VIGA-SR: LLM-BASED EVOLUTIONARY SEARCH FOR SYMBOLIC REGRESSION WITH VISUAL PRIORS GUIDANCE FROM MULTIMODAL REPRESENTATION OF DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression aims to derive concise formulas that both fit data and reveal the mechanisms underlying it, with evolutionary computation (EC) as the widely adopted approach. However, EC-based SR methods must navigate a vast search space to find suitable equations. Prior knowledge can ease this burden, but existing approaches either rely on single-modal information or apply guidance indiscriminately, neglecting rich multimodal insights that human experts naturally exploit to adjust their priors during evolution. We propose ViGa-SR, a framework that automatically discovers the underlying mathematical structure of data and exploits it as prior knowledge to guide LLM-based evolutionary search. The data is first interpolated and rendered through diverse visualization schemes, from which a multimodal LLM (MLLM) progressively identifies the potential substructures, operators, and coupling relationships among the variables in the data; each identified feature is then numerically verified, with its associated constants calibrated. The identification proceeds hierarchically. After each feature is verified, it is stripped from the data, and the MLLM re-examines the simplified data. This process iterates to expose progressively deeper substructures and terminates when no further regularity can be confirmed. The verified prior knowledge guides the subsequent LLM-based evolutionary search and is re-acquired with altered visualization schemes whenever evolution stagnates or population diversity drops. Compared with purely numerical feature extraction, MLLM-based feature extraction is more robust to noise and sample sparsity, and the use of visualizations as the communication medium keeps the entire process transparent and human-readable. On the Feynman dataset and LLM-SRBench, ViGa-SR matches or exceeds state-of-the-art performance, demonstrating strong feature-extraction capability and noise robustness.

1 INTRODUCTION

Symbolic regression (SR) Schmidt & Lipson (2009a); Makke & Chawla (2024); La Cava et al. (2021) aims to discover concise mathematical expressions $f(\mathbf{X})$ from observed data $D = \{\mathbf{X}, Y\}$ by searching the combinatorial space spanned by elementary mathematical operators. Unlike black-box models, SR yields parsimonious functional structures that capture the mechanisms underlying the data, making it valuable for scientific discovery, where interpretability and extrapolation beyond the observed data are critical.

SR paradigms include heuristic search-based methods Koza et al. (1989); Koza (1990; 1994); Keijzer (2003); La Cava et al. (2016) and generative model-based methods Biggio et al. (2021); Valipour et al. (2021); Li et al. (2022); Kamienny et al. (2022); Vastl et al. (2022). However, these methods face fundamental challenges: the discrete and vast search space leads to slow convergence and poor sample efficiency, and evolutionary search is prone to diversity collapse and complexity bloat. Incorporating prior knowledge—mathematical properties such as monotonicity or symmetry, or physical laws such as dimensional consistency Lu et al. (2016); Haider et al. (2023); Tenachi et al. (2023); Schmidt & Lipson (2009b); Brence et al. (2023)—mitigates these issues, but such priors are typically data-independent or extracted by implicit computations, adapt poorly to noise and distribution shift, and cannot be revised during the search.

The recent emergence of LLM-driven SR Liu et al. (2024a); Shojaee et al. (2024); Grayeli et al. (2024); Merler et al. (2024) lets experts express priors in natural language, lowering the barrier

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.

to injecting complex requirements. Yet across this line of work the prior enters the search as *text*: expert domain knowledge, variable semantics, concepts abstracted from search outputs Grayeli et al. (2024), or previously tested expressions with their fitness scores Merler et al. (2024); the systems that close a feedback loop do so without vision—SGA, for instance, validates every hypothesis by physical simulation Ma et al. (2024). Multimodal methods add input channels but not a verified visual prior: ChatSR Li et al. (2025d) encodes tabular observations with a SetTransformer and consumes user-stated priors, with no plot channel; ViSymRe Li et al. (2024) is a supervised model whose deployment consumes no images; LLM-Lex Harvey et al. (2025) evolves plot-proposed skeletons without per-feature verification or decomposition; VIPER-R1 Liu et al. (2025) realigns residuals with an external SR tool rather than a visual re-reading; and the zero-shot system of Li Li et al. (2025b) iteratively proposes and numerically assesses hypotheses, but over video frames rather than tabular data (Appendix C). None of them gates individually read visual features of tabular data by a per-feature numerical criterion before admission; we treat an unverified structural claim as worse than none, because the search must then spend its budget recovering from it.

This paper presents **ViGa-SR**, a framework that discovers the mathematical structure of the data visually and admits only numerically verified coordinates into the search. A multimodal LLM examines renderings of the interpolated data and identifies candidate features: poles, zero sets, composite coordinates, single-variable curves, spreading envelopes, magnitude structure, multiplicative factors, and parallel bands. Every identified coordinate is then judged by a numerical criterion, and only a coordinate that passes its criterion enters the prior knowledge; a rejected claim is simply discarded. The functional form read along a verified coordinate is fitted numerically, and any seed built on a misread form is removed by held-out validation (§4). Identification proceeds hierarchically: each verified feature is stripped from the data, the simplified data is rendered again, and the MLLM re-examines it, until no further regularity can be confirmed. The resulting verified decomposition, together with its calibrated constants, seeds the initial population of an LLM-based evolutionary search and accompanies it as a textual insight block. The extraction stage is also re-invoked during the search, for a fresh independent reading, when the population stagnates on structure it cannot fit.

Four principles govern the design, and together they define the division of labor between the model and the numerics. First, every structural claim originates in the images; the framework maintains no library of candidate answers, and numerical procedures never propose structure. Second, numerical criteria only judge: they verify or reject what the MLLM has identified, and they calibrate constants. Third, no coordinate enters the prior knowledge without passing its verification. Fourth, the MLLM is always consulted; no numerical pre-filter decides on its behalf whether a view is worth reading. In short, the MLLM contributes shape recognition, while numerical procedures carry out all quantitative judgment.

A walk-through makes the division of labor concrete. On one benchmark problem (the case of Figure 2, variables anonymized), the panel of extreme samples shows clean curved stripes; asked what combination of the two axes stays constant along a stripe, the MLLM writes $u = b \cdot c$, and the crossing-localization criterion confirms the claim on the present data. The one-dimensional view along u shows evenly spaced divergences, from which the MLLM reads a periodic form $1/(1-\cos u)$; its constants are fitted numerically, the data is divided by the fitted form, and the dynamic range falls from $116{,}059$ to 3.1. The re-rendered residual exposes a second divergence along a single raw variable, verified as $u = a$; once this layer is stripped, the remaining signal reads as an additive component along $u = b$, and removing it reduces the residual variance from 0.841 to 0.0035. The following round confirms nothing, and the loop terminates. At no point did a numerical routine propose a candidate, and at no point did a visual claim enter the prior unverified; the recovered three-layer decomposition, its constants re-fitted jointly, is what the evolutionary search starts from.

Three consequences follow, and each is measurable. (1) The name channel is closed. Variable names are replaced by anonymous symbols before any processing or model involvement, and the anonymized names are the only ones any model ever sees; a correct answer cannot arrive by recalling a *named* textbook equation—the standing objection to LLM-based SR on Feynman-style benchmarks—so that channel is closed by construction. Shape recall is not: a rendering may still resemble a previously seen figure, but any structure, whether read from the image or recalled from training, must pass its numerical criterion on the present data before entering the prior. (2) Verification makes the visual channel trustworthy. On a 30-problem benchmark spanning rational-pole, trigonometric, transcendental, and smooth families, no verified coordinate was later found to be fabricated structure (Appendix J). Verified structure was extracted on 25 problems, and on 18 of the 30 the seeds assembled from the verified decomposition alone—before any evolution—already reach $R^2 > 0.5$ in at least one examination attempt: an attainable ceiling over repeated examination attempts per problem, not a

single-pass expectation (Appendix J). Each of the eight feature types covers cases that no other type reaches. (3) Hierarchical identification converts a single examination into a decomposition. Stripping each verified feature and re-examining the simplified data recovers multi-component structure that a single reading cannot see, with the decomposition depth decided by the data within a bounded window (three layers by default, six at most). Multiple extraction passes per run, plus up to one re-acquisition on a health trigger, convert per-attempt success probability into per-problem reliability.

To validate the framework, we evaluate ViGa-SR on two standard symbolic regression benchmarks—the Feynman dataset Udrescu & Tegmark (2020); Shojaee et al. (2024) and LLM-SRBench Shojaee et al. (2025)—where it matches the strongest LLM-based baselines without auxiliary expert guidance. The visual channel's cost is bounded by construction: the extraction stage spends tens of MLLM calls per run against the thousand-call text-model budget of the search (§3). The knowledge-guided evolutionary search itself adopts a standard bi-level formulation and is *not* claimed as a contribution; the contribution is the verified visual prior built in front of it and re-acquired inside it. Our contributions are:

- MLLM-based feature extraction with numerical verification. Eight feature types pair a visualization scheme with a question and a numerical criterion that judges the answer. Structure always originates in the images; the criteria only judge—refusals, not penalties.
- Hierarchical identification. Each verified feature is stripped under a numerical acceptance gate, and the MLLM re-examines the simplified data, until no feature type confirms anything—decomposition depth adapts to the data within a safety window (three layers by default, a hard cap of six).
- Verified priors as search guidance, re-acquired adaptively. The verified decomposition enters the search as seed expressions with jointly re-fitted constants and as a textual insight block in operator prompts. When the search stagnates with the signature of an unfit denominator structure, a health monitor re-invokes extraction for a fresh independent reading, at most once per run.

The rest of the paper is organized as follows: §2 describes the framework, §3 the benchmarks and validation studies, and §4 the limitations. Prompt templates, extended results, robustness studies, and related work are in the appendices.

2 METHODOLOGY

2.1 FRAMEWORK OVERVIEW

ViGa-SR couples two stages around an LLM-based evolutionary search. The first stage is MLLM-based feature extraction. The data is interpolated and rendered through diverse visualization schemes, and a multimodal LLM examines these views to identify candidate substructures, operators, and coupling relationships of the underlying equation. Every identified feature is then checked by a numerical criterion, and its constants are calibrated by numerical fitting. The identification proceeds hierarchically: after a feature is verified, its contribution is stripped from the data, and the MLLM re-examines the simplified data, repeating until no further regularity can be confirmed. The output of this stage is a verified decomposition of the data together with calibrated constants, which we call the extracted prior knowledge.

The second stage is the evolutionary search itself. The verified decomposition is assembled into candidate expressions that seed the initial population, and a textual description of the decomposition accompanies the search as guidance. The search follows the standard bi-level formulation in which the LLM proposes discrete structure and numerical optimization fits the constants; we do not claim this backbone as a contribution. The extraction stage is also re-invoked during the search, for a fresh independent reading, when the population stagnates on structure it cannot fit (§2.7). Four principles govern the division of labor between the model and the numerics (§1): structure originates only in the images, numerical criteria only judge, unverified claims are discarded, and the MLLM is always consulted. Figure 1 illustrates the architecture.

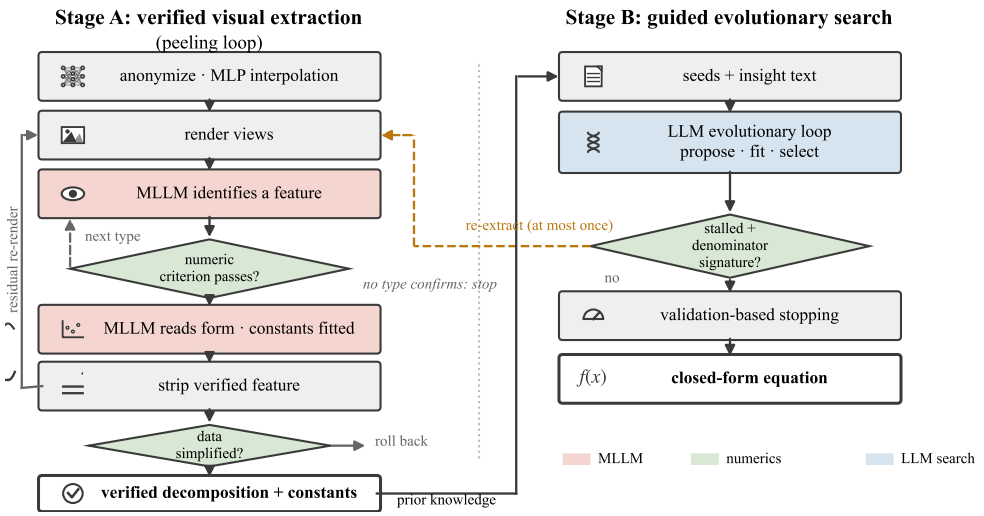


Figure 1: Framework of ViGa-SR. (Left) feature extraction: interpolated data is rendered through diverse visualization schemes, the MLLM identifies candidate features, numerical criteria verify each claim and calibrate its constants, and each verified feature is stripped so the MLLM can re-examine the simplified data. (Right) the verified decomposition seeds and guides the evolutionary search; a health monitor re-invokes extraction when the search stagnates on structure it cannot fit.

2.2 Data Preparation

Variable names are replaced by anonymous symbols $a, b, c, \ldots$. The substitution is a control, not a convenience: it removes the channel through which a correct answer could arrive by recalling a named textbook equation—the standing objection to LLM-based symbolic regression on Feynman-style benchmarks.

The raw samples are interpolated by an MLP (256–128–64, $\tanh$ activations) trained with a gradient-norm penalty. The surrogate's purpose is queryability rather than predictive accuracy: it provides a smooth representation of the data from which every view below can be rendered on demand. It is also where noise robustness is located—the surrogate absorbs the noise, the MLLM reads a smoothed function, and the multimodal stage therefore performs no denoising itself (Appendix D). Rendering from the surrogate rather than the raw samples is also what lets the visual stage tolerate sample sparsity.

2.3 Visualization Schemes

All views are scatter renderings of the interpolated data with anonymized axes; each scheme makes a different kind of structure visible. Five schemes are used: (i) *value-colored pair panels*, which plot each variable pair colored by the response, exposing level sets as color bands; (ii) *one-dimensional views* of the response against each variable, exposing single-variable curves, divergences, spreading envelopes, and parallel bands; (iii) *magnitude views*, the same renderings for $|y|$, exposing amplitude structure that the signed view hides under oscillation; (iv) *quotient views*, which render y/x_i as pair panels and one-dimensional views, exposing multiplicative factors $y = x_i \cdot F(\cdot)$; and (v) *overlay views*, which overlay a fitted candidate curve on the data so the MLLM can compare its reading and correct it. The identification loop decides which scheme is shown when (§2.4–§2.5); adaptive re-acquisition (§2.7) draws on these schemes again for a fresh independent reading when the search stalls.

2.4 Feature Identification and Numerical Verification

The framework recognizes eight feature types, summarized in Table 1, each pairing one visualization scheme with one question asked of the MLLM and one numerical criterion that judges the answer. Together they cover the ways a response can visibly depend on its variables: divergence where a denominator vanishes, a sign-changing zero set where a numerator vanishes, dependence through a

Table 1: The eight feature types. Each pairs a view with the structure the MLLM must identify and the criterion that verifies the claim. Structure originates in the images; the criteria only judge.

Feature type	View	MLLM identification	Numerical verification
Pole	pair panels, extremes highlighted	the factor whose zero set carries the extreme samples	location accuracy of predicted crossings
Zero set	pair panels, zero-crossing samples highlighted	the coordinate whose vanishing carries the sign-changing samples	crossing-localization score of the predicted zero set
Composite coordinate	value-colored pair panels	a combination u of two variables along whose level sets the color is constant	collapse R^2 of y on u
Single-variable curve	one-dimensional views	the variable on which y collapses to one clean curve	collapse R^2 on the raw variable
Spreading envelope	one-dimensional views	the variable along which the spread of y widens	envelope-strength statistic
Magnitude structure	magnitude views	the variable on which $\|y\|$ collapses	collapse R^2 of $\|y\|$
Multiplicative factor	quotient views	the divisor x_i and the coordinate u onto which y/x_i collapses	collapse R^2 of the quotient, with a gain over the undivided view
Parallel bands	one-dimensional views	the variable whose view shows shifted copies of one shared curve	the stripping acceptance gate

combined coordinate or on one raw variable, a spreading envelope, structure in the magnitude alone, a raw variable factoring out multiplicatively, and an additive component repeated as parallel bands. The eight types are generic shape categories, not a candidate equation library: no type stores or retrieves equations, each only pairs a view with a question, and every answer must still earn verification on the present data. Two criteria use a collapse statistic descended from the generalized-symmetry module of AI Feynman 2.0 Udrescu et al. (2020), where it drove a transformation-library search; here it only judges coordinates the MLLM has written down.

The verification criteria share one form: a claim is scored by a statistic that must clear a fixed threshold, calibrated once on offline data and frozen before any benchmark run (worked examples in Appendix B). A failed claim is discarded and the view stays available for the remaining types: every criterion is a refusal rather than a penalty, because a wrong feature in the prior knowledge is worse than none.

Once a feature is verified, the MLLM reads its functional form from the corresponding one-dimensional view, and all constants are calibrated by numerical fitting against the data; an overlay view (§2.3) then lets the MLLM revise a form whose fitted curve visibly departs from the data.

2.5 Hierarchical Identification

Identification proceeds in rounds on the same data. In each round the eight feature types are examined in a fixed order, from the most specific verification to the least specific, and the first type whose claim passes verification is selected for that round. The verified feature is then stripped from the data. When the view shows the feature multiplying the rest of the response, the data is divided by the fitted form; when the view shows the feature added to the rest, the fitted form is subtracted. A numerical acceptance gate checks that the stripping genuinely simplifies the data, measured by the drop in variance, dynamic range, and logarithmic spread; a stripping that fails the gate is rolled back and the round ends. The simplified data is then rendered again for the next round.

The loop terminates when no feature type confirms anything in a round, which we interpret as the absence of further visible regularity. The loop does not stop at its default budget: a round succeeding at the boundary means the simplified data still carried verifiable structure, so the budget extends by one round. Decomposition depth thus adapts to the data within a bounded window—three layers by default, extended while rounds keep succeeding, up to a hard cap of six (Appendix B); problems whose true decomposition exceeds the cap are out of scope.

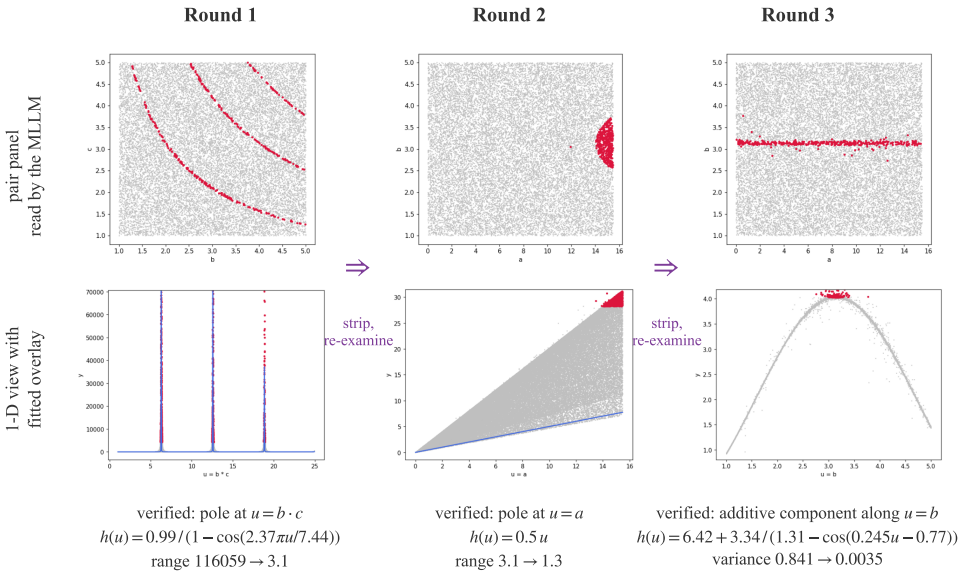


Figure 2: Hierarchical identification on a real examination round (problem `I.30.3_0_0`, variables anonymized). Each column is one round: (top) the pair panel read by the MLLM, with extreme samples highlighted; (bottom) the one-dimensional view along the identified coordinate, with the fitted form overlaid. Round 1 verifies a pole at $u = b \cdot c$ and strips a multiplicative $1/(1 - \cos)$ form (dynamic range $116059 \to 3.1$); round 2 verifies a pole at $u = a$; round 3 strips an additive component along $u = b$ (variance $0.841 \to 0.0035$). Round 4 confirms nothing, and the loop terminates. In other attempts the same coordinate family enters through the zero-set view instead (Appendix J, case (a)).

2.6 From Verified Structure to Search Guidance

The verified decomposition is delivered to the evolutionary search in two forms. The first form is a set of seed expressions. Each seed composes the stripped features back around the remaining signal, and all constants are re-fitted jointly against the training data, which absorbs the small errors accumulated when each feature was calibrated in isolation. When two functional readings of the same feature agree inside the data range but differ outside it, both are kept as alternative seeds; the search decides which generalizes. The seeds enter the initial population alongside conventionally generated candidates. The second form is a textual insight block stating the decomposition in plain language, appended to operator prompts so the LLM can reason over verified structure rather than only inherit it genetically.

The search backbone is standard. Individuals are Python expressions; five LLM operators perform crossover and mutation; every constant in an offspring is re-fitted by numerical optimization; the population is truncated by a parsimony-penalized training error; and a held-out validation split stops a run that has ceased to generalize.

2.7 Adaptive Re-acquisition

The extraction stage is not run only once. A health monitor re-invokes it on the training data when all of the following hold: the best objective has stagnated (relative improvement below 5% across two consecutive checkpoints), the stagnating objective still exceeds half the target variance, and at least half of the six best-ranked individuals contain a division—the empirical signature of a denominator structure that the search keeps writing but cannot fit, which is the setting in which a fresh visual reading is most informative. A first extraction that confirmed features does not disqualify a run: weak or partially misread seeds stall the search all the same. The re-invocation reruns the extraction pipeline with the MLLM re-sampled, an independent attempt that can land where the first reading missed; newly verified features are injected as seeds and appended to the insight block, budget unchanged. Re-acquisition fires at most once per run, bounding the cost to a single extraction pass. A companion monitor handles diversity drops: the best candidates are plotted against the target and the MLLM judges which carry the target's features (Appendix G.4).

3 EXPERIMENTS AND VALIDATION

3.1 EXPERIMENTAL SETTINGS

We use MSE as fitness during evolution and report two validation metrics: *solution rate*, the fraction of runs that exactly recover the ground-truth formula, and *accuracy solution rate*, the fraction achieving $R^2 > 0.999$ regardless of symbolic equivalence. Following prior work Liu et al. (2024a); Shojaee et al. (2024), formulas are represented in code form. We run three experiment sets: (A) the Feynman dataset Udrescu & Tegmark (2020) (the 100 equations from the Feynman Lectures on Physics, excluding the benchmark's 20 bonus equations), (B) LLM-SRBench Shojaee et al. (2025) against state-of-the-art SR methods, and (C) two nonlinear-oscillator case studies. GPT-4o performs visual feature extraction and GPT-4o-mini formula generation and evolution, both via remote API (the extraction stage is robust to the choice of MLLM; Appendix F).

To avoid conflating search capability with benchmark memorization, we strip all auxiliary textual priors (variable semantics, units, problem descriptions) and compare only against methods that ingest zero domain knowledge—pretrained knowledge is common to all LLM-based methods and is not a distinguishing input. Anonymization closes the name channel; any residual shape-level association is measured directly by the LSR-Synth self-check below. Every final result comes from the evolutionary search rather than direct LLM output.

3.2 RESULTS ON DIFFERENT BENCHMARKS

3.2.1 FEYNMAN DATASET

On the Feynman dataset (five runs per equation), ViGa-SR attains a solution rate of 65.2%, the highest among the compared methods that use no human priors (Figure 3); equations with clear visual signatures are recovered exactly. Because this comparison assembles published numbers produced under different budgets, run counts, and solution criteria (Appendix I), we read it as a placement rather than a controlled ranking; the controlled evidence is the ablation (Appendix E), where removing adaptive guidance reduces solution rates modestly while removing all prior knowledge reduces them drastically.



(a) Solution rates

(b) Accuracy rates ($R^2 > 0.999$)

Figure 3: Comparison on the Feynman dataset (noise-free condition; baseline numbers from La Cava et al. (2021) and the original reports, see Appendix I). ViGa-SR achieves highest solution rate among methods using no human priors, with competitive accuracy performance.

3.2.2 LLM-SRBENCH

We evaluate ViGa-SR on LLM-SRBench Shojaee et al. (2025) (LSR-Transform subset, 111 equations, 10 runs each), with a budget of 1,000 LLM calls per problem and baseline numbers taken from the benchmark paper under their own configurations (Table 2; the budget-matching caveat is discussed in §4). ViGa-SR attains the highest solution rate, marginally exceeding LLM-SR (32.02% vs. 31.53%, a difference of roughly five of the 1,110 runs), and remains competitive on Acc@0.1 and NMSE, on par with LLM-SR.

We run a sample of the LSR-Synth subset (6 problems per domain, 24 total, chosen without difficulty screening; single run per problem; semantics stripped; the standard stack of GPT-4o visual extraction with GPT-4o-mini evolution) as a descriptive anti-memorization self-check, not a leaderboard comparison (Table 19). Under the strict criterion ($R^2 > 0.999$) $3/24$ problems are solved, while numerical fit

Table 2: Comparison on LLM-SRBench; rates aggregate all runs (111 problems, 10 runs each). ViGa-SR attains the highest solution rate while remaining competitive on Acc@0.1 and NMSE.

Method	Solution Rate (%)↑	Acc@0.1 (%)↑	NMSE↓
ViGa-SR (Ours)	**32.02**	40.23	0.007
LLM-SR	31.53	39.64	0.009
LaSR	6.31	50.45	0.001
SGA	9.91	8.11	0.232

stays strong (median $R^2 = 0.957$; $17/24$ above 0.9): on equations no model can name, the pipeline still produces strong numerical fits in the median case rather than failing outright (Appendix H).

3.2.3 Nonlinear-Oscillator Case Studies

We adopt the two nonlinear-oscillator equations of Shojaee et al. (2024), integrated with `solve_ivp` (SciPy), as dynamical-system case studies (details in Section A).

Oscillator-1: $\ddot{x} = F\sin(\omega x) - \alpha\dot{x}^3 - \beta x^3 - \gamma x\dot{x} - x\cos x$; Oscillator-2: $\ddot{x} = F\sin(\omega t) - \alpha\dot{x}^3 - \beta x\dot{x} - \delta x\exp(\gamma x)$.

We use population 40 for 500 iterations, repeated ten times. Two prior-knowledge settings are considered: (i) MLLM-extracted features only, the primary configuration without expert priors, and (ii) the same features augmented with the expert priors of Shojaee et al. (2024), included solely so that the placement against that baseline is like-for-like. Setting (i) recovers Oscillator-1 in 4/10 runs and Oscillator-2 in 3/10 runs at about 500 average iterations; setting (ii) raises these to 7/10 and 5/10, at about 300 and 200 iterations.

3.3 Validation Experiments

For interpolation robustness and data-condition analysis, we systematically evaluate four interpolation methods (RBF Spline, GP, Vanilla MLP, MLP with gradient penalty (MLP-GradP)) across data distribution, noise level, and sampling sparsity. MLP-GradP, using three hidden layers (256–128–64) with Tanh activation and gradient-penalty regularization, achieves the smoothest gradients—critical for reliable visual feature extraction—and demonstrates superior impulse-noise resilience (two orders of magnitude lower error than kernel-based approaches). For sampling sparsity, reducing the sample count from 1000 to the 100–300 range degrades interpolation quality by only 1.9–3.8$\times$ relative to the 1000-sample baseline (Appendix D).

For extraction-stage robustness to noise and sparsity, we measure the extraction stage end to end under additive Gaussian noise and uniform subsampling (Appendix D): in single-attempt cells, all three tested problems still yield verified structure at the strongest noise (seed R^2 0.83–0.99) and at 2000 samples, because the MLP surrogate absorbs the degradation before the MLLM reads it. These cells probe whether extraction can still succeed under degraded inputs; the dominant variability is the stochasticity of the visual reading.

For MLLM feature-extraction reliability, we compared multimodal feature extraction against numerical methods across 2700 predictions (diverse distributions, sparsities, and noise levels), scored by a canonical rubric applied identically to both sides. The visual approach outperforms the numerical baselines overall by 11.2%, with the advantage concentrated exactly where finite-difference numerics are fragile, and accuracy remains stable across noise levels, sample sizes, and distributions (Appendix F). Raw first-look accuracy is in any case a lower bound on the guidance the search actually receives: every read claim must still pass its numerical criterion before entering the prior knowledge (Appendix J).

For cost efficiency, the visual channel's cost is bounded by construction, not scaling with the search budget. The accounting is explicit: an identification round costs two scan calls per consulted feature type (double-sampled), about three calls per candidate panel, and four shared calls to read a verified feature's form—roughly nine MLLM calls for a round confirming one panel, sixteen for a fully silent round. Over the Feynman extraction examination (the same 100 equations with 11 duplicate functional forms removed, 89 unique problems), the loop commits about one layer per pass and stays silent on roughly 40% of problems; with two passes and at most one re-acquisition (§2.7), the visual channel costs tens of MLLM calls per run against the 1,000-call text-model budget.

4 LIMITATIONS AND APPLICABILITY

ViGa-SR discovers structure visible in renderings of the data, so its boundaries are those of what a plot can show. Appendix J makes them concrete: structure was verified on 25 of 30 problems (24 in the examination rounds plus III.17.37 in a directed single-problem examination), and the five silent ones separate into correct silence, invisible structure, and perception-limited reading (a sixth, III.17.37, sits at the boundary, its identification succeeding in only about one attempt out of four).

Structure through three or more variables is out of reach of pair-based views. All current views render the response against at most two variables. A response factoring through three or more—`I.13.12` through three, `II.36.38_5_0` through five—cannot be displayed by any current view, and no question recovers what an image does not show. Triple panels or learned projections are the natural extension.

Multiplicative envelopes without poles escape the collapse criterion. The composite-coordinate test verifies a coordinate u by how strongly the response collapses onto u pointwise, which covers additive and pure-composition structure but not a bounded multiplicative envelope $y = h(u) \cdot \mathrm{rest}$. The pole type catches the divergent special case and the quotient views catch single-raw-variable factors, but the general bounded-envelope case escapes all eight criteria; we report this gap explicitly.

Single-attempt variance is the dominant error source. The MLLM's reading is stochastic: the same data can yield a seed at test $R^2 = 0.994$ in one attempt and a diverging seed in another. Multiple passes plus at most one health-triggered re-acquisition mitigate this per-attempt variance at the per-problem level, without eliminating it. Verification also guarantees presence, not usefulness: a verified feature may cover only part of the response, and—as the wider 89-problem examination shows—a verified coordinate can still carry a partly over-read form $h(u)$; such layers simplify the data less than a true form would, their seeds fit poorly, and the validation split removes them, so neither mode yields a confidently wrong prior—yet verification does not guarantee a solution. Finally, per-attempt and per-problem statements must not be conflated: the Appendix J numbers are an attainable ceiling over repeated attempts, and a single pass will frequently do worse.

Anonymization blocks name recall, not shape recall. Replacing variable names with anonymous symbols blocks recall of a named textbook equation, but not *shape* memory: the model may still recognize a familiar curve family in a rendering. The framework's safeguard is structural: recall alone is not actionable, since every claimed structure, whether read from the image or recalled from training, must pass the same numerical criterion on the present data. The LSR-Synth self-check (§3) probes this where recall is useless: the strict-criterion solve rate drops to $3/24$ while numerical fit stays honest (median $R^2 = 0.957$)—a descriptive stress test without a comparator arm, not a measurement of memorization resistance.

Two reporting caveats apply. First, the LLM-SRBench comparison uses our fixed budget against baselines run under their own configurations, a best-effort placement. Second, every candidate undergoes numerical re-fitting of its free scalars (§2.6), so a structurally wrong expression can reach high test R^2 by coefficient optimization alone; we therefore report the strict accuracy solution rate ($R^2 > 0.999$) and inspect the symbolic form.

5 CONCLUSION

This paper presented ViGa-SR, a framework that discovers the mathematical structure of data visually and admits only numerically verified coordinates into the search: an MLLM identifies features of eight types from renderings of interpolated data, numerical criteria judge every claim, and identification proceeds hierarchically until no further regularity is confirmed. The verified decomposition seeds and guides the search, with at most one re-invocation when the search stalls after a silent initial extraction. The extraction stage carries the strongest evidence: across a 30-problem benchmark, no verified coordinate proved fabricated, and on 18 of the 30 a seed built from the verified decomposition alone reaches $R^2 > 0.5$ in at least one attempt before any evolution—an attainable ceiling over repeated attempts, not a single-pass expectation. On the Feynman dataset and LLM-SRBench, ViGa-SR matches the strongest LLM-based baselines, and semantics stripping closes the name channel. Pairing a perceptual model with numerical verification suggests a design pattern that may transfer beyond symbolic regression: perceptual models propose structure, and numerical criteria decide admission.

ETHICS STATEMENT

All experiments use public benchmarks (Feynman, LLM-SRBench) and involve no human subjects, no sensitive or personal data, and no potentially harmful applications. We are not aware of any conflicts of interest or ethical concerns raised by this work.

REPRODUCIBILITY STATEMENT

Section 2 describes the framework, Appendix B provides the pseudocode and the verification criteria, Appendix G provides the prompt templates, and Section 3.1 with Appendix I details the experimental settings and hyperparameters. All benchmarks are publicly available, and the source code is included as anonymized supplementary material.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing of this manuscript, and not for research ideas, methodology, experimental code or results, or scientific conclusions. All AI-assisted text was reviewed for technical accuracy; we take responsibility for the final content of this work.

REFERENCES

Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pp. 879–886, 2014.

Douglas Adriano Augusto and Helio JC Barbosa. Symbolic regression via genetic programming. In *Proceedings. Vol. 1. Sixth Brazilian symposium on neural networks*, pp. 173–178. IEEE, 2000.

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.

Iwo Błądek and Krzysztof Krawiec. Counterexample-driven genetic programming for symbolic regression with formal constraints. *IEEE Transactions on Evolutionary Computation*, 27(5): 1327–1339, 2022.

Jure Brence, Sašo Džeroski, and Ljupčo Todorovski. Dimensionally-consistent equation discovery through probabilistic attribute grammars. *Information Sciences*, 632:742–756, 2023.

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.

Edgar Buckingham. On physically similar systems; illustrations of the use of dimensional equations. *Physical Review*, 4(4):345–376, 1914.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, 2019.

Sihan Chen, Handong Li, Qunbo Wang, Zijia Zhao, Mingzhen Sun, Xinxin Zhu, and Jing Liu. Vast: A vision-audio-subtitle-text omni-modality foundation model and dataset. *Advances in Neural Information Processing Systems*, 36:72842–72866, 2023.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023. URL `https://arxiv.org/abs/2305.01582`.

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, 2020.

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

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

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.

Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47, 2016.

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

Thomas R. Harvey, Fabian Ruehle, Kit Fraser-Taliente, and James Halverson. Symbolic regression with multimodal large language models and kolmogorov arnold networks, 2025. URL `https://arxiv.org/abs/2505.07956`.

Dawei Huang, Chuan Yan, Qing Li, and Xiaojiang Peng. From large language models to large multimodal models: A literature review. *Applied Sciences*, 14(12):5068, 2024.

Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. Sindy-pi: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A*, 476(2242):20200279, 2020.

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

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

Maarten Keijzer and Vladan Babovic. Dimensionally aware genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 1999.

Liron Simon Keren, Alex Liberzon, and Teddy Lazebnik. A computational framework for physics-informed symbolic regression with straightforward integration of domain knowledge. *Scientific Reports*, 13(1):1249, 2023.

Michael F Korns. Accuracy in symbolic regression. In *Genetic Programming Theory and Practice IX*, pp. 129–151. Springer, 2011.

John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, volume 34. Stanford University, Department of Computer Science Stanford, CA, 1990.

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

John R Koza et al. Hierarchical genetic algorithms operating on populations of computer programs. In *IJCAI*, volume 89, pp. 768–774, 1989.

Krzysztof Krawiec and Tomasz Pawlak. Approximating geometric crossover by semantic backpropagation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 941–948, 2013.

Gabriel Kronberger, Fabricio Olivetti de França, Bogdan Burlacu, Christian Haider, and Michael Kommenda. Shape-constrained symbolic regression—improving extrapolation with prior knowledge. *Evolutionary computation*, 30(1):75–98, 2022.

Jiří Kubalík, Erik Derner, and Robert Babuška. Multi-objective symbolic regression for physics-aware dynamic modeling. *Expert Systems with Applications*, 182:115210, 2021.

William La Cava, Kourosh Danai, and Lee Spector. Inference of compact nonlinear dynamic models by epigenetic local search. *Engineering Applications of Artificial Intelligence*, 55:292–306, 2016.

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

Mikel Landajuela, Brenden K. Petersen, Sookyung Kim, Claudio P. Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F. Pettit, and Daniel Faissol. A unified framework for deep symbolic regression. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022.

Da Li, Junping Yin, Jin Xu, Xinxin Li, and Juan Zhang. Visymre: Vision multimodal symbolic regression, 2024. Current version supersedes the v1 meta-learning formulation.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *Science China Information Sciences*, 68(10):200102, 2025a.

Ruikun Li, Yan Lu, Shixiang Tang, Biqing Qi, and Wanli Ouyang. Mllm-based discovery of intrinsic coordinates and governing equations from high-dimensional data, 2025b. URL https://arxiv.org/abs/2505.11940.

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.

Yanjie Li, Jingyi Liu, Min Wu, Lina Yu, Weijun Li, Xin Ning, Wenqiang Li, Meilan Hao, Yusong Deng, and Shu Wei. Mmsr: symbolic regression is a multi-modal information fusion task. *Information Fusion*, 114:102681, 2025c.

Yanjie Li, Lina Yu, Weijun Li, Min Wu, Jingyi Liu, Wenqiang Li, Shu Wei, and Yusong Deng. Chatsr: Multimodal large language models for scientific formula discovery, 2025d. URL https://arxiv.org/abs/2406.05410.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024a.

Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2871–2883, 2024b.

Jiaqi Liu, Songning Lai, Pengze Li, Di Yu, Wenjie Zhou, Yiyang Zhou, Peng Xia, Zijun Wang, Xi Chen, Shixiang Tang, Lei Bai, Wanli Ouyang, Mingyu Ding, Huaxiu Yao, and Aoran Wang. Mimicking the physicist's eye:a vlm-centric approach for physics formula discovery, 2025. URL https://arxiv.org/abs/2508.17380.

Qiang Lu, Jun Ren, and Zhiguang Wang. Using genetic programming with prior formula knowledge to solve symbolic regression problem. *Computational intelligence and neuroscience*, 2016(1): 1021378, 2016.

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.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12585–12602, 2024.

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

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations, 2016.

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.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

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

Ali Narin. Evolutionary reward design and optimization with multimodal large language models. In *Proceedings of the 3rd Workshop on Advances in Language and Vision Research (ALVR)*, pp. 202–208, 2024.

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

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 4442–4450, 2018.

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

Michael D Schmidt and Hod Lipson. Incorporating expert knowledge in evolutionary search: a study of seeding methods. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pp. 1091–1098, 2009b.

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. *arXiv preprint arXiv:2404.18400*, 2024.

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*, 2025.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 404–415, 2006.

Zhilong Song, Qionghua Zhou, Chunjin Ren, Chongyi Ling, Minggang Ju, and Jinlan Wang. Llm-feynman: leveraging large language models for universal scientific formula and theory discovery. *arXiv preprint arXiv:2503.06512*, 2025.

Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search, 2023. URL https://arxiv.org/abs/2205.13134.

Wassim Tenachi, Rodrigo Ibata, and Foivos I Diakogiannis. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4860–4871, 2020.

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

Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. Symbolicgpt: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131*, 2021.

Martin Vastl, Jonáš Kulhánek, Jiří Kubalík, Erik Derner, and Robert Babuška. Symformer: End-to-end symbolic regression using transformer. *arXiv preprint arXiv:2205.15764*, 2022.

Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the genetic and evolutionary computation conference*, pp. 1084–1092, 2019.

Runxiang Wang, Boxiao Wang, Kai Li, Yifan Zhang, and Jian Cheng. Drsr: Llm based scientific equation discovery with dual reasoning from data and experience, 2025a. URL https://arxiv.org/abs/2506.04282.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499, 2024.

Zhifeng Wang, Wanxuan Wu, Chunyan Zeng, and Jialiang Shen. Physioformer: Integrating multimodal physiological signals and symbolic regression for explainable affective state prediction. *PLoS One*, 20(10):e0335221, 2025b.

M-J Willis, Hugo G Hiden, Peter Marenbach, Ben McKay, and Gary A Montague. Genetic programming: An introduction and survey of applications. In *Second international conference on genetic algorithms in engineering systems: innovations and applications*, pp. 314–319. IET, 1997.

Melvin Wong, Thiago Rios, Stefan Menzel, and Yew Soon Ong. Prompt evolutionary design optimization with generative shape and vision-language models. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2024.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, et al. mplug-docowl: Modularized multimodal large language model for document understanding. *arXiv preprint arXiv:2307.02499*, 2023.

Zihan Yu, Guanren Wang, Jingtao Ding, Huandong Wang, and Yong Li. Beyond accuracy and complexity: The effective information criterion for structurally stable symbolic regression, 2026. URL https://arxiv.org/abs/2509.21780.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 543–553, 2023a.

Yanzhe Zhang, Ruiyi Zhang, Jiuxiang Gu, Yufan Zhou, Nedim Lipka, Diyi Yang, and Tong Sun. Llavar: Enhanced visual instruction tuning for text-rich image understanding. *arXiv preprint arXiv:2306.17107*, 2023b.

A Introduction to Benchmark Problems

A.1 Synthetic Benchmark

This section describes the symbolic regression benchmarks employed in our study. To probe how the pipeline behaves across distinct equation families, we assemble a compact yet diverse synthetic suite of ten canonical formulas drawn from R Krawiec & Pawlak (2013), Korn Korns (2011), Keijzer Keijzer (2003), Nguyen Uy et al. (2011), and Livermore Petersen et al. (2019), used in the interpolation-robustness and sensitivity studies of Appendix D. The selection spans simple polynomials (Nguyen-1, Keijzer-15), high-order polynomials (Nguyen-4), elementary transcendental forms involving single sine or cosine (Korn-4, Jin-4/5), composite equations mixing logarithmic, exponential, and trigonometric operators (Livermore-4, Keijzer-4, Korn-15), and hybrid structures that entangle polynomial and non-polynomial components (Vladislavleva-7). The full list is provided in Table 3 and Table 4.

Table 3: Univariate equations selected from different symbolic regression benchmarks.

Name	Equation	Vars	Sampling	Feature
Nguyen-1	$x^3 + x^2 + x$	1	$\mathcal{U}(-1, 1, 20)$	Simple polynomials
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1	$\mathcal{U}(-1, 1, 20)$	Complex polynomials
Korn-4	$-2.3 + 0.13\,\sin(z)$	1	$\mathcal{U}[-50, 50, 10000]$	Simple non-polynomials
Livermore-4	$\log(x + 1) + \log(x^2 + 1) + \log(x)$	1	$\mathcal{U}[0, 2, 20]$	Complex non-polynomials
Keijzer-4	$x^3\, e^{-x} \cos(x) \sin(x) \big(\sin^2(x) \cos(x) - 1\big)$	1	$\mathcal{E}[0, 10, 0.05]$	Hybrids

Table 4: Multivariate equations selected from different symbolic regression benchmarks.

Name	Equation	Vars	Sampling	Feature
Keijzer-15	$\frac{x^3}{5} + \frac{y^3}{2} - y - x$	2	$\mathcal{U}[-3, 3, 20]$	Simple polynomials
Jin-4	$1.5\, e^x + 5.0 \cos(y)$	2	$\mathcal{U}(-3, 3, 100)$	Simple non-polynomials
Jin-5	$6.0 \sin(x) \cos(y)$	2	$\mathcal{U}(-3, 3, 100)$	Simple non-polynomials
Korn-15	$12 - 6\,\frac{tan(x)}{e^y}\big(\ln(z) - \tan(v)\big)$	4	$\mathcal{U}[0, 50, 10000]$	Complex non-polynomials
Vladislavleva-7	$(x-3)(y-3) + 2\sin\big((x-4)(y-4)\big)$	2	$\mathcal{U}[0.05, 6.05, 300]$	Hybrids

A.2 Feynman dataset

The Feynman dataset Udrescu & Tegmark (2020) is a benchmark including 100 equations drawn from the Feynman Lectures on Physics plus 20 bonus equations from classical textbooks. For each equation, it provides data, the true analytic equation, and physical units, enabling standardized testing of symbolic regression algorithms. In this paper, experiments were conducted on the Feynman dataset to compare with other traditional symbolic regression algorithms La Cava et al. (2021) that incorporate neither multimodal information nor any prior knowledge.

Unlike other LLM-based symbolic regression methods that rely on hand-crafted priors from human experts, our evaluation withholds variable names, problem descriptions, and any textual hints. Only features extracted by an MLLM from the plots are fed to the LLM in our experiment on Feynman; supplying further context would conflate search capability with dataset memorization artifacts, thereby compromising benchmark validity Shojaee et al. (2024; 2025).

Beyond these conventional benchmarks, we also evaluate our method on LLM-SRBench Shojaee et al. (2025), introduced as the first comprehensive benchmark for rigorously evaluating LLM-driven scientific equation discovery. It assembles 239 cross-domain problems via two pipelines: LSR-Transform symbolically re-casts 111 classic Feynman equations into unfamiliar mathematical forms by swapping dependent/independent variables, while LSR-Synth auto-generates 128 novel yet physically plausible equations by blending literature-known terms with GPT-4o-invented synthetic terms, followed by solvability checks and expert validation.

A.3 NONLINEAR OSCILLATOR EQUATIONS

We adopt the two synthetic damped oscillators proposed in Shojaee et al. (2024) as practical scientific discovery tasks. The dynamics are governed by second-order ODEs of the form

$$\ddot{x} = f(t, x, \dot{x}),$$

where the nonlinear forcing term f couples position, velocity, and time. The first oscillator

$$\ddot{x} = F\sin(\omega x) - \alpha\dot{x}^3 - \beta x^3 - \gamma x\dot{x} - x\cos x$$

contains amplitude-dependent driving and cubic damping/restoring terms; the second oscillator

$$\ddot{x} = F\sin(\omega t) - \alpha\dot{x}^3 - \beta x\dot{x} - \delta x\exp(\gamma x)$$

replaces the position-dependent drive with an explicit time-periodic force and adds exponential stiffness. Both equations are integrated over $t \in [0, 50]$ s with identical initial conditions $x(0) = 0.5$, $\dot{x}(0) = 0.5$. Parameter settings and initial conditions for the two nonlinear oscillator equations are shown in Table 5.

Table 5: Parameter settings and initial conditions for the two nonlinear oscillator equations.

DATASET	TIME RANGE	INITIAL VALUES	F	α	β	δ	ω
OSCILLATOR 1	(0, 50)	$x_0 = 0.5,\ v_0 = 0.5$	0.8	0.5	0.2	0.5	1.0
OSCILLATOR 2	(0, 50)	$x_0 = 0.5,\ v_0 = 0.5$	0.3	0.5	1.0	5.0	0.5

Below is the prior knowledge provided to the LLM in LLM-SR Shojaee et al. (2024), which we adopt verbatim as the expert-prior component of the augmented configuration in §3.

```
Prior knowledge from LLM-SR of Oscillator1: The goal is to find the
mathematical expression skeleton that represents second state equation of a
custom nonlinear damped oscillator (dv/dt) dynamical system with nonlinear
driving force, restoring force, and damping force, the input are position
(x), velocity (v) and the output is acceleration (a).

Prior knowledge from LLM-SR of Oscillator2: The goal is to find the
mathematical expression skeleton that represents second state equation of a
custom nonlinear damped oscillator (dv/dt) dynamical system with nonlinear
driving force, restoring force, and damping force, the input are time (t),
position (x), velocity (v) and the output is acceleration (a).
```

Design results are shown in Table 6 and Table 7. The LLM-SR rows quote the numbers reported by Shojaee et al. (2024) under their own configuration.

Table 6: Solution rate and average iterations for equation recovery on Oscillator 1.

METHOD	SUCCESSFUL RECOVERIES	AVG. ITERATIONS
LLM-SR	NOT PROVIDED	∼2000
VIGA-SR (MLLM FEATURES ONLY)	4 / 10	∼500
VIGA-SR (MLLM FEATURES + LLM-SR PRIORS)	7 / 10	∼300

Table 7: Solution rate and average iterations for equation recovery on Oscillator 2.

METHOD	SUCCESSFUL RECOVERIES	AVG. ITERATIONS
LLM-SR	NOT PROVIDED	∼400
VIGA-SR (MLLM FEATURES ONLY)	3 / 10	∼500
VIGA-SR (MLLM FEATURES + LLM-SR PRIORS)	5 / 10	∼200

B DETAILED INTRODUCTION TO SUB-COMPONENTS

B.1 INTERPOLATION USING MLP

The objective of this component is to achieve a smooth, accurate and continuous representation of discrete data, ensuring that control-variable plots remain coherent across different noise levels. By

doing so, the MLLM is able to capture the essential features of the data as faithfully as possible. To this end, a variety of interpolation methods can be employed (e.g., RBF spline, Gaussian process, or MLP). Among these, we adopt a feed-forward neural network with gradient-penalty regularization as the default interpolator, as it offers the smoothest gradients and superior robustness to noise. This is achieved by converting the model into a *smooth, interpretable response surface* through penalizing the sensitivity of its predictions with respect to the input variables. Instead of only minimizing MSE during training, we augment the objective with

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{GP}} \mathbb{E}_{\mathbf{x}}[\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_p], \qquad p \in \{1, 2\},$$

where the expectation is taken over the training distribution and λ_{GP} controls the strength of the regularizer. The pseudo-code for this part is given in Algorithm 1. Minimizing this composite loss forces the network to learn functions that (i) fit the data and (ii) vary slowly in regions densely populated by samples, thereby suppressing unwarranted oscillations.

The gradient term is computed via automatic differentiation: each mini-batch is temporarily given requires_grad=True, a backward pass is executed with unit output-gradient, and the resulting input gradient is reduced to a scalar penalty. Because the regularizer is applied *point-wise* rather than on a latent layer, it directly constrains the *input–output map*, yielding a model that is amenable to symbolic post-processing and faithful to prior physical knowledge (e.g. monotonicity, local linearity).

Two training configurations are provided. The first uses a single hold-out split with early-stopping and learning-rate scheduling. The second embeds the same routine within a K-fold cross-validated random search, enabling automated selection of the penalty weight, architecture depth, dropout rate, and optimization parameters; both variants share an identical computational graph. Experiments on the Feynman dataset show that the default configuration in Table 8 already yields high-precision continuous interpolation for most tasks. When this fixed configuration fails to produce an acceptable fit, we apply cross-validation to search for a new hyper-parameter combination over the ranges also listed in Table 8.

Table 8: Hyper-parameters for MLP interpolation with gradient-penalty regularization. Default values are used after cross-validation on the Feynman dataset; search ranges for cross-validation are shown where applicable.

Symbol	Parameter	Default	CV Range
λ_{GP}	`lambda_grad`	0.002	$0,\ 10^{-4} \sim 10^{-2}$
p	`grad_norm`	1	$\{1,\ 2\}$
$\mathbf{L}$	`hidden_layers`	[256, 128, 64]	[128, 64] (if tuned)
α	`lr`	0.0005	$5 \times 10^{-4} \sim 10^{-3}$
δ	`dropout`	0.0	$0.0 \sim 0.3$
B	`batch_size`	64	$\{16,\ 32,\ 64\}$
P	`patience`	100	$50 \sim 200$
γ	`lr_factor`	0.5	–
E	`epochs`	3000	$500 \sim 1000$

Algorithm 1 MLP Training with Gradient Penalty (Default interpolation method in our framework)

Input: data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, hyper-parameters $\lambda_{\text{GP}}, p, B, E, P, \ldots$
Split $\mathcal{D}$ into train/validation/test sets.
Standardize $\mathbf{x}$; build loader $\mathcal{B}$ with batch size B.
Initialize network f_θ, optimizer Adam(θ), scheduler ReduceLROnPlateau.
for $e = 1$ **to** E **do**
  $\text{loss}_{\text{epoch}} \leftarrow 0$
  for each batch $(\mathbf{x}_b, y_b) \in \mathcal{B}$ **do**
    $\mathbf{x}_b \leftarrow \mathbf{x}_b$.detach().requires_grad_($True$)
    $\hat{y}_b \leftarrow f_\theta(\mathbf{x}_b)$
    $\mathcal{L}_{\text{MSE}} \leftarrow \|\hat{y}_b - y_b\|_2^2$
    $\mathbf{g} \leftarrow \nabla_{\mathbf{x}_b} \hat{y}_b$
    $\mathcal{L}_{\text{GP}} \leftarrow \|\mathbf{g}\|_p$
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{MSE}} + \lambda_{\text{GP}} \cdot \mathcal{L}_{\text{GP}}$
    Update θ via Adam w.r.t. $\mathcal{L}$; accumulate $\mathcal{L}_{\text{MSE}}$.
  end for
  Compute validation MSE; update learning rate if plateau.
  if no improvement for P consecutive epochs **then**
    Restore best weights; **break**
  end if
end for
Output: trained f_θ and scaler.

B.2 Automatic Constant Identification and Optimization

Existing research Shojaee et al. (2024); Merler et al. (2024); Harvey et al. (2025) has shown that using LLMs for skeleton design and traditional algorithms for constant optimization can substantially improve performance. However, most existing LLM-based symbolic regression methods directly incorporate constant terms as unknown variables into equation inputs for optimization Shojaee et al. (2024); Harvey et al. (2025). This approach discards the MLLM-extracted priors, since for the same equation skeleton, variations in constant parameters can fundamentally alter equation properties (e.g., $-x^2$ and x^2 have opposite convexity). This paper modifies the conventional constant optimization framework to better preserve the prior knowledge extracted from multimodal information.

The bi-level search framework applied in this paper includes three steps, as demonstrated in Figure 4. In the first step, all non-exponential constant terms are identified from the code-formatted mathematical expressions using SymPy Meurer et al. (2017). In the second step, random perturbations are added to these constants with zero mean and controlled variance (set to 10% of the constant's value in implementation) to introduce randomness and reduce the likelihood of optimization converging to local optima. In the third step, optimization methods (e.g., the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method Fletcher (2013)) refine these constants, starting from the LLM-proposed values, for better precision. To ensure the physical validity of the derived expressions, any constant coefficient whose absolute value falls below a prescribed tolerance (e.g., 0.001) is automatically set to zero.

B.3 Implementation Detail: Data-Driven Parsimony Pressure

This subsection documents an implementation-level refinement of the fitness function. It is *not* part of the paper's central claim (the scheduling of visual evidence, §2); we record it here so that the released code is fully reproducible.

A parsimony pressure (also called *complexity penalty*) multiplies each candidate's MSE by an exponential factor of the formula's structural complexity, $\text{MSE}_{\text{adj}} = \text{MSE} \cdot \exp(\text{complexity}/\text{scale})$, so that bloated expressions that overfit noise are penalized relative to short ones. A fixed $\text{scale} = 30$ for every problem works well when the data are simple (long formulas are overfit risks and should be penalized) but is mismatched on problems with strong variable coupling, where longer formulas are required to express the genuine structure—the static penalty suppresses precisely those candidates the search needs to find.

We replace the static scale with a data-driven function $\text{scale}(K)$, where K is the number of strongly-coupled feature pairs detected from the data before evolution starts (one-time, $\mathcal{O}(n)$ cost). The

Algorithm 2 Automatic Constant Identification and Optimization

Input: population $\mathcal{P} = \{\text{code}_i\}_{i=1}^{n}$, data $(\mathbf{X}, \mathbf{y})$, noise-std σ, trunc-thresholds (τ_a, τ_b)
for each individual code_i in $\mathcal{P}$ **do**
  // 1. extract RHS expression
  $expr \leftarrow \text{extract_expression}(\text{code}_i)$
  // 2. add noise only to non-exponent constants
  $expr' \leftarrow \text{add_noise_to_constants}(expr, \sigma)$
  // 3. parameterize: skip exponents, wrap naked vars, funcs
  $(_, sym_expr, params) \leftarrow \text{find_constants}(expr', \mathbf{symbol_map})$
  // 4. numeric optimization
  $loss(p) = \|lambdify(\mathbf{X}, p) - \mathbf{y}\|^2$
  $opt \leftarrow \text{minimize}(loss, params, \text{method=L-BFGS-B})$
  // 5. truncate extreme values
  $opt \leftarrow \text{clip}(opt, \tau_a, \tau_b)$
  // 6. rewrite code with optimized constants
  $new_code_i \leftarrow \text{rewrite_code}(\text{code}_i, sym_expr, opt)$
  // 7. accept if improves objective
  if $\text{evaluate}(new_code_i) \leq \text{objective}_i$ **then**
    $\text{code}_i, \text{objective}_i \leftarrow new_code_i, \text{evaluate}(new_code_i)$
  end if
end for
return updated population $\mathcal{P}$

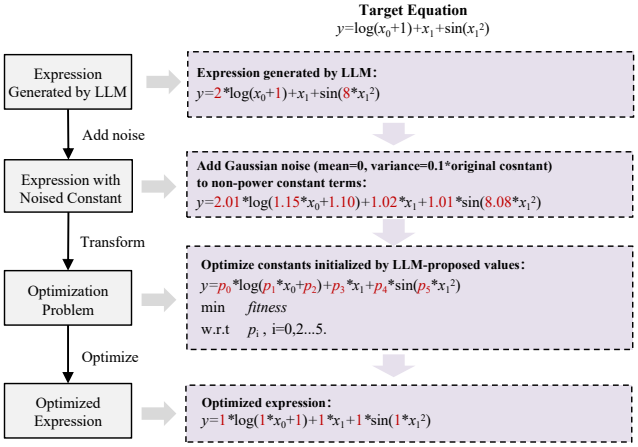


Figure 4: Framework of automatic constant identification and optimization. Non-exponential constants are extracted from equations, perturbed with noise, and optimized to improve accuracy.

function uses a sigmoid centered at $K_0 = 5$ with $\alpha = 0.5$:

$$\text{scale}(K) = 30 \cdot \big(1 + 0.5 \cdot \sigma(0.5(K-5))\big), \quad \sigma(z) = \frac{1}{1+e^{-z}}, \tag{1}$$

which ranges over $[30, 45]$ (a $1.5\times$ window). On data with no detected coupling ($K = 0$), scale $\approx$ 31.1, essentially the static value. On strongly coupled data ($K \geq 8$), scale ≈ 42.3, a 9.2% softer penalty that admits the longer coupled formulas.

The data complexity probe K is computed as follows. Given $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$, we (1) augment $\mathbf{X}$ with sin and cos of each column (so that trig-coupled pairs such as $a\cos b$ are visible to the polynomial test), and (2) for each augmented-feature pair (i, j) fit a 2-D degree-3 polynomial $A_{ij}\,[\xi_i, \xi_j, \xi_i\xi_j, \xi_i^2, \xi_j^2, \xi_i^2\xi_j, \xi_i\xi_j^2, 1]^\top$ via least squares, and compare its residual variance to the marginal 1-D cubic residual variances. A pair is marked coupled if

$$\text{Var}\Big[\mathbf{y} - A_{ij}\hat{\beta}\Big] < 0.5 \cdot \min(\text{Var}_{1\text{D},i}, \text{Var}_{1\text{D},j})\,, \tag{2}$$

and K is the number of such pairs. Pure numpy, no MLP dependency, no new external libraries; the probe runs once before the evolutionary loop starts.

The magnitude of the effect is small. At a typical 5-term formula (complexity $= 10$), the parsimony factor $\exp(10/\text{scale}(K))$ is 1.379 for $K = 0$, 1.357 for $K = 2$, 1.306 for $K = 5$, 1.267 for $K = 8$, and saturates at 1.249 for $K \geq 20$—a 10.5% relative softening at the extreme. This is a small perturbation of the fitness landscape: it is comparable to, and on most problems smaller than, the seed-to-seed variation of the search itself. We therefore do not attribute any headline result to this component, and we do not claim a controlled multi-seed effect for it.

Five files are touched (`code/utils/data_complexity.py` new, 80 lines; one field each in `utils/config.py`, `evolution/evolution_loop.py`, and `evolution/ec_operators.py`; one line in `evolution/evaluate.py` and one call in `experiments/runners/run_main.py`). The flag `-no-complexity-penalty` on `run_main.py` disables the parsimony term entirely.

B.4 Calibration of the Verification Thresholds

The verification criteria of §2.4 share one form: a claim is scored by a statistic that must clear a fixed threshold, and the thresholds were calibrated once on offline data and then frozen before any benchmark run. The calibration data is disjoint from every evaluation benchmark: five rounds of synthetic blind-figure batteries, generated in-house with known ground truth (genuine poles of varying width, no-pole controls, partially occupied panels), in which the MLLM reads the renderings without ever seeing the generating formula and each candidate is scored by its criterion. Thresholds were set so that the batteries yield zero false confirmations while genuine structure passes, and then frozen; no benchmark problem, benchmark equation, or benchmark result was used at any point. Two examples illustrate the calibration. For a composite coordinate u, the collapse R^2 bins the samples by quantiles of u and measures the variance explained by the bin means; a true coordinate reaches the discrete ceiling of this statistic, while coordinates that merely correlate with y fall measurably below it, and the threshold sits in the gap. For a pole, the MLLM-written zero set must place the extreme samples near the predicted crossings with a location accuracy that random or spurious zero sets do not achieve. Table 9 lists the exact statistic and confirm line of every criterion.

The identification loop tries the eight channels in a fixed order on the same residual—pole, zero set, composite coordinate, single-variable curve, spreading envelope, magnitude structure, multiplicative factor, parallel bands—stopping at the first confirmed channel, and stops altogether when no channel confirms or a factor repeats. The default depth is three layers, extendable by one at the budget boundary, with a hard cap of six. Rendering uses dpi $= 110$ for all channels; the highlighted bands are the top (pole) or bottom (zero set) 2% of $|y|$, while the stricter 1% masks enter the criteria. Sampling protocol: each scan and proposal question is issued twice independently, the shape-reading question three times, and escalated candidates three times each; candidate pools are merged and selection is entirely numerical. The first extraction pass uses seed 0 and the second, independent pass seed 1 (seeds merged and deduplicated, at most 24 kept); the training subsample and validation split use seed 0 and the evolution loop seeds its generator with 2024. No temperature or token limit is set on any model call; calls are paced at most one per 5 s, time out at 120 s, and retry up to 5 times with exponential backoff.

Table 9: The numerical criteria behind Table 1: statistic, confirm line, and guards for each feature type, with the stripping acceptance gates. All values were frozen on offline blind-figure batteries before any benchmark run (§B.4).

Feature type	Statistic	Confirm line and guards
Pole	Mann–Whitney AUC of $-\lvert g\rvert$ separating the top-1% $\lvert y\rvert$ samples; $\text{cross} = \min \lvert g\rvert / \text{med}\lvert g\rvert$	AUC ≥ 0.95 *and* cross < 0.05; phantom-pole guard: log-log slope of y vs. $\lvert g\rvert$ below -0.35 (or below -0.25 with contrast ≥ 2); AUC in $[0.95, 0.97)$ triggers a log-log recheck
Zero set	same AUC form on the bottom-1% $\lvert y\rvert$ samples	AUC ≥ 0.95 *and* cross < 0.05; vanish guard: log-log slope above 0.5
Composite coordinate	collapse R^2 of $\text{rank}(y)$ on u, 20 quantile bins	$R^2 \geq 0.99$ (the 20-bin ceiling is ≈ 0.9975); $\geq 95\%$ finite rows, ≥ 6 distinct bin edges
Single-variable curve	collapse R^2 on the raw variable, 20 bins	$R^2 \geq 0.99$
Spreading envelope	per-bin robust scale $s = 1.4826\,\text{MAD}$, 20 bins	$\text{std}(\log s) \geq 2.0$, adjacency correlation ≥ 0.8, normalized-residual spread $\leq$ half of $\text{std}(\log s)$, pointwise collapse < 0.99
Magnitude structure	collapse R^2 of $\lvert y\rvert$ vs. of signed y	$\lvert y\rvert$ collapse ≥ 0.7 *and* at least 0.3 above the signed collapse
Multiplicative factor	collapse R^2 of the quotient y/x_i on u	quotient $R^2 \geq 0.99$ *and* at least 0.3 above the plain-y collapse; $\geq 90\%$ finite quotient rows
Parallel bands	no separate confirm line	the stripping gate (below) is the only judge

Stripping acceptance gates (applied to every layer before the residual is replaced):
multiplicative layer: residual dynamic range below $0.5\times$ the previous residual's (active when the previous range exceeds 2.0) *and* log-std below $0.8\times$; additive layer: residual variance below $0.7\times$; at least 70% finite residual rows; an underflow guard rejects numerically annihilated residuals. Layer constants are re-fitted by Nelder–Mead (8 restarts, 20 for trigonometric forms; `xatol` 10^{-4}, `fatol` 10^{-6}; at most 8 constants; a $\log_{10}$ drift above 2 aborts the layer).

C Related Work

C.1 Symbolic Regression Methods

Symbolic regression, usually regarded as a combinatorial optimization problem Makke & Chawla (2024); Udrescu & Tegmark (2020), is traditionally solved using generative methods or heuristic search-based methods. Generative methods Biggio et al. (2021); Valipour et al. (2021); Li et al. (2022); Kamienny et al. (2022); Shojaee et al. (2023) train generative models using large-scale randomly generated equation-data pairs to predict the corresponding equations for given data. Heuristic search-based methods apply paradigms including genetic programming de Franca & Aldeia (2021); Augusto & Barbosa (2000); Virgolin et al. (2019); Koza (1994); Willis et al. (1997); Cranmer (2023); Arnaldo et al. (2014), Monte Carlo Tree Search Sun et al. (2023), and reinforcement learning Petersen et al. (2019); Mundhenk et al. (2021); Tenachi et al. (2023) to search for expressions using complexity and precision as fitness scores. Unified frameworks such as uDSR Landajuela et al. (2022) combine a pretrained transformer, a risk-seeking RL policy, genetic programming, and MCTS under one interface with shared skeleton and constant handling—the direct precedent for wrapping several search engines around learned priors, onto which the present work grafts a verified visual front-end.

C.2 LLM-based Symbolic Regression

Recently, the rapid development of Large Language Models (LLMs) has introduced a new paradigm for symbolic regression, leveraging their embedded scientific knowledge and powerful code generation capabilities. Several representative methods have been proposed. LLM-SR Shojaee et al. (2024) treats equations as Python programs, using LLMs to generate initial equation skeletons refined through evolutionary loops. FunSearch Romera-Paredes et al. (2024) demonstrates the same LLM-plus-evaluator evolutionary pattern on mathematical and algorithmic problems. LaSR Grayeli et al. (2024) learns and evolves a library of abstract symbolic concepts to guide both LLM-based and genetic programming search. SGA Ma et al. (2024) implements a bi-level optimization framework where LLMs propose equation hypotheses while physical simulations provide validation. ICSR Merler et al. (2024) frames symbolic regression as in-context learning, using previously tested expressions as demonstrations. LLM-Feynman Song et al. (2025) combines LLMs with physical unit constraints, and DrSR Wang et al. (2025a) introduces a dual-reasoning mechanism that continuously refines data understanding and accumulates experience from past search attempts.

A further distinction lies in how prior knowledge originates and is treated. In most LLM-based SR, guidance enters as text—expert domain knowledge, variable semantics, problem descriptions—

but the loop is closing in several ways that should not be conflated. LaSR Grayeli et al. (2024) abstracts concepts *online* from successful search outputs and re-injects them into subsequent search; ICSR Merler et al. (2024) updates its demonstrations during the run from previously tested expressions together with their fitness scores; SGA Ma et al. (2024) validates every generated hypothesis against physical simulation. Two concerns nonetheless remain open for all of them: a textual prior may steer the search toward equations the model already knows, and no *visually read* structural claim is admitted feature by feature under a numerical criterion. ViGa-SR addresses both through its input channel: variables are anonymized before any model call, so no *named* equation can be recalled; priors originate exclusively in renderings of the data; and every identified structure—whether genuinely read from the images or matching something seen in training—must pass its numerical criterion on the present data before entering the search.

C.3 Knowledge-Informed Traditional Symbolic Regression

A prominent direction in symbolic regression research incorporates external expert knowledge into evolutionary search to enhance physical plausibility and generalization Lu et al. (2016); Keren et al. (2023). Prior work encodes mathematical properties—such as monotonicity, convexity, and structural rationality of functional forms—as well as physical laws including dimensional consistency and conservation principles Udrescu & Tegmark (2020); Gupta et al. (2016); Kronberger et al. (2022); Błądek & Krawiec (2022); Yu et al. (2026); Kubalík et al. (2021); Tenachi et al. (2023). Our work advances this line of inquiry by demonstrating how such priors can be autonomously extracted from data visualizations rather than manually specified. AI Feynman 2.0 Udrescu et al. (2020) deserves a specific distinction: its generalized-symmetry module also detects dependence through composite coordinates, and our collapse criterion is related in spirit. The roles differ, however—there, detected symmetries recursively decompose the problem and the module itself searches over a fixed library of symmetry transformations, whereas in ViGa-SR the criterion never proposes anything: it only judges composite coordinates that the MLLM has written down from renderings, and the search over what to try lives in the visual reading, not in a transformation library.

Shape constraints are typically enforced via feasibility checks during evolution. Kronberger et al. Kronberger et al. (2022) apply interval arithmetic to verify monotonicity and convexity across the input domain, discarding infeasible candidates or maintaining separate feasible populations. Blkadek and Krawiec Błądek & Krawiec (2022) employ Satisfiability Modulo Theories (SMT) solvers for formal constraint verification, using counterexamples from verification failures to guide subsequent search. Haider et al. Haider et al. (2023) extend this paradigm by treating shape constraints as additional objectives within a multi-objective GP framework, enabling soft constraint handling through Pareto optimization.

Physical law integration follows similar verification principles. Kubalik et al. Kubalík et al. (2021) adopt multi-objective optimization to balance data fidelity against compliance with conservation laws. Tenachi et al. Tenachi et al. (2023) embed dimensional consistency as grammatical constraints, ensuring that all generated expressions maintain physically valid units by construction—eliminating invalid solutions while simultaneously reducing the search space. Beyond these automated approaches, the SciMED framework Keren et al. (2023) provides multiple scientist-in-the-loop interfaces for domain knowledge injection, including feature grouping, custom loss functions, feature weighting, and explicit equation structure constraints.

The principle that candidate structure should be admitted only after verification has a long history outside symbolic regression. Counterexample-guided inductive synthesis admits a sketched program only when verification produces no refuting counterexample Solar-Lezama et al. (2006); dimensional priors inside genetic programming predate their recent revival by decades Keijzer & Babovic (1999), descending from Buckingham's Pi theorem Buckingham (1914); and equation-learning networks hard-wire operator units into the architecture for extrapolation Martius & Lampert (2016); Sahoo et al. (2018). Cranmer et al. Cranmer et al. (2020) instantiate the reverse division of labor from ours—a trained neural model's internal features *propose* structure that symbolic search then extracts—whereas in ViGa-SR the proposal channel is visual and the numerics only judge.

C.4 Multimodal Large Language Model for Symbolic Regression

Multimodal Large Language Models (MLLMs) Huang et al. (2024) extend the capabilities of LLMs from text to multiple modalities including images Li et al. (2023); Zhang et al. (2023b); Wu et al. (2023); Wang et al. (2024); Ye et al. (2023), audio Chen et al. (2023); Liu et al. (2024b), and video Li et al. (2025a); Maaz et al. (2024); Zhang et al. (2023a), and have demonstrated significant potential in scientific discovery and engineering applications, including aerodynamic shape optimization Wong et al. (2024), health monitoring Wang et al. (2025b), and fluid dynamics Narin (2024).

In symbolic regression, several MLLM-based approaches have been proposed. LLM-Lex Harvey et al. (2025) uses an MLLM to design equation skeletons from plots and a KAN (Kolmogorov–Arnold Network) for multivariate decomposition; a collection of such ansätze forms the population of a genetic algorithm whose free parameters are fitted by standard numerical optimizers, so its visual proposals already evolve under numerical fitting, though without per-feature admission criteria or stripping-based decomposition. MMSR Li et al. (2025c) represents equations using both numeric and symbolic modalities with contrastive learning strategies. ViSymRe Li et al. (2024) was introduced with a meta-learning framework integrating visual, symbolic, and numeric modalities; the current version is a supervised Transformer trained with multi-view random slicing, designed for dataset-only deployment—at inference the model consumes no images. ChatSR Li et al. (2025d) fine-tunes an MLLM for conversational symbolic regression: scientific observations are encoded as a modality by a SetTransformer over tabular data—plots appear nowhere in its pipeline—and users specify prior knowledge in natural language; it demonstrates zero-shot generalization to unseen properties but requires pre-training on millions of synthetic expressions and question–answer pairs. Li et al.'s zero-shot VER system Li et al. (2025b) discovers physical coordinates and governing equations from high-dimensional *video* data: visual locating toolkits propose coordinate candidates, hypotheses are optimized and scored by R^2, an additional confirmation call guards against misidentification, and an experience memory carries knowledge across iterations. VIPER-R1 Liu et al. (2025) trains a VLM-centric model on visual perception, trajectory data, and symbolic reasoning; after producing an ansatz, the model invokes an external symbolic-regression tool for Symbolic Residual Realignment (SR^2), a numerically driven residual-correction pass that does not re-read the visuals.

Mapping this space along four axes—where structure proposals originate, what the numerics do, how residuals are revisited, and whether the data is re-read visually—leaves one intersection unoccupied. ChatSR's priors are user-stated text over tabular encodings; LaSR's concepts originate from search outputs; ICSR's demonstrations are whole-expression precedents gated by fitness rather than per-feature criteria; SGA's validation is simulation rather than visual reading; VER assesses hypotheses numerically but discovers coordinates in video rather than tabular renderings; VIPER-R1's residual realignment is post-ansatz and non-visual; LLM-Lex evolves plot-proposed skeletons without stripping or per-feature verification. ViGa-SR occupies the remaining cell: priors originate in renderings of the present data, are admitted feature by feature only after a numerical criterion passes, are stripped so the simplified data is rendered and read again, and are re-acquired visually when the search stalls (§2.7). We treat MLLM-derived mathematical features as explicit, interpretable priors whose accuracy can be independently validated and whose influence on search behavior is measurable through ablation.

C.5 Sparse Identification of Governing Equations

For dynamical systems, sparse identification discovers governing equations by linear-in-coefficients regression over a fixed candidate library (SINDy Brunton et al. (2016)), extended to jointly learned coordinates with autoencoders Champion et al. (2019) and to rational, denominator-containing implicit forms Kaheman et al. (2020). This is the canonical equation-discovery line for trajectory data, and it shares our goal of recovering interpretable governing structure. We do not include it as a baseline on the oscillator tasks of §3 because the task formulation differs on both input and output: following Shojaee et al. (2024), the case study is a static regression from state–time samples $(t, x, \dot{x})$ to the acceleration, seeking the single second-state equation, whereas sparse identification consumes time-ordered state trajectories, estimates the derivatives itself, and identifies the full vector field. Its candidate library is also a human prior: covering the oscillators' $\sin / \cos / \exp$ terms and couplings such as $x \exp(\gamma x)$ already requires expert library design, the exact input class our comparison withholds. Where trajectory data and a well-matched library are available, SINDy-family methods are strong and appropriate baselines; the denominator structure our health monitor targets is

precisely the case SINDy-PI addresses within its formulation, and a comparison under matched input assumptions is valuable future work.

D Robustness Experiments

This appendix presents complete robustness analysis of the ViGa-SR framework, including both interpolation-level resilience and end-to-end symbolic regression performance under challenging conditions. All experiments used fixed random seeds for full reproducibility.

D.1 Interpolation Method Selection and Rationale

Four interpolation methods were systematically evaluated across diverse data conditions: RBF Spline, Gaussian Process (GP), Vanilla MLP, and MLP with Gradient Penalty (MLP-GradP). Each method exhibits distinct strengths:

- **RBF Spline**: Achieves lowest overall MSE on clean, uniformly-sampled low-dimensional data, with fast training time
- **Gaussian Process**: Provides well-calibrated uncertainty estimates, making it suitable for small datasets where confidence quantification is critical
- **Vanilla MLP**: Offers fastest inference for high-dimensional problems with dense sampling
- **MLP-GradP**: Emerges as the *most stable method across all conditions*, with consistently smooth gradients essential for reliable visual feature extraction and superior resilience to impulse noise (two orders of magnitude lower MSE than kernel-based approaches)

After extensive testing, MLP-GradP was selected as the default interpolation method for the ViGa-SR framework due to its unmatched robustness across noise levels, sampling distributions, and sparsity regimes. However, the framework is designed to support any interpolation method—users may select alternatives based on their specific data characteristics and requirements.

D.2 Baseline Method Comparison

We first compare the four interpolation methods across three complementary metrics: overall reconstruction quality (MSE), goodness of fit (R^2), and gradient smoothness (variance of the interpolated derivative). As shown in Figure 5, RBF Spline achieves the lowest overall interpolation error on clean, uniformly sampled data, while MLP-GradP provides the smoothest gradients—a property critical for reliable visual feature extraction by the MLLM.

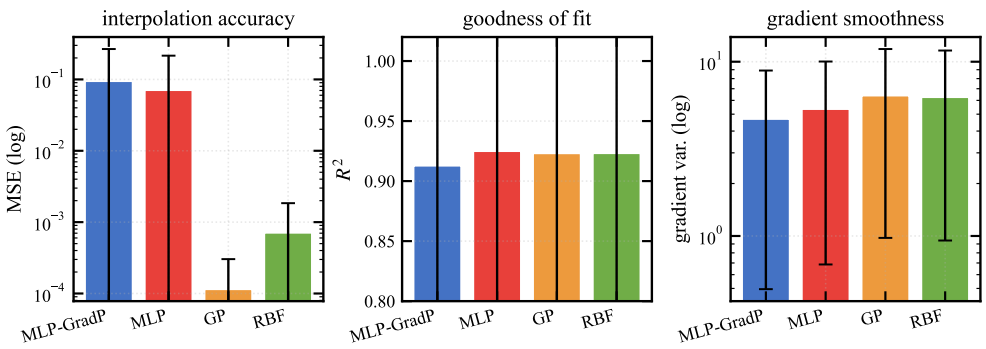


Figure 5: Baseline interpolation method comparison across three metrics: (a) MSE on log scale, (b) R^2 goodness of fit, and (c) gradient variance (smoothness). Results are averaged across 7 test functions and 3 random initializations. RBF Spline achieves the lowest overall MSE, while MLP-GradP provides the smoothest gradients for reliable visual feature extraction.

D.3 Sampling Distribution Robustness

Scientific data rarely follows ideal uniform sampling. We evaluate robustness across three input distributions: uniform, log-scaled, and Gaussian. As reported in Figure 6, MLP-based methods maintain more consistent performance than kernel-based approaches under distribution shift, with MLP-GradP exhibiting the smallest relative degradation.

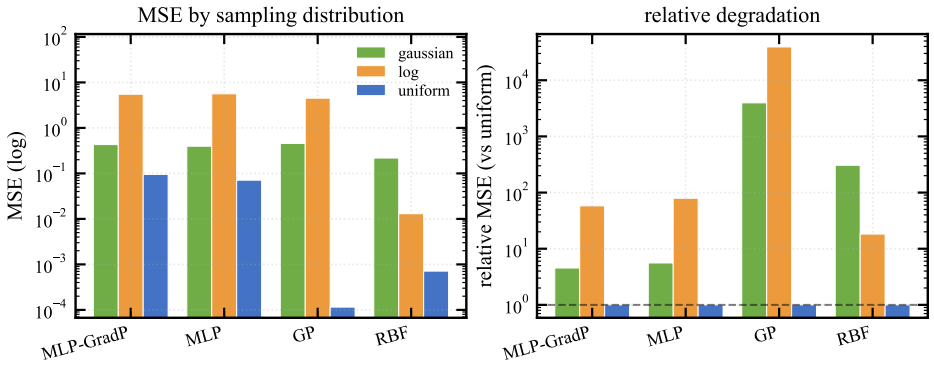


Figure 6: Distribution shift robustness evaluation. (a) MSE under uniform, log-scaled, and Gaussian sampling distributions. (b) Relative MSE normalized to uniform distribution performance, showing degradation under distribution shift. MLP-based methods maintain more consistent performance than kernel-based approaches.

D.4 Noise Robustness

Real-world experimental data is invariably corrupted by measurement noise. We test robustness across Gaussian white noise, colored AR(1) noise, and impulse noise. Results in Figure 7 show that MLP-based methods, particularly MLP-GradP, exhibit substantially better resilience to impulse noise (two orders of magnitude lower MSE) compared to kernel-based approaches.

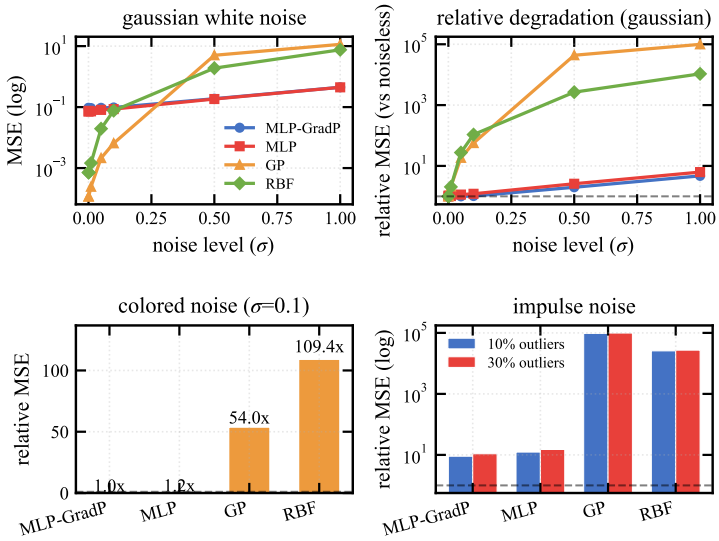


Figure 7: Complete noise robustness analysis across all noise types. (a) Gaussian white noise intensity sweep from $\sigma = 0$ to 1.0. (b) Relative performance degradation normalized to noiseless baseline. (c) Colored AR(1) noise degradation relative to white noise at matched intensity. (d) Impulse noise relative performance at 10% and 30% contamination rates. MLP-based methods demonstrate substantially better resilience to impulse noise (two orders of magnitude lower MSE) compared to kernel-based approaches.

D.5 Sample Sparsity and Dimension Robustness

High-dimensional scientific problems often suffer from limited sampling density. We characterize the dimension-sample size tradeoff for 2D through 4D problems in Figure 8. Even in the sparse 100–300 sample range, MLP-GradP maintains interpolation quality within 1.9–3.8× of the 1000-sample baseline, with graceful degradation as sparsity increases.

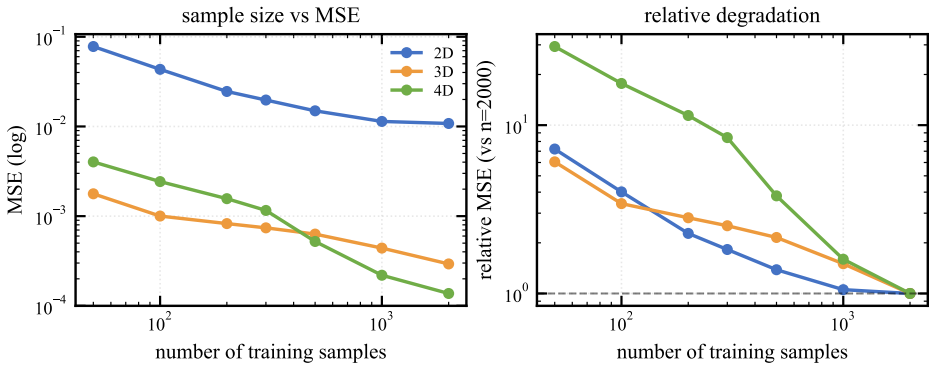


Figure 8: Dimension vs. sample size tradeoff analysis. (a) Full MSE vs. sample size curves for 2D, 3D, and 4D problems on log-log scale. The 100–300 sample range shows performance within 1.9–3.8× degradation relative to the 1000-sample baseline. (b) Relative MSE normalized to the 2000-sample reference, showing consistent convergence rates across dimensions.

D.6 Gradient Penalty Ablation Study

The gradient penalty regularization coefficient λ controls the fundamental tradeoff between interpolation fidelity and gradient smoothness. The ablation study in Figure 9 characterizes this tradeoff: larger λ monotonically improves gradient smoothness at the cost of slightly increased MSE. The selected operating point $\lambda = 0.002$ balances these competing objectives.

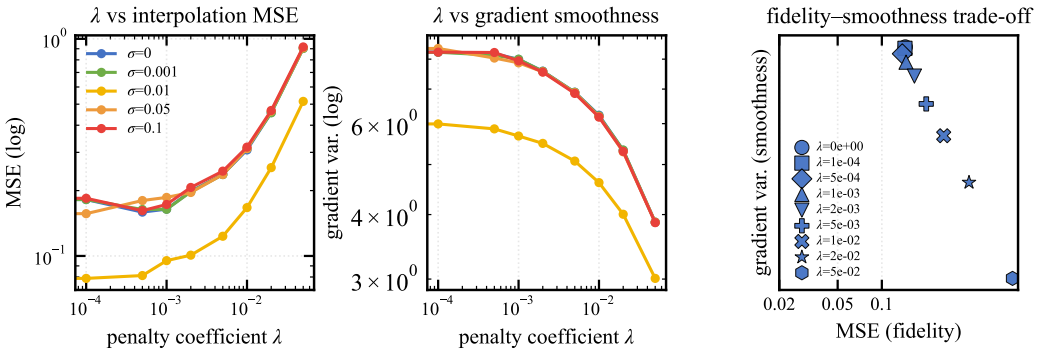


Figure 9: Gradient penalty ablation study characterizing the fidelity-smoothness tradeoff. (a) MSE as a function of regularization strength λ across noise levels. (b) Gradient variance vs. λ, showing monotonic improvement in smoothness with larger penalty. (c) The fidelity-smoothness Pareto frontier: each point represents a different λ value. The operating point $\lambda = 0.002$ used throughout the paper represents a balanced compromise between interpolation accuracy and gradient stability.

D.7 End-to-End Noise Robustness on Feynman Dataset

Beyond interpolation-level validation, we evaluated the complete ViGa-SR pipeline on the Feynman dataset under increasing noise conditions. The results were obtained by injecting Gaussian noise with standard-deviation levels of 0.01 and 0.001 into the target variable, following the same settings as La Cava et al. (2021). The noise-free baseline results are presented in the main text. Under target

noise $\sigma = 0.001$, ViGa-SR retains a solution rate of 43.1% and an accuracy solution rate of 0.83; at $\sigma = 0.01$ these stand at 31.2% and 0.31, with the degradation concentrated in the strict accuracy metric. This robustness is mainly attributed to the gradient penalty during MLP interpolation and the subsequent curve smoothing, which allow the extraction stage to read the core data features even in the presence of noise.

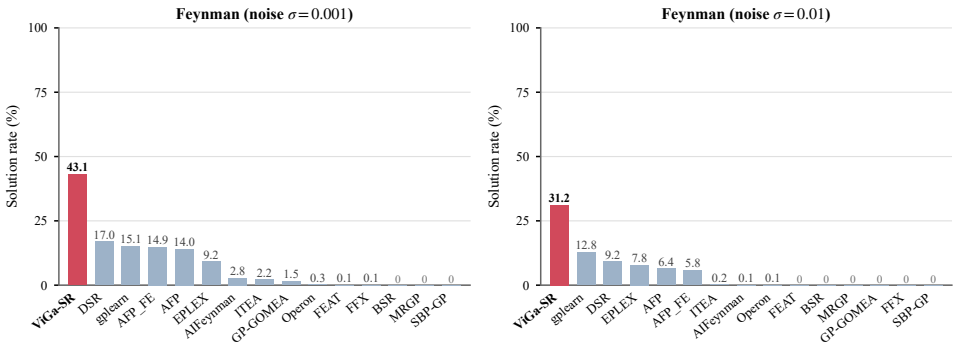


Figure 10: Solution rates under noisy conditions on the Feynman dataset. Noise-free results are provided in the main text.

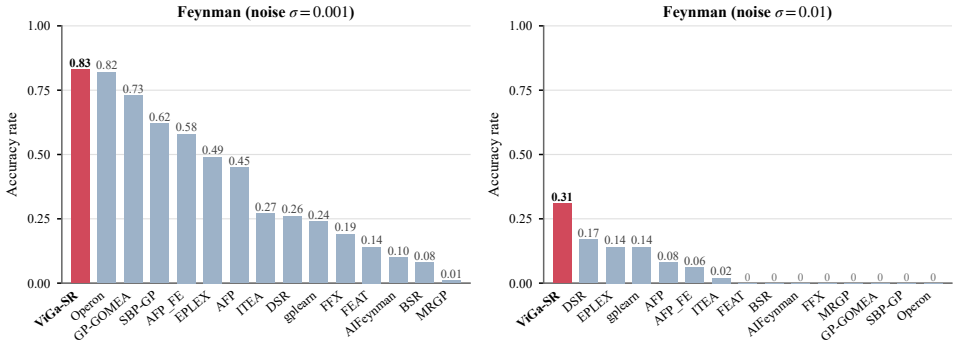
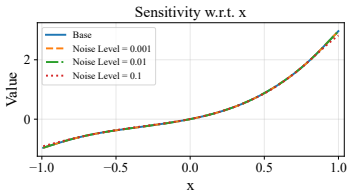
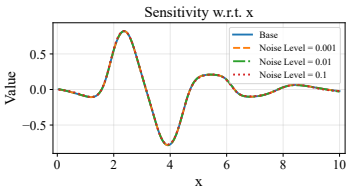


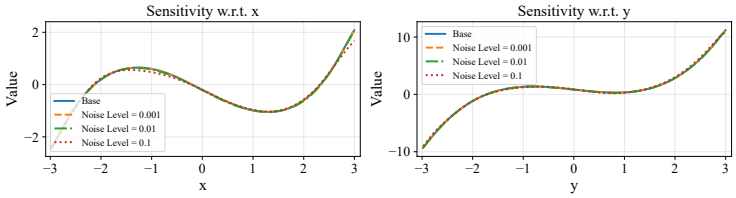
Figure 11: Accuracy solution rates ($R^2 > 0.999$) under noisy conditions on the Feynman dataset. Noise-free results are provided in the main text.



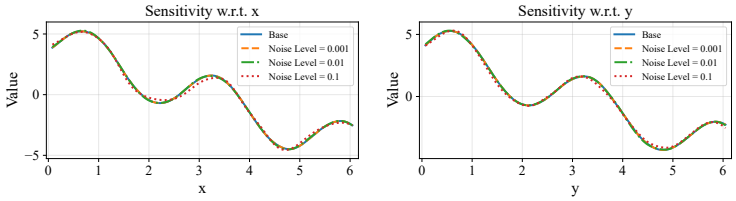
Nguyen 1: $x^3 + x^2 + x$



Keijzer 4: $x^3\, e^{-x} \cos(x) \sin(x) \left(\sin^2(x) \cos(x) - 1\right)$



Keijzer 15: $\dfrac{x^3}{5} + \dfrac{y^3}{2} - y - x$



Vladislavleva 7: $(x-3)(y-3) + 2\sin\big((x-4)(y-4)\big)$

Figure 12: Sensitivity plots produced by the gradient-penalty MLP under increasing noise levels. **Solid black**: noise-free data; **gray dash-dot**: $\sigma = 0.001$; **gray dash**: $\sigma = 0.01$; **light-gray dot**: $\sigma = 0.1$. Despite large noise, the smoothed curves preserve monotonicity, extrema and symmetry, enabling reliable MLLM feature extraction.

D.8 Extraction-Stage Robustness to Noise and Sparsity

The interpolation studies above measure the surrogate in isolation; here we measure the extraction stage end to end. Three problems with reliable verified structure (poles on `II.11.27` and `II.27.18`; a composite coordinate on `III.10.19`) are subjected to two degradation grids: additive Gaussian noise at $\sigma/\mathrm{std}(y) \in \{0, 0.01, 0.05, 0.1\}$ on the full training set, and uniform subsampling to 8000 and 2000 points (10% and 2.5% of the full set) under no noise. On each degraded training set the full extraction stage is run once, and the resulting seed expressions are scored on the clean held-out test split (Figure 13). Two observations matter. First, there is no monotone degradation: at the strongest noise level all three problems still yield verified structure with seed test R^2 between 0.83 and 0.99, and at 2000 samples all three still yield verified structure, two of them with $R^2 > 0.78$. The MLP surrogate absorbs the degradation exactly as designed, and the MLLM reads the smoothed function rather than the raw samples. Second, the dominant variability is the stochasticity of the visual reading, not the data condition: empty attempts occur at every noise level including zero, and successes occur at every noise level including the strongest. This is the same single-attempt variance characterized in Appendix J, and it is what the production configuration's multiple independent extraction passes are designed to absorb.

E Details of Ablation Study

In the main text, we demonstrated the impact of different components in our framework on the Feynman dataset. Figure 14 summarizes the aggregate ablation: removing adaptive guidance reduces solution rates modestly, while removing all prior knowledge causes a pronounced collapse.

Here, we conduct a detailed analysis based on equation types by categorizing them into the classes shown in Table 10 and Table 11. The classification of the Feynman dataset used here is not officially provided; rather, it is a comprehensive categorization based on the nesting depth of functions and

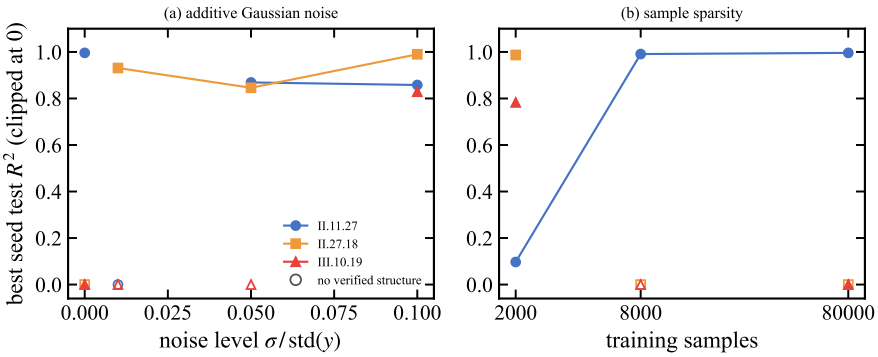
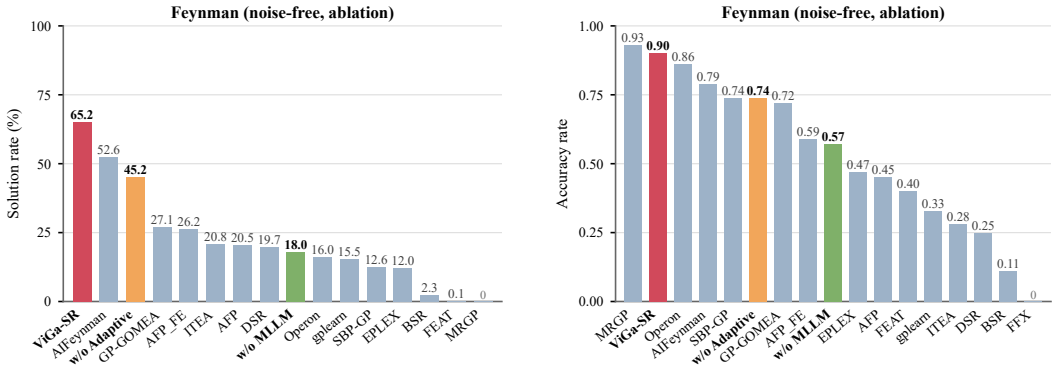


Figure 13: Extraction-stage robustness. Each point is one complete run of the extraction stage on a degraded training set, scored by the best seed's R^2 on the clean held-out test split (negative values clipped to 0). (a) Additive Gaussian noise. (b) Uniform subsampling. Open markers at 0 denote attempts where no feature passed verification. Neither noise nor sparsity produces monotone degradation; empty and successful attempts occur at every condition, consistent with single-attempt reading variance being the dominant error source.



(a) Solution rates

(b) Accuracy rates ($R^2 > 0.999$)

Figure 14: Ablation study on the Feynman dataset: removing adaptive guidance reduces solution rates modestly; removing all prior knowledge causes a pronounced collapse.

the performance of the algorithm presented in this paper, aimed at investigating how our algorithm behaves across different types of ground-truth equations. The categorization covers the same 100 equations as the headline Feynman evaluation of §3 ($53 + 35 + 12$), so the class-wise rates are a size-weighted breakdown of the headline figure; class-wise rates are rounded to integers, so weighted aggregates can differ from the headline figures by a few tenths of a point.

Table 10: Solution rates of different equation categories in the Feynman dataset in ablation studies.

CAT.	EXAMPLES	NUMBER	FEATURES	MULTIMODAL FEATURES	FULL ALG.	W/O ADAP.	W/O MLLM
A	I. 34.8: $\frac{qvB}{p}$ II. 24.16: $\epsilon c E_f^2$ II. 15.4: $\frac{p_d E_f}{\cos\theta}$ I. 11.19: $x_1 y_1 + x_2 y_2 + x_3 y_3$ ...	53	NO NESTING, SIMPLE	OBVIOUS	94%	66%	32%
B	I. 10.7: $\frac{m_0}{\sqrt{1-v^2/c^2}}$ I. 12.11: $q(E_f + Bv\sin\theta)$ II. 35.21: $n_\rho m \tanh(\frac{mB}{k_b T})$ III. 8.54: $\sin^2(\frac{E_n t}{h/2\pi})$ ...	35	SHALLOW NESTING, SIMPLE	PARTIALLY OBVIOUS	42%	29%	4%
C	II. 36.38: $\frac{mH}{k_b T} + \frac{m\alpha M}{\epsilon c^2 k_b T}$ III. 9.52: $\frac{p_d E_f t \sin^2\phi}{h/2\pi\cdot\phi^2}$ WHERE $\phi = \frac{(\omega-\omega_0)t}{2}$ ...	12	NESTING, HIGH-DIM (>6), COMPLEX	NOT OBVIOUS	6%	0%	0%

Table 11: Accuracy solution rates of different categories of equations in the Feynman dataset under ablation studies.

CAT.	EXAMPLES	NUMBER	FEATURES	MULTIMODAL FEATURES	FULL ALG.	W/O ADAP.	W/O MLLM
A	I. 34.8: $\frac{qvB}{p}$ II. 24.16: $\epsilon c E_f^2$ II. 15.4: $\frac{p_d E_f}{\cos\theta}$ I. 11.19: $x_1 y_1 + x_2 y_2 + x_3 y_3$ ...	53	NO NESTING, SIMPLE	OBVIOUS	99%	88%	75%
B	I. 10.7: $\frac{m_0}{\sqrt{1-v^2/c^2}}$ I. 12.11: $q(E_f + Bv\sin\theta)$ II. 35.21: $n_\rho m \tanh(\frac{mB}{k_b T})$ III. 8.54: $\sin^2(\frac{E_n t}{h/2\pi})$ ...	35	SHALLOW NESTING, SIMPLE	PARTIALLY OBVIOUS	90%	69%	47%
C	II. 36.38: $\frac{mH}{k_b T} + \frac{m\alpha M}{\epsilon c^2 k_b T}$ III. 9.52: $\frac{p_d E_f t \sin^2\phi}{h/2\pi\cdot\phi^2}$ WHERE $\phi = \frac{(\omega-\omega_0)t}{2}$ ...	12	NESTING, HIGH-DIM (>6), COMPLEX	NOT OBVIOUS	55%	30%	10%

From the results presented in Table 10, it is evident that when both MLLM guidance and adaptive guidance are activated, our algorithm performs well on Class A equations, essentially reconstructing

the true equations. However, performance declines for Class B equations, which we attribute to the insufficient prominence of multimodal features in certain equations due to nesting. For Class C equations, it is nearly impossible to obtain the correct equations end to end; the extraction-stage audit in Appendix J shows what the extraction stage alone nevertheless recovers on such problems (e.g., a verified pole family on `III.9.52` whose seed reaches test $R^2 = 0.868$), which is the channel through which this class narrows.

When adaptive MLLM guidance is removed—that is, the search runs on the initially extracted features alone, with no stagnation-triggered re-reading of the data during evolution—the solution rate for Class A equations drops sharply. This indicates that the MLLM's initial reading is not always sufficient; without a fresh independent reading when the search stalls, uncorrected misreadings adversely affect the outcome. For Class B equations, the performance impact is more contained, which we attribute to the inherently less prominent multimodal features of Class B, where search performance is already limited and thus re-reading has less to recover. For Class C equations, recovery becomes infeasible either way, indicating that complex nesting renders multimodal features too obscure for the MLLM to identify the correct functional characteristics from the outset.

When the mathematical features provided by the MLLM are completely removed—that is, searching with the LLM without any prior knowledge—some equations in Class A can still be successfully reconstructed. We attribute this to the presence of overly simple equations in Class A, which the LLM can generate through random sampling. Class B equations are severely affected, as they are more complex and thus difficult to recover without prior knowledge. For Class C equations, relying solely on the LLM fails to reconstruct any equation. Results in Table 11 reveal similar trends, with the only difference being that our framework, leveraging bi-level search, can identify equations that are not identical to the original ones but exhibit high accuracy; such accuracy is often achieved at the expense of complexity, a trade-off discussed in §4.

F Performance of Different MLLMs in Mathematical Feature Extraction

The framework developed in this paper relies on an MLLM to perform mathematical feature extraction from curve plots; this section examines how sensitive that extraction is to the choice of MLLM. It presents the accuracy of different MLLMs in extracting mathematical features on the Feynman dataset, rated by human experts and categorized into three levels: completely correct, partially correct, and completely incorrect. This section follows the same division of the Feynman dataset as detailed in the ablation study ($n = 53/35/12$ problems per column); results are shown in Table 12. Two scope notes apply. First, the rated task is the free-form per-variable description prompt of Appendix G §G.6, which predates the deployed eight-type channel protocol; the table therefore evidences insensitivity of *description-quality* to the model choice, not per-channel confirmation rates of the deployed pipeline. Second, the deployed extraction model (GPT-4o) is not among the four rated models, so the table does not by itself establish robustness of the deployed configuration.

Table 12: Performance comparison of different MLLMs on different equation categories of the Feynman dataset.

MLLM	Correctness	No nesting (n=53)	Shallow nesting (n=35)	Complex nesting (n=12)
GPT-4o-mini	Completely Correct	95%	32%	0%
	Partially Correct	5%	63%	40%
	Completely Incorrect	0%	5%	60%
GPT-5	Completely Correct	97%	36%	0%
	Partially Correct	3%	60%	35%
	Completely Incorrect	0%	4%	65%
gemini-3 (pro-preview)	Completely Correct	96%	35%	0%
	Partially Correct	4%	63%	40%
	Completely Incorrect	0%	2%	60%
grok-4 (fast-reasoning)	Completely Correct	94%	30%	0%
	Partially Correct	6%	62%	37%
	Completely Incorrect	0%	8%	63%

Results demonstrate that, due to our adoption of the Chain-of-Thought (CoT) approach to gradually guide the MLLM in extracting information from plots, description accuracy varies little across the four tested models—whereas it is strongly correlated with the characteristics of the equations themselves. Within the scope noted above (description task, deployed model not rated), we read this as evidence that the framework is not brittle to the MLLM choice; a per-channel comparison of the deployed eight-type protocol across models, GPT-4o included, remains future work. For instance, for equations that do not possess strong multimodal features, such as I.13.12: $Gm_1m_2(1/r_2 - 1/r_1)$ and I.18.4: $(m_1r_1 + m_2r_2)/(m_1 + m_2)$, it is difficult to obtain effective information from their plots regardless of which MLLM is employed. Conversely, for equations with strong multimodal features, such as I.12.11: $q(E_f + Bv\sin\theta)$ and I.11.19: $x_1y_1 + x_2y_2 + x_3y_3$, even when the equations are high-dimensional and moderately complex, the MLLM can still reliably extract mathematical features and effectively guide equation design. The plot of an equation with strong multimodal features is shown in Figure 15.

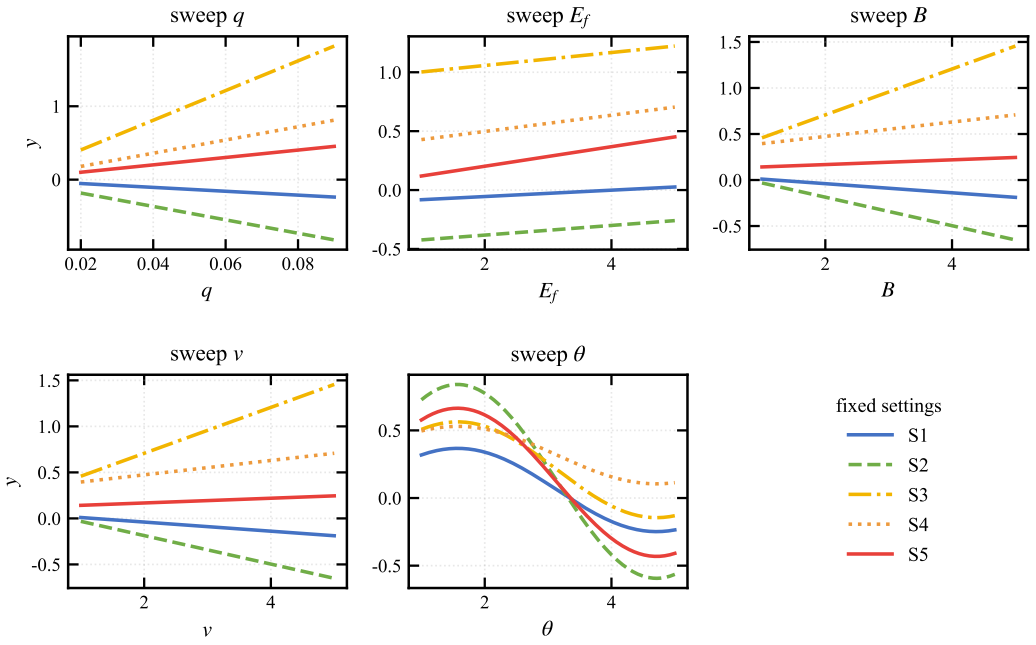


Figure 15: Sensitivity plot of the I.12.11 equation in the Feynman dataset (possessing strong multimodal features).

Furthermore, we observed that more powerful MLLMs tend to describe mathematical features in greater detail than the prompts require. Free-form descriptions of this kind carry a structural limitation: abstract characterizations such as *monotonic decreasing with an asymptotic decay* are difficult for a downstream LLM to convert into concrete equation-design choices, whether or not they are accurate. This observation is part of what motivated the deployed protocol, in which every claim is a concrete, coordinate-anchored statement that a numerical criterion judges before it can influence the search. Taking I.13.12 ($G * m_1 * m_2 * (1/r_2 - 1/r_1)$) as an example, a representative description generated by GPT-5 is presented below. Note that to prevent memorization-based shortcuts, variable names in the plots provided to the LLM were replaced with a, b, c, ... instead of their actual names, where a represents G, b represents m_1, c represents m_2, d represents r_2, and e represents r_1.

```
Insights from GPT-5 of equation I.13.12:

For independent variable G, m_1, m_2: this function is non-linear and
monotonic for the majority of curves (convex increasing or convex decreasing
rather than periodic), the function is likely to apply basic operators
including exponential and power (polynomial) to this variable, and the power
of this independent variable is likely to be odd, the relationship between
this variable and other variables could be multiplication, The confidence
level of the above description is 70%.

For independent variable r_2: this function is non-linear and monotonic
```

```
decreasing with an asymptotic decay (consistent with reciprocal or
exponential decay rather than periodic or linear), the function is likely to
apply basic operators including reciprocal (negative power) and exponential
decay to this variable, and the power of this independent variable is likely
to be odd, the relationship between this variable and other variables could
be division, The confidence level of the above description is 85%.

For independent variable r_1: this function is non-linear and monotonic with
saturating/concave-down behavior for most curves (rapid change at small
values then leveling off), the function is likely to apply basic operators
including exponential (saturation) and logarithmic transforms to this
variable, and the power of this independent variable is likely to be odd,
the relationship between this variable and other variables could be
multiplication, The confidence level of the above description is 75%.
```

F.1 Controlled MLLM Feature Extraction Evaluation

To rigorously validate MLLM performance independent of equation recovery, we conducted controlled experiments on synthetic functions where ground-truth mathematical features are known exactly. We evaluated feature extraction accuracy over 20 test functions from several families (polynomial, exponential, periodic, and hybrid) spanning 2–5 dimensions, crossed with three noise levels ($\sigma = 0, 0.01, 0.1$), three sample sizes ($n = 100, 500, 1000$), and three sampling distributions (uniform, log-scaled, Gaussian); each of the resulting 540 conditions is queried for five feature types (monotonicity, linearity, periodicity, exponential shape, curvature classification), yielding 2700 individual feature predictions. Scoring is categorical and programmatic: each prediction is compared against the exact ground-truth category under a fixed rubric (exact category match counts as correct; near-misses do not), and the identical rubric is applied to both the MLLM and the numerical baseline. The overall results are presented in Table 13 and Figure 16.

Two scope disclosures apply to this subsection. First, the five feature types evaluated here are generic shape categories, not the eight channel-specific claims of the deployed protocol (Table 1); the mapping is partial—curvature and monotonicity inform several channels, while pole localization and composite-coordinate claims have no counterpart here—so these numbers characterize visual shape reading in general, not per-channel confirmation rates of the deployed pipeline. Second, the comparison against numerical methods is not input-matched: the numerical baseline differentiates the raw (possibly noisy) data, whereas the MLLM reads an MLP-interpolated rendering of the same data, so part of the gap on noise-sensitive features reflects the interpolation smoothing rather than the visual reading itself. A matched-surface comparison (numerical differentiation on the same interpolated surface the MLLM sees) remains future work.

Table 13: Complete MLLM feature extraction accuracy by feature type across all conditions. The MLLM's advantage over numerical methods is concentrated in geometric shape features (curvature, monotonicity); for operator-family features (linear, periodic, exponential) accuracy is solid but not perfect. Parenthesised values are Wilson 95% CI half-widths.

Feature Type	Accuracy (%)	Number of Tests
Linearity	77.0 (±3.6)	540
Periodicity	77.6 (±3.5)	540
Exponential Shape	71.1 (±3.9)	540
Monotonicity Direction	68.7 (±3.9)	540
Curvature Classification	71.3 (±3.9)	540
Overall	**73.1** (±1.7)	**2700**

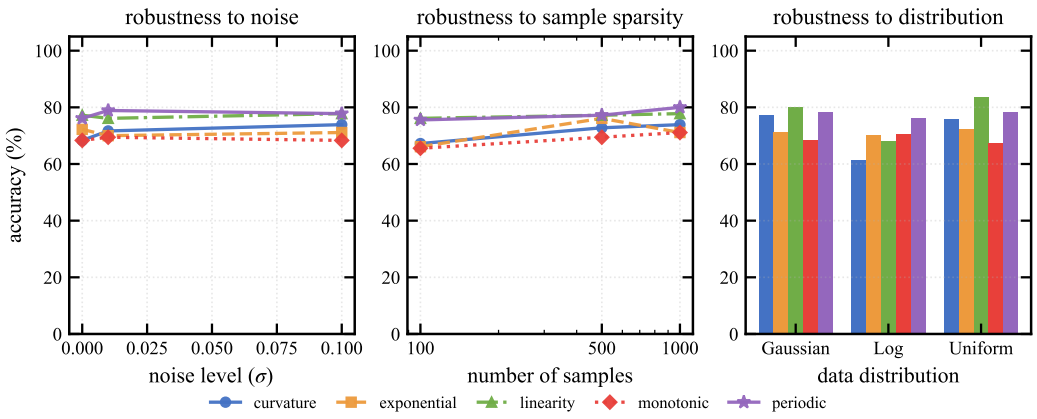


Figure 16: Complete MLLM feature extraction performance breakdown across all experimental conditions. Geometric shape features (curvature, monotonicity) are where the MLLM most clearly outperforms numerical differentiation, while operator-family features remain reliably detected across conditions.

Under strict category-level scoring, no single feature reaches perfect accuracy; the MLLM's distinctive strength is instead robustness on shape features. In particular, curvature classification—where finite-difference second derivatives are highly noise-sensitive—reaches 71.3%, far above the numerical baseline (25.4%), and monotonicity direction (68.7%) likewise exceeds it (57.8%). These shape features are precisely the ones that numerical estimators handle poorly, which is why visual reasoning provides complementary value as search guidance.

F.2 Noise Robustness

As shown in Table 14, performance remains stable across noise levels ($\sigma = 0$ to 0.1) for all feature types, with no degradation—and even slight improvement in curvature detection—as noise increases, since the MLP interpolation smooths noisy inputs before visualization.

Table 14: MLLM feature extraction accuracy across noise levels.

Noise Level	Monotonic	Linear	Periodic	Exponential	Curvature
$\sigma = 0.0$	68.3%	77.2%	76.1%	72.2%	68.3%
$\sigma = 0.01$	69.4%	76.1%	78.9%	70.0%	71.7%
$\sigma = 0.1$	68.3%	77.8%	77.8%	71.1%	73.9%

F.3 Sample Size Robustness

Table 15 shows that accuracy degrades only mildly as samples drop from 1000 to 100, confirming resilience to sparse sampling.

Table 15: MLLM feature extraction accuracy across sample sizes.

Sample Size	Monotonic	Linear	Periodic	Exponential	Curvature
$n = 100$	65.6%	76.1%	75.6%	66.1%	67.2%
$n = 500$	69.4%	77.2%	77.2%	76.1%	72.8%
$n = 1000$	71.1%	77.8%	80.0%	71.1%	73.9%

F.4 Distribution Shift Robustness

Table 16 reports consistent performance across uniform, log-scaled, and Gaussian distributions, with log-scaled inputs showing the only notable sensitivity (linearity and curvature).

Table 16: MLLM feature extraction accuracy across input sampling distributions.

Distribution	Monotonic	Linear	Periodic	Exponential	Curvature
Uniform	67.2%	83.3%	78.3%	72.2%	75.6%
Log-scaled	70.6%	67.8%	76.1%	70.0%	61.1%
Gaussian	68.3%	80.0%	78.3%	71.1%	77.2%

F.5 Dimension Scaling

As shown in Table 17, overall accuracy stays in the 60.0%–79.4% range across dimensions; the dip at 3D reflects the small number of 3D test functions rather than a systematic scaling failure, and 4D–5D remain strong, indicating reliable scaling to moderately high-dimensional problems.

Table 17: MLLM feature extraction accuracy by problem dimension. Parenthesised values are Wilson 95% CI half-widths; the wider interval at 3D ($\pm$4.7) reflects its smaller test count.

Dimension	Overall Accuracy (%)	Number of Features
2D	73.8 ($\pm$2.6)	1080
3D	60.0 ($\pm$4.7)	405
4D	79.4 ($\pm$3.4)	540
5D	75.0 ($\pm$3.3)	675

F.6 Comparison with Numerical Methods

Table 18 summarizes the head-to-head accuracy of numerical methods and the MLLM across the 2700 predictions, scored by a canonical rubric applied identically to both sides. A direct visual comparison between MLLM-based feature extraction and traditional numerical methods is provided in Figure 17. The complete evaluation dataset is available as supplementary material in `mllm_feature_accuracy_results.csv` (see Appendix G for the prompt template).

Table 18: Feature Extraction Accuracy: Numerical Methods vs. Multimodal LLM, scored by a canonical rubric applied identically to both sides. The visual approach's advantage concentrates on geometric shape features (curvature, monotonicity) that numerical differentiation estimates poorly; operator-family features are read at parity.

Feature Type	Numerical Methods	MLLM (Vision)	Advantage
Curvature	25.4%	71.3%	+45.9%
Exponential	75.4%	71.1%	-4.3%
Linearity	78.9%	77.0%	-1.9%
Monotonicity	57.8%	68.7%	+10.9%
Periodic	72.4%	77.6%	+5.2%
Overall	62.0%	73.1%	+11.2%

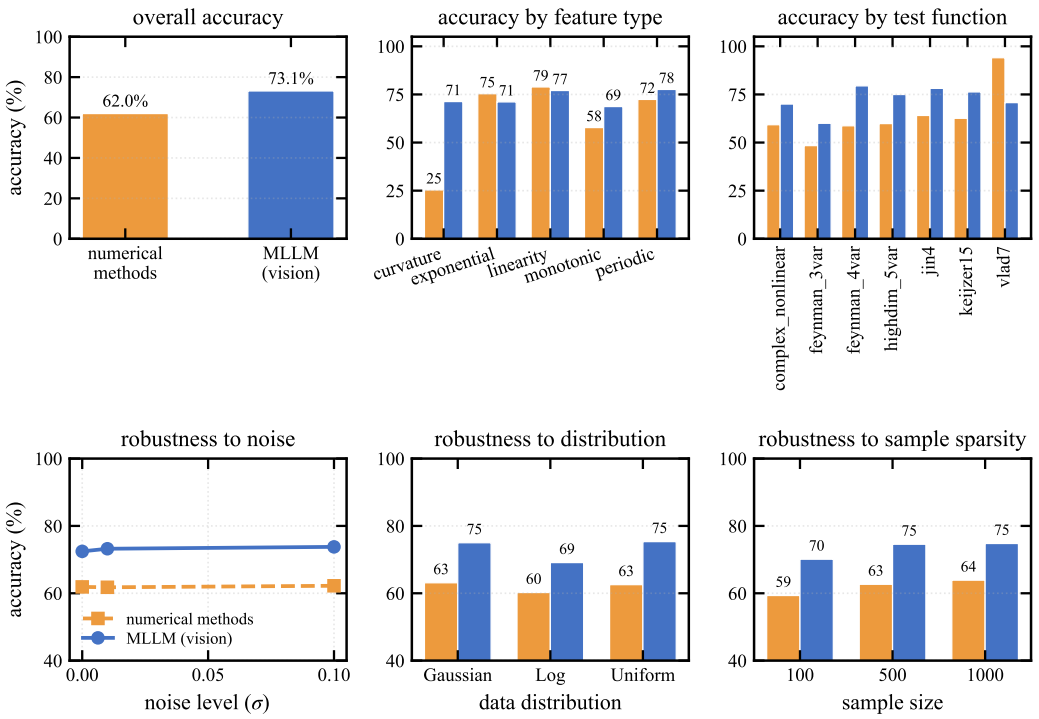


Figure 17: Head-to-head comparison between numerical gradient-based methods and MLLM visual feature extraction.

G PROMPTS

This appendix collects the principal prompt templates used in the framework, transcribed verbatim from the deployed code. The code additionally contains minor variants that reuse the same scaffolding with only the panel description adjusted—the single-panel (two-variable) form of each scan question, and the auxiliary overlay and feedback calls of the proposal loop—which are omitted here for brevity. §G.1 gives the eight type-specific scan questions, one per feature type of Table 1; §G.2 the proposal and revision loop that turns a confirmed panel into a symbolic coordinate; §G.3 the shape-reading prompt applied to every verified layer; §G.4 the two reflection prompts used inside the search; §G.5 the LLM backbone's operator prompts; §G.6 the feature-description prompt used only for the MLLM comparison of Appendix F. In all templates, `{A}`, `{B}` are the anonymized axis-variable names of the panel under examination, `{n}` the number of subplots, `{layout}` a textual list of the variable pairs, `{div}` the trial divisor, and `{scores}` a numeric report of the candidates' criterion values. All MLLM calls use the served model's default sampling parameters: no temperature or max-tokens field is set anywhere in the pipeline. Every scan and proposal question is issued twice independently (the shape-reading prompt of §G.3 three times, escalated candidates three times each) and the candidate pools are merged; the numerical criteria alone decide what is accepted—there is no voting.

G.1 THE EIGHT TYPE-SPECIFIC SCAN QUESTIONS

G.1.1 POLE

```
This image shows scatter samples of the same physical system across all {n}
pairs of input variables ({layout}; each subplot title gives its two variable
names, format "x-axis variable / y-axis variable").
Red points are the small fraction of samples with the LARGEST |y| (most extreme
system response); gray points are ordinary samples.

Check each of the {n} subplots: in which subplots do the red points arrange
into clean, recognizable geometric lines or curves (e.g. straight lines,
hyperbolas, periodic stripes, sharp arc boundaries)?
Distinguish carefully:
```

```
- A global density tilt of red points toward a corner or edge is NOT geometric
  structure; only explicit lines/curves/arc boundaries count.
- If the gray points themselves form a curve (the data is a trajectory) and
  the red points are just a segment of that gray curve with no new trend
  beyond it, that also does NOT count as the red points having their own
  structure.

Output ONLY one JSON object, no other text:
{"structured_panels": ["xvar/yvar", ...], "notes": "one sentence, optional"}

- structured_panels lists only subplots with explicit line/curve/arc-boundary
  structure, using the variable names from the subplot titles.
- If no subplot has explicit structure, output {"structured_panels": []}.
- Do not estimate any numeric values.
```

G.1.2 ZERO SET

Identical scaffolding to the pole scan, with the highlighted population inverted:

```
This image shows scatter samples of the same physical system across all {n}
pairs of input variables ({layout}; each subplot title gives its two variable
names, format "x-axis variable / y-axis variable").
Blue points are the small fraction of samples where the response is essentially
ZERO (smallest |y|); gray points are ordinary samples.

Check each of the {n} subplots: in which subplots do the blue points arrange into
clean, recognizable geometric lines or curves (e.g. straight lines, hyperbolas,
periodic stripes, sharp arc boundaries)?
Distinguish carefully:
- A global density tilt of blue points toward a corner or edge is NOT geometric
  structure; only explicit lines/curves/arc boundaries count.
- If the gray points themselves form a curve (the data is a trajectory) and
  the blue points are just a segment of that gray curve with no new trend
  beyond it, that also does NOT count as the blue points having their own
  structure.

Output ONLY one JSON object, no other text:
{"structured_panels": ["xvar/yvar", ...], "notes": "one sentence, optional"}

- structured_panels lists only subplots with explicit line/curve/arc-boundary
  structure, using the variable names from the subplot titles.
- If no subplot has explicit structure, output {"structured_panels": []}.
- Do not estimate any numeric values.
```

G.1.3 COMPOSITE COORDINATE

```
This image shows scatter samples of the same physical system across all {n}
pairs of input variables ({layout}; each subplot title gives its two variable
names, format "x-axis variable / y-axis variable").
Every point is COLORED by the system's response y -- the colorbar shows the
RANK of y (blue = smallest responses, red = largest, pale = middle), so color
differences are visible uniformly across the whole range.

Check each of the {n} subplots: in which subplots does the COLOR arrange into
clean, recognizable bands, stripes, or ridge/valley lines -- i.e. the color
stays (roughly) CONSTANT along some curve in the panel?
Distinguish carefully:
- Random salt-and-pepper color mixing does NOT count; only coherent
  bands/stripes/ridges count.
- Structure that merely follows the data cloud's density or edges (empty
  corners, cloud outline) is NOT color structure.
- A smooth large-scale gradient COUNTS if its iso-color lines are clean curves.

Output ONLY one JSON object, no other text:
{"structured_panels": ["xvar/yvar", ...], "notes": "one sentence, optional"}

- structured_panels lists only subplots with coherent iso-color structure,
  using the variable names from the subplot titles.
- If no subplot has such structure, output {"structured_panels": []}.
- Do not estimate any numeric values.
```

G.1.4 SINGLE-VARIABLE CURVE

```
This image shows the response y of a dynamical system plotted against EACH of
its input variables (one subplot per variable, variable name on the x-axis).
Red points are the samples with the LARGEST |y|.

The samples may lie on a TRAJECTORY (the system evolves in time), so do NOT
```

```
look for 2-D bands -- instead check each subplot on its own:
- In which subplots does y look like a CLEAN single-valued function of the
  x-axis variable: a smooth curve (rising, decaying, oscillating, saturating)
  with little vertical spread?
- Or a clear DIVERGENCE: y blowing up as the variable approaches a specific
  value?
- A roundish / vertical-spread cloud where many different y values occur at
  the same x does NOT count.

Output ONLY one JSON object, no other text:
{"structured_vars": ["variable_name", ...], "notes": "one sentence, optional"}

- structured_vars lists only subplot x-axis variables showing a clean curve or
  divergence.
- If no subplot shows such dependence, output {"structured_vars": []}.
- Do not estimate any numeric values.
```

G.1.5 SPREADING ENVELOPE

```
This image shows the response y of a system plotted against EACH of its input
variables (one subplot per variable, variable name on the x-axis). Red points
are the samples with the LARGEST |y|.

Do NOT look for a clean curve. Instead check the vertical THICKNESS (spread)
of the point cloud in each subplot:
- In which subplots does the spread change SMOOTHLY and STRONGLY along the
  x-axis -- a fan / trumpet / cone shape, a cloud pinched at one end and wide
  at the other, or an oscillating curve whose AMPLITUDE grows or decays
  along x?
- A cloud of uniform thickness does NOT count; neither does a thin clean curve.
- When in doubt, NAME the variable anyway: a downstream numeric check verifies
  every claim, so a missed fan is worse than an extra name.

Output ONLY one JSON object, no other text:
{"fan_vars": ["variable_name", ...], "notes": "one sentence, optional"}

- fan_vars lists only subplot x-axis variables showing a clear fan/envelope.
- If no subplot shows one, output {"fan_vars": []}.
- Do not estimate any numeric values.
```

G.1.6 MAGNITUDE STRUCTURE

```
This image shows the MAGNITUDE |y| of the response of a system plotted against
EACH of its input variables (one subplot per variable, variable name on the
x-axis, LOG y-axis). Red points are the samples with the LARGEST |y|.

The response itself flips sign (e.g. oscillates), so its magnitude is plotted.
Check each subplot on its own:
- In which subplots does |y| form a clean, narrow band or curve along the
  x-axis variable -- e.g. repeated arches of the same shape, a smooth decay or
  growth -- with little vertical thickness?
- A few DISTINCT overlapping thin bands (several separate smooth trajectories
  stacked on top of each other) still count as structure -- that is not the
  same as a uniform thick cloud. A roundish / uniformly thick cloud where |y|
  fills the whole range at the same x does NOT count.
- When in doubt, NAME the variable anyway: a downstream numeric check verifies
  every claim, so a missed structured variable is worse than an extra name.

Output ONLY one JSON object, no other text:
{"structured_vars": ["variable_name", ...], "notes": "one sentence, optional"}

- structured_vars lists only subplot x-axis variables showing a clean |y|
  dependence.
- If no subplot shows such structure, output {"structured_vars": []}.
- Do not estimate any numeric values.
```

G.1.7 MULTIPLICATIVE FACTOR

Identical scaffolding to the composite-coordinate scan, on quotient-colored panels:

```
This image shows scatter samples of the same physical system across all {n}
pairs of input variables ({layout}; each subplot title gives its two variable
names, format "x-axis variable / y-axis variable").
Every point is COLORED by the system's response y DIVIDED BY the raw variable
{div} -- the colorbar shows the RANK of the quotient y/{div} (blue = smallest
quotient, red = largest, pale = middle), so color differences are visible
uniformly across the whole range.
```

```
Check each of the {n} subplots: in which subplots does the COLOR arrange into
clean, recognizable bands, stripes, or ridge/valley lines -- i.e. the color
stays (roughly) CONSTANT along some curve in the panel?
Distinguish carefully:
- Random salt-and-pepper color mixing does NOT count; only coherent
  bands/stripes/ridges count.
- Structure that merely follows the data cloud's density or edges (empty
  corners, cloud outline) is NOT color structure.
- A smooth large-scale gradient COUNTS if its iso-color lines are clean curves.

Output ONLY one JSON object, no other text:
{"structured_panels": ["xvar/yvar", ...], "notes": "one sentence, optional"}

- structured_panels lists only subplots with coherent iso-color structure,
  using the variable names from the subplot titles.
- If no subplot has such structure, output {"structured_panels": []}.
- Do not estimate any numeric values.
```

G.1.8 PARALLEL BANDS

```
This image shows the response y of a physical system plotted against EACH of
its input variables (one subplot per variable, variable name on the x-axis).
Red points are the samples with the LARGEST |y|.

Check each subplot: in which subplots do the points form several clean,
roughly PARALLEL curve bands -- the SAME curve shape repeated at different
vertical offsets, like shifted copies of one curve stacked across the panel?
Distinguish carefully:
- A SINGLE clean curve with little vertical spread does NOT count here (that
  is a different kind of structure). But the bands may be tightly packed:
  several strands hugging each other while tracking one shared curve shape DO
  count -- narrow gaps between bands are still bands, and a modest overall
  vertical thickness does not turn them into a single curve.
- A diffuse/roundish cloud does NOT count.
- A fan that SPREADS does NOT count: if the vertical width of the point set
  changes strongly along the x-axis, or the bands clearly diverge/converge,
  it is not a parallel family. The bands must share one shape and stay
  roughly parallel.
- The bands need not be perfectly separated; a clear stacked-band impression
  with one shared curve shape is enough. When in doubt, NAME the variable
  anyway: a downstream numeric check verifies every claim, so a missed
  structured variable is worse than an extra name.

Output ONLY one JSON object, no other text:
{"structured_vars": ["variable_name", ...], "notes": "one sentence, optional"}

- structured_vars lists only subplot x-axis variables showing a parallel-band
  family.
- If no subplot shows such structure, output {"structured_vars": []}.
- Do not estimate any numeric values.
```

G.2 CANDIDATE PROPOSAL AND REVISION

Once a panel is confirmed, the framework asks for symbolic candidates, scores them numerically, and lets the MLLM revise against an overlay of its own best candidate. The pole channel's loop is canonical. The proposal prompt:

```
This panel shows scatter samples of a physical system: red points are the
samples with the LARGEST |y|, gray points are ordinary samples. The red points
arrange into a geometric structure in this {A}/{B} plane ({A} on the x-axis,
{B} on the y-axis; read positions and spacings off the axis tick values).

Your task: first OBSERVE carefully, then WRITE 1-3 candidate symbolic
expressions g({A}, {B}) whose ZERO SET g = 0 should coincide with the red
structure (the red points should lie exactly where |g| is smallest).

Observe (fill the "observation" field):
1. COUNT the separate red lines/branches. One? Several? Many repeating ones?
2. INVARIANT ALONG ONE LINE: pick 2-3 points lying ON one red line and read
   their coordinates off the axes. Then test simple combinations of {A} and
   {B} at those points -- the sum, the difference, the product, the ratio,
   ... -- which combination has (nearly) the SAME value at every point on the
   line? Report the combination and its value (e.g. "{A}*{B} = 6.3 on the
   first line"). If NO raw combination stays constant, one variable may sit
   INSIDE a function of the other -- test forms like {A}*cos({B}),
   cos({A})*{B}, {A}*sqrt({B}), {A}*exp(-{B}) the same way.
3. If the pattern REPEATS over several lines: report that combination's value
   on EACH line (e.g. 3.1, 6.3, 9.4, ...). Are the values evenly spaced? By
```

```
   how much?

Then write the candidates, using what you measured: g must be zero exactly
where your invariant combination takes the values you listed (all of them at
once). Examples of the LEVEL of answer expected: if the red points form a
vertical line at {A}~2, a candidate is "{A}-2"; if they form stripes repeating
with period ~6.28 along the product {A}*{B}, a candidate could be
"cos({A}*{B})-1" (zero wherever {A}*{B} = 2*pi*k); if the red curve bends like
a sideways cosine (drifting with {A} while curving back and forth along {B}),
a candidate could be "{A}*cos({B})-2". Invent the form yourself from what you
SEE. CRITICAL: your g must be zero on ALL the red lines AT ONCE -- if you
counted several repeating lines, a single-line equation can only match one of
them and will be scored low.

Rules:
- Use ONLY the variables {A}, {B}, numeric constants, + - * / ** (or ^) and
  the functions cos sin tan exp sqrt abs log, plus the constant pi.
- Write each candidate as a bare expression (e.g. "cos({A}*{B})-1"), NOT an
  equation -- no "=" sign.
- 1 to 3 candidates, most likely first.
- Output ONLY one JSON object, no other text:
{"observation": "what you counted and measured",
 "candidates": [{"expr": "...", "why": "a few words"}],
 "need_extra_var": false}
- If you believe the structure fundamentally cannot be written with {A}, {B}
  alone, set "need_extra_var": true and still give your best attempts.
```

The zero-set channel uses the same observe-then-write protocol with the task reframed ("the blue points should lie exactly where $|g|$ is smallest. Since the RESPONSE vanishes on the blue structure, your g is most likely a NUMERATOR factor of the response"); the composite-coordinate and multiplicative-factor channels ask directly for the coordinate ("find the COMPOSITE COORDINATE $u(\{A\},\{B\})$ whose LEVEL CURVES ($u = \text{const}$) trace the iso-color bands—i.e. y depends on $\{A\}$ and $\{B\}$ mostly through this one combination u"; for the quotient view, "the quotient $y/\{\text{div}\}$ depends on $\{A\}$ and $\{B\}$ mostly through this one combination u (equivalently $y \sim \{\text{div}\} \cdot h(u)$)").

The revision prompts feed the numeric scores back. The overlay prompt (pole channel):

```
The BLUE curve(s) in this panel show the ZERO SET of your best candidate so
far:
  g = {best_expr}   (data score: AUC={best_auc:.3f}; AUC=1.0 would mean the blue
  lines pass exactly through the red points)
All your candidates so far scored:
{scores}
The red points are the target structure, same as before.

LOOK at the mismatch: are the blue lines in the wrong position, wrong
orientation, wrong spacing, wrong curvature -- or are there too FEW blue lines
(the red structure may have several branches while your g has only one)?
If the blue lines are straight/axis-aligned but the red lines CURVE, re-measure
before rewriting: pick 2-3 points ON one red line, read their coordinates, and
find which simple combination of {A} and {B} stays constant along the line --
sum, difference, product, ratio, or one variable wrapped in a function of the
other ({A}*cos({B}), cos({A})*{B}, {A}*sqrt({B}), ...). Your g should be built
from THAT combination.
Then WRITE 1-3 IMPROVED candidate expressions g({A}, {B}) whose zero set would
pass through ALL the red lines at once.{family_note}

Same rules: ONLY the variables {A}, {B}, numeric constants, + - * / ** (or ^),
functions cos sin tan exp sqrt abs log, constant pi. Bare expressions, no "="
sign.
Output ONLY one JSON object:
{"observation": "the mismatch you see",
 "candidates": [{"expr": "...", "why": "a few words"}],
 "need_extra_var": false}
```

and, after a failed round, the feedback prompt admits one extra variable:

```
Your previous candidate(s) for the red structure in the {A}/{B} panel were
scored against the data:
{scores}
(AUC: 1.0 would mean the red points lie exactly on the zero set of g, 0.5
means unrelated. cross: how close g actually gets to 0 relative to its typical
magnitude -- a real zero set must reach < 0.05.)

The structure was NOT confirmed. Try again with better expressions. You may
now also use ONE extra variable beyond {A} and {B}, chosen from:
{extra}.{family_note}
Use ONLY the variable names listed above ({A}, {B}, plus optionally ONE from
```

```
the extra list) -- do not invent names like x or y. Bare expressions, no "="
sign.
Same JSON output format as before:
{"observation": "what you think was wrong",
 "candidates": [{"expr": "...", "why": "a few words"}],
 "need_extra_var": false}
```

Two further pole-channel variants cover borderline and three-variable cases: a log-log re-plot on which the MLLM double-checks the shape family (a product $A \cdot B = \text{const}$ becomes a slope--1 line, a ratio keeps slope $+1$), and slice panels conditioned on a third variable $\{Z\}$ for structures whose zero set moves with Z. Their full text follows the same protocol and is included in the released code.

G.3 Reading the Shape of a Verified Layer

Every verified coordinate u is re-plotted against the response, and the MLLM reads the outer dependence $h(u)$ and classifies the layer as multiplicative or additive. The canonical prompt (pole variant):

```
This 1-D view shows the response y of a physical system plotted against the
COMPOSITE coordinate u = {u_expr} (a combination of the raw variables).
Red points are the samples with the LARGEST |y|.

Earlier analysis established that the extreme samples lie where a factor of u
vanishes. Now describe the OUTER dependence: what does y look like AS A
FUNCTION OF u?

Observe (fill "observation"): the overall shape -- repeating
arches/divergences? a monotone rise or decay? saturation? a straight line? no
clean dependence? If there are divergences or extrema, READ THEIR u-POSITIONS
off the axis (e.g. "divergences at u ~ -1.6, 1.6, 4.7") -- your h(u) MUST
diverge/behave at exactly those positions, so build it from a function that
vanishes/peaks there. COUNT the divergences: if they repeat at (roughly)
EVENLY SPACED u-positions, h MUST be a periodic form (e.g.
"1/(1-cos(a*u+b))", "1/sin(a*u+b)**2") -- a rational denominator like
1/(u-a)**2 can only produce ISOLATED divergences and can never match a
repeating pattern; more than two divergences -> go periodic.

Then WRITE 1-3 candidate expressions h(u) for this dependence, using ONLY the
variable u, numeric constants, + - * / ** (or ^), functions cos sin tan exp
sqrt abs log, and the constant pi. Bare expressions, no "=" sign. Write the
phase/scale constants INSIDE function arguments as rough placeholder NUMBERS
(e.g. "1/cos(2*u+1)**2") -- every constant is refit numerically afterwards,
so rough values are fine; never use LETTERS for constants. The form must be
able to place its divergences/extrema at the positions you read. If the curve
is a spike/arch/decay ON TOP OF a nonzero floor (it flattens to a nonzero
level at one or both ends), write the SUM of floor and shape (e.g.
"0.3 + 1/(u-1)**2") -- a bare spike form like 1/(u-a) cannot reproduce the
floor.

Also classify the layer:
- "mul": y looks like h(u) TIMES something else (y = h(u)*rest).
  Divergence/arch shapes are this type; write h INCLUDING its denominator,
  e.g. "1/(1-cos(u))".
- "add": y looks like h(u) PLUS something else (y = h(u)+rest).

Output ONLY one JSON object, no other text:
{"observation": "the shape you see",
 "candidates": [{"expr": "...", "why": "a few words"}],
 "mode": "mul" | "add", "clean_structure": true}
If there is no clean dependence of y on u, set "clean_structure": false and
give an empty candidate list.
```

Per-channel variants are mechanical substitutions of the establishing sentence: the zero-set variant states that the response VANISHES where a factor of u is zero and asks for periodic forms with zeros (`1-cos(a*u+b)`, `sin(a*u+b)`); the composite-coordinate variant states that y depends on the raw variables mostly through this single coordinate u (color constant along $u = \text{const}$ curves); the single-variable and magnitude variants refer to the clean 1-D dependence of y (resp. $|y|$) on the raw variable u; the multiplicative-factor variant reads the quotient, with $y = \{\text{div}\} \cdot h(u) \cdot \text{rest}$; the parallel-band variant appends "This layer is additive by construction: report mode `"add"`." An overlay variant feeds back the numerically fitted curve and its residual spread for one revision round.

G.4 Reflection Prompts During the Search

When population diversity drops, the best individual of each cluster is plotted against the target and the MLLM judges them:

```
Please judge which function in the plot tends to better fit the feature of the
target function and answer in following way for each independent variable:
{For independent variable (name of the independent variable),  These functions
(numbers in the legend) tends to perform badly and These functions (numbers in
the legend) perform good. The confidence level of the above description
is {}% }
Concisely analysis each independent variable separately of each function .
```

When the best objective stagnates, the current best candidate is plotted against the target and the MLLM is asked to refresh the insight block:

```
Please summarize feature of the target function plotted using red line in the
figure on the basis and in the same format of following contents:
<current insight block>
make corrections if necessary, dont revise
Besides, based on the plot, what change could benefit current best candidate:
<current best function>
to approach the target, provide some concise candidates functions that may
better fit the target if you can. the form of output should be: Here are some
candidates which may better fit the target function: 1. function 1 ;
2. function 2
```

G.5 LLM Backbone Operator Prompts

The evolutionary backbone builds its prompts by string concatenation; angle brackets mark the filled fields: <task> the problem statement, <func> the function name, <inputs>/<outputs> the variable lists, <io> input–output formatting information, and <other> the auxiliary information slot through which the insight block enters (included with probability insight_prob $= 0.6$). Every operator shares the tail: "First, describe your new function and main steps in one sentence. The description must be inside a brace. Next, implement it in Python as a function named <func>. This function should accept <n> input(s): <inputs>. The function should return 1 output(s): <outputs>. <io> <other>". The operator-specific bodies:

Initialization (i1). "<task>" followed directly by the shared tail.

Crossover 1 (e1). "<task> I have <k> existing function with their codes as follows: <functions> Please help me create a new function as concise as possible which includes the common unary operator from the above functions." plus the shared tail.

Crossover 2 (e2). "<task> I have <k> existing functions with their codes as follows: <functions> Please help me create a new function as concise as possible which does not includes the common unary operator from the above functions. Firstly, identify the common backbone idea in the provided functions. Secondly, based on the backbone idea describe your new function in one sentence." (the tail's "First…Next" ordering is adjusted accordingly).

Mutation 1 (m1). "<task> I have one function with its code as follows. Function description: <desc> Code: <code> Replace several binary and unary operators randomly using different operators (e.g., turn + into *, - into / etc.). Use more * and / rather than + and - in the designed formula." plus the shared tail.

Mutation 2 (m2). Same header, then: "Change the power of independent variables, or add constant to independent variables. (e.g., turn a into a**2, a**2 into a**(0.5), sin(a) into sin(a+1), a*b into a*(b+1) etc.)" plus the shared tail.

Mutation 3 (m3). Same header, then: "For complex expressions composed of more than five binary operators, delete some parts of this function and creat a concise function, especially parts containing overly large constants compared with other parts, keep all independent variables in the function" plus the shared tail.

G.6 Feature-Description Prompt for the MLLM Comparison

The human-rated MLLM comparison of Appendix F predates the eight-type protocol and uses a free-form per-variable description prompt:

```
Observe these pictures, on each picture, there are multiple curves of an
independent variable. Describe and summarize the commonalities among the
majority of the curves in each picture:  Form the output in the following way
for each independent variable and fill your answer in the {}, delete {} for
your final output:
'For independent variable {fill the name of the independent variable}: this
function is {linear?, periodic?, symmetry? describe the most important
feature}, the function is likely to apply basic operators including {unary
operators which can generate the figure} to this variable, and the power of
this independent variable is likely to be {1 if linear, odd if asymmetric,
even if symmetric; negative if y increase quickly when this variable
approcahes 0.}, the relationship between this variable and other variables
could be {addition(if all curves are strictly parallel in the plot of this
variable); multiplication (if all curves are not parallel in the plot of
this variable); division (If y increase quickly when this variable
approcahes 0) The confidence level of the above description is {}% (lower
if features are unclear for this independent variable). }'
Only give answer in text form rather than mathematical form (do not just say
yes or no) and do not provide any other descriptions.
```

H Other Supplementary Information

All experiments in this paper were conducted on a single computer:

- CPU: AMD Ryzen 9 8945HX with Radeon Graphics
- GPU: NVIDIA GeForce RTX 5060 Laptop GPU

GPT-4o was used for MLLM visual feature extraction and GPT-4o-mini for LLM formula generation and evolution, both accessed via remote API calls. The computational speed depends on network latency; consequently, our approach incurs a higher time cost than methods that skip LLM dialogue (which is why we cannot reuse the parameter settings of the traditional genetic-algorithm-based symbolic regression algorithms shown in Appendix I). We believe this is a shared limitation of any symbolic regression method that relies on LLM interaction, and once the LLM is deployed locally, the overhead due to API calls and network communication disappears.

Two reproducibility disclosures apply. First, model versions were not pinned: all API calls were made between August and September 2026 against the providers' default snapshots for `gpt-4o` and `gpt-4o-mini`, so provider-side updates during the experimental window are an uncontrolled source of drift. We state this as an explicit limitation; re-running the full evaluation against pinned, dated snapshots remains future work. Second, per-run monetary cost and end-to-end wall-clock latency were not logged during the experimental campaign, so we cannot report them; the dominant cost drivers are nonetheless identifiable from the configuration (two extraction passes per run plus at most one health-triggered re-acquisition, and the LLM-operator calls per generation, listed in Appendix I).

H.1 Extended Discussion of the LSR-Synth Self-Check

Table 19: Convenience sample of the anti-memorization LSR-Synth subset (6 problems per domain, 24 total; single run, GPT-4o visual extraction and GPT-4o-mini evolution, variable semantics stripped). "Solved" counts test $R^2 > 0.999$ (accuracy criterion, not exact symbolic match). Reported as a robustness self-check, not a baseline comparison.

Domain	#	Solved (R^2>0.999)	Median R^2	Mean R^2
Chemical reaction (chem_react)	6	1/6	0.962	0.672
Population growth (bio_pop_growth)	6	1/6	0.944	0.691
Materials science (matsci)	6	1/6	0.994	0.986
Physical oscillator (phys_osc)	6	0/6	0.917	0.729
All LSR-Synth	**24**	**3/24**	**0.957**	**0.770**

The aggregate in Table 2 covers only LSR-Transform. To directly address the concern that success on Feynman-style benchmarks may reflect memorization of familiar forms, we additionally evaluate a convenience sample of the anti-memorization LSR-Synth subset—6 problems from each of its

four domains (24 total), run under our own protocol with all variable semantics stripped (Table 19). We report this as a robustness self-check rather than a leaderboard: because we sample a subset under our own configuration, the numbers are not directly comparable to the full-set baseline figures in the benchmark paper, and we therefore make no head-to-head claim. Two observations matter. First, under the strict accuracy criterion ($R^2 > 0.999$) only $3/24\,(12.5\%)$ of the Synth problems are solved—exactly what one expects if the method *searches* from data rather than recalling named equations; a high Synth solution rate would in fact be the more suspicious outcome. We report this accuracy-based criterion rather than exact symbolic match here, and therefore do *not* draw a direct numerical comparison to the exact solution rate in Table 2: the Synth subset is a single-run sample under our own configuration, so the two figures are not measured on the same footing. Second, numerical fit remains strong (median test $R^2 = 0.957$; $17/24$ problems exceed $R^2 = 0.9$), with the mean (0.77) pulled down by a few genuine failures (e.g. `PO3` and one chem/bio overfit with negative R^2) rather than uniform mediocrity. We read this gap descriptively: it is consistent with a method that searches from data rather than recalling named equations, though a single-arm self-check without a comparator arm cannot by itself measure memorization resistance.

I Experiment Settings

I.1 ViGa-SR Configuration

Table 20 collects the deployed configuration of ViGa-SR for reproducibility. Values are reported as deployed in the experimental campaign. Two items could not be recovered from the run records and are stated honestly rather than guessed: the train/validation/test split proportions of the Feynman evaluation and the exact equivalence checker used for the strict solution criterion follow the protocol of the respective source benchmarks La Cava et al. (2021); Shojaee et al. (2025) but were not independently logged by us.

Table 20: Deployed ViGa-SR configuration. All values are as used in the reported experiments.

Component	Setting	Notes
Visual extraction model	GPT-4o	provider default snapshot, Aug–Sep 2026
Evolution model	GPT-4o-mini	provider default snapshot, Aug–Sep 2026
Extraction passes per run	2 (seeds 0 and 1)	plus at most 1 health-triggered re-acquisition
Seeds per pass	up to 24	assembled from verified decompositions
Interpolator	MLP-GradP, 256-128-64	gradient penalty $\lambda = 0.002$ (App. D)
LLM operators	i1, e1, e2, m1, m2, m3	crossover/mutation prompts (App. G)
Insight probability	0.6	probability an operator call consults the insight block
Evolution random seed	2024	fixed for the reported runs
API rate limit	1 call / 5 s	120 s timeout, up to 5 retries
LSR-Transform budget	1000 LLM calls/problem	10 independent runs per problem
Oscillator case studies	population 40, 500 iterations	10 runs (descriptive, App. A)
Feynman population/generations	*not logged*	could not be recovered from run records; stated honestly rather than guessed
Train/val/test split	*not logged*	follows the source-benchmark protocol; see text

I.2 Settings of Other Methods

The results and settings of the other methods in this paper are taken directly from La Cava et al. (2021), as shown in Table 21.

Table 21: Hyper-parameters of other symbolic regression algorithms tested on the Feynman dataset (0 and 0.01 noise).

Algorithm	Key Parameter	Symbol	Value	Description
AFP	popsize	N_{pop}	1000	population size
	generations	G	250	max generations
	op_list	$\mathcal{O}$	[+, *, exp, log, sqrt, sin, cos]	primitive set
AFP_FE	idem	idem	idem	same as AFP
AIFeynman	BF_try_time	T_{BF}	600 s	brute-force budget
	NN_epochs	E_{NN}	400	NN-training epochs
	ops_file	$\mathcal{F}_{\text{ops}}$	14ops.txt	14-function set
BSR	treeNum	K	3	# trees in additive model
	itrNum	I	5000	MCMC iterations
	val	ν	100	prior variance scale
DSR	batch_size	B	1000	policy-gradient batch
EPLEX	popsize	N_{pop}	1000	population
	generations	G	250	generations
	op_list	$\mathcal{O}$	[+, *, exp, log, sqrt, sin, cos]	primitives
FEAT	pop_size	N_{pop}	1000	population
	gens	G	250	generations
	lr	η	0.1	GD learning-rate on constants
FFX	deterministic	–	–	no stochastic hyper-params
GP-GOMEA	popsize	N_{pop}	1000	population
	init_max_height	$h_{\max}$	6	max initial tree depth
	functions	$\mathcal{O}$	[+, *, /, exp, log, sqrt, sin, cos]	primitives
	linear_scaling	ℓ	True	automatic linear scaling
gplearn	population_size	N_{pop}	1000	population
	generations	G	500	generations
	function_set	$\mathcal{O}$	[add, mul, div, log, sqrt, sin, cos]	primitives
ITEA	term_limit	$t_{\max}$	15	max terms per model
	exponents	$\mathcal{E}$	$[-5, 5]$	integer power range
	trans_functions	$\mathcal{T}$	[Id, Sin, Cos, Exp, Log]	unary transforms
MRGP	popsize	N_{pop}	1000	population
	G	G	250	generations
	rt_cross	p_{C}	0.8	crossover rate
	rt_mut	p_{M}	0.2	mutation rate
Operon	population_size	N_{pop}	500	population
	max_length	$L_{\max}$	50	max program length
	allowed_symbols	$\mathcal{S}$	[add, mul, aq, exp, log, sin, tanh, const, var]	primitives
	local_iterations	I_{LM}	5	Levenberg–Marquardt steps
SBP-GP	popsize	N_{pop}	1000	population
	functions	$\mathcal{O}$	[+, *, /, exp, log, sqrt, sin, cos]	primitives
	linear_scaling	ℓ	True	automatic linear scaling

J Detailed Per-Problem Analysis of Structure Discovery

This appendix examines the feature-extraction stage problem by problem on the 30-problem benchmark used throughout this work (21 problems from the Feynman-derived LSR-Transform collection spanning rational-pole, trigonometric, transcendental, and smooth families, and 9 harder problems from the LLM-SRBench synthetic collections; the per-problem IDs are exactly those listed in Table 22). Two scope disclosures matter. First, the 21 LSR-Transform members also belong to the 111-equation leaderboard set of §3, so this audit set is *not* independent of the leaderboard evaluation. Second, the audit set is distinct from the 100-equation Feynman evaluation set of §3, its Feynman-derived members entering through LSR-Transform. The examination isolates the extraction stage from the evolutionary search: the framework is run on each training set with anonymized variables and GPT-4o as the MLLM, and we record which features were identified, which of them passed numerical verification, and how well the resulting seed expressions fit held-out test data before any evolution takes place. The seed R^2 is therefore a direct measurement of the quality of the extracted prior knowledge, not of the search.

Because the MLLM reading is stochastic, each problem was examined in several independent attempts (six examination rounds as the visualization schemes were refined, plus directed single-problem examinations). Table 22 reports, for every problem, the most informative identified features and the best seed test R^2 obtained across attempts. We stress the framing: this table is an *attainable-ceiling* measurement of the extraction stage across repeated attempts, not the expected output of a single extraction pass, and it deliberately uses test data to score the seeds because its purpose is to audit what the extraction stage *can* produce. A second caveat follows from how the rounds were run: the six rounds span the development period of the pipeline (August–September 2026), so the schemes

being read were not identical across rounds, and this audit is *not* a frozen-pipeline measurement. A frozen-code, held-out audit of the deployed pipeline—fixed pass counts, per-type confirmation and rejection rates, and a false-confirmation interval on problems never used during refinement—remains future work; the numbers below should be read as an upper envelope over development versions. No production decision is made this way: within a run, the choice among seeds and the early stopping of the search are governed by the held-out validation split, and test data is touched only for final reporting. The per-attempt variation is itself analyzed below; the production configuration performs two independent extraction passes before the search and up to one more on a health trigger, in each of several independent runs, precisely because single-attempt variance is the dominant error source.

Table 22: Per-problem structure discovery on the 30-problem benchmark. "Identified features" lists the most informative verified features across attempts, in the form feature type: coordinate. The best seed R^2 is measured on held-out test data before any evolution; a dash means no feature passed verification in any attempt. Entries marked † are measured on the training data (directed single-problem examinations where no test split was scored).

Problem	Family	Identified features (verified)	Best seed R^2
II.11.28_0_0	rational pole	pole: $u{=}b$; pole: $u{=}a$; pole: $u{=}a\,b$	0.994
II.11.27_0_0	rational pole	pole: $u{=}b$	0.989
I.13.12_3_0	rational pole	—	—
II.21.32_2_0	rational pole	pole: $u{=}a$	0.972
II.34.2_1_0	rational pole	multiplicative factor: quotient y/a collapses onto $u{=}b\,c$	$\approx 1^\dagger$
II.36.38_5_0	rational pole	—	—
I.30.3_0_0	trigonometric	pole: $u{=}b\,c$; pole: $u{=}a$; pole: $u{=}b$	0.972
II.15.5_0_0	trigonometric	pole: $u{=}\lvert c{-}1.57\rvert\,\lvert c{-}4.71\rvert$	0.000
III.17.37_0_0	trigonometric	pole: $u{=}a\cos b$ (exact zero set)	$0.92^\dagger$
I.50.26_0_0	trigonometric	—	—
III.9.52_0_0	trigonometric	pole family: $u{\approx}(f{-}e)^2/c,\ h{\propto}\lvert u\rvert^{-1.08}$	0.868
I.29.16_0_0	trig. smooth	— (correct: no exploitable structure)	—
I.44.4_4_0	transcendental	spreading envelope: $u{=}a$; pole: $u{=}a/c$	0.000
II.35.18_2_1	transcendental	pole: $u{=}a-0.5\,b$	0.255
I.40.1_4_0	transcendental	pole: $u{=}a\,d$	0.252
I.15.3t_0_0	transcendental	pole: $u{=}b+6.02\,c-c^{3.72}$	0.000
I.12.2_3_0	transc. smooth	pole: $u{=}a$	0.907
II.6.15a_1_0	smooth control	pole: divergence in one combined variable	0.820
III.10.19_1_1	smooth control	composite coordinate: $u{=}a/b$	0.963
I.48.2_1_0	smooth control	— (correct: response nearly constant)	—
II.27.18_1_0	smooth control	pole: $u{=}a/b$	1.000
PO22	SRBench	pole: $u{=}a+c$	0.641
PO5	SRBench	pole: $u{=}\cos(10c)-b$; pole: $u{=}b-e^{-a}$	0.484
PO28	SRBench	spreading envelope: $u{=}b$	0.775
PO10	SRBench	parallel bands: shared curve along $u{=}x$	$0.973^\dagger$
CRK0	SRBench	pole: $u{=}a\,b$	0.982
CRK21	SRBench	pole / magnitude structure	0.685
CRK23	SRBench	single-variable curve: $u{=}b$	1.000
MatSci10	SRBench	composite coordinate: $u{=}a{+}b$; two further poles	0.880
BPG12	SRBench	magnitude structure	0.008

Figure 19 walks through four representative extraction rounds, showing for each the panel the MLLM actually read, the response along the coordinate it wrote down, and the numerically fitted form of $h(u)$. Case (a), I.30.3 (truth $I_0\sin^2(n\theta/2)/\sin^2(\theta/2)$), is a bounded response — the ratio is the squared Fejér kernel, finite everywhere, so there is no true divergence for the pole family to lock onto — but the zero-set view makes the exact numerator visible: the MLLM writes $u=\theta\cdot n$ (crossing-localization AUC 0.9996) and reads a periodic form that the fitter pins down as $h(u)=1{-}\cos(\pi u/3.1416)=1{-}\cos u$, which *is* $2\sin^2(n\theta/2)$ up to a constant; the log-spread of the response along u drops from 1.92 to 0.57. In other attempts of the same problem the same coordinate family enters through the pole (extreme-sample) view instead, as in the rounds of Figure 2; the channel that fires first varies across attempts, the coordinate does not. Case (b), II.37.1 (truth $\mathrm{mom}\,(1{+}\chi)\,B$), shows the channels composing: the zero-set view identifies $u=\mathrm{mom}\cdot B$ (AUC 0.9899) with $h(u)=u$, and the residual after stripping then collapses onto $u=\chi$ (collapse $R^2=0.9975$) with

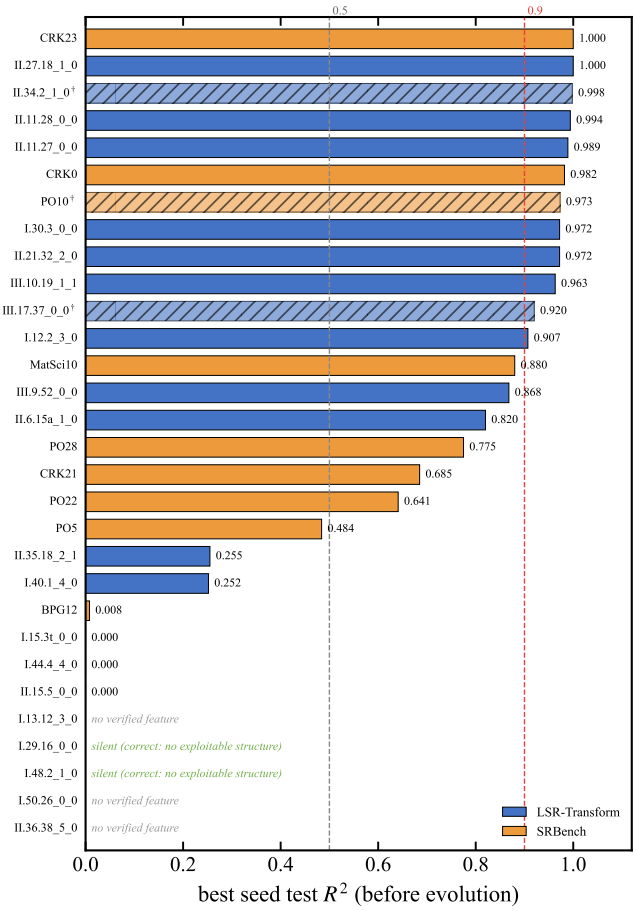


Figure 18: Best seed test R^2 before any evolution, one bar per benchmark problem, sorted. Blue bars are the 21 LSR-Transform problems, orange bars the 9 LLM-SRBench problems. A seed is the expression assembled from the verified decomposition with jointly re-fitted constants, so the bar height measures the quality of the extracted prior knowledge alone. The bottom five entries yielded no verified feature in any attempt; two of them (green) are smooth or near-constant responses where silence is the correct answer. Hatched bars are measured on the training data (directed single-problem examinations where no test split was scored); all other bars use held-out test data.

$h(u) = 2 + u$; the composition $\mathrm{mom} \cdot B \cdot (2 + \chi)$ is exact up to constants the fitter re-fits, and the residual variance falls from 1.33 to 0. Case (c), II.11.27 (truth $n\alpha/(1-n\alpha/3) \cdot \epsilon E_f$), shows a composite coordinate carrying a hidden pole: the zero-set view verifies $u = n \cdot \alpha$ (AUC 0.9997) — precisely the product whose value 3 is the true singularity — and the response along u is read as periodic; the variance drops from 0.503 to 0.115. Case (d), II.21.32 (truth $q/[4\pi\epsilon r\,(1-v/c)]$), is the control case for verification: the pole view verifies $u = \epsilon \cdot r$ (AUC 0.9875), a genuine factor of the true denominator, and the earlier framework's two fabricated layers on this problem are no longer produced — the current criteria reject them while keeping this factor (variance $2.4 \times 10^{-3} \to 9.5 \times 10^{-4}$).

Verified structure was extracted on 24 of the 30 problems in at least one attempt of the examination rounds; a 25th, III.17.37, was verified only in a directed single-problem examination (*The remaining gaps* below). Each feature type of Table 1 owns cases that no other type reaches. Poles account for the majority (the rational-pole and trigonometric families), the composite-coordinate test owns the smooth dependence through a variable combination (III.10.19, MatSci10), the single-variable test owns the trajectory problem whose pair views degenerate into curves (CRK23), the spreading-envelope test owns the envelope problems (PO28, I.44.4), the magnitude test owns sign-flipping oscillations (BPG12, CRK21), the multiplicative-factor test owns the quotient structure invisible to every pointwise criterion (II.34.2, where the undivided collapse R^2 is 0.397 and the divided one is 0.998), and the parallel-band test owns the additive structure of PO10, whose single-variable views show shifted copies of one curve (variance $18.9 \to 0.5$ after stripping). The zero-set test owns the sign-changing numerators of bounded responses, where nothing diverges and only the pattern of sign changes betrays the structure (I.30.3, whose zero set along $u = \theta \cdot n$ is the exact numerator factor $2\sin^2(n\theta/2)$ up to a constant; Figure 19a). This division is the empirical justification for maintaining eight distinct visual tests rather than one general one: the tests do not steal cases from each other, because each verification criterion is shaped to reject the structures its test is not responsible for.

Across all examination rounds, no numerically verified coordinate was later found to be fabricated structure. Two scope notes keep this statement honest. First, it is an accepted-feature check, not a completeness result: rejected claims were not logged with denominators during the examination rounds, so we report no proposal/rejection counts, and the statement covers only what verification *admitted*. Second, the guarantee covers coordinates, not functional forms: the MLLM can write down a genuinely verified coordinate yet over-read its form $h(u)$, so a layer passes its coordinate criterion while carrying a partly fabricated form; such layers simplify the data less than a true form would, and the seeds built from them fit poorly and are removed by the validation split (§4). The two further failure modes that do occur are both benign. A verified feature can be real yet useless for extrapolation — PO5 and I.44.4 yield seeds whose test R^2 is near zero or negative because the verified envelope describes only part of the response; the evolutionary search discards such seeds by validation. And a verified feature can be assembled into seeds whose jointly re-fitted constants land in a poor basin (II.11.28 in one attempt), which the same validation removes. Neither mode produces a confidently wrong prior.

The dominant source of failure is the stochasticity of the visual reading, not the numerical criteria. II.11.28 produced a seed at test $R^2 = 0.994$ in one attempt and a diverging seed in another, with the same data and the same criteria; II.6.15a fired in three attempts out of six. Aggregating attempts, 18 of the 30 problems obtained a seed with $R^2 > 0.5$ at least once (15 scored on held-out test data; II.34.2, III.17.37, and PO10 scored on training data), and 12 obtained $R^2 > 0.9$ under the same convention (9 on test data, the same 3 on training data), both counts derivable directly from Table 22. This distribution is what the production configuration exploits: multiple independent extraction passes per run, plus several independent runs, convert a per-attempt success probability into a per-problem one.

Five problems never yielded verified structure in any examination. Two are correct silence: I.29.16 is a smooth four-variable response and I.48.2 is nearly constant, and the framework's declaration that nothing is visible is the right answer for both. Two are out of reach of the current pair-based views: I.13.12 factors through a three-variable combination and II.36.38_5_0 through a five-variable one, and no two-variable panel can display either. The last, I.50.26_0_0, is perception-limited. A sixth problem, III.17.37, sits at the boundary of this group: its exact zero set ($u = a\cos b$) was verified in a directed examination (Table 22), but the identification succeeds in only about one attempt out of four, which is again the multiple-attempt regime rather than a missing capability.

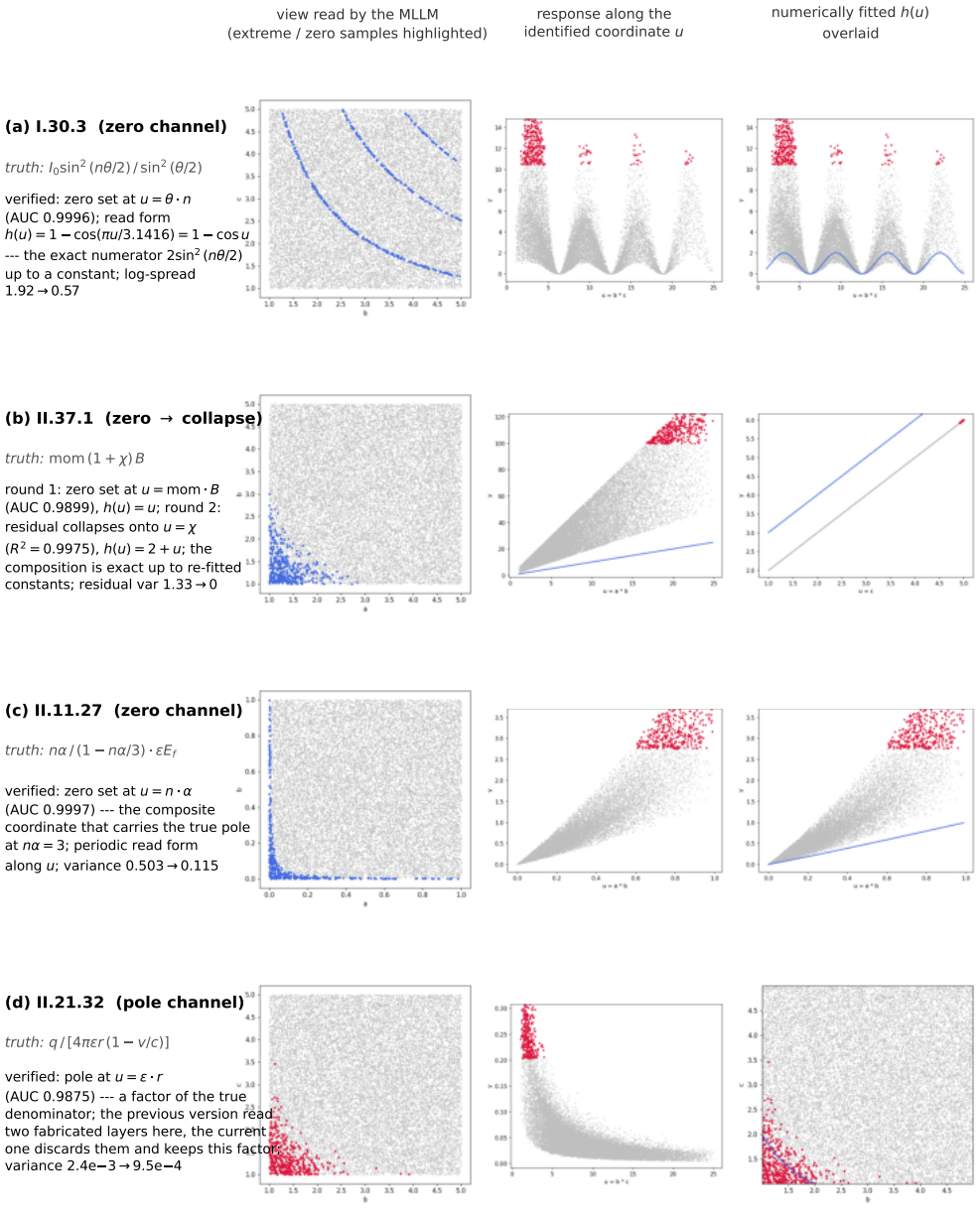


Figure 19: Four representative extraction rounds (anonymized variables, GPT-4o). Each row shows, left to right: the panel the MLLM read (extreme or zero samples highlighted), the response along the coordinate u the MLLM wrote down from that panel, and the numerically fitted $h(u)$ overlaid. Row (a) is a zero-set discovery of the exact numerator of a bounded response (no true divergence exists); row (b) shows a zero-set layer and a collapse layer composing into the exact formula; row (c) shows a composite coordinate that carries the true singularity; row (d) shows a verified denominator factor that the previous framework had buried under fabricated layers. All quoted scores (AUC, R^2, variance ratios) are produced by the numerical verification stage, not by the MLLM.