MULTI-DATASET SYMBOLIC REGRESSION: BENCHMARK CONSTRUCTION AND REFERENCE METHODS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing symbolic regression methods focus on discovering equations from a single dataset, whereas scientific experiments often require identifying a shared symbolic form from multiple datasets collected under different experimental configurations. To address this gap, we formulate the multi-dataset symbolic regression (MDSR) problem as the discovery of shared functional forms across datasets generated under varying experimental conditions, such as different parameter settings. We construct a benchmark suite for MDSR, comprising a real-world experimental benchmark from three underdamped oscillators and a synthetic benchmark generated from 59 transformed physics-derived problems under varying conditions. Unlike conventional symbolic regression benchmarks, our synthetic benchmark includes test datasets with unseen parameter values, controlled Wasserstein shifts in parameter distributions, and varying noise levels, enabling systematic evaluation of how shared functional forms discovered from training sets generalize under parameter-distribution shifts and observation noise. We further provide two reference solvers, a PySR-based Multi-stage Selection Symbolic Regression method (MSSR) and a Large-Language-Model-based Multi-Dataset Symbolic Regression method (LLM-MDSR), adapted from LLM-SR to support multiple datasets. Experimental results show that MSSR can rediscover shared functional forms equivalent to the theoretical formula on the real-world benchmark. On the synthetic benchmark, without specified parameters, MSSR identified shared functional forms with $R^2 > 0.9999$ for 48 of 59 benchmark problems (81.3 %), while LLM-MDSR discovered shared functional forms with $R^2 > 0.9999$ for 52 of 59 benchmark problems (88.1 %), on the training set.

1 INTRODUCTION

Symbolic Regression (SR) can automatically discover a set of functional forms that accurately fit experimental data, resulting in much more interpretable and concise equations compared to a DNN Zhang et al. (2024). Unlike traditional regression, SR can attain the most appropriate form of equation and the parameters of these equations without requiring a predefined functional form Dong & Zhong (2025). SR has emerged as a research hotspot by substantially reducing the labor-intensive effort required to uncover physical laws Angelis et al. (2023). Udrescu and Tegmark introduced the Feynman SR database, which consists of 100 problems from *The Feynman Lectures on Physics*, and developed AI Feynman, an SR system that successfully rediscovered the equations for the problems from data Udrescu & Tegmark (2020). La Cava et al. proposed SRBench, a benchmark of 252 diverse regression problems, and used it to evaluate 14 SR methods La Cava et al. (2021). Imai Aldeia et al. extended SRBench into a living benchmark, refining evaluation metrics and incorporating trade-offs among predictive accuracy and model complexity. Cranmer et al. presented a benchmark of real empirical equations designed to reflect scientific discovery from noisy, imperfect experimental data, and developed PySR, an SR tool effective for empirical equation discovery Cranmer (2023). Lemos, P. et al., using the past 30 years' worth of observational data, first trained a graph neural network, then used this network to produce more data to train the SR model, and obtained a result that can be proven to be equal to that of newton's law of gravitation Lemos et al. (2023). Following recent advancements in LLMs, LLMs have also been adopted for symbolic regression (SR) Wang et al. (2025); Xia et al. (2026); Grayeli et al. (2024); Li et al. (2026); Guo et al. (2025); Zhang et al. (2026). Shojaee, P. et al. developed a SR method utilizing the LLM's knowledge and data gener-

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.

ation capacity to discover symbolic forms from data Shojaee et al. (2025a). They also constructed a new standard benchmark for LLM-based SR methods that is designed to prevent memorization of formulas by LLMs, leading to overestimates of LLM-SR-based methods' performance Shojaee et al. (2025b). However, these benchmarks and methods are designed for single-dataset SR which find equations capable of describing only one dataset and cannot find shared patterns across datasets.

In practice, experimental sciences such as physics often require a shared functional form across multiple datasets collected under different experimental parameter settings. Recent work by Russeil et al. proposed multi-view SR (MvSR) Russeil et al. (2024), attempting to obtain a unified analytical formula for multiple experimental datasets. However, MvSR was evaluated on only three dataset collections without known ground-truth formulas, making it difficut to verify the structural correctness of the discovered equations. In addition, when evaluating MvSR on our benchmarks, we were unable to obtain satisfactory results. One possible explanation is that the method uses Operon Burlacu et al. (2020), a genetic programming-based SR framework, to search directly for shared functional forms that fit all datasets simultaneously, which may substantially increase the difficulty of the search. Despite this progress, research on multi-dataset SR remains relatively limited, and there is still a lack of standardized benchmark datasets for systematic evaluation.

To address these limitations, we formulate the Multi-Dataset Symbolic Regression (MDSR) problem as discovering a shared symbolic structure across datasets collected under different experimental configurations. We create an entirely new benchmark suite for MDSR, comprising a synthetic benchmark and a real-world experimental data. Unlike conventional symbolic regression benchmarks, our synthetic benchmark includes test datasets with unseen parameter values, controlled Wasserstein shifts in parameter distributions, and varying noise levels, enabling systematic evaluation of how shared functional forms discovered from training sets generalize under parameter-distribution shifts and observation noise. We further propose two reference solvers, a PySR-based symbolic regression with multi-stage selection symbolic regression and a multi-dataset symbolic regression based on a large language model by extending LLM-SR to handle multiple datasets with varying parameter values.

2 Multi-dataset Symbolic Regression

In experimental sciences such as physics, researchers often conduct multiple experiments under different parameter configurations to identify shared patterns across datasets. Here, we distinguish independent variables from parameters: independent variables vary within each dataset, whereas parameters remain fixed within each dataset but may vary across datasets. We often fix the parameters in an experiment and record the values of the dynamic variables to make a dataset. Then, to find the universal relation between the dynamic variables, we repeat the experiment under different parameter configurations, thereby recording multiple datasets to form a dataset collection. For instance, we may fix the length of a pendulum and measure how its amplitude changes with time to make a dataset. Repeating this experiment with different pendulum lengths then produces a dataset collection.

In other words, if there is a symbolic relation $y = \Phi_\theta(X, C; \theta)$, where $X \in \mathbb{R}^m$ are independent variables and $\theta \in \mathbb{R}^k$ are a set of optimizable coefficients (or, in short, coefficients). The coefficients θ may vary between experiments and are typically estimated by fitting each individual dataset. The experimental configuration parameters (or, in short, parameters) C are determined by the experimental conditions. For a given experiment, they remain constant within the corresponding dataset but may vary across different experiments. However, such configuration parameters are typically not explicitly considered in single-dataset symbolic regression.

For the i-th experiment, let the experimental parameters be denoted by C_i. The corresponding experimental dataset is written as $D_i = \{(X_{ij}, y_{ij})\}_{j=1}^{N_i}$, where j indexes the observations and N_i denotes the number of samples in the dataset. Using this formulation, datasets can be generated under multiple parameter configurations. The resulting dataset collection is $\mathcal{D} = \{D_i\}_{i=1}^{S}$, where S denotes the number of datasets in the collection.

The goal of multi-dataset symbolic regression is to recover a shared functional form that can simultaneously describe all datasets in $\mathcal{D}$. For each candidate shared functional form Φ, the optimizable coefficients θ_i are fitted separately for each dataset. The optimal symbolic structure is therefore

defined as

$$\Phi^* = \underset{\Phi \in \mathcal{F}}{\operatorname{argmin}} \left(L \left(\left\{ \min_{\theta_i} E\left(\Phi_f\left(X_i,\, C_i,\, \theta_i\right), y_i\right) \mid i \in [1 \ldots n] \right\}_{i=1}^{S} \right) \right) \tag{1}$$

where $\mathcal{F}$ denotes the search space of candidate symbolic expressions, E denotes a loss function such as NMSE, and L denotes an aggregation function, such as the mean, used to summarize a candidate equation's performance across multiple datasets. Thus, Eq. 1 aims to identify a shared functional form that captures the underlying mathematical relationship across all datasets while allowing the optimizable coefficients to vary across datasets.

3 Multi-dataset Symbolic Regression Benchmarks

3.1 Real-world Benchmark Dataset

We constructed three real-world benchmark dataset collections from three underdamped-oscillation experiments involving spring-mass systems, the small-angle pendulum, and RLC circuits. Every dataset collection contains eight datasets. Underdamped oscillation is a classic dynamical behavior of second-order linear systems, occurring in a wide range of physical setups. These three systems, as well as other underdamped oscillatory systems, share the following mathematical form:

$$X(t) = Ae^{-\beta t}\cos((\sqrt{\omega_0^2 - \beta^2})t - \phi) \tag{2}$$

The Small-Angle Pendulum We suspended the smartphone from two parallel strings of equal length and allowed it to swing back and forth. The two-string configuration suppressed rotation of the phone about the strings, allowing its motion to approximate that of a simple pendulum. The total mass of the oscillator is $308g$. The angular velocity of the pendulum oscillator is measured using the output of the phone's gyroscope (X-axis), with a sampling frequency of 50 Hz. PhyPhox Csernovszky et al. (2024) is employed to record the sensor data. By varying the length of the string, we repeated the experiment eight times to obtain eight datasets.

The Spring Oscillator In this experiment, the mobile phone is suspended from a spring and used as the oscillator. Its built-in accelerometer is utilized to measure the vertical acceleration at a frequency of 10Hz. PhyPhox Csernovszky et al. (2024) is employed to record the sensor data. By combining different springs to vary the spring constant, we repeated the experiment eight times to obtain eight datasets.

Underdamped RLC Oscillator As shown in Section A.1 of the Appendix, we constructed an RLC underdamped oscillation circuit and a corresponding data acquisition system. Using the SPDT switches in the setup, we selected different numbers of inductors connected in series and capacitors connected in parallel, thereby varying the oscillation period of the RLC circuit. The experiment was repeated eight times, producing eight datasets.

3.2 Synthetic benchmark

Although conducting a real-world experiment for every problem would be ideal, it is impractical to perform experiments and collect data for all problems used to evaluate MDSR methods. Therefore, we construct a new benchmark for multi-dataset symbolic regression based primarily on synthetic datasets. For some of the experiments, the values of the parameters can be measured very accurately, like the mass of an object. We call these parameters specific parameters. Symbolic regression should be able to find the explicit relation between the specific parameters, independent variables, and the target variable. For other experiments, some of the parameters may be hard to measure or not well understood. Symbolic regression should also find the symbolic relation between independent variables and the target variable. These parameters are usually expressed implicitly using coefficients.

Unlike traditional symbolic regression settings, MDSR distinguishes independent variables from dataset-level parameters for two purposes. First, varying parameters across datasets allows us to simulate data collection under different experimental configurations. Second, it enables the construction of test sets whose parameter distributions differ from those of the training sets, providing a direct way to evaluate generalization to unseen configurations.

3.2.1 DATASET METADATA GENERATION

Synthetic data were generated from 59 textbook physics problems (P01–P59) Moebs et al. (2016); Flowers et al. (2019); Newton (1701); Shojaee et al. (2025b); Imai Aldeia et al. (2025). Given the analytical expressions and sampling specifications, the steps are as follows:

To make this benchmark useful for LLM-based symbolic regression methods, we first transform the original textbook equations into less common mathematical forms by rearranging each equation to solve symbolically for one of the original independent variables as the new target variable. This can prevent potential LLM memorization of formulas. This is inspired by Shojaee, P. et al., LLM-SRBench Shojaee et al. (2025b). 25 of the equations directly used the transformed Feynman equations from LLM-SRBench.

We then identify the parameters and independent variables according to their physical meanings and conventional usage, and determine their ranges by taking mathematical singularities and constraints into account.

For each problem, we fixed a parameter configuration, sampled a specified number of independent-variable values, and computed the corresponding target values using the ground-truth function to form one dataset. We then repeated this process with different parameter configurations sampled from the parameter ranges, producing multiple datasets that together form the dataset collection for that problem.

To evaluate the extrapolation capability and robustness of the discovered formulas, we constructed dedicated test sets with parameter-distribution shifts of varying Wasserstein distances from the training set. Here, $\boldsymbol{\theta}_i \in \mathbb{R}^d$ denotes the baseline parameter vector associated with the i-th dataset, sampled from the original parameter distribution and used to generate the corresponding training dataset. The shifted parameter vector is constructed by perturbing $\boldsymbol{\theta}_i$ along a prescribed direction $\boldsymbol{u}$, i.e.,

$$\boldsymbol{\theta}_i^r = \boldsymbol{\theta}_i + r\boldsymbol{u}, \qquad r \in \{r_1, \ldots, r_n\}, \tag{3}$$

where r controls the magnitude of the parameter-distribution shift. In our experiments, $\boldsymbol{u}$ was set to a unite vector, and $r \in \{0, 0.25, 0.4\}$, where $r = 0$ corresponds to the unshifted setting. To evaluate the effect of noise on both training and testing data, we constructed additional noisy datasets, with the noise level L defined as the noise level when noisy data is generated by:

$$y_{noise} = y_{groundtruth}\left(1 + \mathcal{N}(0, L)\right). \tag{4}$$

Noise levels $L \in \{0\%, 1\%, 3\%, 5\%\}$ and $L \in \{0\%, 1\%, 3\%, 5\%, 10\%\}$ were applied to the training and test sets, respectively.

3.2.2 PROMPT DESIGN FOR LLM-BASED SYMBOLICREGRESSION

To prevent the LLM from relying on prior knowledge to directly derive or recall textbook formulas, we used prompts that deliberately omitted problem-specific descriptions. In contrast to LLM-SR, each prompts contains only the names of the independent variables, the target variables, the specific syntax requirements of the output, and the structure of the evaluation function. Hence, apart from the names of the variables, the LLM can only know limited information. The Prompt can be seen in Table 13 in Appendix.

4 MULTI-STAGE SELECTION SYMBOLIC REGRESSION METHOD

Most symbolic regression (SR) tools perform regression separately on individual datasets. To overcome this limitation, we develop MSSR, a PySR-based method with a multi-stage selection strategy for identifying shared functional forms that can simultaneously describe multiple datasets, as illustrated in Figure 1. Specifically, the method proceeds as follows:

(1) Use PySR to perform symbolic regression on each dataset.

(2) Assess the complexity and accuracy of the resulting sets of equations. For each dataset, we select the top L_1 equations based on these criteria and pass them to the next stage. In practice, we select candidate equations with a complexity range from 3 to 40 since most of the equations have complexities within this range.

(3) Replace the coefficients in the candidate formulas with symbolic constants to obtain candidate shared functional forms. As shown in Figure 1, the formula $0.84e^{0.002x_0}\cos(4.24x_0 + 0.20)$ is transformed into $p_0e^{p_3x_0}\cos(p_2x_0 + p_1)$.

(4) Merge the candidate shared functional forms from the same dataset collection. Calculate the similarity between candidate equations, and get rid of redundant ones using SymPy.

(5) For each candidate shared functional form, perform curve fitting using all datasets from the same dataset collection with curve_fit in SciPy, and compute the average fitting error (AvgMSE) with Eq. 6 in the Appendix.

(6) Rank the formulas by their AvgMSE, and select the top n formulas with the lowest AvgMSE as shared functional forms.

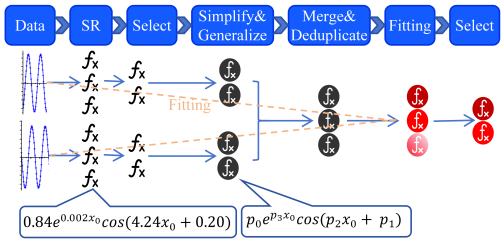


Figure 1: Schematic diagram of multi-stage selection symbolic regression

5 Large-Language-Model-based Multi-Dataset Symbolic Regression

Shojaee et al. proposed LLM-SR, a symbolic regression framework that leverages scientific priors encoded in large language models to discover equations from data Shojaee et al. (2025a). In LLM-SR, candidate equations are generated as python functions. Building on this framework, we adapt LLM-SR to the multi-dataset setting and develop LLM-MDSR.

The original LLM-SR finds one equation that is evaluated on one dataset during each iteration, and uses the NMSE as the criterion for the quality of the equation found. Our goal, however, is to discover unified analytical formulas that explain multiple datasets rather than a single dataset. We therefore regard an equation as successful only if it fits all datasets in the collection well. Therefore, as shown in Figure 2, we changed the scoring criterion. Specifically, each generated expression is evaluated on all datasets in the collection, and the mean of the two largest NMSE values is used as one of the scoring criteria. To discourage the LLM from producing unnecessarily complex equations, we also include equation complexity as an additional criterion. The complexity is computed by first converting the Python functions into symbolic expressions, then simplifying them and calculating their complexity using SymPy. We combine both of these criteria in the following way:

$$score = -\log(\text{avg}(worst2nmse)) - \lambda C \tag{5}$$

where, $wost2nmse$ is the worst two NMSE fitting results, C is the complexity derived by SymPy, and λ is a superparameter with an empirical value of 0.04.

In the original LLM-SR, each island evolves independently and exchanges information every four hours. Specifically, the weaker half of all islands was replaced by the stronger half. However, we found that this mechanism spreads high-quality formulas too slowly and may reduce diversity across islands. To address this issue, we maintain a pool of the top 10 equations discovered so far. During each iteration, with a probability of 20%, two equations are sampled from this pool and provided to the LLM as guidance. Doing this would ensure the efficient spread of the good equations while also preserving the diversity of the island by making them independent 80% of the time.

As shown in Figure 2, for each individual problem, the problem prompt, as shown in Table 13 in appendix , would be used to instruct the LLM to generate the initial formula. The equation is then fitted using all training datasets in the dataset collection, after which we calculate the NMSE. Then, the complexity of the equation and the two worst NMSE values are combined using Eq. 5 to calculate the score. The generated equation is then registered in an island. If the equation is better than any one of the 10-best equations, the top-10 equations list gets updated. Then an island is randomly selected, and equations from its highest-scoring cluster are used to construct an intra-island experience guidance prompt, as shown in Table 11 in the appendix, which is combined with the problem prompt to form a new prompt. In addition to the problem prompt, there is a 20% probability of two formulas being randomly selected from the top-10 equation list to construct a global elite guidance prompt, as shown in Table 12 in the appendix. The LLM then uses the resulting

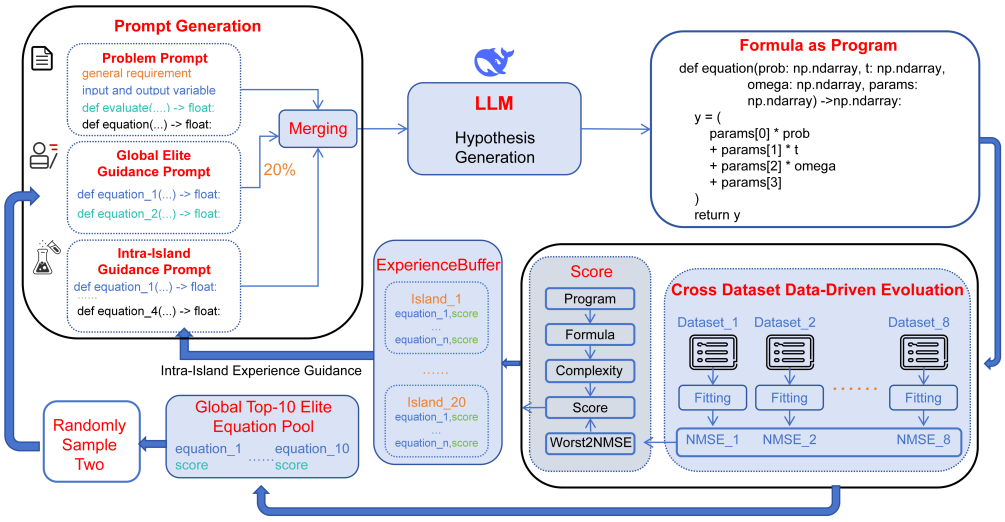


Figure 2: The LLM-MDSR framework, consisting of three main steps: prompt generation, LLM formula generation,cross dataset data-driven evoluation, score calculation, intra-island experience guidance prompt generation and global elite guidance prompt generation.

prompt to generate an improved formula. This iterative process is repeated, allowing equations to continuously evolve and ultimately identify a shared functional form across multiple datasets.

6 Experiment

6.1 Real-World Experimental Benchmark Experiments

6.1.1 Experimental Results of MSSR for Real-World Experimental Benchmark

We apply the MSSR to the pendulum, spring, and RLC damped oscillation dataset collections to derive shared functional forms. We do not specify paramters, allowing them to be expressed implicitly through fitted coefficients, since we lack the experimental apparatus needed to measure them accurately, especially due to the damping coefficient. The discovered shared functional forms are then fitted to the each dataset of the corresponding problem with SciPy's curve_fit, and the average fitting error is calculated through Eq. 6 in the Appendix.

Table 1 presents the shared functional forms successfully discovered by MSSR, along with their complexities and the two lowest average MSEs for the three oscillation problems. Compared with the formulas found by PySR, shown in Table 5 in the Appendix, the formulas with the smallest fitting errors obtained by MSSR are not only more concise but also exhibit a consistent structural form across the three collections. Moreover, some of them can be derived from the theoretical formulas Eq. 2.

Theoretical analysis shows that the three oscillation systems share a common theoretical formula. We therefore merge the datasets from all three collections into a single dataset collection and apply the proposed MSSR method for symbolic regression. Table 2 presents the top three equations with the smallest average fitting errors. As shown in Table 2, MSSR successfully identifies a shared functional form for the combined dataset collection. These results confirm that MSSR is capable of recovering the correct shared functional forms of the underlying damped oscillation dynamics directly from the data from multiple experiments of the same category, without relying on prior physical knowledge. Moreover, it can be readily shown that the discovered formula is consistent with the theoretical formula, as shown in Eq. 2. Figure 6 in the Appendix shows the fitting results for Equation 1 in Table 2, $p_0 p_1^{x_0} \sin(p_2 + p_3 x_0)$, using the first dataset from the RLC oscillation dataset collection. The plots compare the regression curves obtained by MSSR with the original data curves and the curves with the smallest regression error obtained by PySR alone. These figures

Table 1: Shared functional forms obtained by MSSR for the pendulum, spring, and RLC dataset collections.

Dataset	No.	Equation	AvgMSE	Complexity
Pendulum	1	$p_0(p_1 + p_2^{x_0}) \sin(p_3 + p_4 x_0)$	3.163×10^{-4}	18
	2	$(p_0 + p_1 e^{p_2 x_0}) \sin(p_3 + p_4 x_0)$	3.223×10^{-4}	21
Spring	1	$p_0 e^{p_1 x_0} \sin(p_2 + p_3 x_0)$	9.600×10^{-4}	17
	2	$p_0 p_1^{x_0} \sin(p_2 + p_3 x_0)$	9.600×10^{-4}	18
RLC	1	$p_0^{(p_1+p_2 x_0)^{p_3}} \cos(p_4 + p_5 x_0)$	4.059×10^{-5}	24
	2	$p_0[p_1 + \cos(p_2 + p_3 x_0)] e^{p_4 x_0}$	4.121×10^{-5}	19
	3	$p_0 e^{p_1 x_0} \sin(p_2 + p_3 x_0)$	4.139×10^{-5}	17

Table 2: The shared functional forms found by MSSR for a combined dataset collection comprising three underdamped oscillation problems.

No.	Equation	AvgMSE
1	$p_0 p_1^{x_0} \sin(p_2 + p_3 x_0)$	6.872×10^{-4}
2	$p_0 (p_1^{x_0})^{p_2} \sin(p_3 + p_4 x_0)$	7.108×10^{-4}
3	$p_0 e^{p_1 x_0} \sin(p_2 + p_3 x_0)$	7.176×10^{-4}

show that the regression curves of the shared functional form discovered by MSSR closely match both the original data and the best-fit curves obtained by PySR alone.

6.1.2 Comparison with Multi-view Symbolic Regression(MvSR)

To the best of our knowledge, except for our proposed method, MvSR Russeil et al. (2024) is the only existing symbolic regression method designed to regress shared functional forms from multiple datasets. We use MvSR to identify a unified formula from the real-world experimental benchmark. In the experiment, we used the code provided by the author on https://github.com/erusseil/ MvSR-analysis. The unary and binary operators were configured the same as in PySR, with 5,000 iterations and a population size of 3,000. Since MvSR can only generate one equation at a time, we ran it six times to obtain six equations for a fair comparison. As shown in Table 6 in the Appendix, MvSR fails to attain the correct equation form.

6.1.3 Minimum Number of Iterations Required for MSSR to Identify the Correct Formula

We reduced the iterations of PySR in MSSR to only three rounds and obtained shared functional forms for the three dataset collections using MSSR, as shown in Table 7 in the Appendix. It can be readily shown that Formula No. 1 in Table 7 is consistent with the theoretical one in Eq. 2. This demonstrates that, with only three iterations of PySR, MSSR is able to discover a shared functional form for the combined dataset collections from three underdamped oscillation problems. In contrast, with the same limited number of iterations, most individual datasets fail to yield meaningful formulas when regressed independently using PySR. This demonstrates that MSSR has a low requirement for the quality of symbolic regression on individual dataset: as long as a good formula is found in one dataset, it can be immediately identified and adopted as the unified analytical formula for the entire datasets. Therefore, MSSR can improve the efficiency of formula discovery by reducing the number of iterations.

6.2 Experiments on the Synthetic Benchmark

6.2.1 Results on Noise-Free Training Dataset

We conducted three experiments on the noise-free training dataset. In one experiment, specified parameters were provided to MSSR during PySR symbolic regression, representing a scenario in which the experimental parameters can be freely adjusted and accurately measured. In the other two

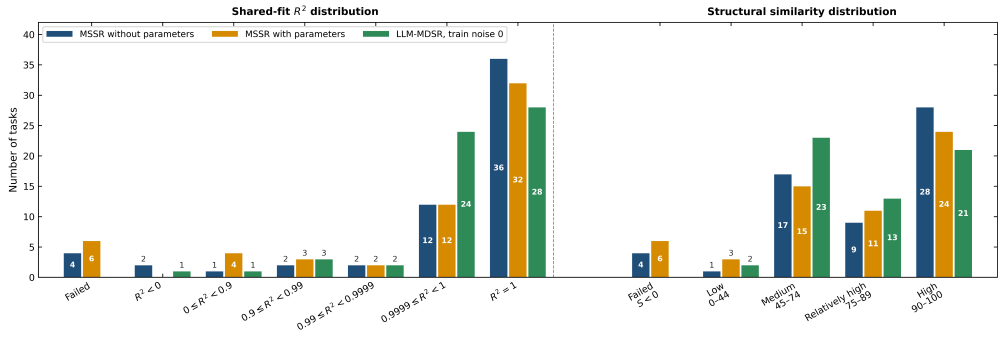


Figure 3: Distribution of R^2 and Structural Similarity Levels on the Noise-Free Training Dataset With and Without Specified Parameters. Failure denotes cases in which MSSR failed to generate any valid shared functional forms.

experiments, no specified parameters were provided, and MSSR and LLM-MDSR were evaluated. Figure 3 shows the distribution of R^2 values and structural similarity levels on the training datasets, both with and without specified parameters. When no specified parameters were provided, MSSR discovered shared functional forms with $R^2 = 1$ for 36 of the 59 benchmark problems (61.0%) with $R^2 \geq 0.9999$ for 48 problems (81.3%), and with $R^2 \geq 0.99$ for 50 problems (84.7%). For LLM-MDSR, 28 of the 59 problems (47.4%) achieved $R^2 = 1$. Although fewer problems reached exact recovery, a larger proportion, 52 of 59 problems (88.1%), achieved $R^2 \geq 0.9999$, compared with 48 of 59 problems (81.3%) for MSSR. With specified parameters, MSSR identified the shared functional form achieving $R^2 = 1$ for for 32 of the 59 benchmark problems (54.2%), $R^2 \geq 0.9999$ for 44 problems (74.6%) and $R^2 \geq 0.99$ for 46 problems (77.9%). These results are all lower than those obtained by MSSR without specified parameters.

We used GPT-5.6 Sol to evaluate structural similarity between the discovered and data-generating equations. Although its scores may be imperfect and underestimate MSSR, they provide useful reference points; see the Appendix for details. As shown in Figure 3, when no specified parameters were introduced, GPT-5.6 Sol reported high structural similarity scores for 28 of the 59 equations (47.4%) using MSSR and for 21 of the 59 equations (35.5%) using LLM-MDSR. When MSSR was applied with parameters, 24 equations (40.6%) received high structural similarity scores.

MSSR without the specified parameters achieves better performance than with specified parameters, probably due to the latter's relation being harder to find in the search space. Specifically, optimizable coefficients can achieve a low MSE through numerical optimization alone, whereas specified parameters generally require discovering a more exact symbolic form, which is much less likely.

6.2.2 Result under Noise in the Training and TestDatasets

To evaluate the generalization ability of the shared functional forms discovered by MSSR and LLM-MDSR when no specific parameters were introduced, we first obtained shared functional forms from training datasets with noise levels $L_{train} \in \{0\%, 1\%, 3\%\}$. Each discovered shared functional form was then fixed, and only its dataset-specific coefficients were refitted to test datasets with $L_{test} \in \{0\%, 1\%, 3\%, 5\%, 10\%\}$. The results are shown in Figure 4.

Figure 4 shows that as training noise increases, we do see a drop in performance across multiple testing sets with different noises. As the noise level increases in the test sets, the success rate at $R^2 \geq 0.9999$ decreases across all methods. Moreover, across all training noise levels, the success rates of the equations discovered by LLM-MDSR and MSSR decline at a rate similar to that of the original data-generating equations, shown in purple. This suggests that the decrease in success rate is mainly caused by the increasing uncertainty in the data. Overall, MSSR achieved lower R^2 values than LLM-MDSR, partly because MSSR, both with and without specified parameters, failed to discover valid shared functional forms or produced $R^2 < 0$ for six problems, whereas LLM-MDSR did so for only one. Another possible reason is that LLM-MDSR can obtain some prior knowledge from variable names, whereas MSSR cannot.

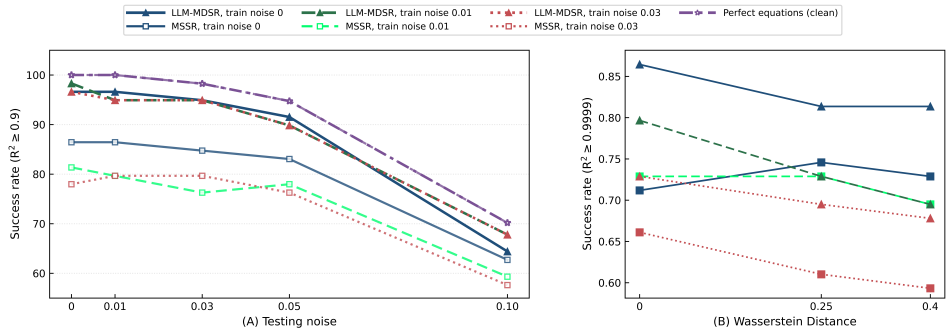


Figure 4: Success rates of MSSR and LLM-MDSR in achieving $R^2 \geq 0.9$ on the training and test datasets across different noise levels, and $R^2 \geq 0.9999$ on the test datasets across different Wasserstein distances. The x-axis denotes the noise level in (A) and the Wasserstein distance in (B), while the y-axis shows the task success rate, defined as the percentage of equations satisfying the thresholds specified in the legend.

6.2.3 Results on Test Set at Different Wasserstein Distance

To evaluate the structural generalization ability of MSSR and LLM-MDSR under shifts in the parameter distribution, we fixed the shared functional forms discovered from training datasets with noise levels $L \in \{0, 0.01, 0.03\}$. We then refitted their dataset-specific coefficients to test datasets with Wasserstein distances $r \in \{0, 0.25, 0.40\}$ from the training distribution. The results are shown in Figure 4. Overall, except for MSSR with zero training noise, the success rate of achieving $R^2 \geq 0.9999$ decreases as the Wasserstein distance increases.

6.2.4 Synthetic Benchmark Validity Discussion

Figure 4 shows that LLM-MDSR achieves higher success rates across different noise levels and Wasserstein distances. Figure 3 further shows that LLM-MDSR is less likely to fail. However, the same figure also indicates that its structural similarity scores are lower than those of MSSR. From this, we may infer that the LLM in LLM-MDSR is not simply reciting or directly deriving equations from textbook formulas; otherwise, it would have achieved higher structural similarity. Although the LLM has the names of the variables, the equation itself is transformed into an uncommon form, and the prompt provided no additional information about the original problem. This made it hard for the LLM to simply search for the equation or deduce the equation itself. In addition, we change the experimental parameters to make a testing set that is statistically different from the training set to measure its generalization capacity and made a standard prompt for the LLM-MDSR. Taken together, these results demonstrate the potential of the proposed synthetic benchmark to serve as a standard benchmark for multi-dataset symbolic regression.

7 Conclusion and Discussion

In this work, we formulated the multi-dataset symbolic regression (MDSR) problem and constructed two benchmarks for MDSR: a real-world experimental benchmark collected from three underdamped oscillators and a synthetic benchmark generated from 59 transformed physics-derived problems under varying conditions. We proposed two reference methods, a PySR-based Multi-stage Selection Symbolic Regression method (MSSR) and a Large-Language-Model-based Multi-Dataset Symbolic Regression method (LLM-MDSR), and validated the effectiveness of both the methods and the benchmarks. Future work will expand the benchmark by introducing synthetic dataset collections from domains such as chemistry, biology, and finance, while also incorporating a broader range of real-world experimental datasets suitable for multi-dataset symbolic regression. Currently, MSSR does not utilize intrinsic knowledge in the data, such as units or data shapes. LLM-MDSR also leaves considerable room for improvement. For example, combining local evolutionary search within each island with LLM-guided evolution could reduce the number of LLM calls while improving search efficiency and diversity.

AI USE STATEMENT

In this work, we used generative AI tools as part of LLM-MDSR to generate candidate formulas expressed as Python functions and to assist in evaluating structural similarity between discovered formulas and the corresponding data-generating equations. These AI-generated formulas and evaluations were used only as components of the proposed method and experimental analysis.

We did not use generative AI tools to generate research ideas, design the benchmark, fabricate or modify data, conduct the core numerical experiments, or make scientific claims without author verification. Generative AI tools were also used to assist with language polishing and improving the clarity of the manuscript. All AI-assisted outputs were reviewed by the authors. The generated Python functions were executed and evaluated using our experimental pipeline, and the structural-similarity assessments were inspected as reference scores rather than treated as definitive ground truth.

We take responsibility for the final content of this work, including all text, claims, code, experimental results, and artifacts produced with the aid of generative AI.

REFERENCES

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

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon c++: An efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pp. 1562–1570. Association for Computing Machinery, 2020. doi: 10.1145/3377929.3398099. URL https://doi.org/10.1145/3377929.3398099.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023. URL https://arxiv.org/abs/2305.01582.

Z. Csernovszky, M. Hömöstrei, and K. Kurucz. Study of damped oscillations using phyphox and arduino controlled hall-sensor. *Journal of Physics: Conference Series*, 2693(1):012004, 2024. doi: 10.1088/1742-6596/2693/1/012004. URL https://doi.org/10.1088/1742-6596/2693/1/012004.

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

Paul Flowers, Klaus Theopold, Richard Langley, and William R. Robinson. *Chemistry 2e*. OpenStax, 2019. URL https://openstax.org/details/books/chemistry-2e.

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.

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

Gabriel S. Imai Aldeia, Hao Zhang, Geoffrey Bomarito, Miles Cranmer, Anderson Fonseca, Bogdan Burlacu, William G. La Cava, and Fábio O. de França. Call for action: Towards the next generation of symbolic regression benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2529–2538. Association for Computing Machinery, 2025. doi: 10.1145/3712255.3734309. URL https://doi.org/10.1145/3712255.3734309.

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.

Pedro Lemos, Natasha Jeffrey, Miles Cranmer, Shirley Ho, and Peter Battaglia. Rediscovering orbital mechanics with machine learning. *Machine Learning: Science and Technology*, 4 (4):045002, 2023. doi: 10.1088/2632-2153/acfa63. URL https://doi.org/10.1088/2632-2153/acfa63.

Yuchen Li, Handing Wang, and Yaochu Jin. Evosr-llm: Evolutionary symbolic regression guided by large language models. *IEEE Transactions on Evolutionary Computation*, 2026.

William Moebs, Samuel J. Ling, and Jeff Sanny. *University Physics*, volume 1–3. OpenStax, Rice University, 2016. URL https://openstax.org/details/books/university-physics-volume-1.

Isaac Newton. Scala graduum caloris. *Philosophical Transactions*, 22(270):824–829, 1701. doi: 10.1098/rstl.1700.0082. URL https://doi.org/10.1098/rstl.1700.0082.

E. Russeil, F. O. de França, K. Malanchev, B. Burlacu, E. Ishida, M. Leroux, C. Michelin, G. Moinard, and E. Gangler. Multiview symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 961–970. Association for Computing Machinery, 2024. doi: 10.1145/3638529.3654087. URL https://doi.org/10.1145/3638529.3654087.

Parshin Shojaee, Kourosh Meidani, Sarthak Gupta, Amir Barati Farimani, and Chandan K. Reddy. LLM-SR: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=m2nmp8P5in.

Parshin Shojaee, Nam Nguyen, K. Meidani, Amir Barati Farimani, K. D. Doan, and Chandan K. Reddy. LLM-SRBench: A new benchmark for scientific equation discovery with large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55325–55359. PMLR, 2025b. URL https://proceedings.mlr.press/v267/shojaee25a.html.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631. URL https://doi.org/10.1126/sciadv.aay2631.

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

Zeyu Xia, Jun Zhu, and Dong Yan. Functionevolve: Structure-guided symbolic regression with llms. *arXiv preprint arXiv:2606.07704*, 2026.

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. Llm-meta-sr: In-context learning for evolving selection operators in symbolic regression. *IEEE Transactions on Evolutionary Computation*, 2026.

M. Zhang, S. Kim, P. Y. Lu, and M. Soljačić. Deep learning and symbolic regression for discovering parametric equations. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11): 16775–16787, 2024. doi: 10.1109/TNNLS.2023.3297978. URL https://doi.org/10.1109/TNNLS.2023.3297978.

A Appendix

A.1 Underdamped RLC Oscillator

As shown in Figure 5 in the Appendix, inductors and capacitors are used to construct an RLC underdamped circuit. To accurately record the voltage variation over time, a measurement device was constructed using an ESP32, ADS1115, and TL081,as shown in Figure 5. The Hall-effect sensor used in previous studies Csernovszky et al. (2024) was not adopted because it has low accuracy, is easily affected by environmental conditions, and only measures voltage indirectly. Since the

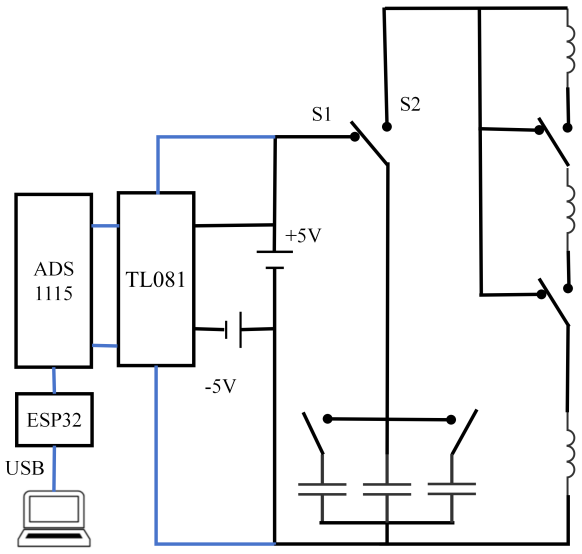


Figure 5: Experimental setup of the RLC underdamped oscillation circuit and the data acquisition system.

ADS1115 may introduce small leakage currents into the circuit, potentially distorting the capacitor's voltage, a TL081 voltage follower was added to provide isolation and ensure accurate measurements. This ensures that the voltage presented to the ADC is an exact replica of the capacitor voltage, thereby preserving measurement accuracy. It also ensures that there will be no unwanted current in the RLC circuit due to its gigantic. Using the SPDT switches, we selected different numbers of inductors connected in series and capacitors connected in parallel, thereby varying the oscillation period of the RLC circuit.

A.2 Evaluation criteria

Candidate shared functional forms for MSSR are selected based on AvgMSE, which is defined as follows:

$$\text{AvgMSE} = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\left(y_{ij} - \Phi(\mathbf{x}_{ij};\boldsymbol{\theta}_i)\right)^2. \tag{6}$$

We also use the mean coefficient of determination, denoted by $\overline{R^2}$, to evaluate the overall performance of a discovered equation across multiple datasets in a problem:

$$\overline{R^2} = \frac{1}{S}\sum_{i=1}^{S}\left[1 - \frac{\sum_{j=1}^{N_i}\left[y_{ij} - \Phi(\mathbf{x}_{ij};\boldsymbol{\theta}_i)\right]^2}{\sum_{j=1}^{N_i}\left(y_{ij} - \bar{y}_i\right)^2}\right], \qquad \bar{y}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} y_{ij}. \tag{7}$$

Here, S is the number of datasets in the collection, N_i is the number of observations in the i-th dataset, $(\mathbf{x}_{ij}, y_{ij})$ is the j-th observation in that dataset, Φ denotes the candidate shared functional form, and $\boldsymbol{\theta}_i$ denotes the dataset-specific coefficients obtained by fitting Φ to the i-th dataset. For simplicity, we hereafter refer to the mean $\overline{R^2}$ value simply as R^2.

A.3 Experimental Configuration

A.3.1 Experimental Configuration for MSSR

This experiment uses the Python-based open-source symbolic regression tool PySR (version 1.5.9) for regression. The reported formula complexity and regression error are taken directly from the output of PySR, and the tool's configuration parameters are shown in Table 3.

Table 3: PySR configuration for MSSR.

Setting	Real-world	Synthetic
Binary ops.		$+, -, \times, \div,$
Unary ops.	$\cos, \sin, \exp, \sqrt{\cdot}$	$\cos, \sin, \exp, \sqrt{\cdot}, \log, \tanh, \text{atanh}, \text{acos}$
Iterations	100	400
Population size	50	40
Max nodes	40	50
Optimizer		Levenberg–Marquardt

Table 4: LLM-MDSR configuration.

Setting	Value
Iteration	1000
Number of islands	200
LLM name and version	DeepSeek-v4-flash
Size of elite equation pool	10
Elite equation sample size	2
Probability of using elite-equation prompt	20%

A.3.2 Experimental Configuration for LLM-MDSR

LLM-MDSR was implemented by modifying the publicly available LLM-SR source code on https://github.com/deep-symbolic-mathematics/LLM-SR. Its detailed configuration is presented in Table 4.

A.4 Additional Experiments

A.4.1 Experimental Results of PySR for Real-World Benchmark Dataset

We perform symbolic regression separately on the data from each individual experiment using PySR to obtain its corresponding regression formula. The regression error loss of each resulting formula is then recorded. Table 5 reveals several limitations of PySR. Even for experiments of the same type, the resulting formulas vary significantly, and many do not align with physical intuition.

A.4.2 The Structural Similarity Evaluation

Although the equations found by symbolic regression might be mathematically right, it is still possible to express it in a non-standard form. Therefore, different expressions do not necessarily imply an incorrect analytical relation. However, manually inspecting all of the equations and proving their mathematical correctness is tiresome and is subject to non-objective judgement. Hence, to increase

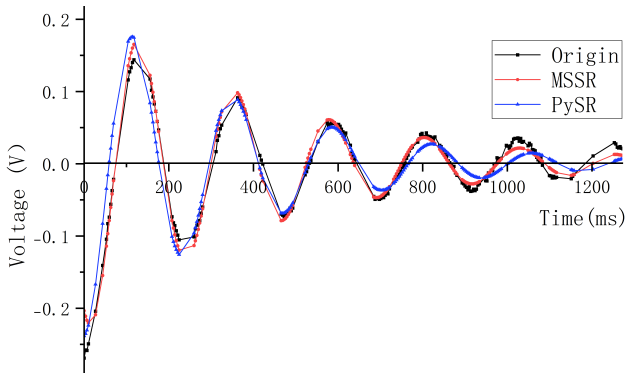


Figure 6: Comparison of regression curves obtained by PySR and MSSR with original data for the RLC circuit.

Table 5: The two formulas with the lowest loss discovered by PySR for each dataset collection.

Sample	Equation	Loss
Spring1	$2.229566(0.9814985^{0.16313623x_0})^{1.1720469}\cos(8.75574753x_0 + 5.44302534)$	3.42×10^{-3}
	$2.229566(0.9814985^{0.18056753x_0})\cos(8.75574753x_0 + 5.44302534)$	3.77×10^{-3}
Spring2	$-1.3757831(0.99652505^{x_0})\sin(8.7520344x_0 - 0.46065983)$	1.02×10^{-3}
	$-0.9968428^{x_0}\sin(8.7525867x_0 - 0.51184926)$	5.58×10^{-3}
Pendulum1	$\sin\left(\frac{0.54373837\sin(4.6098614x_0 - 1.69987164)}{1.0231901^{1.0930687x_0 - 0.73064494}} + 0.57940745\right)$	4.79×10^{-4}
	$\frac{0.54373837\sin(4.6098614x_0 - 1.69987164)}{1.0231901^{1.0930687x_0 - 0.3687468}} + 0.57886916$	4.80×10^{-4}
Pendulum2	$0.37066284(0.97777015^{x_0} + 0.09776211)\sin(2.4348734x_0 - 1.65820966)$	1.01×10^{-4}
	$0.36975053(0.9846499^{x_0})\sin(2.4344996x_0 - 1.65786781)$	1.92×10^{-4}
RLC1	$-0.2921444e^{-2.241475x_0^{0.6062856}}\cos[x_0(0.92931217x_0 + 26.854767) - 0.38011467]$	1.14×10^{-5}
	$-0.23230167(0.096422985^{x_0})\cos(27.800209x_0 - 13.03372)$	5.39×10^{-5}
RLC2	$-0.11169767e^{(6.652285\times10^{-13})x_0 - 1.330338x_0}\sin(19.786373x_0 + 1.0972509) - 0.0017898591$	1.54×10^{-5}
	$-(-0.003333423 + \frac{0.029812329}{x_0 + 0.12841907})\sin(19.85194x_0 + 1.1311423)$	4.91×10^{-5}

Table 6: Shared functional forms discovered by MvSR regression for the real-world dataset collections.

Type	Formula	Loss
Pendulum	$\frac{\cos(AX_1 - B)}{-CX_1 + \cos(DX_1 + e)}$	4.450×10^{-3}
	$\cos(AX_1 + B)$	9.043×10^{-3}
	$-\sin(AX_1 + B)$	9.394×10^{-3}
Spring	$\exp(\exp(\sin(\cos(AX_1))) - \cos(\exp(-BX_1)))$	3.210×10^{-1}
	$-\sin\left(\frac{\sin(AX_1 - B)}{-CX_1 - D}\right)$	2.928×10^{-1}
	$\frac{\sin(AX_1 + B)}{-CX_1 - D}$	2.748×10^{-3}
RLC	$X_1\exp(-BX_1)$	1.682×10^{-3}
	$-AX_1 + (\exp(-BX_1))^{-C}$	1.535×10^{-3}
	$(\exp(A^{BX_1}))^{-CX_1^2}$	1.579×10^{-3}

the efficiency of the evaluation of structural similarity in a subjective manner, we use LLM to evaluate the similarity between equations discovered by symbolic regression and the original ones. We used GPT-5.6 Sol to give a score for every equation to describe the structural similarity of the discovered equation to that of the original one, with 100 being a perfect match. The prompt used for LLM-based scoring is provided as the Table 10. The LLM evaluated the structural similarity according to the criteria presented in Table 8. It is important to note that GPT-5.6 Sol may itself be subject to bias and may be influenced by certain characteristics of the equations, and is therefore not always reliable, as discussed below.

A.4.3 Relation Between Structural Similarity Score and NMSE for MSSR

Figure 7 shows that the relation between structural similarity score and NMSE on the noise-free training dataset for MSSR without specified parameters. The results indicate that lower NMSE values weakly correspond to higher structural similarity scores. This suggests that MSSR can recover the general structure of the underlying equations, but its structural recovery performance is less consistent than its numerical fitting performance. One possible reason is that the lowest-average-MSE selection criterion we used may not be sufficiently sensitive to distinguish between equations with similar numerical performance but different structures. It is also worth noting that some of the equations failed to fit; these will be discussed later in this paper.

A.4.4 Relation between Structural Similarity and Formula Complexity for MSSR

Figure 8 demonstrates the distribution of the structural similarity and simplified formula complexity for MSSR on the training datasets, with and without specified parameters. We can see that the MSSR is capable of discovering formulas with high R^2 values that describe data from multiple datasets, regardless of the complexity of the original formula. Interestingly, the structural similarity score remains high for simple formulas but decreases as the equations become more complex. This

Table 7: the shared functional forms for the combined collections obtained by MSSR after reducing PySR iterations to three rounds.

No.	Type	Equation	AvgMSE
1	Pendulum	$p_0(e^{p_1 x_0})^{p_2} \cos(p_3 + p_4 x_0)$	1.351×10^{-3}
2	Pendulum	$p_0 x_0 e^{p_1 x_0} \sin(p_2 + p_3 x_0)$	1.816×10^{-2}
3	Pendulum	$x_0 \sin(p_0 e^{p_1 x_0}) \sin(p_2 + p_3 x_0)$	1.816×10^{-2}
4	Spring	$p_0 + p_1 p_2^{x_0} \sin(p_3 x_0)$	6.115×10^{-2}
5	Spring	$p_0 p_1^{x_0} \sin(p_2 x_0)$	6.116×10^{-2}
6	Pendulum	$p_0 p_1^{x_0} x_0 (p_2 x_0 + \cos(p_3 x_0))$	6.969×10^{-2}

Table 8: Criteria for different levels of structural similarity.

Score	**Level**	**Equivalence interpretation**	**Rule**
90–100	High	The formulas may be strictly equivalent, or the core structure can be exactly recovered by parameter specialization.	High structural consistency
75–89	Relatively High	The main structures are similar, but non-negligible topological or redundant differences remain.	Relatively high structural consistency
45–74	Medium	Some variable dependencies or local operators are preserved, but the formulas are not equivalent overall.	Partial structural correspondence
0–44	Low	Key variable dependencies or core operator topologies are clearly inconsistent.	Structural non-equivalence

suggests that, for complex equations, MSSR is more likely to find alternative equations that describe the data equally well than to recover the exact form of the underlying equations.

A.4.5 Accuracy Change Relative to the Noise-Free Baseline across Training-Set Noise Levels for MSSR

Figure 9 shows the relation between train set noise and accuracy change relative to the noise-free baseline for MSSR. Where 0 is perfectly matching the results of the clean dataset; a positive number means the noisy dataset performs better than clean dataset, and vice versa. We again see that, on a clean test set, or when testing noise is 0, adding noise in the training set does not change the typical fitting average R^2 scores significantly over the range of training noise we are investigating. What is interesting is that we actually see an improvement on noisy testing sets when the training set also got some noise. The improvement got better as testing noise increases, implying that adding some noise to the training set actually makes the equations in general more versatile. We think it is because the noise forced the MSSR to find more versatile equations. Therefore, for the equations successfully found, adding training noise actually makes them more versatile. Still, the benefit is small, and adding noise is obviously getting diminishing returns from 0.01 to 0.03, and we can also observe that as noise increases, the number of equations producing poor results also increases, and dramatically, with a lot of equations getting extraordinarily bad average R^2 scores (values below 0). This is more evident as you increase the relative noise level; for instance, our experiment using a noise level of 5% shows that 21 out of 30 failed to find an average score less than 0. When noise level reach 10%, not a single equation was found.

A.5 Discussion on the Validity of LLM-Based Formula Structural Similarity Scores.

To verify how effective GPT-5.6 Sol is in providing structural similarity scores of shared functional forms, we chose some of thescores to verify their validity manually. Specifically, we chose equations with an average R^2score of 1 but with a similarity score lower than 90 and those that have similarity scores higher than 90 but haveR^2 scores lower than 0.9. We found 17 for MSSR without

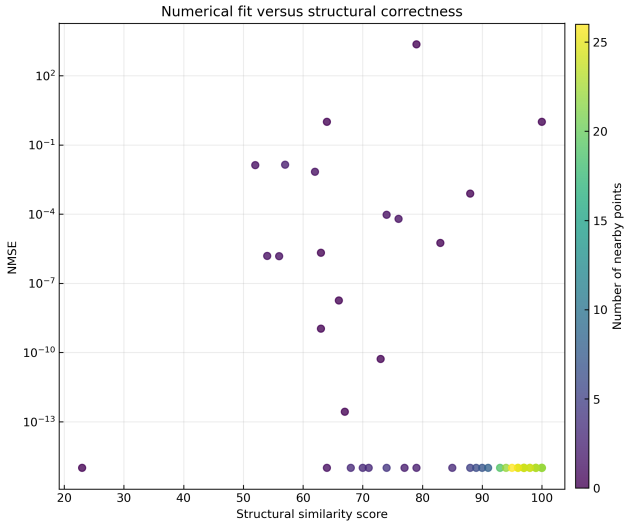


Figure 7: The relation between Structural similarity score and NMSE for MSSR without specified parameters. The x-axis is structural similarity derived through SymPy. The y-axis is the NMSE score on a logarithmic scale. The number of points in the vicinity is described bythe colorbar.

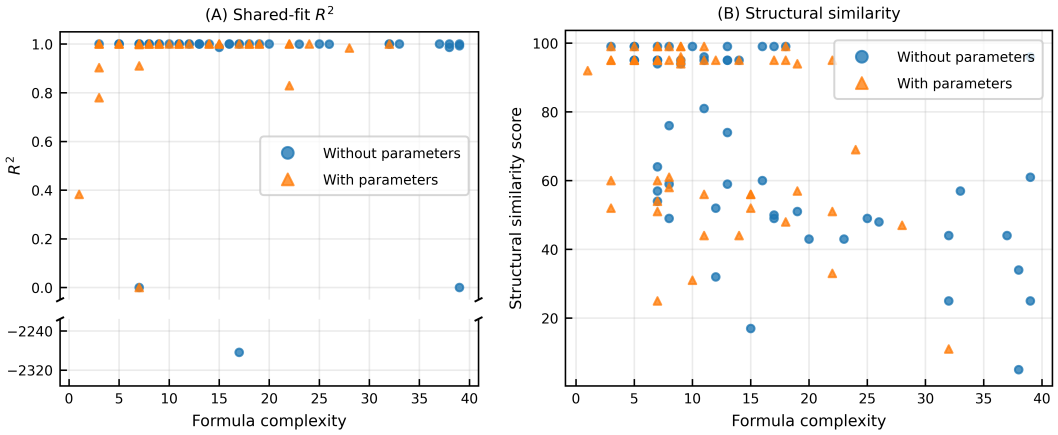


Figure 8: The distribution of the structural similarity and formula complexity. The x-axis is formula complexity derived from SymPy; the y-axis is R^2 (A) and the similarity score (B)

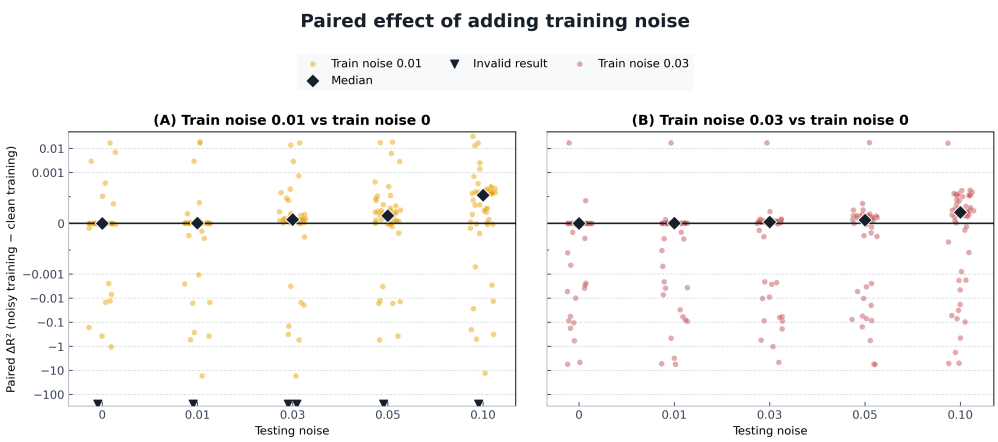


Figure 9: Accuracy Change Relative to the Noise-Free Baseline at Different Training-Set Noise Level. The plot on the left is for the train set with noise 0.01; the plot on the right is the train set with noise 0.03

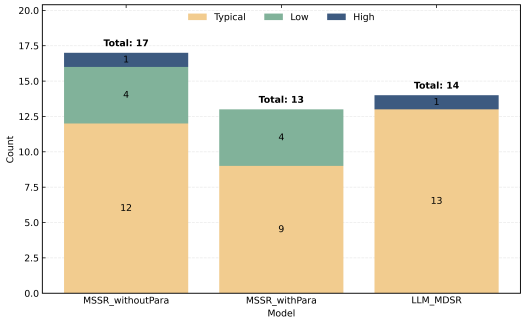


Figure 10: Agreement between LLM-Based Structural Similarity Evaluation and Human Assessment (High: Overestimation; Low: Underestimation; Typical: Consistent)

parameters, 13 for MSSR with parameters and 14 for LLM_MDSR. As can be shown in Figure 10, LLM underestimated 4 out of 17(23.5%) equations of MSSR without parameters, whereas there it overestimated only 1(5.8%). One notable example of the equations MSSR without parameters found was $p_0\,[E_n^{p_1}\,(p_3\omega^{p_4}+p_2)]^{p_5}$ the equation that generated it is $2\sqrt{\frac{E_n}{m(\omega^2+\omega_0^2)}}$. Here, ω_0 and m are parameters represented implicitly by the fitted coefficients, whose specific values are determined separately for each dataset. One can easily see that they are equivalent, but the LLM only gave a score of 77. The LLM's evaluation of MSSR with parameters underestimated 4 out of 13 (30.7%) equations. The LLM-MDSR meanwhile only have one overestimated equation. Specifically, it found $\frac{p_0\exp(p_1E_n)}{p_2+E_n}+p_3E_n$where the equation that generated it is $\frac{E_nh}{2\pi J_z g_J mom}$. Here, g_J are parameters; h, J_z, mom and π are constants. The equation suffered extreme fitting losses, almost to infinity, yet the LLM gave a high structural similarity score of 89. All in all, the MSSR was generally underestimated, where the LLM-MDSR was more stable, though sometimes overestimated. Hence, although at times GPT-5.6 Sol is not reliable, it still gave a relatively subjective answer.

A.6 Failure Analysis of MSSR and LLM-MDSR

We analyzed the cases in which MSSR and LLM-MDSR failed to generate any valid shared functional forms. For MSSR, failure generally occurred when none of the candidate formulas satisfied the selection criteria. For LLM-MDSR, failure was defined as the case in which the generated formula yielded an $R^2 = -\infty$ on the training datasets. Table 9 lists the problems for which MSSR and LLM-MDSR failed to generate any valid shared functional forms.

Table 9: Problems where MSSR and LLM-MDSR failed to produce a valid shared functional form.

MSSR				LLM-MDSR			
WP	N0	N1	N3	N0	N1	N3	Problem
	✓						Relativistic mass increase $v = c\sqrt{1 - m_0^2/m^2}$
✓							Newton cooling $t = -\frac{1}{k} \ln \frac{T - T_a}{T_0 - T_a}$
✓	✓		✓				Bernoulli pressure-speed $v = \sqrt{2(P_0 - P)/\rho}$
✓	✓	✓					Lorentzian resonance $f = f_0 + \gamma\sqrt{A/(S - B_0) - 1}$
✓	✓	✓					Radioactive decay $t = \ln(N_0/N)/\lambda$
✓							Uniform acceleration velocity $t = (v - v_0)/a$
✓						✓	Simple harmonic oscillator $t = (\arccos(x/A) - \phi)/\omega$
			✓				Hooke law with force offset $x = (F_0 - F)/k$
			✓				Light Scatter $E = \frac{\sqrt{3}\sqrt{P_{wr}/(c\epsilon)}(\omega_0^2 - \omega^2)}{2\sqrt{\pi}\omega^2 r}$
				✓			Zeeman Energy-Level Model $B = \frac{E_n h}{2\pi J_z g_J mom}$
					✓		Magnetic-Field Magnitude Energy Model $B_x = \sqrt{\frac{E_n^2}{mom^2} - B_y^2 - B_z^2}$
6	4	1	4	1	1	1	

Table 10: Prompt for LLM-Based Structural Similarity Evaluation

```
Please evaluate the GenerationFormula (the formula used to generate the data) and formula (the
formula obtained through symbolic regression) fields in the uploaded spreadsheet.  Be sure to
consider ParameterRange, FixedConstantValues, IndependentVars, and Target.  ParameterRange specifies
the parameters whose values may vary, while FixedConstantValues specifies the constants used in
GenerationFormula.  Assign a quantitative score ranging from 0 to 100, where 100 indicates the highest
structural similarity and 0 indicates the lowest.  Additionally, classify the structural similarity
into one of four levels:  High, Relatively High, Medium, or Low, and provide a justification for the
evaluation.
```

A.6.1 Failure Analysis of MSSR

Table 9 contains problems where MSSR failed to produce a valid shared functional form. A few things might be causing the catastrophic failure of these problems. One of the reasons might be that our MSSR currently lacks the ability to automatically block invalid values that can cause out-of-domain coefficients. This means that some equations get discarded early on in the evolutionary process, as a result letting inferior equations survive the evolution process, or even when MSSR does manage to find the correct symbolic structure, it might still fail on certain datasets and end up having extremely bad R^2 scores due to just a few outliers. Examination of some of the results (notably the uniform-acceleration velocity and radioactive decay) proves that the averaging MSE across datasets can make the scores susceptible to extreme outliers. It is possible in a future study to use a more proficient aggregation method.

A.6.2 Failure Analysis of LLM-MDSR

From Table 9, we can see that LLM-MDSR have exactly one problem that failed to find any shared functional form for every one of the noise levels respectively. Specifically, there are two that are related to Electromagnetism, one related to harmonic oscillators. It is possible that in the harmonic oscillator case, the use of arccos is so uncommon in physics that the LLM never guessed the operator arccos, hence the low result. Still, one can see that LLM-MSSR has far fewer failed problems than MSSR.

Table 11: Intra-Island Experience Guidance Prompt

```
instruction_prompt = (
"You are a symbolic regression assistant discovering mathematical "
"laws for scientific systems.\n\n"
"Several versions of a mathematical function are provided.  "
"The LAST versioned function is the function that you must improve.\n\n"
"Rewrite the mathematical body of that final function and return "
"exactly ONE COMPLETE Python function definition.\n\n"
"Prefer equations using no more than 4 dataset-specific parameters "
"unless additional parameters are clearly necessary.\n\n"
"Requirements:\n"
"- The formula complexity must not exceed "
f"{MAX_FORMULA_COMPLEXITY} mathematical operations.\n"
"- Copy the function name exactly from the LAST function in the prompt.\n"
"- Copy the complete function signature exactly, including all arguments "
"and type annotations.\n"
"- Do not use the name or signature of an earlier function version.\n"
"- Improve or replace the mathematical expression in the body.\n"
"- Use params[i] as free numerical parameters when appropriate.\n"
"- Use only NumPy-compatible mathematical operators and functions.\n
"- The function must contain executable Python code.\n"
"- The function must contain an explicit return statement.\n"
"- Never return a docstring-only or empty function.\n"
"- Never use undefined variables.\n"
"- Do not include explanations or Markdown code fences.\n"
"- Do not use control-flow constructs inside the function:  "
"if/elif/else, conditional expressions, for or while loops, "
"comprehensions, try/except/finally, with, match/case, "
"break, continue, or raise.\n\n"
"- Division and reciprocal relationships are allowed and should be"
"considered when physically meaningful.\n"
"- Independent variables may appear in the denominator.\n"
"- Do not avoid division merely because it may introduce singularities,"
" the supplied data already satisfy the stated domain restrictions.\n"
"Output format:\n"
"- Output exactly one complete Python function.\n"
"- Use the name and complete signature of the LAST function in the prompt.\n"
"- Put the function body on lines following the function header.\n"
"- End the function with an explicit return statement.\n"
)
```

Table 12: Global Elite Guidance Prompt.

```
# GLOBAL REFERENCE FORMULAS:
# The formulas below are randomly selected from the current
# globally high-scoring formulas across all islands.
# They are references for exploration, not templates that must be copied.
# Analyze them internally to identify useful directions, including:
# - effective mathematical structures and operator combinations;
# - useful parameterizations and variable interactions;
# - structures shared by several high-scoring formulas;
# - weaknesses that may still be improved.
#
# Generate a genuinely new and improved formula following these directions.
# Do not copy any original candidate or best formula.
# Do not produce an algebraically equivalent version of any listed formula.
# Renaming variables, reordering terms, factoring, expanding, or replacing
# expressions with equivalent identities does not count as improvement.
# The new formula must contain a meaningful structural or mathematical change.
# Analyze internally and output only the new indented function body.
#
```

Table 13: Prompt for Problem P20

```
"""
Task: discover one shared
physical law from multiple
datasets.
All datasets describe the same
underlying physical process.
Input:
prob: transition probability
t: interaction time
omega: driving angular frequency
Output:
target: target physical quantity
Find ONE mathematical expression
target = f(prob, t, omega, params)
that can describe every dataset.
SHARED-STRUCTURE REQUIREMENT:
All datasets MUST use exactly the
same symbolic equation structure.
For dataset k:
target_k = f(
prob,
t,
omega,
params_k
)
where the mathematical form of f
is identical for every dataset,
but params_k may have different
numerical values.
Therefore:
SAME:
mathematical operators
functional form
arrangement of variables
parameter positions
interpretation of parameters

MAY DIFFER:
numerical values of params
Do NOT:
- create different formulas for
different datasets
- use dataset IDs
- use conditional branches for
individual datasets
- hard-code numerical values for
particular datasets
- fit each dataset with a
different symbolic structure
- replace the physical relation
with unnecessarily complicated
high-order polynomial
approximations
The equation may contain
physically meaningful nonlinear
relationships
between prob, t, and omega.
Possible dataset-dependent
physical quantities may be
represented using
params[i]. These parameters are
optimized independently for each
dataset.
These are free numerical
parameters, not additional
independent variables.
Use params[i] for dataset-specific
numerical parameters.
The evolved equation MUST:
- use prob, t, and omega as
independent variables
- use params[i] for adjustable
dataset-specific parameters
- return a valid NumPy-compatible
prediction
- contain executable code
- contain an explicit return statement
- not return None
Prefer the simplest physically meaningful shared
equation that explains
all datasets.
""" import numpy as np
from scipy.optimize import
minimize
MAX_NPARAMS = 10
INVALID_LOSS = 1.0e100
VARIANCE_EPS = 1.0e-15
@evaluate.run
def evaluate(data: dict) ->
float:
"""
Evaluate one shared equation
structure on one dataset.
The symbolic equation is shared
across all datasets.
Numerical params are optimized
independently for this dataset.
"""
inputs = np.asarray(
data["inputs"],
dtype=float,
)
outputs = np.asarray(
data["outputs"],
dtype=float,
).reshape(-1)
prob = inputs[:, 0]
t = inputs[:, 1]
omega = inputs[:, 2]
output_variance = np.var(outputs)
def objective(params):
try:
with np.errstate(all="ignore"):
prediction = equation(
prob,
t,
omega,
params,
)
if prediction is None:
return INVALID_LOSS
prediction = np.asarray(
prediction,
dtype=float,
)
prediction = np.broadcast_to(
prediction,
outputs.shape,
)
if not
np.all(np.isfinite(prediction)):
return INVALID_LOSS
residual = prediction - outputs
mse = np.mean(
residual ** 2
)
if not np.isfinite(mse):
return INVALID_LOSS
# If the target is almost
constant, NMSE is unstable.
if output_variance <= VARIANCE_EPS:
return float(mse)
# Normalize by target variance so
datasets with different
# numerical scales contribute
comparably.
nmse = mse / output_variance
if not np.isfinite(nmse):
return INVALID_LOSS
return float(nmse)
except Exception:
# evaluate() is called separately for each dataset.
# Therefore every dataset gets its own optimized
parameter values,
# while equation() keeps exactly the same
symbolic structure.
initial_params = np.ones(
MAX_NPARAMS,
dtype=float,
)
try:
result = minimize(
objective,
initial_params,
method="BFGS",
)

final_loss = float(result.fun)

except Exception:
return None
if (
not np.isfinite(final_loss)
or final_loss ¿= INVALID_LOSS
):
return None

# LLM-SR maximizes the returned score.
return -final_loss @equation.evolve
def equation(
prob: np.ndarray,
t: np.ndarray,
omega: np.ndarray,
params: np.ndarray,
) -> np.ndarray:
"""
Shared equation:
target = f(prob, t, omega, params)
Keep ONE common symbolic structure
for every dataset.
Only the numerical values of
params may differ between
datasets.
Args:
prob:
Transition probability.
t:
Interaction time.
omega:
Driving angular frequency.
params:
Dataset-specific numerical
parameters.
Returns:
Predicted target quantity.
IMPORTANT:
Replace the equation body with a
candidate mathematical law.
The function MUST contain
executable NumPy-compatible code
and MUST explicitly return the
prediction.
"""
y = (
params[0] * prob
+ params[1] * t
+ params[2] * omega
+ params[3]
); return y
```