  $\mathcal{P}_t \leftarrow \mathrm{PySR}(\mathcal{D}_{\mathrm{tr}}; \pi_t)$
  Evaluate each $f \in \mathcal{P}_t$ on $\mathcal{D}_{\mathrm{tr}}$ and $\mathcal{D}_{\mathrm{val}}$
  $m_t \leftarrow \mathrm{Evaluate}(\mathcal{P}_t, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}})$
  $r_t \leftarrow \mathrm{Reviewer}(c, \mathcal{P}_t, m_t)$
  $z_t \leftarrow \mathrm{NumericalGate}(m_t, r_t, m^\star)$
  $f^\star, m^\star \leftarrow \mathrm{NumericalUpdateBest}(f^\star, m^\star, \mathcal{P}_t, r_t, z_t)$
  $\mathcal{M}_t \leftarrow \mathrm{MemoryBuilder}(\mathcal{M}_{t-1}, \pi_t, \mathcal{P}_t, m_t, z_t, r_t)$
end for
$f^\star \leftarrow \mathrm{AblationPrune}(f^\star, \mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}})$
return $f^\star$

3.2 LLM-PYSR OVERVIEW

LLM-PySR is a closed-loop SR framework that uses LLM agents to control the search behavior of PySR rather than directly generate final equations. This design is related to recent agentic LLM frameworks that use reasoning, feedback, and memory to improve iterative decision making (Yao et al., 2022; Shinn et al., 2023; Madaan et al., 2023). Given a training set $\mathcal{D}_{\mathrm{tr}}$, an optional validation set $\mathcal{D}_{\mathrm{val}}$, and task context c including variable semantics and units, LLM-PySR maintains an experience memory $\mathcal{M}_t$ over search iterations.

At iteration t, a search-space planner proposes a search plan

$$\pi_t = (\mathcal{E}_t, \mathcal{X}_t, \mathcal{O}_t, \mathcal{U}_t, d_t),$$

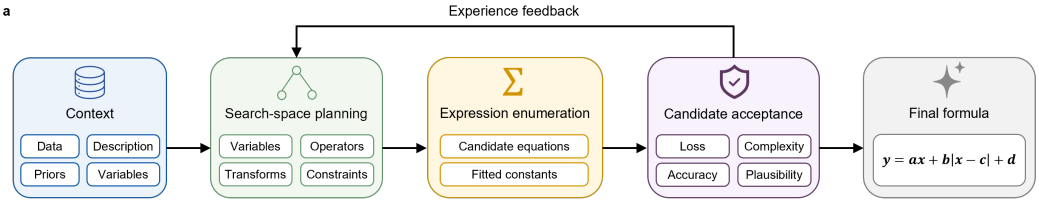
where $\mathcal{E}_t$ denotes engineered features, $\mathcal{X}_t$ denotes selected input variables, $\mathcal{O}_t$ denotes selected base operators, $\mathcal{U}_t$ denotes custom operators, and d_t denotes the maximum search depth. The planner is instructed to propose no more than five inputs, whereas the sanitizer removes unavailable and duplicate inputs. PySR then searches for candidate expressions under the resulting search configuration and returns a candidate pool $\mathcal{P}_t$.

Each candidate is evaluated programmatically using training and validation $\mathrm{ACC}_{0.1}$ and NMSE, together with PySR loss, PySR score, and expression complexity. The reviewer agent additionally

returns semantic annotations and a bounded score, collectively denoted by r_t. Its categorical label is discarded; candidate retention and incumbent updating instead follow fixed rules that combine numerical conditions with the reviewer score as a bounded auxiliary signal. The resulting evaluations are summarized in the experience memory to guide subsequent search-space planning. The expanded workflow and complete prompts for the search-space planner, candidate reviewer, and experience-memory agent are provided in Appendix B and C.

Briefly, candidates are first ranked into a small numerical elite set using validation accuracy, training accuracy, PySR score, NMSE, and loss. Outside the bootstrap iteration, an elite candidate is retained when at least two of three fixed tests are satisfied: training accuracy is within 0.002 of the incumbent, validation accuracy is within 0.002 of the incumbent, and either NMSE or loss is no more than 5% worse than the incumbent. A predefined relaxation is activated only after two consecutive iterations without a retained candidate. All thresholds are task-independent. Complete sanitization constraints, bounded plan-update rules, search-budget schedules, candidate-retention thresholds, and incumbent-update pseudocode are provided in Appendix D.

a



b

Roles	Search-space planning	Expression enumeration	Candidate acceptance	Representative method
Search controller	LLM	Numeric	LLM + Numeric	LLM-PySR **Ours** LangLaw
Equation author	LLM	LLM	LLM + Numeric	LLM-PySR ablation
Candidate decider	LLM	Numeric	LLM	LLM-PySR ablation
Full-stack LLM	LLM	LLM	LLM	LLM-SR, DrSR
No-LLM search	Numeric	Numeric	Numeric	PySR

Figure 1: Role specifications for LLM-assisted SR. (a) The closed-loop pipeline separates three decision points: where to search, which expressions to generate, and which candidates to retain. (b) The search controller, equation author, and candidate decider form a controlled role-reassignment ablation. Relative to the search controller implemented as LLM-PySR, the equation author reassigns candidate construction to the LLM, whereas the candidate decider reassigns candidate retention to the LLM. Full-stack LLM search and no-LLM numerical search provide external baselines at the two ends of the role spectrum.

3.3 Role specifications and common pipeline

To systematically evaluate the decision-authority boundaries of large language models in equation discovery, we establish the search-controller configuration, implemented as LLM-PySR, as the reference pipeline shown in Figure 1. An LLM plans the variables, engineered features, operators, and search depth; deterministic checks sanitize the plan; PySR enumerates expression structures and fits constants; numerical evaluation governs candidate retention; an experience agent summarizes the resulting evidence; and post-hoc pruning selects a compact final expression.

We define two matched variants by reassigning one decision point. Equation author replaces PySR structure enumeration with LLM-proposed equation skeletons while retaining numerical constant fitting. Candidate decider replaces numerical retention and incumbent updating with the LLM reviewer's judgments; upstream planning, candidate evaluation, and search budgets remain shared. The three configurations therefore constitute a controlled role-reassignment ablation.

External baselines cover the remaining role settings. Standalone PySR represents no-LLM numerical search; LLM-SR and DrSR represent full-stack methods in which the LLM constructs candidate expressions; and LangLaw is the closest external search controller, configuring input features, tree depth, and search budgets while leaving expression enumeration and constant fitting to PySR.

Role	$ACC_{0.1}\uparrow$	NMSE↓	Token↓	Time(s)↓
Search controller (LLM-PySR)	**0.958±0.165**	**0.034±0.023**	91,273±74,873	**493±485**
Search controller (LangLaw)	0.956±0.229	**0.015±0.068**	**67,865±23,114**	659±246
Equation author	0.892±0.259	0.387±2.050	449,627±315,861	909±787
Candidate decider	0.952±0.183	0.030±0.135	102,527 ± 89,787	569±681
No-LLM (PySR)	0.922±0.237	0.104±0.170	——	**7.73±3.22**
Full-LLM (LLM-SR)	0.892±0.270	0.051±0.028	2,208,135±464,128	28,128±8,085
Full-LLM (DrSR)	0.665±0.385	0.169±0.068	2,835,147±893,329	613±138

Table 1: Numerical performance and LLM footprint on the AI-Feynman benchmark. Values are means and standard deviations across the 74 benchmark equations. Best performances are shown in bold fonts, and the best wall-clock time for LLM-based methods is shown with underline.

4 EXPERIMENTS

4.1 BENCHMARKS AND EXPERIMENT SETTINGS

We evaluated the seven systems across five role categories mentioned above on two synthetic benchmark suites, AI-Feynman and a seven-task complex-formula stress test, comparing formula-recovery performance, robustness, and computational efficiency.

(1) AI-Feynman dataset (Udrescu & Tegmark, 2020). The AI-Feynman dataset comprises 100 physics equations derived from the Feynman Lectures on Physics. These equations represent compact analytical relationships commonly encountered in physical sciences. To ensure sufficient training data for SR, equations with fewer than 500 samples were removed, leaving 74 equations for evaluating all compared methods.

(2) DrSR-aligned complex formula suite (Wang et al., 2025). To facilitate direct comparison with full-stack LLM search methods, we further evaluated the five roles on the representative formula-recovery tasks adopted in the DrSR study. These tasks cover 7 nontrivial symbolic expressions from physics, biology, chemistry, and materials science, including nonlinear oscillator dynamics, bacterial growth, stress–strain behaviour, and representative LSR-Transform and LSR-Synth cases from LLMSR-bench (Shojaee et al., 2025b). Because this suite contains only a small number of representative cases, we use it as a formula-level stress test for complex symbolic recovery rather than as a large-scale statistical benchmark.

For each benchmark equation, at most 500 samples were randomly selected for training, and all remaining samples were used for testing. In addition, for search controller, equation author and candidate decider that follows our controlled pipeline, the 500 training samples were further split into 250 training samples and 250 validation samples. The validation set was used only for search feedback, model selection and experience accumulation, and the final reported metrics were computed on the held-out test set.

Because the compared frameworks use non-equivalent computational units, we retained their method-specific search configurations rather than matching nominal iteration counts. LLM-SR used the recommended maximum of 2,500 generated samples, whereas DrSR used 10 iterations with four samples per iteration. To ensure a fair and controlled comparison across all role variants, we did not perform extensive hyperparameter tuning. LLM-PySR and LangLaw were each run for at most 10 controller iterations with identical PySR settings: a population size of 128, an initial budget of 200 evolutionary iterations, and an allowable range of 100–500 iterations for subsequent searches. The equation-author, candidate-decider, and no-LLM PySR variants used the same numerical backend settings wherever applicable. We standardized the language-model backbone to GPT-5.1 for all LLM-based methods (Singh et al., 2025) and measured realized computational cost after execution rather than treating nominal iterations as a shared budget. Total token usage was computed as the sum of input and output tokens over all LLM calls. Wall-clock runtime was recorded under the same hardware and execution environment, with one core of Intel Core Ultra 9 275HX processor, backend of PySR v1.5.10, juliapkg v0.1.23 and juliacall v0.9.26.

Following DrSR, we evaluated SR performance using Accuracy under Error Tolerance (ACC_τ) and Normalized Mean Squared Error (NMSE). In all experiments, we set τ=0.1. Given test predictions

Role	Node Complexity↓	Structral Similarity↑	Symbolic Exact↑
Search controller (LLM-PySR)	15.05	**61.0%**	11.3%
Search controller (LangLaw)	16.46	58.0%	2.8%
Equation author	18.91	56.6%	12.3%
Candidate decider	21.96	57.4%	11.3%
No-LLM (PySR)	20.18	47.4%	4.2%
Full-LLM (LLM-SR)	20.19	59.3%	**15.5%**
Full-LLM (DrSR)	**10.78**	59.1%	12.7%

Table 2: Complexity and structural performance on the AI-Feynman benchmark. Values are means and standard deviations across the 74 benchmark equations. Best performances are shown in bold fonts, and the best wall-clock time for LLM-based methods is shown with underline.

y_{pred} and ground-truth values y_{true} , ACC_τ is defined as:

$$ACC_\tau = \frac{1}{n}\Sigma_{i=1}^{n}\mathrm{I}(|y_{\text{pred},i} - y_{\text{true},i}| \leq \tau|y_{\text{true},i}|) \tag{2}$$

where $\mathrm{I}(\cdot)$ is the indicator function. ACC_τ measures the fraction of predictions whose relative error is bounded by the prescribed tolerance and therefore reflects the ability of a discovered equation to provide bounded-error predictions.

NMSE measures the overall prediction error normalized by the variance of the target variable:

$$NMSE = \frac{\frac{1}{n}\Sigma_{i=1}^{n}(y_{\text{pred},i} - y_{\text{true},i})^2}{\frac{1}{n}\Sigma_{i=1}^{n}(y_{\text{true},i} - \overline{y_{\text{true}}})^2} \tag{3}$$

Lower NMSE and higher ACC_τ indicate better predictive performance.

Additionally, we use three metrics to assess formula complexity and agreement with the ground truth: node complexity counts the nodes in the canonical expression tree; structural similarity measures the normalized similarity between canonical expression-tree token sequences, with numerical constants replaced by a common token; and symbolic exact indicates whether the discovered and ground-truth formulas are identical after symbolic canonicalization.

4.2 Main Results on Benchmarks

Table 1 reports predictive performance and computational cost on AI-Feynman. The three matched variants form a controlled role-reassignment ablation that isolates the consequences of assigning the LLM to different decision points. LLM-PySR and the candidate decider achieved comparable mean performance, with $ACC_{0.1}$ values of 0.958 and 0.952 and NMSE values of 0.034 and 0.030, respectively. In contrast, the equation author achieved an $ACC_{0.1}$ of 0.892 and an NMSE of 0.387 while using 4.9 times as many tokens as LLM-PySR. Thus, assigning candidate-structure generation to the LLM was associated with lower aggregate performance and a substantially larger language-model footprint, whereas assigning candidate-retention authority provided no consistent improvement over deterministic numerical gating. At the system level, LLM-PySR and LangLaw also achieved a more favorable observed accuracy–token trade-off than LLM-SR and DrSR, attaining higher mean $ACC_{0.1}$ and lower mean NMSE while consuming only 2.4–4.1% as many tokens.

In addition, Table 2 reports formula complexity and structural agreement with the ground truth on AI-Feynman. DrSR produced substantially simpler expressions than all other methods, but this simplicity was accompanied by markedly lower predictive accuracy, suggesting that its search may favor overly simple formulas. Methods that rely on the LLM to propose candidate equation structures achieved relatively high symbolic-exact rates, yet even the best-performing method on this metric, LLM-SR, recovered the exact ground-truth expression in only 15.5% of cases. This result indicates that, even for well-known benchmark problems, current LLMs cannot reliably infer the correct equation structure directly from the provided context. Moreover, these occasional exact recoveries did not translate into the highest average structural similarity: LLM-PySR achieved the best structural similarity of 61.0%. This discrepancy suggests that when an LLM fails to identify the correct structure, its proposed skeleton may constrain or misdirect the subsequent symbolic search.

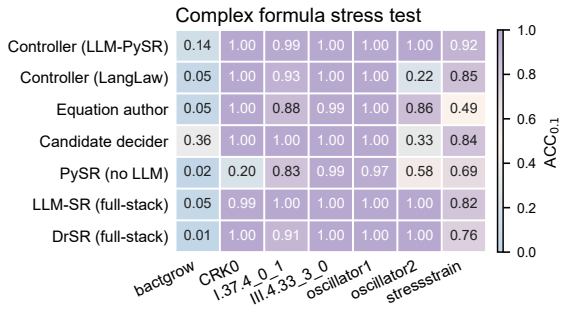


Figure 2: $ACC_{0.1}$ heat map across seven complex stress tests.

Overall, LLM-PySR achieved the strongest balance across these criteria, combining relatively low expression complexity with high predictive accuracy and the highest average structural agreement with the ground truth.

Despite this shared controller-level advantage, LLM-PySR and LangLaw exhibit different efficiency profiles. LLM-PySR dynamically adapts the scope and budget of PySR searches, including the variables, operators, transformations, engineered features, and complexity levels exposed to the numerical engine. Simple problems can therefore be addressed with fewer search rounds and within smaller symbolic spaces. This adaptive allocation is consistent with the wall-clock results: LLM-PySR required 493 seconds on average, making it the fastest LLM-assisted method in Table 1, whereas LangLaw required 659 seconds. LangLaw, in contrast, maintains a more compact experience pool and exposes a narrower set of PySR decisions to the language model. It consequently used fewer tokens than LLM-PySR, 67,865 versus 91,273, but had less opportunity to tailor the numerical search to each problem. Because PySR performs its computation locally while repeated language-model calls introduce API-related latency, total runtime depends on both the number of LLM interactions and the efficiency with which the local search is configured. The comparison therefore shows that token usage alone does not determine runtime: LangLaw minimizes the language-model footprint, whereas LLM-PySR achieves the more favorable overall balance between adaptive control and search time.

The seven complex formula-recovery tasks further distinguish the breadth of the two controller implementations. As shown in Fig. 2, LLM-PySR outperformed LangLaw on four tasks and matched it on the remaining three, yielding a mean $ACC_{0.1}$ of 0.864 compared with 0.721 for LangLaw. The largest difference occurred on oscillator2, where LLM-PySR achieved 1.00 while LangLaw reached 0.22. LangLaw remained highly accurate on several standard scientific relations, confirming the general value of using an LLM to configure numerical symbolic search. Its weaker performance on the more structurally demanding cases is consistent with its more compact experience representation and narrower control interface. By allowing the controller to revise operators, transformations, engineered features, and complexity choices in response to accumulated numerical evidence, LLM-PySR can redirect the search more flexibly when the initial configuration is insufficient.

The remaining role specifications reveal complementary failure modes. For the generative roles, including the equation author and the full-stack LLM methods, performance was strongly associated with the familiarity of the target structure to the base model. These methods achieved near-perfect accuracy on several standard equations, such as CRK0 and oscillator1, but degraded sharply on atypical nonlinear relations such as bactgrow and stressstrain. This pattern suggests that direct equation generation can prematurely restrict exploration to familiar templates. The candidate decider exposed a different vulnerability. Although PySR generated its candidate pool, assigning final retention authority to the language model produced unstable choices on cases such as oscillator2. Language models can recognize broad structural plausibility, but they are less reliable at distinguishing candidates whose residual differences require fine-grained numerical comparison.

Taken together, these results favor assigning the LLM to search-space planning while retaining numerical procedures for expression enumeration, fitting, and candidate retention. We next examine whether this division of labour also yields interpretable relations on real-world scientific data.

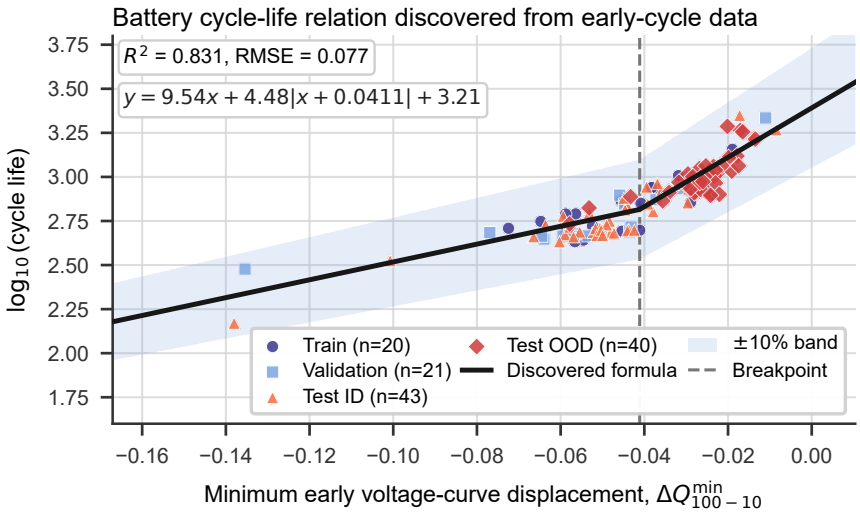


Figure 3: Application of the search-controller framework to lithium-ion battery cycle-life prediction. The discovered hinge-like expression relates early voltage-curve displacement, $\Delta Q_{100-10,\min}$, to $\log_{10}$ cycle life.

4.3 Real-world Scientific Discovery

4.3.1 Scientific discovery in battery lifetime prediction

To further showcase the scientific discovery ability of LLM-PySR, we applied LLM-PySR to battery lifetime prediction with limited data: 124 commercial lithium iron phosphate/graphite cells cycled under fast-charging conditions, with lifetimes of 150–2,300 cycles (Severson et al., 2019). Inputs comprised 11 statistical indicators (e.g., variance, minimum, and mean) of discharge-curve evolution over the first 100 cycles (e.g., $\Delta Q_{100-10}(V)$), alongside capacity degradation, internal resistance, and temperature measurements. The target was cycle life to 80% of nominal capacity. We retained the original cell partitions but repurposed the primary test set for validation: 41 training cells supported PySR structure search and constant fitting, 43 validation cells guided LLM experience feedback and deterministic gating, and 40 secondary-test cells provided independent generalization evaluation.

Given only raw discharge-voltage measurements and semantic context, LLM-PySR autonomously pinpointed the minimum change in the discharge curve between cycles 10 and 100 as the governing coordinate, returning a compact analytical relationship shown in Figure 3:

$$\log_{10}(N_{\text{life}}) = 9.535x + 4.475|x + 0.0411| + 3.208 \tag{4}$$

where x is the minimum value of $\Delta Q_{100-10,min} = Q_{100}(V) - Q_{10}(V)$, which is typically negative. On a 40-cell independent test set, this relation achieved an RMSE of 0.0772 and R^2 = 0.8314. To render the physical meaning explicit, we define the displacement magnitude $d = -\Delta Q_{100-10,min}$, representing the magnitude of the most negative early cycle voltage-curve shift. The equation can be rewritten as:

$$\log_{10}(N_{\text{life}}) = 3.208 - 9.535d + 4.475|0.0411 - d| \tag{5}$$

The expression defines a piecewise-linear decay: larger early voltage-curve displacements predict shorter cycle life, with a change in slope at $d \approx 0.0411$ Ah. This mathematical formulation aligns well with established electrochemical domain knowledge: while early capacity fade is often negligible and visually imperceptible, underlying degradation modes such as the loss of active material in the negative electrode may silently shift the voltage curves long before overt capacity loss appears (Dubarry et al., 2012; Anseán et al., 2016). We interpret the breakpoint as an empirical transition between mild and severe early degradation regimes. The result is therefore valuable as a compact, testable relation between an early electrochemical signal and subsequent cycle life.

4.3.2 THERMODYNAMIC CONSISTENCY IN ENANTIOSELECTIVE ELECTROCATALYSIS

We use the enantioselective electrocatalysis dataset of von Münchow et al. (von Münchow et al., 2025) as a positive-control task for feature identification and analytical-relation recovery. We provide the 33 numerical variables in the released table as candidate inputs and use ee_{exp} as the response. Because the inputs include $\Delta\Delta G_{\mathrm{exp}}$, a thermodynamic re-expression of ee_{exp}, this task is not de novo prediction or discovery of a new law. It instead tests whether LLM-PySR can select the response-equivalent coordinate from a high-dimensional table and recover its expected transformation.

Given the variable semantics and numerical observations, LLM-PySR selected $\Delta\Delta G_{\mathrm{exp}}$ and returned

$$ee_{\mathrm{exp}} = -99.995 \tanh(0.674\, \Delta\Delta G_{\mathrm{exp}}). \tag{6}$$

According to transition-state theory (Eyring, 1935), the enantiomeric ratio satisfies $ER = \exp(-\Delta\Delta G/RT)$, and $ee = 100(ER-1)/(ER+1)$, which together yield

$$ee = -100 \tanh\left(\frac{\Delta\Delta G}{2RT}\right). \tag{7}$$

The recovered coefficient 0.674 corresponds to $1/(2RT)$ at approximately $T = 373\,\mathrm{K}$, matching the $100^{\circ}\mathrm{C}$ reaction temperature used in the experimental study.

This task also illustrates the importance of feature selection in scientific symbolic regression. Among the 33 candidate variables, $\Delta\Delta G_{\mathrm{exp}}$ is the only one that directly encodes ee_{exp} through a known analytical relation. LLM-PySR isolated this response-equivalent coordinate and recovered a bounded one-variable thermodynamic expression. In contrast, all competing methods incorporated multiple additional variables into their final equations. Although some of these variables may be correlated with ee_{exp} in this finite dataset, they are unnecessary for expressing the underlying thermodynamic mapping and make the resulting equations more difficult to interpret. This situation is common in real scientific datasets, where a candidate feature table may contain many descriptors whose relevance to the response is unknown. By controlling feature selection and construction during the search, LLM-PySR was able to exclude superfluous inputs and concentrate the symbolic search on the variable carrying the directly relevant information, making the recovered equation more likely to preserve the underlying scientific relation rather than merely fit the observed samples. Complete expressions and evaluation metrics for all methods on the real-world scientific problems are reported in Appendix E.

5 DISCUSSION

This work treats role assignment, rather than LLM inclusion alone, as a central design choice in language-model-assisted symbolic regression. We construct matched configurations in which the LLM acts as a search controller, equation author, or candidate decider. Across 74 AI-Feynman equations and seven complex formula-recovery tasks, search control provided the strongest observed accuracy–cost trade-off: LLM-PySR and LangLaw outperformed the generation-centered baselines while using less than one-twentieth of their tokens. LLM-PySR's broader control over variables, transformations, operators, and search budgets further improved complex-formula recovery and reduced runtime. The battery and electrocatalysis cases show that this allocation can also produce compact, scientifically inspectable relations, although the former reflects a parsimony–accuracy trade-off and the latter is a thermodynamic positive control rather than de novo law discovery.

These conclusions remain conditional on the evaluated model, numerical backend, and datasets. All LLM-based experiments use the same language-model backbone, and LLM-PySR relies on PySR for structural coverage and constant fitting; other backbones or symbolic engines may produce different role trade-offs. Moreover, restricting variables and operators can exclude the correct structure, introducing pruning bias when the planner's prior is inaccurate. The two real-world studies are also limited in scale and do not establish broad scientific generalization. Future work should test additional backbones and numerical engines, stronger noise and distribution shifts, and adaptive safeguards against premature search-space restriction. Overall, our results support using learned models to guide where hypotheses are sought while retaining explicit numerical procedures for testing which hypotheses survive.

AI USE STATEMENT

In this work, GPT-5.1 was used as the underlying language model in the proposed LLM-PySR framework and in the evaluated LLM-based baselines, as described in the experimental settings. In addition, we used generative AI tools for assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis or interpreting results. We have not used generative AI tools for generating synthetic data sets or conceptual frameworks, designing or providing feedback on research methodology or experiments, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, and editing the paper to improve readability. We have reviewed all AI-assisted work. We checked that the datasets reformatted by the LLM were consistent with the original datasets and that the statistical analysis tools provided by the LLM were correct. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors have identified no ethical concerns associated with this work.

REFERENCES

Dimitrios Angelis, Filippos Sofos, and Theodoros E Karakasidis. Artificial intelligence in physical sciences: Symbolic regression trends and perspectives: D. angelis et al. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, 2023.

D Anseán, M Dubarry, A Devie, BY Liaw, VM García, JC Viera, and M González. Fast charging technique for high power lifepo4 batteries: A mechanistic analysis of aging. *Journal of Power Sources*, 321:201–209, 2016.

Christopher J Bartel, Samantha L Millican, Ann M Deml, John R Rumptz, William Tumas, Alan W Weimer, Stephan Lany, Vladan Stevanović, Charles B Musgrave, and Aaron M Holder. Physical descriptor for the gibbs energy of inorganic crystalline solids and temperature-dependent materials chemistry. *Nature communications*, 9(1):4168, 2018.

Deaglan J Bartlett, Harry Desmond, and Pedro G Ferreira. Exhaustive symbolic regression. *IEEE Transactions on Evolutionary Computation*, 28(4):950–964, 2023.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of machine learning research*, 3(Nov):463–482, 2002.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

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

Fabricio O de França, Marco Virgolin, M Kommenda, MS Majumder, M Cranmer, G Espada, L Ingelse, A Fonseca, M Landajuela, B Petersen, et al. Srbench++: principled benchmarking of symbolic regression with domain-expert interpretation. *IEEE transactions on evolutionary computation*, 2024.

Fabrício Olivetti de França. A greedy search tree heuristic for symbolic regression. *Information Sciences*, 442:18–32, 2018.

Matthieu Dubarry, Cyril Truchot, and Bor Yann Liaw. Synthesize battery degradation modes via a diagnostic and prognostic model. *Journal of power sources*, 219:204–216, 2012.

Henry Eyring. The activated complex in chemical reactions. *The Journal of chemical physics*, 3(2): 107–115, 1935.

Yifeng Guan, Chuyi Liu, Dongzhan Zhou, Lei Bai, Wan-jian Yin, Jingyuan Li, and Mao Su. Discovery of interpretable physical laws in materials via language-model-guided symbolic regression. *arXiv preprint arXiv:2602.22967*, 2026.

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

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

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

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 94743–94765, 2025.

Terrell Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Brenden K Petersen, et al. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. *Advances in Neural Information Processing Systems*, 34:24912–24923, 2021.

Akhil S Nair, Lucas Foppa, and Matthias Scheffler. Materials-discovery workflow guided by symbolic regression for identifying acid-stable oxides for electrocatalysis. *npj Computational Materials*, 11(1):150, 2025.

Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M Ghiringhelli. Sisso: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Physical Review Materials*, 2(8):083802, 2018.

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

Kristen A Severson, Peter M Attia, Norman Jin, Nicholas Perkins, Benben Jiang, Zi Yang, Michael H Chen, Muratahan Aykol, Patrick K Herring, Dimitrios Fraggedakis, et al. Data-driven prediction of battery cycle life before capacity degradation. *Nature energy*, 4(5):383–391, 2019.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

Harsha Vaddireddy, Adil Rasheed, Anne E Staples, and Omer San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Physics of Fluids*, 32(1), 2020.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Tristan von Münchow, Neeraj Kumar Pandit, Suman Dana, Philipp Boos, Sven Erik Peters, Josselin Boucat, Yi-Ru Liu, Alexej Scheremetjew, and Lutz Ackermann. Enantioselective c–h annulations enabled by either nickel-or cobalt-electrocatalysed c–h activation for catalyst-controlled chemodivergence. *Nature Catalysis*, 8(3):257–269, 2025.

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

Tairan Wang, Jianyu Hu, Runhai Ouyang, Yutao Wang, Yi Huang, Sulei Hu, and Wei-Xue Li. Nature of metal-support interaction for metal catalysts on oxide supports. *Science*, 386(6724):915–920, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

A Theoretical Analysis

We provide a statistical interpretation of the role separation used in this research to clarify the potential trade-offs from different LLM roles and actions.

Let $\mathcal{D} = (x_i, y_i)_{i=1}^n$ be sampled from an unknown distribution P, and let $R(f) = E_{(x,y)\sim P}[L(f(x), y)]$ denote the population risk under a bounded loss L. Let $\hat{R}_n(f)$ be the empirical risk. In a standard PAC-style uniform convergence bound (Valiant, 1984; Bartlett & Mendelson, 2002), for a fixed symbolic hypothesis class $\mathcal{F}$, with probability at least $1 - \delta$:

$$\sup_{f\in\mathcal{F}} |R(f) - \hat{R}_n(f)| \lesssim \sqrt{\frac{C(\mathcal{F}) + \log(1/\delta)}{n}} \tag{8}$$

where $C(\mathcal{F})$ denotes an appropriate statistical complexity measure of the class, such as pseudo-dimension, covering complexity or a Rademacher-complexity-based term. If an algorithm returns an approximate empirical-risk minimizer $\hat{f} \in \mathcal{F}$ satisfying

$$\hat{R}_n(\hat{f}) \leq \inf_{f\in\mathcal{F}} \hat{R}_n(f) + \eta \tag{9}$$

then the corresponding excess risk over the best function in $\mathcal{F}$ obeys

$$R(\hat{f}) - \inf_{f\in\mathcal{F}} R(f) \lesssim \sqrt{\frac{C(\mathcal{F}) + \log(1/\delta)}{n}} + \eta. \tag{10}$$

In SR, the additional term η is important. Unlike convex empirical-risk minimization, SR relies on finite-budget combinatorial search and constant optimization; hence the returned expression need not be the empirical optimum of the specified symbolic class. We therefore interpret η as a finite-budget search error.

Let $\mathcal{F}_0$ denote the full symbolic class accessible to an unconstrained PySR run under the global variable set, operator set and maximal expression complexity. At an LLM-PySR iteration, the LLM

planner returns a search specification ϕ, including selected variables, allowed operators, engineered transformations and complexity constraints. This induces a restricted symbolic class $\mathcal{F}_\phi \subseteq \mathcal{F}_0$. The full-class oracle is $f_0^* \in \arg\min_{f \in \mathcal{F}_0} R(f)$. Because LLM-PySR searches only inside $\mathcal{F}_\phi$, it may exclude the full-class optimum. We define the pruning bias as $\mathcal{B}_\phi = \inf_{f \in \mathcal{F}_\phi} R(f) - \inf_{f \in \mathcal{F}_0} R(f)$, this term is zero only when the LLM-induced restricted class retains a population-risk minimizer, or at least a near-optimal expression, from the full symbolic class.

If ϕ is selected from data, an additional model-selection cost must also be paid. We denote this cost by Λ_ϕ. For a finite set of possible search specifications, Λ_ϕ can be taken as $\log(1/\pi(\phi))$ under a prior over specifications $\pi(\phi) > 0, \Sigma_{\phi \in \Phi} \pi(\phi) \leq 1$. This term accounts for the fact that a data-dependent search space is not a free prior.

Combining these terms, LLM-PySR admits the following decomposition:

$$R(\hat{f}_\phi) - R(f_0^*) \lesssim \underbrace{\mathcal{B}_\phi}_{\text{pruning bias}} + \underbrace{\sqrt{\frac{C(\mathcal{F}_\phi) + \Lambda_\phi + \log(1/\delta)}{n}}}_{\text{statistical estimation error}} + \underbrace{\eta_\phi(N_{\text{eval}})}_{\text{finite-budget search error}} \tag{11}$$

where N_{eval} denotes the expression-evaluation budget used by the symbolic search engine.

This decomposition clarifies why merely reducing the search space is not sufficient. Search-space restriction decreases the statistical complexity term only if

$$C(\mathcal{F}_\phi) < C(\mathcal{F}_0) \tag{12}$$

For monotone complexity measures, this follows from $\mathcal{F}_\phi \subseteq \mathcal{F}_0$. Another intuitive indicator expression skeleton count number $|S_K| \approx O(K^L)$ shows the number of admissible expression structures grows with the number of possible operators:

$$|S_{K_\phi}| \sim O(K_\phi^L), |S_{K_0}| \sim O(K_0^L), K_\phi < K_0 \tag{13}$$

where K_ϕ and K_0 represent the number of possible operators of LLM-PySR and PySR, and L represents the skeleton length. However, it may simultaneously increase the pruning bias $\mathcal{B}_\phi$. Therefore, LLM-PySR is advantageous only when the reduction in statistical and search complexity outweighs the approximation loss caused by excluding parts of the full symbolic class.

For an unconstrained PySR baseline, the corresponding bound is:

$$R(\hat{f}_0) - R(f_0^*) \lesssim \sqrt{\frac{C(\mathcal{F}_0) + \log(1/\delta)}{n}} + \eta_0(N_{\text{eval}}). \tag{14}$$

Thus, LLM-PySR is theoretically favoured over unconstrained PySR under the sufficient condition:

$$\mathcal{B}_\phi + \sqrt{\frac{C(\mathcal{F}_\phi) + \Lambda_\phi + \log(1/\delta)}{n}} + \eta_\phi(N_{\text{eval}}) < \sqrt{\frac{C(\mathcal{F}_0) + \log(1/\delta)}{n}} + \eta_0(N_{\text{eval}}). \tag{15}$$

This condition exposes the central bias–complexity–search trade-off. Plain PySR has no pruning bias, because it searches the full symbolic class, but it generally pays a larger statistical complexity penalty and may suffer a larger finite-budget search error. LLM-PySR accepts a nonzero pruning bias in exchange for a lower-complexity symbolic class and a potentially easier search problem. This advantage is most pronounced when semantic priors remove many irrelevant expressions while preserving the region containing a near-optimal formula.

We can view the risk bound of equation author with the same inequality:

$$R(\hat{f}_{\text{author}}) - R(f_0^*) \lesssim \mathcal{B}_{\text{author}} + \sqrt{\frac{C(\mathcal{F}_{\text{author}}) + \Lambda_{\text{author}} + \log(1/\delta)}{n}} + \eta_{\text{author}}(N_{\text{eval}}) \tag{16}$$

In the equation-author role, the LLM proposes M_{author} equation skeletons, inducing a structural class whose complexity $C(\mathcal{F}_{\text{author}})$ may be substantially smaller than $C(\mathcal{F}_\phi)$ or $C(\mathcal{F}_0)$. However, because these skeletons are discrete samples from the LLM prior, this compression can increase the pruning bias $\mathcal{B}_{\text{author}}$ by excluding the true or near-optimal structure. This is consistent with our observation that equation authorship performs well on familiar forms but degrades on atypical relations. Search control instead uses the LLM to restrict the search space while leaving structural enumeration and constant fitting to PySR, preserving broader coverage and a more favorable balance among pruning bias, statistical complexity, and finite-budget search error.

For the candidate decider role, the symbolic search space may remain the same as search controller because PySR still generates candidate expressions. However, the final selection is delegated to the LLM. This introduces an additional selection-noise term:

$$R(\hat{f}_{\text{decider}}) - R(f_0^*) \lesssim \mathcal{B}_\phi + \sqrt{\frac{C(\mathcal{F}_\phi) + \Lambda_\phi + \log(1/\delta)}{n}} + \eta_\phi(N_{\text{eval}}) + \zeta_{LLM}. \tag{17}$$

considering candidate decider inherits all other parts from search controller other than the term ζ_{LLM}. This term introduces additional selection noise, which captures the risk gap between the LLM-selected candidate and the numerically best candidate in the pool. This decomposition also explains why the candidate decider can exhibit average performance close to that of the search controller: in the matched setting, most error terms are shared, and the performance gap is mainly governed by ζ_{LLM}. When LLM judgment agrees with numerical selection, this gap is small; when the LLM is insensitive to fine residual differences or overweights semantic plausibility, ζ_{LLM} can increase and lead to unstable or suboptimal selection.

Thus, the advantage of LLM-PySR is not simply that it reduces the search space. Rather, it uses semantic priors to reduce operator-induced symbolic complexity while preserving numerical enumeration within the restricted class and avoiding LLM-based final selection. This role separation seeks a favourable balance among pruning bias, statistical complexity and finite-budget search error, while preventing semantic plausibility from overriding numerical evidence.

B Detailed LLM-PySR Workflow

C Detailed prompt templates in LLM-PySR

Agent A (Search controller) Prompt template:

Listing 1: System prompt used by Agent-A (search controller).

```
Task: You are LLM-A in a symbolic-regression loop.
Context: {context or "Unknown physical background. Use variable
    meanings + data patterns only."}
Variable meanings: {json.dumps(variable_desc, ensure_ascii=False)}
Data snapshot (first up to 100 rows): {json.dumps(data_snapshot if
    isinstance(data_snapshot, dict) else {}, ensure_ascii=False)}
Experience memory: {json.dumps(memory, ensure_ascii=False)}
Previous round reviews_with_formula: {json.dumps(reviews_with_formula
    or [], ensure_ascii=False)}
LLM-C recommended maxdepth (previous round): {
    llm_c_recommended_maxdepth if llm_c_recommended_maxdepth is not None
     else "null"}
Allowed base operators: {global_operators}

Return strict JSON with keys:
- engineered_features: [{"name":"u1","expression":"...","reason
    ":"...","dimension_check":"pass/unknown"}]
- selected_input_features: ["x1","x2","u1"]
- selected_operators: ["+","*","log", "..."]  # choose from
    allowed_base_operators
- custom_unary_operators: {"f1":"..."}
```

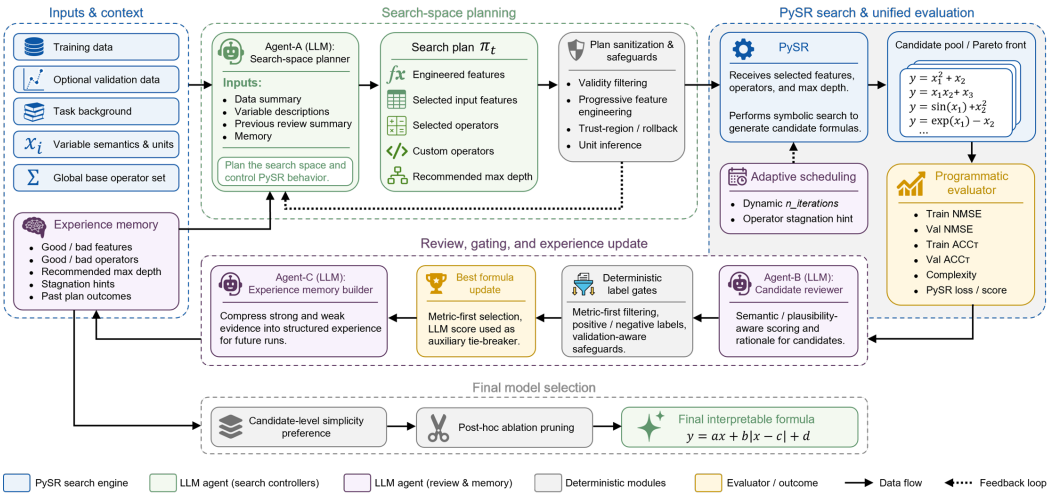


Figure 4: Detailed workflow of LLM-PySR algorithm. LLM-PySR operates as an iterative search-controller framework for symbolic regression. Given data, task context, variable semantics, an admissible operator set and accumulated experience memory, an LLM planner proposes a bounded search plan specifying candidate variables, engineered features, operators, transformations and expression depth. This plan is sanitized by deterministic safeguards and executed by PySR, which enumerates symbolic expressions and fits constants. Candidate formulas are assessed by a unified programmatic evaluator, while deterministic metric gates assign retention labels based on predictive error, tolerance accuracy, complexity and stability. LLM-based review is used only as auxiliary semantic feedback, and the accepted numerical outcomes are summarized into experience memory for the next iteration. After the iterative search, simplicity-guided selection and post-hoc ablation pruning yield the final compact interpretable formula.

```
- custom_binary_operators: {"f2":"..."}
- recommended_maxdepth: <int_or_null>
- decision_evidence: {
    "operators": [{"op":"sin","source":"pattern|memory|context","
    evidence":"..."}],
    "features": [{"name":"u1","source":"pattern|memory|context","
    evidence":"..."}]
  }

Rules:
1) Prefer only high-confidence engineered features.
2) A custom operator should combine >=2 base operators.
3) Do not exceed {max_custom_ops} custom operators total.
4) Only use operators from allowed list in expressions.
4.1) Custom operator constraints:
   - Unary custom operator must be expression over x only (form f(x)).
   - Binary custom operator must be expression over x,y only (form f(x,
    y)).
   - Do not reference dataset variable names (u1, v, E_n, etc.) inside
    custom operator definitions.
   - Do not call custom operators inside custom operator definitions (
    no f1(...), f2(...)).
   - Do NOT output assignments like "f1(x)+x" as an operator definition
    . The definition itself must be only the RHS expression.
   - Valid examples:
     unary: "exp(-abs(x))", "sin(x) + x^2"
     binary: "x*log(1+abs(y))", "sin(x) + cos(y)"
   - Invalid examples:
     "f1(x)+x", "f2(u1, x)", "x + z", "f3(x)"
   - If uncertain, output empty custom operators: {}.
```

```
5) selected_input_features are the final variables to pass to PySR.
    Keep them minimal.
5.2) selected_input_features must contain at most 5 variables total.
   If uncertain, choose fewer variables instead of adding weak ones.
5.1) Feature-engineering budget:
   - Hard limit: at most 3 engineered features.
   - Output only the top-3 most important engineered features ranked by
     expected impact on PySR fit quality.
   - If you are uncertain, output fewer features instead of filling to
    3.
   - Start from atomic/low-granularity features first (pairwise
    distance, ratio, product, square).
   - Avoid directly constructing an almost-complete final law in one
    step.
6) Use Experience memory as hard guidance:
   - Prioritize operators marked effective.
   - Consider expanding to suggested operators when current fit is poor
    .
   - Avoid operators marked ineffective unless there is clear new
    evidence.
   - Prefer recommended feature-engineering patterns and custom unary/
    binary operators.
7) Analyze internal relation patterns directly from the provided first
    -100-row data snapshot.
8) For selected_operators: prefer a stable compact set.
   If memory already supports prior operators, change at most 1-2
    operators this round.
9) Dimension-check policy:
   - Do NOT output "fail". Use only "pass" or "unknown".
   - If variable meanings do NOT provide explicit physical units,
    prefer "unknown".
   - For sin/cos/tan/log/exp on normalized or unit-unknown variables,
    use "pass" or "unknown".
10) selected_operators must be a concise subset of
    allowed_base_operators.
11) Prefer operators that are repeatedly supported by data_snapshot and
     experience_memory.
11.1) Never add "re"/"im" as operators. They are exporter wrappers and
    should not increase complexity.
12) Progressive update policy:
   - Keep high-performing engineered features/operators from prior
    rounds unless strong contradictory evidence appears.
   - Avoid replacing many features/operators at once.
13) Single-plan policy:
   - Return only one final plan.
   - Do not output conservative_plan or expansion_plan.
14) Every selected operator/engineered feature should have a traceable
    evidence source in decision_evidence.
15) Decide recommended_maxdepth in [6, 40]. Prefer LLM-C recommendation
     unless strong contrary evidence appears.
```

Agent B (Candidate reviewer) Prompt template:

Listing 2: System prompt used by Agent-B (candidate reviewer).

```
Task: You are LLM-B. Analyze and score candidate symbolic formulas.
Context: {context or "Unknown background"}
Variable meanings: {json.dumps(variable_desc, ensure_ascii=False)}
Current best score: {current_best_score}
Current best metrics: {json.dumps(current_best_metrics, ensure_ascii=
    False)}
Candidates:
{json.dumps(candidates, ensure_ascii=False)}
```

```
Return strict JSON object with this schema (no extra text):
{
  "reviews": [
    {
      "index": <int>,
      "subscores": {
        "interpretability": <float_0_to_1>,
        "acc_quality": <float_0_to_1>,
        "nmse_quality": <float_0_to_1>,
        "pysr_metrics_quality": <float_0_to_1>,
        "fit_quality": <float_0_to_1>,
        "simplicity": <float_0_to_1>,
        "variable_semantics_consistency": <float_0_to_1>
      },
      "score": <float_0_to_1>,
      "label": "positive|negative",
      "reason": "<short>"
    }
  ]
}

Rules:
1) Score each candidate directly; do not hard-reject whole formulas by
    a global dimension gate.
2) Metric meanings:
   - ACC: higher is better (accuracy under tolerance).
   - val_acc: higher is better; treat this as the primary reliability
    signal for final score.
   - NMSE: lower is better (normalized mean square error).
   - loss (from PySR): lower is better.
   - sr_score (from PySR): higher is better, but only a secondary tie-
    breaker.
3) Score based on:
   - physical interpretability,
   - ACC/val_acc value (most important): larger is better,
   - NMSE value: smaller is better,
   - PySR metrics (loss lower; use sr_score only when other metrics are
     close).
4) Reason must explicitly mention ACC/NMSE/loss/sr_score and why the
    score is assigned.
5) The final positive/negative label is determined by system-side
    deterministic gates.
   You still need to return a reasonable label and score, but the
    system may override label.
6) In comparison reasoning, prioritize val_acc/ACC first, then NMSE/
    loss, then interpretability.
   Use sr_score only for tie-breaking when the above metrics are close.
7) Score direction must follow metric direction:
   - if ACC/val_acc are higher while NMSE and loss are lower, assign a
    higher score.
   - if ACC/val_acc are lower while NMSE and loss are higher, assign a
    lower score.
8) Strict anti-overrating guard (must follow):
   - Let reliability = val_acc when available, otherwise ACC.
   - If reliability < 0.10, score must be <= 0.45.
   - If 0.10 <= reliability < 0.20, score must be <= 0.55.
   - If 0.20 <= reliability < 0.40, score must be <= 0.65.
   - If 0.40 <= reliability < 0.60, score must be <= 0.75.
   - If 0.60 <= reliability < 0.80, score must be <= 0.85.
   - If 0.80 <= reliability < 0.90, score must be <= 0.92.
   - If reliability >= 0.90, score can be in [0.0, 1.0] according to
    other metrics.
9) High-score gate:
   - score > 0.85 is only allowed when BOTH ACC >= 0.60 and val_acc >=
    0.60 (or ACC >= 0.60 if val_acc missing).
```

```
10) Never give high score to low-quality fit:
   - If reliability < 0.60 and (NMSE is large or loss is large relative
     to better candidates),
     keep score conservative (typically <= 0.75).
```

Agent C (Experience reflector) Prompt template:

Listing 3: System prompt used by Agent-C (experience reflector).

```
Task: You are LLM-C. Produce concise, correctable experience for the
    next LLM-A round.
Context: {context or "Unknown physical background"}
Variable meanings: {json.dumps(variable_desc, ensure_ascii=False)}
Candidate briefs from this iteration:
{json.dumps(candidate_briefs or [], ensure_ascii=False)}
Strong-evidence channel:
{json.dumps(strong_pos, ensure_ascii=False)}
Weak-evidence channel:
{json.dumps(weak_channel, ensure_ascii=False)}
Return strict JSON:
{
  "memory": [],
  "guidance": {
    "good_feature_combinations": ["..."],
    "bad_feature_combinations": ["..."],
    "good_operators": ["..."],
    "bad_operators": ["..."],
    "recommended_maxdepth": <int_or_null>
  }
}
Rules:
1) Keep guidance concise and high precision. Only output these 4
    categories.
2) good_feature_combinations: at most 2 combinations.
3) Use strong channel as hard evidence; weak channel only as tentative
    hints.
4) Experience can be wrong. You must revise previous conclusions if
    current evidence contradicts them.
5) If no reliable evidence exists for an item, do not output it.
5.1) Treat re(expr)/im(expr) as exporter wrappers (real/imag part), not
     as extra operator complexity.
5.2) good_operators and bad_operators must be operator names from this
    exact set only:
{json.dumps(GLOBAL_BASE_OPERATORS, ensure_ascii=False)}.
Do NOT output natural-language phrases such as "avoid division_by_xxx".
5.3) good_feature_combinations and bad_feature_combinations must be
    expression snippets over known variables only.
Do NOT output plain-language advice in these fields.
6) If evidence is enough, set guidance.recommended_maxdepth in [6, 40].
6) Previous good combo memory (may contain stale ideas):
{json.dumps(prior_combos, ensure_ascii=False)}
```

D Programmatic Sanitization and Candidate-Retention Rules

This section specifies the programmatic safeguards used in the reported LLM-PySR experiments. We distinguish three operations that were summarized more coarsely in the main text: (i) sanitizing a planner-proposed search specification, (ii) assigning positive or negative feedback labels to the candidate pool, and (iii) updating the incumbent formula. These operations should not be conflated. In particular, the label returned in the raw LLM review is discarded and replaced by Algorithm 2. However, the reviewer score is an input to both Algorithm 2 and Algorithm 3. The reported imple-

mentation is therefore a fixed *programmatic hybrid rule*: numerical safeguards have priority, but the rule is not a numerical-only gate.

Notation. For candidate i, let a_i, v_i, n_i, $\tilde{n}_i$, ℓ_i, p_i, c_i, and s_i denote training $\mathrm{ACC}_{0.1}$, validation $\mathrm{ACC}_{0.1}$, training NMSE, validation NMSE, PySR loss, PySR score, expression complexity, and the reviewer score, respectively. A superscript $\star$ denotes the corresponding metric of the current incumbent. Reviewer scores are clipped to $[0, 1]$; reviews with a missing or non-numeric score are discarded. The validation split is available for all three matched role configurations in the reported experiments.

Admissible base operators. Before any experiment-level exclusions, the global base-operator list is

$$\mathcal{O}_{\mathrm{base}} = \{\text{addition},\ \text{subtraction},\ \text{multiplication},\ \text{division},\ \text{exponentiation},\ \texttt{sqrt}, \texttt{log}, \texttt{log10}, \texttt{exp}, \texttt{sin}, \texttt{cos}, \texttt{tan}, \texttt{abs}, \texttt{min}, \texttt{max},\ \texttt{asin}, \texttt{acos}, \texttt{atan}, \texttt{sinh}, \texttt{cosh}, \texttt{tanh}, \texttt{asinh}, \texttt{acosh}, \texttt{atanh}\}.$$

The operator list exposed to the planner is obtained by removing any experiment-level exclusions from this set.

Scope of the input-count instruction. The planner prompt requests at most five selected input variables, but the reported programmatic implementation does not truncate a nonempty valid selection to five. The hard programmatic rule is therefore the availability and deduplication rule in Table 3; the five-variable limit is a prompt-side instruction except in the empty-selection fallback. This distinction is stated explicitly to separate prompt guidance from enforced sanitization.

Search-budget scheduler. Let F_t and O_t be the numbers of selected inputs and base operators, and let A_t be the most recent training $\mathrm{ACC}_{0.1}$. The unclipped target budget is initialized as

$$B_t = 140 + 45\max(0, F_t - 3) + 12\max(0, O_t - 8).$$

It is multiplied by 1.9, 1.5, or 1.2 when $A_t < 0.1$, $0.1 \le A_t < 0.3$, or $0.3 \le A_t < 0.6$, respectively, and by 0.75 when $A_t \ge 0.9$. If the last accuracy fails to exceed the preceding accuracy by 10^{-3}, the target is additionally multiplied by 1.15. After clipping B_t to $[100, 500]$, the next budget is

$$N_{t+1} = \mathrm{clip}_{[100,500]}\left(\mathrm{round}(0.6N_t + 0.4B_t)\right).$$

The initial budget used in the reported experiments is $N_1 = 200$.

Algorithm 2 overwrites the positive/negative label proposed by the reviewer. Its first acceptance branch after bootstrap, $i \in E \wedge Q \ge 2$, is numerical-only. The second branch uses the reviewer score as a bounded hint when only one numerical quality vote is available. After two consecutive rounds without a positive label, the third branch relaxes this score-hint requirement but retains the elite, numerical-quality, and PySR-score conditions. The resulting labels are used to separate strong and weak evidence for the experience-memory agent; they are not, by themselves, the complete incumbent-update rule.

The two update attempts in Algorithm 3 are executed in the displayed order, so the positive candidate is compared with the possibly updated simplest candidate. Whenever the incumbent changes, the sanitized search plan from that round becomes the new plan anchor. Iteration terminates early only when the incumbent simultaneously reaches reviewer score $s^\star \ge 0.99$ and validation $\mathrm{ACC}_{0.1} \ge 0.99$, subject to the minimum-iteration requirement. After the iterative loop, the selected expression is further simplified by the post-hoc ablation-pruning procedure described in the main text.

E Comparative Results on Real-World Scientific Problems

We further compare the final expressions returned by all methods on two real-world scientific datasets: early prediction of lithium-ion battery cycle life (Severson et al., 2019) and recovery of the thermodynamic relation between free-energy difference and enantiomeric excess in enantioselective electrocatalysis (von Münchow et al., 2025). These cases complement the synthetic benchmarks by testing whether a method returns an expression that is not only predictive on held-out observations, but also compact enough to inspect scientifically. Table 4 reports $\mathrm{ACC}_{0.1}$, NMSE, and the number of distinct input variables in each final expression. The complete expressions are listed below.

Table 3: Programmatic sanitization and bounded-update rules applied to a raw planner proposal. "Remove" means that the invalid item is omitted while the remaining plan is retained. Unless otherwise noted, the limits are fixed across tasks.

Plan component	Enforced rule	Action on violation
Output schema	The planner output must satisfy the typed JSON schema for engineered features, selected inputs, base operators, custom operators, and maximum depth.	Reject the planner response and report a validation failure.
Selected base operators	Retain only operators in the admissible global list after applying experiment-level include/exclude settings. Remove duplicates while preserving order. User-forced admissible operators are then inserted.	Remove unknown, excluded, or repeated entries.
Custom operators	At most three unary and binary custom operators are considered in total. Each definition must contain at least two distinct admissible base operators. A unary definition may reference only x; a binary definition may reference only x and/or y. Dataset-specific identifiers and nested calls to other custom operators are forbidden. Accepted operators are renamed sequentially as $f_1, f_2, \ldots$.	Truncate to the first three proposals and remove invalid definitions.
Engineered-feature syntax	Feature names and expressions must be nonempty. Names and normalized expressions must be unique. Expressions using an experiment-level excluded operator are forbidden. A reported dimension status outside {`pass`, `unknown`} is reset to `unknown`; this field is not used as a numerical acceptance test.	Remove invalid or duplicate features; normalize an unsupported dimension status.
Engineered-feature materialization	A proposed feature name may not overwrite an existing column. Its expression must evaluate successfully to a real, finite vector with exactly the same number of rows as the input table.	Skip features that collide with an existing column, raise an evaluation error, have the wrong length, or contain a non-finite value.
Engineered-feature granularity	A feature is rejected as overly coarse when it simultaneously (i) contains division and a nonlinear operation from {`sqrt`, exponentiation, `log`, `exp`} and (ii) uses at least three variables. It is also rejected when it contains at least ten operator occurrences and uses at least three variables.	Remove the overly coarse feature.
Per-round feature budget	No more than three genuinely new engineered features are admitted in one round. Once an incumbent plan anchor exists, the bounded-update rule retains the anchor features and admits at most one additional feature, with at most three engineered features retained in total.	Truncate new features in proposal order.
Selected PySR inputs	Retain only selected variables that exist in the transformed dataset, omit the target, and remove duplicates while preserving order. If no valid input remains, use the first five available non-target columns.	Remove unavailable or repeated inputs; invoke the five-column fallback only when the sanitized set is empty.
Maximum tree depth	Clip the planner-proposed PySR maximum depth to the closed interval $[6, 40]$. If no valid depth is supplied, use the numerical runner default, which is clipped to the same interval at execution.	Clip or use the clipped default.
Base-operator drift	Once an incumbent plan anchor exists, allow at most two edit operations (additions followed by removals) to the selected base-operator set relative to the anchor in one round.	Truncate edits after the two-edit budget is exhausted.
Recovery after stagnation	If no candidate has received a positive programmatic label for two consecutive rounds and an incumbent plan anchor exists, replace the new proposal with a copy of that anchor before the remaining sanitization rules are applied.	Roll back to the incumbent plan anchor.
PySR search budget	Clip the number of PySR evolutionary iterations to $[100, 500]$. The reported initial value is 200; subsequent values are scheduled from the current feature count, operator count, and recent training accuracy.	Clip to the admissible interval.
Returned PySR candidates	Re-evaluate each expression uniformly on the training and validation sets. Remove expressions that contain an excluded operator after engineered features are expanded, and remove constant-only expressions.	Remove the candidate before review and retention labeling.

E.1 Battery cycle-life prediction

The target is $\log_{10}(N_{\text{life}})$. To keep the expressions readable, we use the following abbreviations:

$$q_{\min} = \texttt{delta_q_100_10_min}, \quad q_{\text{var}} = \texttt{delta_q_100_10_var},$$
$$s_{\text{cap}} = \texttt{capacity_fade_slope_2_100}, \quad b_{\text{cap}} = \texttt{capacity_fade_intercept_2_100},$$
$$C_2 = \texttt{discharge_capacity_cycle_2}, \quad T_{\text{int}} = \texttt{temp_integral_2_100},$$
$$R_{\min} = \texttt{min_ir_2_100}, \quad \Delta R = \texttt{ir_diff_100_2},$$
$$t_{\text{ch}} = \texttt{avg_charge_time_first_5}.$$

LLM-PySR.

$$\widehat{y} = 0.18800634 q_{\min} + 0.08823834\,|q_{\min} - 0.3875072| + 2.7426913. \tag{18}$$

Algorithm 2 Programmatic candidate-retention labels used for experience construction

Require: Candidate metrics $\{(a_i, v_i, n_i, \ell_i, p_i, s_i)\}_{i=1}^{N}$; incumbent metrics $(a^\star, v^\star, n^\star, \ell^\star, p^\star, s^\star)$; consecutive no-positive count h
Ensure: Programmatic labels $z_i \in \{\text{POSITIVE}, \text{NEGATIVE}\}$

Rank candidates lexicographically in descending order of $(v_i, a_i, p_i, -n_i, -\ell_i)$.
Let $K = \max(2, \min(4, N))$, and let E contain the first $\min(K, N)$ ranked candidates.
Let p_{med} be the median finite PySR score in the pool.
for each candidate i with a valid reviewer score s_i **do**
  $z_i \leftarrow$ NEGATIVE.
  if $s^\star < 0$ **then**
    {bootstrap state}
    $b \leftarrow 0.05$ if $h < 2$, otherwise $b \leftarrow 0.01$.
    if $i \in E$ and $a_i \geq b$ and $v_i \geq b$ and $s_i \geq 0.35$ **then**
      $z_i \leftarrow$ POSITIVE.
    end if
  else
    $G_a \leftarrow [a_i \geq \max(0.01, a^\star - 0.002)]$.
    $G_v \leftarrow [v_i \geq \max(0.01, v^\star - 0.002)]$.
    $G_n \leftarrow [\neg\text{finite}(n^\star)]$ or $[n_i \leq 1.05 n^\star]$.
    $G_\ell \leftarrow [\neg\text{finite}(\ell^\star)]$ or $[\ell_i \leq 1.05 \ell^\star]$.
    $G_p \leftarrow [p_i \geq 0.8 p^\star]$ if $p^\star > 10^{-12}$; otherwise $G_p \leftarrow [p_i \geq p_{\text{med}}]$.
    $G_s \leftarrow [s_i \geq \max(0.35, s^\star - 0.05)]$.
    $Q \leftarrow \mathbb{I}(G_a) + \mathbb{I}(G_v) + \mathbb{I}(G_n \vee G_\ell)$.
    if $i \in E$ and $Q \geq 2$ or $(Q \geq 1 \wedge G_s \wedge G_p)$ or $(h \geq 2 \wedge Q \geq 1 \wedge G_p)$ **then**
      $z_i \leftarrow$ POSITIVE.
    end if
  end if
end for
return $\{z_i\}_{i=1}^{N}$

Table 4: Final-model performance and variable count on two real-world scientific problems. The number of variables counts distinct measured or tabulated inputs in the returned expression and does not count fitted constants. Lower NMSE is better. Because several methods attain $\text{ACC}_{0.1} = 1$, NMSE and expression structure are needed to distinguish their error profiles and scientific interpretability.

Problem	Method	Variables	$\text{ACC}_{0.1} \uparrow$	NMSE$\downarrow$
Battery cycle life	LLM-PySR	**1**	1.000	0.169492688492
	LLM-SR	9	1.000	0.034658196410
	DrSR	9	1.000	**0.034658196028**
	PySR	4	1.000	0.114347693146
	LangLaw	7	1.000	0.095899300099
Enantioselective electrocatalysis	LLM-PySR	**1**	1.000	6.65×10^{-9}
	LLM-SR	9	0.423076923	0.117789391073
	DrSR	12	0.192307692	0.693930451000
	PySR	2	0.769230769	0.009836130474
	LangLaw	**1**	1.000	$\mathbf{1.9754 \times 10^{-9}}$

LLM-SR. After algebraically expanding the outer subtraction, the returned expression is

$$\widehat{y} = 2.7939691 - 0.009369131 q_{\min} - 0.11576148 q_{\text{var}} + 0.0014997437 s_{\text{cap}} - 0.048894435 b_{\text{cap}} + 0.081122356 C_2 - 0.03383006 T_{\text{int}} - 0.018372048 R_{\min} - 0.017604721 \Delta R + 0.031506652 t_{\text{ch}}. \tag{19}$$

Algorithm 3 Incumbent selection and update in the reported implementation

Require: Candidate pool P, programmatic labels z_i, reviewer scores s_i, and incumbent $f^\star$ with metrics $(a^\star, v^\star, \tilde{n}^\star, \ell^\star, p^\star, s^\star)$
Ensure: Updated incumbent $f^\star$

Define $\mathrm{AdmissibleUpdate}(i)$ as follows:
  $H_i \leftarrow$ all of $(a_i, v_i, \tilde{n}_i, \ell_i, p_i, s_i)$ are finite, $a_i \geq a^\star - 0.01$, and $v_i \geq v^\star - 0.01$.
  $S_i \leftarrow [s_i \geq s^\star + 0.02]$.
  $V_i \leftarrow [v_i \geq v^\star + 0.01] \wedge [s_i \geq s^\star - 0.02]$.
  $T_i \leftarrow [|s_i - s^\star| \leq 0.01] \wedge [|v_i - v^\star| \leq 0.005]$ and candidate i is lexicographically better in $(-\tilde{n}_i, -\ell_i, p_i)$, using 10^{-12} as the equality tolerance.
  Return $H_i \wedge (S_i \vee V_i \vee T_i)$.

Let $F = \{i : a_i \geq a^\star - 0.01,\ v_i \geq v^\star - 0.01\}$.
if $F \neq \varnothing$ **then**
  Select $j \in F$ by minimizing $(c_j, -v_j, -a_j, -p_j)$.
  Use the review score for j when available; otherwise use $s^\star$.
  if $\mathrm{AdmissibleUpdate}(j)$ **then**
    $f^\star \leftarrow f_j$ and replace all incumbent metrics.
  end if
end if

Let $P^+ = \{i : z_i = \text{POSITIVE}\}$.
if $P^+ \neq \varnothing$ **then**
  Select $k \in P^+$ by maximizing $(v_k, s_k, -\tilde{n}_k, -\ell_k, p_k)$.
  if $\mathrm{AdmissibleUpdate}(k)$ **then**
    $f^\star \leftarrow f_k$ and replace all incumbent metrics.
  end if
end if
return $f^\star$

DrSR.

$$\begin{aligned}\widehat{y} = {} & 2.79396979625 + 0.00150675242993 s_{\mathrm{cap}} - 0.0489142351767 b_{\mathrm{cap}} + 0.0811423116135 C_2 \\ & + 0.0315088619397 t_{\mathrm{ch}} - 0.0338293499066 T_{\mathrm{int}} - 0.018374757752 R_{\mathrm{min}} - 0.0176064605374 \Delta R \\ & - 0.00938111727564 q_{\mathrm{min}} - 0.115764199563 q_{\mathrm{var}}. \end{aligned} \tag{20}$$

PySR.

$$\begin{aligned}\widehat{y} = {} & 2.6897624 + 0.025488654 (q_{\mathrm{min}} + 1.945337) \Big[1.5561504 - q_{\mathrm{var}} \\ & - T_{\mathrm{int}} + \sin\Big(q_{\mathrm{min}} - T_{\mathrm{int}} - \big[q_{\mathrm{min}} - (-T_{\mathrm{int}} + \Delta R - \sin(\Delta R \cos \Delta R)) \big] \Big) \Big]. \end{aligned} \tag{21}$$

LangLaw.

$$\begin{aligned}\widehat{y} = {} & 2.7959538 - 0.03674738 \, (0.9424673)^{\Delta R / s_{\mathrm{cap}}} \\ & \times \left[-t_{\mathrm{ch}} + R_{\mathrm{min}} - \left(b_{\mathrm{cap}} - \frac{q_{\mathrm{var}}}{0.4692554} + t_{\mathrm{ch}} - T_{\mathrm{int}} - \frac{0.31669474}{t_{\mathrm{ch}} - 1.1518941} \right) \right]. \end{aligned} \tag{22}$$

Interpretation. All five methods achieve $\mathrm{ACC}_{0.1} = 1$, so the tolerance metric is saturated on this task and should not be interpreted as evidence that the models have identical residual errors. LLM-SR and DrSR obtain the lowest NMSE, and their nearly identical nine-variable linear expressions indicate that both methods converged to essentially the same multivariate predictor. LLM-PySR instead selects only the minimum early-cycle voltage-curve displacement. Its NMSE of 0.1695 is higher than those of the multivariate baselines; under the NMSE definition used in the main paper, it corresponds to $R^2 = 1 - \mathrm{NMSE} \approx 0.831$. The result should therefore be understood as a parsimony–accuracy trade-off rather than as numerical dominance.

The structure of Eq. equation 18 is directly inspectable. Writing $x = q_{\min}$, it is equivalent to

$$\widehat{y} = \begin{cases} 0.099768\,x + 2.7768843, & x < 0.3875072, \\ 0.27624468\,x + 2.7084983, & x \geq 0.3875072. \end{cases} \tag{23}$$

Thus the model identifies a single breakpoint at which the sensitivity of predicted log cycle life to early voltage-curve displacement increases. Both segments have positive slope: larger values of $q_{\min}$, corresponding to a less severe negative displacement of the early discharge curve, predict longer cycle life. This agrees qualitatively with the interpretation that a larger-magnitude early voltage-curve shift is a marker of degradation before large capacity loss becomes directly visible. Because this experiment uses the standardized feature representation indicated by the task identifier `battery_cycle_life_log_zscore`, the numerical breakpoint in Eq. equation 23 is initially expressed in the standardized coordinate

$$x = \frac{q_{\min} + 0.0487505129}{0.0197171640}.$$

Applying this inverse transformation gives the raw-coordinate expression

$$\widehat{y} = 9.53516134 q_{\min} + 4.47520444\,|q_{\min} + 0.04110997| + 3.20753531, \tag{24}$$

with a breakpoint at $q_{\min} = -0.04110997$. This is the same hinge relation reported in the main text, before rounding of its fitted constants.

The comparison exposes the practical distinction between predictive and scientific utility. The nine-variable LLM-SR and DrSR models minimize squared error more effectively, whereas the one-variable LLM-PySR relation is easier to visualize, falsify, and connect to an early degradation coordinate. PySR and LangLaw occupy intermediate error levels but return substantially more nested expressions involving four and seven variables, respectively.

E.2 Thermodynamic consistency in enantioselective electrocatalysis

This experiment is a positive-control task rather than de novo prediction of enantioselectivity. The candidate table already contains $g = \Delta\Delta G_{\text{exp}} = \texttt{delG_exp}$, which is a thermodynamic re-expression of the response ee_{exp}. The test is whether a method can select this governing coordinate from 33 candidate inputs and recover the correct analytical transformation. We use $\mu = \texttt{Dipolemoment}$, $h = \texttt{HOMO}$, $l = \texttt{LUMO}$, $o_1 = \texttt{NBO_O1}$, $c_j = \text{NBO}_{Cj}$, $n_{12} = \texttt{NBO_N12}$, and $r_{\min}, r_{\max} = \texttt{R1_Bmin}, \texttt{R1_Bmax}$.

LLM-PySR.

$$\widehat{ee}_{\text{exp}} = -99.9949 \tanh(0.6743717 g). \tag{25}$$

PySR.

$$\begin{aligned} \widehat{ee}_{\text{exp}} = \mu + \Big| & 61.849492858894(-g)^{0.8862578} \\ & - \Big| g(\mu + 0.18122451) - (-1.5210397 g)^{3.0601606} + 0.648141 \Big| \Big|. \end{aligned} \tag{26}$$

LLM-SR.

$$\begin{aligned} \widehat{ee}_{\text{exp}} = {} & 1489.7394 h - 611.8758 o_1 + 27.154155 c_4 + 472.90781 n_{12} + 671.7787 l - 117.72453 c_3 \\ & + 2950.1376\, l o_1 - 28605.013 c_{10}^2 + 255.28208 c_5 c_{10} + 5670.2506 c_{11} c_{10}. \end{aligned} \tag{27}$$

DrSR.

$$\begin{aligned} \widehat{ee}_{\text{exp}} = {} & 199.769081011 - 73.1183489338 h + 6.9072945022 l - 246.228305636 o_1 \\ & + 96.6084628241 c_3 + 95.0348972177 c_4 - 7.8644911623 c_5 + 40.5428403378 n_{12} c_{11} \\ & + 100.884987566 \frac{c_{10}}{\mu} - 98.60359531 \log(1 + r_{\min} r_{\max}). \end{aligned} \tag{28}$$

LangLaw.

$$\widehat{ee}_{\exp} = 2g\Big\{ g^2\big[g(0.2643308g + 2.5093734) + 7.338444\big] + 0.8848076g - 33.59938 + \frac{0.00026152283}{g + 0.4220596}\Big\}. \tag{29}$$

Interpretation. LLM-PySR and LangLaw are the only methods that both select $\Delta\Delta G_{\exp}$ as their sole input and attain $\mathrm{ACC}_{0.1} = 1$. LangLaw has the numerically smallest NMSE, 1.98×10^{-9}, compared with 6.65×10^{-9} for LLM-PySR; both errors are effectively negligible at the scale of the response. Their analytical implications are nevertheless different. The LangLaw expression is a high-order rational polynomial with a pole at $g = -0.4220596$. It is unbounded and does not encode the saturation of enantiomeric excess, so its excellent interpolation error does not by itself establish thermodynamic validity.

In contrast, Eq. equation 25 matches the known relation

$$ee = -100 \tanh\left(\frac{\Delta\Delta G}{2RT}\right), \tag{30}$$

obtained from $ER = \exp(-\Delta\Delta G/RT)$ and $ee = 100(ER - 1)/(ER + 1)$ (Eyring, 1935). The fitted prefactor 99.9949 recovers the physical bound of 100%, while the coefficient 0.6743717 corresponds to

$$T = \frac{1}{2R(0.6743717)} \approx 373.1\ \mathrm{K}$$

when $R = 1.9872 \times 10^{-3}\,\mathrm{kcal\,mol^{-1}\,K^{-1}}$. This agrees with the $100^{\circ}\mathrm{C}$ reaction temperature. The expression is bounded, has the correct zero at $\Delta\Delta G = 0$, and has the expected sign and saturation behavior.

PySR combines g with the dipole moment and attains intermediate predictive performance. LLM-SR and DrSR do not select the direct thermodynamic coordinate; instead, they combine nine and twelve electronic or structural descriptors and exhibit substantially larger held-out errors. The comparison therefore shows that the main advantage of LLM-PySR in this positive-control task is not merely a small NMSE. It is the recovery of the correct response-equivalent coordinate and the physically constrained analytical map relating that coordinate to enantiomeric excess.

E.3 Implications for real-world use

These two examples illustrate complementary forms of scientific usefulness. On the battery dataset, LLM-PySR sacrifices some squared-error performance in exchange for a one-variable hinge relation that exposes a testable change in sensitivity. On the electrocatalysis dataset, it achieves both essentially exact prediction and recovery of the established thermodynamic law. Thus, the real-world results do not show that LLM-PySR uniformly minimizes prediction error. Rather, they place it at a favorable point on the accuracy–parsimony–interpretability frontier: the returned expressions remain competitive in predictive accuracy while being substantially easier to inspect and, in the thermodynamic positive control, scientifically correct outside the narrow sense of interpolation error.