SURFACEBENCH: CAN SELF-EVOLVING LLMS FIND THE EQUATIONS OF 3D SCIENTIFIC SURFACES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Equation discovery from data is a core challenge in machine learning for science, requiring the recovery of concise symbolic expressions that govern complex physical and geometric phenomena. Recent approaches with large language models (LLMs) show promise in symbolic regression, but their success often hinges on memorized formulas or overly simplified functional forms. Existing benchmarks exacerbate this limitation: they focus on scalar functions, ignore domain grounding, and rely on brittle string-matching metrics that fail to capture scientific equivalence. We introduce SURFACEBENCH, the first comprehensive benchmark for symbolic surface discovery. SURFACEBENCH comprises 199 surfaces across 18 categories of symbolic complexity, spanning explicit, implicit, and parametric forms. Each task includes ground-truth equations, variable semantics, and synthetically sampled 3D data. Many surfaces are novel or synthetically constructed to resist memorization, yet remain grounded in scientific domains such as fluid dynamics, robotics, electromagnetics, and geometry. To evaluate discovery quality, we pair symbolic checks with geometry-aware metrics (Chamfer, Hausdorff, Earth Mover's), ensuring that models are judged by the structures they reproduce rather than their algebraic syntax. Our experiments reveal that state-of-the-art frameworks, while occasionally successful on specific families, fail to generalize consistently across all representations. SURFACEBENCH thus establishes a challenging and diagnostic testbed for equation discovery research, enabling principled progress in symbolic generalization, data-driven induction, and geometry-aware reasoning with LLMs. We release the code at the link: `https://anonymous.4open.science/r/surfacebench-183B`

1 INTRODUCTION

Recovering the governing equations of complex 3D surfaces from sampled data is a foundational challenge across science, engineering, and design. In real-world applications, ranging from medical imaging and materials science to robotics and aerospace, surfaces are often observed only as dense point clouds or measurement grids, without access to their underlying generative equations. Symbolic representations offer interpretable and compact descriptions of geometric structures, capturing physical, biological, or structural laws that can be generalized, simulated, and optimized. Yet, despite progress in scientific modeling and equation discovery, symbolic recovery of surface equations has remained underexplored due to challenges such as multi-output coupling, implicit constraints, and nonlinear transformations.

Surfaces are not abstract constructs but central to high-stakes domains where precise geometry determines outcomes. In physics and engineering, they define optical wavefronts, fracture boundaries, and aerodynamic flow separation. In biology and medicine, they describe cortical folds, arterial structures, and protein energy landscapes. In robotics and navigation, they capture terrains, obstacle boundaries, and safe operating zones. Across these fields, surface equations are not aesthetic artifacts—they are functional tools for analysis, optimization, and design.

Crucially, surfaces exhibit properties that scalar functions cannot capture: (i) *Multi-output coupling:* Physical constraints tie multiple observables together, e.g., toroidal confinement surfaces in plasma physics enforce algebraic identities across (x, y, z) that cannot be modeled independently; (ii) *Latent coordinate systems:* Many laws are compact only in reparameterized coordinates (e.g., spherical

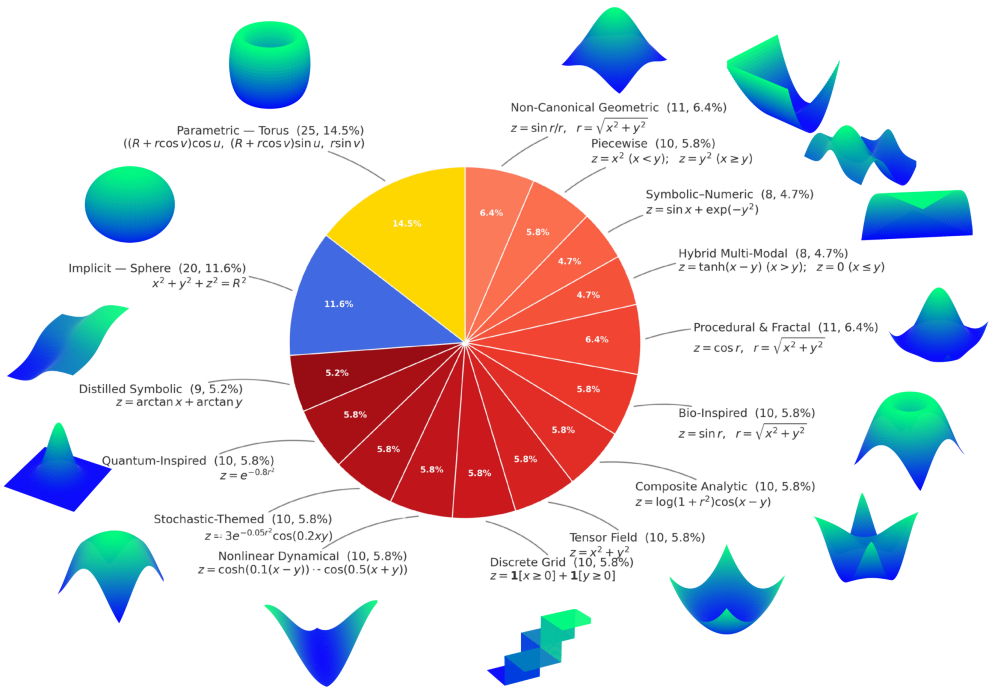

Figure 1: Overview of the 18 surface categories in SurfaceBench. The central pie chart shows the distribution of 199 benchmark surfaces across categories, with percentages indicating their relative proportions. Surrounding the chart are representative 3D renderings and canonical equations for each category, including parametric (e.g., torus), implicit (e.g., sphere), symbolic-numeric hybrids, fractal/procedural forms, bio-inspired surfaces, and higher-order analytic composites. Together, these categories span diverse structural complexities, functional forms, and scientific motivations, highlighting the breadth and richness of SurfaceBench as a diagnostic benchmark for symbolic surface discovery.

charts in planetary motion or angular systems in molecular orbitals), requiring inference of hidden variables for interpretability; (iii) *Topological and geometric richness:* Real-world structures exhibit holes, folds, discontinuities, and symmetries, as in protein folding, crack propagation, or fluid vortices (phenomena beyond simple scalar relations); (iv) *Invariant equivalence classes:* Distinct symbolic forms, such as implicit vs. parametric equations of a sphere, can yield the same geometry. Evaluating at the surface level thus mirrors how scientific laws are judged (i.e., by the structures they reproduce, not merely by algebraic syntax). These properties make symbolic surface discovery both more demanding and more realistic than traditional symbolic regression benchmarks, which focus on scalar outputs (e.g., $y = f(x)$). While such benchmarks have been valuable, they do not reflect the multivariate, structurally constrained, and geometrically meaningful equations required for real-world science and engineering. Bridging this gap demands benchmarks that explicitly test reasoning, compositionality, and invariance handling in recovering surface equations.

To address this need and create a more challenging evaluation setting for recent LLM-based equation discovery frameworks (Shojaee et al., 2025b), we introduce SURFACEBENCH, a benchmark suite for symbolic surface discovery in scenarios where memorization fails but structure-aware reasoning is essential. SURFACEBENCH is scientifically grounded, drawing from forms in optics, mechanics, biology, and geometry, while deliberately altering parameterizations to resist trivial recall. Unlike scalar benchmarks, it encodes multi-output coupling, latent coordinate systems, and symbolic non-uniqueness, providing a richer and more realistic testbed. Ultimately, SURFACEBENCH functions as a diagnostic tool: solving it requires reasoning about invariances, compositional structures, and geometry-aware consistency—the core capabilities needed for interpretable scientific AI. Spanning 18 diverse categories of surfaces, from non-canonical algebraic forms to fractals, symbolic–numeric hybrids, and topologically rich manifolds, it captures both synthetic complexity and real-world relevance (see Figure 1).

The primary contributions of this work are as follows:

- **First benchmark for symbolic surface discovery.** We introduce the first systematic benchmark of 199 explicit, implicit, and parametric surfaces across 18 categories, establishing a new setting for equation discovery beyond scalar functions.
- **Geometry-aware evaluation protocol.** We design evaluation metrics (Chamfer, Hausdorff, Earth Mover's) that assess equivalence at the surface level, moving beyond brittle string/AST comparisons and aligning with how scientists judge analytic models.
- **Stress-testing reasoning over memorization.** By including transformed, constraint-driven, and non-canonical surface families, SURFACEBENCH explicitly distinguishes true structural reasoning from rote equation recall, enabling rigorous diagnosis of LLM generalization.
- **Scientifically motivated and interpretable.** Each surface family is derived from real scientific phenomena in physics, materials science, biology, and engineering, ensuring that benchmark tasks are not only challenging but also grounded, interpretable, and practically relevant.

2 SURFACEBENCH: DESIGN AND MOTIVATION

Symbolic discovery of scientific surfaces introduces challenges that extend well beyond conventional symbolic regression. Unlike benchmarks focused on recovering single-output functions $y = f(x)$, surface equations demand reasoning over multivariate structure, hidden parameterizations, and representational non-uniqueness. SURFACEBENCH is explicitly designed to foreground these challenges, serving as a diagnostic testbed that separates rote memorization from genuine symbolic reasoning.

2.1 MULTI-OUTPUT COUPLING

Surface equations are inherently multivariate. In parametric form, latent coordinates (u, v) map into coupled outputs (x, y, z), where components share terms and satisfy algebraic constraints. For example, in a torus, $(x, y, z) = ((R + r\cos v)\cos u,\ (R + r\cos v)\sin u,\ r\sin v)$ satisfies $x^2 + y^2 = (R + r\cos v)^2$. Successful recovery thus requires identifying the underlying coupled mechanism rather than treating outputs as independent regressions. This enforces structural consistency, compositional reuse of symbolic components, and robustness against trivial memorization.

2.2 LATENT COORDINATE SYSTEMS

Many scientific surfaces admit compact laws only in reparameterized coordinates. For instance, spheres or molecular orbitals are concise in spherical harmonics or angular charts but appear algebraically entangled in Cartesian space. In practice, observations are provided as samples in $\mathbb{R}^3$, requiring inference of hidden variables that simplify the representation. Discovery therefore entails uncovering usable coordinate charts and respecting reparameterization invariances, rather than piecemeal fitting of local composites.

2.3 SYMBOLIC NON-UNIQUENESS

Multiple algebraically distinct expressions may describe the same surface. A sphere can be expressed implicitly as $x^2+y^2+z^2 = R^2$ or parametrically as $(R\sin\phi\cos\theta,\ R\sin\phi\sin\theta,\ R\cos\phi)$. Trigonometric identities, affine transformations, and coordinate reparameterizations further multiply equivalent forms. Conventional evaluation via string or AST matching penalizes these valid alternatives, while pointwise regression error rewards overfit surrogates that interpolate samples without capturing structure. SURFACEBENCH addresses this by evaluating at the geometry level, treating hypotheses as equivalent if they induce the same surface up to admissible transformations.

2.4 GEOMETRY-AWARE EVALUATION

To operationalize geometry-level equivalence, SURFACEBENCH renders candidate and ground-truth surfaces, samples them as point clouds, aligns them under similarity transforms, and computes object-space distances. We adopt Chamfer Distance (capturing average fidelity) and Hausdorff Distance (capturing worst-case deviation). These measures respect symbolic non-uniqueness and

emphasize the true scientific objective: discovering laws that reproduce geometric structure, not merely equations that minimize sample error.

2.5 Expressivity and Richness

Finally, SurfaceBench probes aspects of reasoning that scalar benchmarks cannot. Surfaces may exhibit folds, holes, discontinuities, or fractal oscillations, requiring expressive forms that combine periodic, conditional, or compositional elements. Categories include non-canonical algebraic surfaces, fractal and procedural manifolds, symbolic–numeric hybrids, and topologically rich parametric embeddings. This diversity ensures that models must reason about symmetry, separability, and invariance, rather than memorize canonical formulas. In doing so, SurfaceBench provides a rigorous and scientifically grounded setting for advancing symbolic discovery.

2.6 Problem Setup

We introduce **SurfaceBench**, a benchmark designed to evaluate LLM-based methods for *data-driven symbolic discovery of 3D surfaces*. As shown in Figure 3, a surface discovery task is defined as follows: given a dataset $\mathcal{D}$ of sampled points, the objective is to recover a symbolic hypothesis h that compactly represents the underlying surface with both high geometric fidelity and scientific plausibility. The hypothesis h may take one of three standard forms: an *explicit* function $z = f(x, y)$, an *implicit* relation $g(x, y, z) = 0$, or a *parametric* mapping $\mathbf{r}(u, v) = (x(u, v), y(u, v), z(u, v))$. The discovery process mirrors how human scientists iteratively refine models: LLMs generate candidate forms, evaluate them against data- and geometry-based feedback, and progressively refine hypotheses.

Formal Objective. Given $\mathcal{D} = \{(x_i, y_i, z_i)\}_{i=1}^{N}$ (or parametric samples $\{(u_j, v_j, \mathbf{r}_j)\}$, or signed-distance/occupancy evaluations), together with context $\mathcal{C}$ specifying domains and evaluation rules, the goal is to find a symbolic h such that

$$h^\star \;=\; \arg\min_{h \in \mathcal{T}} \; \mathcal{L}_{\text{geo}}(\mathcal{D}; h, \mathcal{C}) \;+\; \lambda\, \mathcal{L}_{\text{simp}}(h),$$

where $\mathcal{L}_{\text{geo}}$ measures geometric fidelity (e.g., Chamfer/Hausdorff distance, level-set consistency), and $\mathcal{L}_{\text{simp}}$ encourages interpretability via symbolic parsimony (e.g., operator count or expression depth). By spanning explicit, implicit, and parametric hypotheses—and evaluating across both in-domain and out-of-distribution regimes—SurfaceBench stresses challenges absent in scalar equation discovery, such as multi-output coupling, topological variation, and representational non-uniqueness.

Dataset Construction. The benchmark surfaces are curated through a multi-stage process:

1. *Base sampling:* We manually select seed surfaces from diverse scientific and mathematical domains (analytic geometry, implicit physical models, and parametric shapes from graphics).
2. *Compositional rewrites:* We apply symbolic transformations such as nesting trigonometric/exponential functions, coordinate reparameterizations (e.g., polar or affine transforms), and controlled combinations of functional components Shojaee et al. (2025b), yielding families of related but distinct variants.
3. *Filtering and validation:* Each candidate surface is manually visualized, with discontinuous, degenerate, or unnecessarily convoluted examples discarded. The retained corpus contains only smooth, interpretable, and scientifically meaningful surfaces.

This procedure ensures both variety and quality: surfaces are either drawn directly from scientific exemplars or generated through controlled perturbations that preserve interpretability.

Representation Categories. Surfaces in SurfaceBench are organized by their analytic representation:

- **Explicit:** functions $z = f(x, y)$, i.e., height fields over the xy-plane.
- **Implicit:** zero-level sets of $g(x, y, z) = 0$.

- **Parametric:** vector maps $\mathbf{r}(u, v) = (x(u, v), y(u, v), z(u, v))$ with two parameters.

These forms correspond to canonical surface definitions in geometry. Notably, explicit surfaces are special cases of implicit ones (e.g., $g(x, y, z) = f(x, y) - z$). By requiring models to handle all three conventions, SurfaceBench evaluates not only symbolic regression ability but also representation robustness.

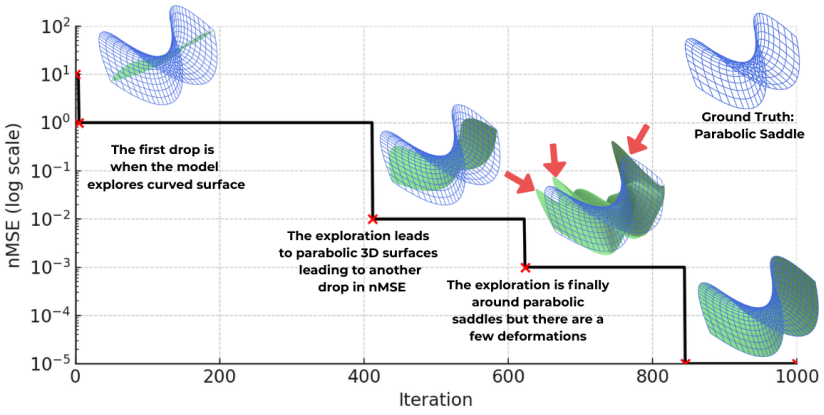

Figure 2: Iterative discovery of a parabolic saddle surface. nMSE decreases as the model progresses from flat to curved surfaces, then to parabolic forms, and finally converges to the true saddle, with intermediate candidates showing minor deformations.

3 Experimental Setup

3.1 Benchmark Methods

We evaluate SurfaceBench using a representative set of symbolic regression frameworks that capture both classical and LLM-driven approaches. Together, these baselines span evolutionary search, neural guidance, and language-model–based discovery.

LaSR (Grayeli et al., 2024). LaSR is a neural-guided symbolic regression system that combines probabilistic grammars with learned search heuristics. It produces closed-form expressions directly and has demonstrated success on large-scale scalar regression tasks. For SurfaceBench, we adapt LaSR to handle multi-output targets and implicit formulations.

SGA (Holland, 1975; Koza, 1992). The Simple Genetic Algorithm (SGA) is a classical genetic programming method that evolves symbolic expressions through mutation and recombination. While computationally expensive and prone to expression bloat, it provides a reference point for how traditional search-based methods perform on symbolic surface discovery.

LLM-SR (Shojaee et al., 2025a). LLM-SR leverages pretrained large language models to generate symbolic programs conditioned on sampled data. Candidate equations are produced through prompting and then ranked by their fit to observed samples, offering a direct measure of how well LLM priors alone can recover surface laws.

OpenEvolve (Sharma, 2025). OpenEvolve is an open-ended discovery framework that couples LLM-based proposal generation with evolutionary refinement. By iteratively updating a pool of candidate programs based on novelty and fitness, it provides a stronger exploratory baseline than direct prompting.

PySR (Cranmer, 2023). PySR is a widely used evolutionary symbolic regression library in scientific applications. It searches over mathematical expression trees using mutation, crossover, and simplification rules. As a mature and well-optimized non-LLM baseline, PySR provides an important point of comparison for evaluating LLM-based methods.

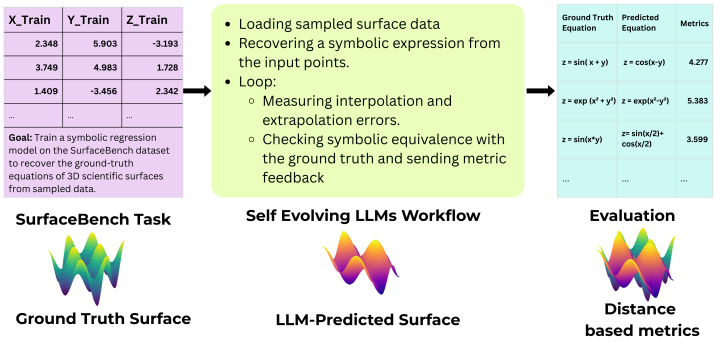

Figure 3: Overview of the SurfaceBench evaluation pipeline. Given sampled 3D surface data, self-evolving LLMs recover candidate symbolic equations. Predicted expressions are compared to ground truth using both regression-style errors (NMSE), symbolic equivalence checks, and geometry-aware distance metrics: Chamfer and Hausdorff.

3.2 Evaluation Metrics

Prior equation discovery benchmarks typically rely on scalar regression metrics (e.g., normalized mean squared error on $y = f(x)$) or exact string/AST matches between candidate and ground-truth formulas. Such measures are insufficient for surfaces, where multiple algebraically distinct forms (explicit, implicit, parametric) may describe the same geometry. To address this, SurfaceBench introduces a domain-specific evaluation suite that combines *geometry-aware distances*, *symbolic equivalence*, and *scale-invariant regression error*.

Geometry-aware metrics. Candidate and ground-truth surfaces are rendered as point clouds and aligned under a similarity transform. We adopt two standard object-space distances:

$$\text{Chamfer}(P,Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2^2 \;+\; \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|_2^2, \tag{1}$$

$$\text{Hausdorff}(P,Q) = \max\left\{\sup_{p\in P}\min_{q\in Q}\|p-q\|_2,\;\sup_{q\in Q}\min_{p\in P}\|q-p\|_2\right\}. \tag{2}$$

Chamfer Distance (CD) captures average geometric fidelity, while Hausdorff Distance (HD) highlights worst-case deviations. These metrics assess equivalence at the *surface level*, avoiding penalties for symbolic non-uniqueness.

Symbolic accuracy. Following LLM-SRBench Shojaee et al. (2025b), we measure *Symbolic Accuracy* using an LLM-based equivalence check that incorporates algebraic simplifications and parameter rescalings. This provides a principled but flexible way to judge whether the recovered equation is symbolically equivalent to the ground truth.

Normalized Mean Squared Error (NMSE). To maintain comparability with scalar-function benchmarks, we include NMSE as a regression-style measure of pointwise fit:

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

We also report a threshold-based accuracy score:

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

Together, these metrics capture complementary aspects of performance: NMSE links to prior symbolic regression practice, Symbolic Accuracy measures algebraic recovery, and Chamfer/Hausdorff distances ensure that discovered equations faithfully reproduce the intended scientific surfaces.

4 RESULTS

Table 1: Comparison of LLM-based equation discovery methods on SURFACEBENCH. Performance is reported across explicit, implicit, and parametric forms using symbolic accuracy (SA), normalized mean squared error (NMSE), Chamfer distance, and Hausdorff distance. SGA, LaSR, and LLM-SR cannot handle parametric equations by design, which is indicated with dashes. While LLM-based methods achieve partial success on explicit and implicit surfaces, they consistently lag behind the PySR baseline, highlighting the difficulty of generalizing to structurally complex equations.

	Explicit				Implicit				Parametric			
Base LLM	**SA ↑**	**NMSE ↓**	**Chamfer ↓**	**Hausdorff ↓**	**SA ↑**	**NMSE ↓**	**Chamfer ↓**	**Hausdorff ↓**	**SA ↑**	**NMSE ↓**	**Chamfer ↓**	**Hausdorff ↓**
SGA												
GPT4o-mini	0.20	2.86	8.26	16.53	0.16	1.38	2.96	6.72	–	–	–	–
Llama-3.1-8B	0.10	3.73	9.82	18.48	0.12	1.43	3.01	7.81	–	–	–	–
Qwen-7B	0.10	4.29	5.19	13.25	0.10	1.57	3.05	8.26	–	–	–	–
LaSR												
GPT4o-mini	0.35	2.87	4.30	11.00	0.06	3.48	5.04	10.07	–	–	–	–
Llama-3.1-8B	0.30	3.21	3.68	14.21	0.10	2.81	4.67	9.78	–	–	–	–
Qwen-7B	0.30	2.96	4.18	12.84	0.06	3.08	4.92	10.06	–	–	–	–
LLM-SR												
GPT4o-mini	0.30	2.57	7.08	24.17	0.10	1.54	2.20	5.25	–	–	–	–
Llama-3.1-8B	0.20	2.62	7.44	29.29	0.13	1.74	3.01	9.05	–	–	–	–
Qwen-7B	0.25	2.38	6.99	28.83	0.02	1.61	1.51	10.6	–	–	–	–
OpenEvolve												
GPT4o-mini	0.50	0.98	2.69	4.88	0.07	0.71	1.85	4.96	0.80	0.30	1.22	2.01
Llama-3.1-8B	0.40	0.99	3.17	5.08	0.02	0.99	2.96	5.02	0.65	0.32	1.62	2.57
Qwen-7B	0.40	1.25	3.23	5.82	0.04	0.92	2.35	5.92	0.75	0.32	1.63	2.06
PySR												
Non-LLM	0.65	0.0011	0.13	0.41	0.10	0.6138	2.52	5.53	0.70	0.0020	0.70	1.80

Note: SGA, LaSR and LLM-SR methods cannot handle parametric equations due to their design.

The main table 1 reports results over all the surfaces by representation: explicit, implicit, and parametric, with four metrics: Symbolic Accuracy, nMSE, Chamfer, and Hausdorff. Two signals stand out. Explicit surfaces attain the highest Symbolic Accuracy. Implicit surfaces attain the lowest geometric distances, reflected in stronger Chamfer and Hausdorff. Notably, the explicit results reveal that models are recovering the correct structural family but not a geometrically tight instance. Symbolic structure is often right, yet coefficients or latent symmetries remain imperfectly calibrated, which elevates Chamfer and Hausdorff distances despite strong Symbolic Accuracy. This points to a pipeline gap: after structure discovery, a targeted geometric calibration step is needed to lock scale, shift, rotation, and curvature so structural exploration translates to gain in distance based metrics.

Secondly, the implicit category results show the converse pattern. Distance-driven search brings the discovered surface equations close to the target even when the algebraic form is not fully faithful, producing strong Chamfer and Hausdorff metrics with lower Symbolic Accuracy. Together, these adjustments align geometric proximity with algebraic fidelity. Finally, the parametric surface results show consistently good performance across metrics by both non-LLM and LLM based openevovle framework. We conduct a comprehensive ablation to understand the robustness of these methods. As noted previously, the LaSR, SGA and LLM-SR methods do not have algorithmic design for parametric equation discovery. Hence, we do not include the parametric equations for our ablations.

5 ANALYSIS

5.1 OUT OF DOMAIN PERFORMANCE

We define OOD strictly as a range shift in the evaluation grid. If a model is trained on inputs sampled from [-5,5] along each axis, OOD tests use the non-overlapping exterior bands $[-10, -5] \cup [5, 10]$. This isolates extrapolation from interpolation: models must extend learned structure beyond the training support rather than reuse local trends. We keep all other factors—category, representation, and rendering—fixed, and assess geometry in object space using Chamfer/Hausdorff distances after Sim(3) alignment, ensuring that improvements reflect genuine extrapolative fidelity rather than token-level or coordinate artifacts. We report both (i) absolute OOD errors and (ii) generalization gaps (OOD minus in-distribution error) to quantify robustness.

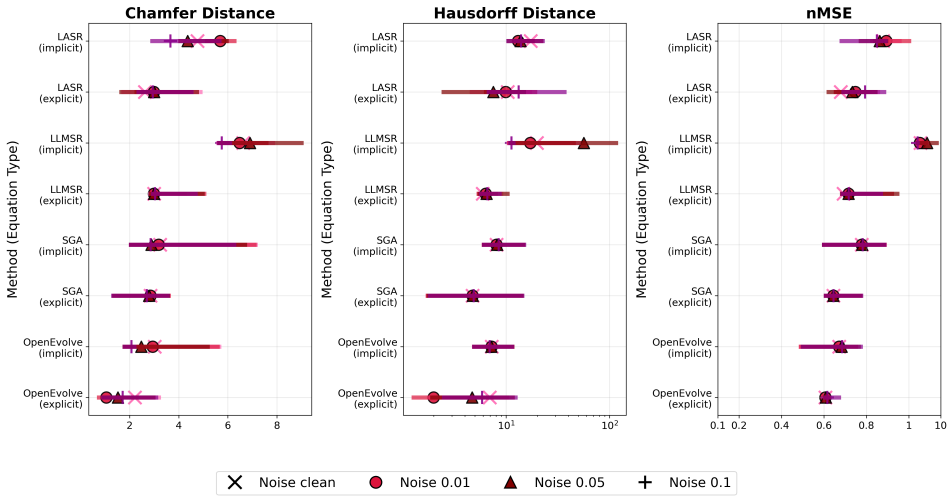

Figure 4: Noise sensitivity across Chamfer Distance, Hausdorff Distance, and nMSE. Performance degrades with increasing Gaussian noise, with OpenEvolve showing the most robustness, especially for explicit surfaces, while LaSR and LLM-SR are more sensitive.

5.2 Noise Sensitivity

To evaluate the robustness of our symbolic regression models under realistic data conditions, we conducted a comprehensive noise sensitivity analysis across two state-of-the-art language models: GPT-4o-mini and LLaMA-3.1-8B. The experiment was designed to systematically assess how model performance degrades when exposed to varying levels of data corruption, simulating real-world scenarios where training data may contain measurement errors, sensor noise, or other sources of uncertainty. We selected 13 representative equations spanning diverse mathematical domains including nonlinear dynamical systems, quantum-inspired surfaces, stochastic processes, and hybrid multi-modal symbolic surfaces, ensuring broad coverage of the symbolic regression problem space. The experimental design employed a factorial approach with three noise levels (1%, 5%, and 10% Gaussian noise). Performance was evaluated using both Chamfer distance and Hausdorff distance metrics on in-domain test data, providing complementary measures of geometric fidelity between predicted and ground truth.

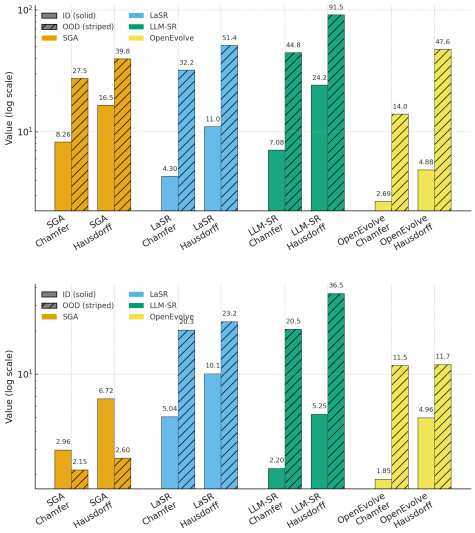

Figure 5: Results of out-of-domain (OOD) performance across explicit and implicit categories of equations.

5.3 Impact of Domain Knowledge

We examine whether lightweight domain priors—e.g., hints about plausible coordinate charts (spherical/cylindrical), conservation or symmetry constraints, or canonical basis functions from optics and mechanics—improve recovery. SurfaceBench is grounded in real scientific forms but perturbs parameters, compositions, and coordinate mappings to thwart rote memorization; thus, the role of priors is to guide structure discovery (e.g., shared factors across outputs, separability, invariances) rather than to supply an answer key. In practice, providing such priors narrows the search space toward mechanisms consistent with the target domain, and when coupled with geometry-aware scoring, prioritizes concise laws that faithfully reproduce the surface over ad-hoc composites that only interpolate samples.

Table 2: Comparison of Normal vs Specialized prompts on Chamfer and Hausdorff distances (↓ better). Δ = Normal − Specialized. Positive Δ indicates improvement.

Method	Explicit						Implicit					
	Chamfer			Hausdorff			Chamfer			Hausdorff		
	Normal	Spec.	Δ	Normal	Spec.	Δ	Normal	Spec.	Δ	Normal	Spec.	Δ
SGA	7.76	6.22	+1.54	16.09	12.22	+3.87	3.01	2.76	+0.25	7.60	6.26	+1.34
LaSR	4.05	4.04	+0.01	12.68	10.04	+2.64	4.88	4.24	+0.64	9.97	9.24	+0.73
LLM-SR	7.17	5.77	+1.40	27.43	20.77	+6.66	2.24	2.07	+0.17	8.30	5.07	+3.23
OpenEvolve	2.93	1.16	+1.77	4.98	4.16	+0.82	2.41	1.82	+0.59	4.99	4.82	+0.17

6 Related Work

Symbolic regression and equation discovery. Classical symbolic regression (SR) methods such as PySR Cranmer (2023) use evolutionary search to fit scalar functions but scale poorly to complex domains and often rediscover shallow forms. Recent work integrates large language models (LLMs) into this process: LLM-SR Shojaee et al. (2025b) and LaSR Grayeli et al. (2024) combine LLM priors with evolutionary search to generate equation skeletons or libraries of reusable components. OpenEvolve Sharma (2025), AlphaEvolve Novikov et al. (2025), and AI Scientist Lu et al. (2024) extend this idea to iterative optimization pipelines that couple LLM reasoning with execution feedback. These advances highlight the promise of LLM-driven discovery, but their evaluation remains focused on single-output scalar functions.

Benchmarks. Datasets such as the Nguyen and Strogatz suites Matsubara et al. (2022); Cava et al. (2021) provide synthetic nonlinear functions, while Feynman-I Udrescu & Tegmark (2020) compiles physics-inspired analytic expressions. LLM-SRBench Shojaee et al. (2025b) recently expanded this space with 239 problems designed to probe generalization beyond memorization. Although influential, these benchmarks are restricted to scalar outputs and explicit forms, and thus cannot test reasoning about multi-output coupling, implicit surfaces, or parametric representations that are essential in real-world scientific contexts.

Geometry and language–geometry models. Surface learning benchmarks in geometry processing focus on reconstruction from point clouds or meshes. Classical techniques such as Poisson reconstruction (Kazhdan et al., 2006) remain widely used, while neural approaches like Points2Surf (Erler et al., 2020) and SALD (Atzmon & Lipman, 2021) leverage implicit neural fields for higher fidelity. Evaluation is typically performed with Chamfer and Hausdorff distances (Fan et al., 2017), and more recent work such as STITCH (Jignasu et al., 2024) introduces topology-aware constraints to capture connectivity and cycles. Meanwhile, emerging LLM–geometry systems (Mews et al., 2025; Li et al., 2025) extend language models to procedural 3D reasoning, but they operate on geometric primitives or latent fields rather than producing interpretable analytic equations.

SurfaceBench fills this gap as the first benchmark for *symbolic surface discovery*, bridging SR and geometry by requiring recovery of explicit, implicit, and parametric forms. It emphasizes multi-output coupling, latent coordinate systems, and representation non-uniqueness, enabling evaluation of both symbolic interpretability and geometric fidelity.

7 Conclusion

We introduce **SurfaceBench**, the first comprehensive benchmark for LLM-driven *symbolic discovery of 3D surfaces*, encompassing **199** tasks across **18** categories and three representation types: *explicit*, *implicit*, and *parametric*. SurfaceBench provides a standardized and geometry-aware evaluation protocol (Chamfer/Hausdorff in object space, alongside Symbolic Accuracy and NMSE) and accommodates diverse hypothesis formats spanning expression strings and executable programs. Extensive experiments with state-of-the-art discovery frameworks and multiple LLM backbones reveal that, despite occasional successes on specific families, no method consistently excels across all representations or evaluation regimes; overall performance remains far from saturation, underscoring substantial headroom for advances in structure discovery, parameter calibration, and invariance handling. We envision that the *SurfaceBench* datasets and evaluation protocol will serve as a common foundation for future research, catalyzing progress in automated discovery of surface equations and deepening our understanding of LLMs' geometric and symbolic reasoning in scientific settings.

REFERENCES

Matan Atzmon and Yaron Lipman. Sald: Sign agnostic learning with derivatives. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=7EDgLu9reQD.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales, 2021. URL https://arxiv.org/abs/2106.06427.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio Oliveira de França, Manuel Virgolin, Yaochu Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. *arXiv preprint*, 2021. URL https://arxiv.org/abs/2107.14351.

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

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Michael Wimmer, and Niloy J. Mitra. Points2surf: Learning implicit surfaces from point cloud patches. *arXiv preprint arXiv:2007.10453*, 2020. URL https://arxiv.org/abs/2007.10453.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 605–613, 2017. doi: 10.1109/CVPR.2017.654.

Amin Grayeli, Anant Sehgal, Octavio Costilla-Reyes, Miles D. Cranmer, and Shibabrat Chaudhuri. Symbolic regression with a learned concept library. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2409.09359.

John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

Anushrut Jignasu, Ethan Herron, Zhanhong Jiang, Soumik Sarkar, Chinmay Hegde, Baskar Ganapathysubramanian, Aditya Balu, and Adarsh Krishnamurthy. STITCH: Surface reconstruction using implicit neural representations with topology constraints and persistent homology. *arXiv preprint arXiv:2412.18696*, 2024. URL https://arxiv.org/abs/2412.18696.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)*, pp. 61–70. Eurographics Association, 2006. doi: 10.2312/SGP/SGP06/061-070.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5.

Zongzhao Li, Jiacheng Cen, Bing Su, Wenbing Huang, Tingyang Xu, Yu Rong, and Deli Zhao. Large language-geometry model: When LLM meets equivariance (EquiLLM). *arXiv preprint arXiv:2502.11149*, 2025. URL https://arxiv.org/abs/2502.11149.

Cheng Lu, Chi Lu, Robert T. Lange, Jakob N. Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2408.06292.

Yusuke Matsubara, Naoki Chiba, Ryota Igarashi, Takahiro Taniai, and Yasushi Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *arXiv preprint*, 2022. URL https://arxiv.org/abs/2206.10540.

Maximilian Mews, Ansar Aynetdinov, Vivian Schiller, Peter Eisert, and Alan Akbik. Don't mesh with me: Generating constructive solid geometry instead of meshes by fine-tuning a code-generation LLM. *arXiv preprint arXiv:2411.15279*, 2025. doi: 10.48550/arXiv.2411.15279. URL https://arxiv.org/abs/2411.15279. AI for Content Creation Workshop @ CVPR 2025.

Alexey Novikov, Nhat Vu, Michael Eisenberger, Emile Dupont, Po-Sen Huang, Alexander Z. Wagner, Sergei Shirobokov, Boris Kozlovskii, Francisco J. R. Ruiz, Arshak Mehrabian, M. Pawan Kumar, Abelina See, Shibabrat Chaudhuri, Guy Holland, Andrew Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2506.13131.

Animesh Sharma. Openevolve: an open-source evolutionary coding agent. GitHub repository, 2025. URL https://github.com/codelion/openevolve.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models, 2025a. URL https://arxiv.org/abs/2404.18400.

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, 2025b. URL https://arxiv.org/abs/2504.10415.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. URL https://arxiv.org/abs/1905.11481.

8 APPENDIX

A. EVALUATION DETAILS

A.1. GEOMETRIC FIDELITY

Unlike symbolic regression tasks over 1D functions, SURFACEBENCH evaluates the quality of discovered surface equations primarily through their *geometric agreement* with the ground-truth surfaces. Our evaluation emphasizes spatial accuracy rather than symbolic form, since different algebraic expressions may represent geometrically equivalent surfaces.

We employ three complementary metrics: **Chamfer Distance (CD)**, **Hausdorff Distance (HD)**, and **Normalized Mean Squared Error (NMSE).**

Bidirectional Chamfer distance (CD).

$$\mathrm{CD}(P,Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2^2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|_2^2. \tag{3}$$

This estimates the average nearest-neighbor discrepancy in both directions, providing a stable measure of typical geometric error; by construction it can underweight sparse but severe mismatches, which motivates pairing it with a worst-case metric.

Hausdorff distance (HD).

$$\mathrm{HD}(P,Q) = \max\Big\{\sup_{p\in P}\min_{q\in Q}\|p-q\|_2,\ \sup_{q\in Q}\min_{p\in P}\|q-p\|_2\Big\}. \tag{4}$$

This yields an upper bound on the geometric deviation between the two surfaces—if $\mathrm{HD} \le \varepsilon$, every point on either surface lies within ε of the other. Its sensitivity to outliers is well known; in practice we also report a high-percentile variant (e.g., HD_{95}, the 95^{th} percentile of the one-sided nearest-neighbor distances) to separate systematic mismatch from isolated artifacts.

Normalized Mean Squared Error (NMSE).

$$\mathrm{NMSE} = \frac{\sum_{i=1}^{N_{\text{test}}}(\hat{z}_i - z_i)^2}{\sum_{i=1}^{N_{\text{test}}}(z_i - \bar{z})^2}, \tag{5}$$

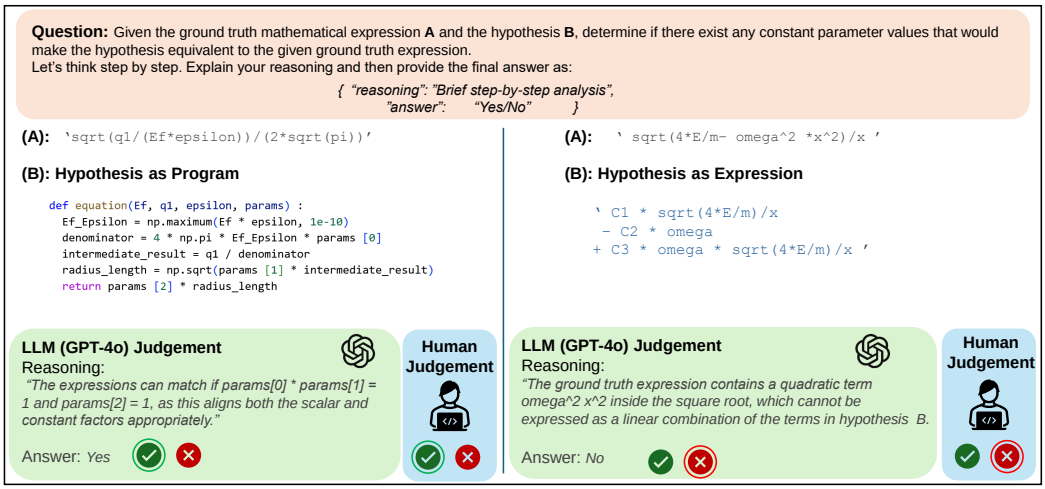

Figure 6: Symbolic assessment in equation discovery with GPT-4o as evaluator

Given test samples (x_i, y_i, z_i), the predicted surface values $\hat{z}_i$ are compared against true values z_i via: where $\bar{z}$ is the mean of ground-truth values. NMSE captures scale-invariant accuracy in the predicted scalar field, and is analogous to metrics used in curve regression benchmarks.

Together, these metrics provide a **balanced assessment**: NMSE captures numerical prediction accuracy, CD measures average geometric similarity, and HD highlights worst-case divergences. This combination allows us to evaluate not only whether an equation fits sample points, but whether it recovers the *entire surface geometry* faithfully.

A.2. Symbolic Accuracy

In addition to geometry-based metrics, we also report **Symbolic Accuracy** to provide a complementary view of equation discovery performance. For this, we adopt the evaluation methedology introduced in Shojaee et al. (2025b), which leverage GPT-4o as an automated evaluator for accessing mathematical equivalance between predicted and ground-truth hypotheses.

Traditional exact-match metrics (e.g., recovery rate, tree edit distance) are insufficient in our setting, as many surface equations admit multiple algebraically equivalent representations. The LLM-based evaluator provides a more flexible and semantically meaningful assessment of symbolic equivalence, operating across diverse formats (strings, trees, and executable forms).

We follow the same preprocessing pipeline as Shojaee et al. (2025b), including normalization of constants and removal of auxiliary infromation, and rely on GPT-4o's judgement of equivalance. This ensures comparability with prior symbolic regression benchmarks while complementing our primary focus on geometric fidelity.

As shown in Figure 6, the symbolic assessment provides a complementary view of equation discovery performance.

Table 3: Implementation details of LLM-based scientific equation discovery methods.

Method	Parameters
OpenEvolve(Sharma, 2025)	Temperature τ = 0.8 Iterations = 1000 e = 4 parallel evaluators Time limit T = 30s per program hypothesis,
SGA	PyTorch-based implementation of model and `torch.nn.Module` class Mean square error loss for data-driven feedback in agentic search Adam optimizer in PyTorch for differential parameter optimization of equation skeletons
LaSR (Biggio et al., 2021)	Iterations = 25 Cycles per iteration = 550 Populations = 10 Population size = 33 Maximum size = 30 Operators: $+$, $*$, $-$, $/$, $\wedge$, exp, log, sqrt, sin, cos, tan, cosh LLM weights: llm_mutate =0.005, llm_crossover =0.005, llm_gen_random =0.005 Top-K = 20 concepts from library Default configuration of PySR for parameter optimization
LLM-SR (Shojaee et al., 2025a)	Temperature τ = 0.8 Batch size b = 4 equation programs per prompt e = 4 parallel evaluators Time limit T = 30s per program hypothesis, Memory limit M = 2GB m = 10 islands for population diversity through search k = 2 in-context examples per prompt Maximum 10 parameters per equation skeleton BFGS optimizer from Scipy for parameter optimization of equation skeletons

B. Implementation Details

For a comprehensive evaluation, we implement three state-of-the-art LLM-guided scientific equation discovery baselines, each tested on the SurfaceBench datasets using three different LLM backbones: an open-source model (Llama-3.1-8B-Instruct), a closed-source model (GPT-4o-mini), and a proprietary model (Qwen-2.5-7B-Instruct).

B.1. Parameters

Table 3 presents the key implementation details for each discovery agentic method. We adopt most of the hyperparameters from the original implementation for these methods. We have only changed some hyperparameters in different baselines that affect the number of LLM calls in the search framework. This is to make sure we have a fair comparison across baseline discovery frameworks with same access budget to LLMs. In our experiments, all baseline frameworks have 1k calls to LLMs (per problem) through the discovery process and equivalent number of calls to Non-LLM method.

B.2. Prompts

For our experiments, we employ two types of prompts. The first set consists of *generic prompts*, which provide minimal surface description and are applicable across all surface categories. The second set comprises *specialized prompts*, which include detailed information about the specific surface category, its domain, or structural characteristics. This distinction allows us to evaluate the impact of category-specific guidance on the performance of LLM-based equation discovery.

B.2.1. Generic prompts

B.2.1.1. LLM-SR

1. Instruction prompt.

```
You are a symbolic regression assistant. Your goal is to recover the symbolic equation z = f(x,y)
that describes a 3D surface from sampled data.

Input variables:
- x: horizontal coordinate
- y: vertical coordinate
Output variable:
```

```
- z: height or surface value at (x,y)

Optional observations:
- Discontinuities or piecewise behaviours may be present.

Generate a closed-from symbolic expression for z = f(x,y) using common mathematical functions
(e.g. sin, log, exp, tanh, polynomials, np.where).

Explain your reasoning briefly.
```

2. Evaluation specification prompt.

```
import numpy as np

#Initialize parameters
MAX_NPARAMS = 10
params = [1.0]*MAX_NPARAMS


def evaluate(data: dict) -> float:
   """ Evaluate the equation on data observations."""

   # Load data observations
   inputs, outputs = data['inputs'], data['outputs']
   X = inputs

   # Optimize parameters based on data
   from scipy.optimize import minimize
   def loss(params):
      y_pred = equation(*X, params)
      return np.mean((y_pred - outputs) ** 2)

   loss_partial = lambda params: loss(params)
   result = minimize(loss_partial, [1.0]*MAX_NPARAMS, method='BFGS')

   # Return evaluation score
   optimized_params = result.x
   loss = result.fun

   if np.isnan(loss) or np.isinf(loss):
      return None
   else:
      return -loss
```

3. Equation example specification as Python programming function.

```
### Function Examples
def equation_v0($INPUT_VAR[0], ..., $INPUT_VAR[N], params):

      """ Mathematical function for {$OUTPUT_VAR_DESC}
      Args:
      $INPUT_VAR[0]: A numpy array representing observations of {$INPUT_VAR_DESC[0]}.
      ...
      $INPUT_VAR[N]: A numpy array representing observations of {$INPUT_VAR_DESC[N]}.
      params: Array of numeric constants or parameters to be optimized

      Return: A numpy array representing {$OUTPUT_VAR_DES} as the result of applying the
      mathematical function to the inputs.
      """

      # Equation example 1 logic as function body
      ...

def equation_v1($INPUT_VAR[0], ..., $INPUT_VAR[N], params):
   # Equation example 2
   ...


### Function to be completed
def equation($INPUT_VAR[0], ..., $INPUT_VAR[N], params):
   """ Improvement version of equation_v0 and equation_v1 """
```

B.2.1.2. OPENEVOLVE

The following prompts are used in our implementation of OpenEvolve for scientific equation discovery tasks, following the original implementation of OpenEvolve's public code repository (`https://github.com/codelion/openevolve`), which includes:

System prompt for task.

```
You are a symbolic regression assistant. Your goal is to recover the symbolic equation z = f(x,y)
that describes a 3D surface from sampled data.

Input variables:
- x: horizontal coordinate
- y: vertical coordinate
Output variable:
- z: height or surface value at (x,y)

Optional observations:
- Discontinuities or piecewise behaviours may be present.

Generate a closed-from symbolic expression for z = f(x,y) using common mathematical functions
(e.g. sin, log, exp, tanh, polynomials, np.where).

Explain your reasoning briefly.
```

Evaluator prompt.

```
You are an expert code reviewer.
```

B.2.1.3. LASR

We use the default prompts from LaSR's (**?**) public code repository (`https://github.com/trishullab/LibraryAugmentedSymbolicRegression.jl`), which includes:

1. The LLMINIT prompt, which is used in an LLM-augmented initialization operation.
2. LLMMUTATION prompt is used to mutate an expression based on a set of concepts.
3. LLMCROSSOVER prompt is used to construct a new expression from the crossover of two sampled expressions based on a set of concepts.
4. LLM Concept Abstraction prompt in CONCEPTABSTRACTION function, which extracts a natural language concept from current trends of hypotheses at each iteration.
5. LLM Concept Evolution prompt in CONCEPTEVOLUTION function, which creates a new concept that follows a set of ideas in the current library.

In the following, we provide examples of these prompts.

1. LLMINIT prompt.

```
<System prompt>
You are a helpful assistant that proposes a mathematical expression by following three provided
suggestions.
An expression must consist of the following variables: {{variables}}. All constants will be
represented with the symbol C. Each expression will only use these operators: {{operators}}.

<User prompt>
Suggestion 1: {{assump1}}
Suggestion 2: {{assump2}}
Suggestion 3: {{assump3}}

Propose {{N}} expressions that would be appropriate given the suggestions. Provide short
commentary for each of your decisions. End with a JSON list that enumerates the proposed
expressions following this format:
```json
["expr1",
 "expr2",
 ...
 "expr{{N}}"
]
```
```

2. LLMMUTATION prompt.

```
<System prompt>
You are a helpful assistant that mutates a mathematical expression by following a few provided
suggestions. You will be given three suggestions and a single reference expression to mutate.
An expression must consist of the following variables: {{variables}}. All constants will be
represented with the symbol C. Each expression will only use these operators: {{operators}}.

<User prompt>
Suggestion 1: {{assump1}}
Suggestion 2: {{assump2}}
Suggestion 3: {{assump3}}
Reference Expression: {{expr}}
```

```
Propose {{N}} expressions that would be appropriate given the suggestions and references. Provide
short commentary for each of your decisions. End with a JSON list that enumerates the proposed
expressions following this format:
```json
["expr1",
 "expr2",
 ...
 "expr{{N}}"
]
```
```

3. LLMCROSSOVER prompt.

```
<System prompt>
You are a helpful assistant that recombines two mathematical expressions by following a few
provided suggestions. You will be given three suggestions and two reference expressions to
recombine.
An expression must consist of the following variables: {{variables}}. All constants will be
represented with the symbol C. Each expression will only use these operators: {{operators}}.

<User prompt>
Suggestion 1: {{assump1}}
Suggestion 2: {{assump2}}
Suggestion 3: {{assump3}}
Reference Expression 1: {{expr1}}
Reference Expression 2: {{expr2}}

Propose {{N}} expressions that would be appropriate given the suggestions and references. Provide
short commentary for each of your decisions. End with a JSON list that enumerates the proposed
expressions following this format:
```json
["expr1",
 "expr2",
 ...
 "expr{{N}}"
]
```
```

4. LLM Concept Abstraction prompt.

```
<System prompt>
You are a helpful assistant that hypothesizes about the underlying assumptions that generated a
list of good and bad mathematical expressions in detailed ways. My ultimate goal is to discover
what assumptions generated the observed good mathematical expressions and excludes the bad
mathematical expressions. Focus more on the good expressions, their mathematical structure, and
any relation to physical concepts. Note that capital C represents an arbitrary constant

<User prompt>
Good Expression 1: {{gexpr1}}
Good Expression 2: {{gexpr2}}
Good Expression 3: {{gexpr3}}
Good Expression 4: {{gexpr4}}
Good Expression 5: {{gexpr5}}

Bad Expression 1: {{bexpr1}}
Bad Expression 2: {{bexpr2}}
Bad Expression 3: {{bexpr3}}
Bad Expression 4: {{bexpr4}}
Bad Expression 5: {{bexpr5}}

Propose {{N}} hypotheses that would be appropriate given the expressions. Provide short commentary
for each of your decisions. Do not talk about topics related to the simplicity or complexity of
the expressions. I want ideas that are unique and interesting enough to amaze the world's best
mathematicians. End with a JSON list that enumerates the proposed hypotheses following this format:
```json
["hyp1",
 "hyp2",
 ...
 "hyp{{N}}"
]
```
```

5. LLM Evolution prompt.

```
<System prompt>
You are an insightful assistant skilled in logical reasoning and deduction. Your task is to
analyze a set of ideas and infer nontrivial conclusions that logically follow from them. The
ultimate goal is to uncover underlying principles or properties of the hidden expressions. Focus
on providing logical conclusions that are unique, interesting, and profound.

<User prompt>
Idea 1: {{idea1}}
Idea 2: {{idea2}}
Idea 3: {{idea3}}
Idea 4: {{idea4}}
Idea 5: {{idea5}}
```

```
Based on these ideas, deduce {{N}} logical conclusions or hypotheses that directly follow from
them. Provide a brief explanation for each conclusion, highlighting the logical connections
between the ideas. Avoid discussing topics related to the simplicity or complexity of the
expressions. Conclude with a JSON list that enumerates the proposed conclusions in the following
format:
```json
["Conclusion 1",
 "Conclusion 2",
 ...
 "Conclusion {{N}}"
]
```
```

B.2.1.4. SGA

The following prompts are used in our implementation of SGA **?** for scientific equation discovery tasks, following the original implementation SGA's public code repository (`https://github.com/PingchuanMa/SGA`), which includes:

System prompt for task.

```
You are a symbolic regression assistant. Your goal is to recover the symbolic equation z = f(x,y)
that describes a 3D surface from sampled data.

Input variables:
- x: horizontal coordinate
- y: vertical coordinate
Output variable:
- z: height or surface value at (x,y)

Optional observations:
- Discontinuities or piecewise behaviours may be present.

Generate a closed-from symbolic expression for z = f(x,y) using common mathematical functions
(e.g. sin, log, exp, tanh, polynomials, np.where).

Explain your reasoning briefly.
```

Code formatting prompt for scientific discovery task.

````
### PyTorch Tips
1. When working with tensors, always use PyTorch's operators (such as 'torch.exp', 'torch.cos',
'torch.sqrt', ...) to ensure compatibility and optimal performance.
2. In PyTorch, operator input arguments must be tensors, not floats.

### Code Requirements
1. The only library allowed is PyTorch. Follow the format provided by the user examples.
2. Annotate the size of the tensor as comment after each tensor operation. For example, # (B, 3,
3).
3. Separate the code into parameters that can be tuned with differentiable optimization and the
symbolic expression represented by PyTorch code.
5. The proposed code must strictly follow the structure and function signatures below:

```python
import torch
import torch.nn as nn


class SymbolicEquation(nn.Module):

   def __init__(self, {$PARAM_INPUTS}):
      """
      Define trainable continuous parameters for differentiable optimization.
      Tentatively initialize the parameters with the default values in args.

      Args:
         {$PARAM_DESCRIPTION}
      """
      super().__init__()
      {$PARAM_INIT}

   def forward(self, {$INPUT_VARIABLES}) -> torch.Tensor:
      {$FORWARD_FUNCTION_DESCRIPTION$}
```

### Solution Requirements

1. Analyze step-by-step what the potential problem is in the previous iterations based on the
feedback. Think about why the results from previous iterations mismatched with the ground truth.
Do not give advice about how to optimize. Focus on the formulation of the scientific equation.
Start this section with "### Analysis". Analyze all iterations individually, and start the
subsection for each iteration with "#### Iteration N", where N stands for the index. Remember to
analyze every iteration in the history.

2. Think step-by-step what you need to do in this iteration. Think about what is needed to improve
performance. If the analysis suggests specific functional forms or constraints, think about how
these will be incorporated into the symbolic equation. Think about how to separate your algorithm
into a continuous parameter part and a symbolic expression model part. Describe your plan in
pseudo-code, written out in great detail. Remember to update the default values of the trainable
parameters based on previous optimizations. Start this section with "### Step-by-Step Plan".

3. Output the code in a single code block "```python ... ```" with detailed comments in the code
block. Do not add any trailing comments before or after the code block. Start this section with
"### Code".
````

Context prompt for each scientific problem.

```
### Context

The objective is to construct a mathematical expression that accurately maps input variables to a
target output based on a provided dataset. The task involves filling in a code block to define a
symbolic expression or model that minimizes the difference between predicted and ground-truth
outputs. The code block defines a class with two functions: one for parameters within the
expression and another for generating or modifying the symbolic structure of the expression.
Feedback is provided in the form of metrics measuring the error between the model's predictions
and the ground-truth values, as well as guidance on structural improvements to the symbolic
expression.

The expression represents {$OUTPUT_VAR_DESC}, given data on {$INPUTS_DESC}.
```

8.1 SURFACEBENCH EQUATIONS FOR EACH SCIENTIFIC CATEGORIES

Table 4: Benchmark equations.

Category	ID	Equations
Non-Canonical 3D Geometric Surfaces	NCGS1	$\frac{\sin(x^2+y^2)}{1+x^2+y^2}$
	NCGS2	$\frac{x^2-y^2}{1+x^2+y^2}$
	NCGS3	$\operatorname{atan2}(x,y)\,\exp\big(-(x^2+y^2)\big)$
	NCGS4	$\tanh\big(\sin(xy)\big)$
	NCGS5	$\log\big(1+x^2+y^2\big)\,\sin(x-y)$
	NCGS6	$\exp\big(\sin(x^2+y^2)\big)$
	NCGS7	$\frac{\cos(x^2+y)}{1+\lvert xy\rvert}$
	NCGS8	$\sinh(xy)\,\exp(-y^2)$
	NCGS9	$\frac{\sin\big(\sqrt{x^2+y^2}\big)}{\log(1+x^2)}$
	NCGS10	$x\,\exp\big(-x^2-y^2\big)\,\cos y$
	NCGS11	$\frac{\sin(xy)}{1+x^2+y^2}$
Piecewise Regime Surfaces	PRS1	x^2 if $x<y$ else y^2
	PRS2	$\sin(x)$ if $x<0$ else $\exp(y)$
	PRS3	xy if $xy>0$ else $-xy$
	PRS4	x^2+y^2 if $x<y$ else x^2-y^2
	PRS5	$\cos(x)$ if $\lvert x\rvert<1$ else $\exp(-y^2)$
	PRS6	x^3 if $y>0$ else $-y^3$
	PRS7	$\lvert x-y\rvert+\sin(x)$
	PRS8	$\sin(x+y)$ if $x^2+y^2<1$ else 0
	PRS9	$\tanh(x)$ if $x>y$ else $\cos(y)$
	PRS10	xy if $\lvert x-y\rvert<0.5$ else $\sin(x-y)$
Symbolic-Numeric Composite Surfaces	SNCS1	$\sin(x)+\exp(-y^2)$
	SNCS2	$\tanh(xy)+x^2$
	SNCS3	$\exp(-x^2-y^2)+\cos(3x)$
	SNCS4	$\alpha\sin(\beta x)+\gamma\log(1+y^2)$
	SNCS5	$\sinh(x)-\tanh(y)$
	SNCS6	$\sin(x^2+y^2)\,\exp(-\sqrt{x^2+y^2})$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
	SNCS7	$\tanh(x)\ \log(1+y^2)$
	SNCS8	$\cos(xy)+\exp(-x^2+y)$
	SNCS9	$\sin(2x)+\alpha\exp(-y^2)$
	SNCS10	$\beta\cos(x)+\gamma\sin(y^2)$
Hybrid Multi-Modal Symbolic Surfaces	HMMSS1	x^2 if $x<0$ else $\sin(y)$
	HMMSS2	$\log\big(1+\lvert x\rvert\big)$ if $y<$ 0 else $\exp(-y^2)$
	HMMSS3	$x^2+\sin(y)$ if $xy>$ 0 else $\big(-x^2-\cos y\big)$
	HMMSS4	$\tanh(x-y)$ if $x>y$ else 0
	HMMSS5	$\lvert xy\rvert+\sin(x-y)$
	HMMSS6	x^2 if $y>0$ else $\cos(y^2)$
	HMMSS7	$\sin(xy)$ if $x^2+y^2<$ 1 else $\log(1+x^2)$
	HMMSS8	$\tanh(x+y)$ if $xy<$ 0 else $\sin(x-y)$
	HMMSS9	x if $x>y$ else $y^2+\sin(x)$
Procedural & Fractal Surfaces	PFS1	$\sin(5x)\cos(5y)$
	PFS2	$\cos(x^2y^2)+0.2\ \sin\big(5\sqrt{\lvert x\rvert+\lvert y\rvert}\big)$
	PFS3	$\sin(xy)+0.5\ \sin(3x+5y)$
	PFS4	$e^{-0.1(x^2+y^2)}\ \sin(xy)$
	PFS5	$\sin(x^3+y^3)$
	PFS6	$xy\ \cos\big(\sqrt{x^2+y^2}\big)$
	PFS7	$\sin(2^x x)\ \cos(2^y y)$
	PFS8	$e^{-\lvert x-y\rvert}\ \sin\big(3(x+y)\big)$
	PFS9	$\sum_{i=1}^{4}\frac{\sin(2^i x)}{i}$
	PFS10	$\tanh(xy)\ \cos\big(\sqrt{x^2+y^2}\big)$
Bio-Inspired Morphological Surfaces	BIMS1	$\sin\big(\sqrt{x^2+y^2}\big)$
	BIMS2	$\exp(-x^2-y^2)\ \cos(3x)$
	BIMS3	$\tanh(x+y)\ \sin(xy)$
	BIMS4	$\log(1+x^2+y^2)\ \cos(xy)$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
	BIMS5	$x^2 + y^2 - \sin(2x + 2y)$
	BIMS6	$\cos(2x)\ \cos(2y)$
	BIMS7	$\frac{\sin(x^2 + y^2)}{1 + x^2 + y^2}$
	BIMS8	$\tanh(x^2 - y^2)$
	BIMS9	$\exp\big(-\lvert xy\rvert\big)\ \sin(x + y)$
	BIMS10	$\cos(xy) + 0.1\,(x^2 + y^2)$
Complex Composite Surfaces	CSS1	$\log\big(1 + x^2 + y^2\big)\ \cos(x - y)$
	CSS2	$\frac{\sin(x) + \cos(y)}{1 + x^2 + y^2}$
	CSS3	$e^{-0.1\lvert xy\rvert}\ \tanh(x + y)$
	CSS4	$\frac{x^2 y - y^2}{1 + x^2}$
	CSS5	$\sqrt{1 + x^2 + y^2}\ \sin(xy)$
	CSS6	$\frac{e^x + e^{-y}}{1 + \lvert x - y\rvert}$
	CSS7	$\begin{cases} x^2 + y^2, & \text{if } x + y < 0, \\ \sin(x + y), & \text{if } x + y \geq 0 \end{cases}$
	CSS8	$\frac{\cos\big(\sqrt{x^2 + y^2}\big)}{1 + e^{-xy}}$
	CSS9	$\sinh(x^2 - y^2)\, e^{-0.1(x+y)^2}$
	CSS10	$\arctan(xy) + 0.2\, e^{-x^2 - y^2}$
Tensor Field Surfaces	TFS1	$x^2 + y^2$
	TFS2	$\sin(x)\ \cos(y)$
	TFS3	$\exp(-x^2 - y^2)$
	TFS4	$x\, y$
	TFS5	$\tanh(x + y)$
	TFS6	$\cos(x^2 + y^2)$
	TFS7	$\log(1 + x^2 + y^2)$
	TFS8	$x^2 - y^2$
	TFS9	$\sin(x\, y)$
	TFS10	$\exp\big(-\lvert x - y\rvert\big)$
Discrete Symbolic Grid Surfaces	DSGS1	$\sin(i) + \cos(j)$
	DSGS2	$(-1)^i\,(-1)^j$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
	DSGS3	$\mathrm{mod}(i, 3) + \mathrm{mod}(j, 2)$
	DSGS4	$\left\lfloor \sqrt{i^2 + j^2} \right\rfloor$
	DSGS5	$\sin(ij) + i - j$
	DSGS6	$\cos(i + j)$
	DSGS7	$\mathrm{mod}(i^2 + j^2,\, 5)$
	DSGS8	$\tanh(i - j)$
	DSGS9	$\lfloor \sin(i^2 + j^2) \rfloor$
	DSGS10	$\mathrm{mod}(ij,\, 4)$
Non-Linear Dynamical System Surfaces	NLDSS1	$\cosh\big(0.1(x - y)\big) - \cos\big(0.5(x + y)\big)$
	NLDSS2	$e^{-0.05(x^2+y^2)}\,(x^2 - y)\cos(y)$
	NLDSS3	$\log(1 + x^2)\sin(y) - \log(1 + y^2)\cos(x)$
	NLDSS4	$\sqrt{1 + 0.1(x^2 + y^2)}\,\sin\big(0.5(x - y)\big)$
	NLDSS5	$\tanh\big(0.2(x^2 - y^2)\big)$
	NLDSS6	$0.3(xy) - 0.2\sin(x + y)\,e^{-0.05(x^2+y^2)}$
	NLDSS7	$\dfrac{x^2 \sin(y)}{1 + 0.2y^2}$
	NLDSS8	$\sinh(0.2x)\,e^{-0.1y^2}$
	NLDSS9	$\arctan(xy) - 0.3\sin(x - y)$
Stochastic Process Surfaces	SPS1	$3\,e^{-0.05(x^2+y^2)}\cos(0.2xy) + 0.1x$
	SPS2	$2.2\sin(0.3x + 0.2y)\,\big(1 - e^{-0.1x^2}\big)$
	SPS3	$1.8\cos(0.4xy)\,e^{-0.1x^2} + 0.3y^2$
	SPS4	$2\sin(0.7x)\,e^{-0.05y^2} + 0.5xy$
	SPS5	$3\,(1 - e^{-0.15x^2})\cos(0.3y) + 0.2x$
	SPS6	$2.5\tanh(0.2xy) + 0.4\sin(0.5x + y)$
	SPS7	$1.5e^{-0.1(x^2+y^2)}\sin(0.6x) + 0.3y$
	SPS8	$4\cos(0.4x)\,\big(1 - e^{-0.05y^2}\big) + 0.1x^2$
	SPS9	$2x^2 e^{-0.2\lvert y\rvert} + 1.5\sin(0.3xy)$
	SPS10	$3\sin(0.5x)\,e^{-0.1y^2} + 0.2xy\cos(y)$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
Quantum Inspired Surfaces	QIS1	$e^{-0.8(x^2+y^2)}$
	QIS2	$x^2\, e^{-(x^2+y^2)}$
	QIS3	$(x^2+y^2)\, e^{-0.9(x^2+y^2)}$
	QIS4	$e^{-0.4(x^2+y^2)}\,\big(1.1+\cos(5x)\big)$
	QIS5	$\big(\cos(1.5x)\cos(1.5y)\big)^2$
	QIS6	$\sin^2\big(3\arctan(\frac{y}{x})\big)\, e^{-\sqrt{x^2+y^2}}$
	QIS7	$1-\tanh(x^2+y^2-4)$
	QIS8	$(x^2-y^2)^2\, e^{-0.7(x^2+y^2)}$
	QIS9	$\sin^2(x+y)\,\cos^2(x-y)$
	QIS10	$\frac{1+x^2}{1+(x^2+y^2)^2}$
Surrogate-Distilled Symbolic Approximations	SDSA1	$x^3+y^3-3xy+\sin(x)$
	SDSA2	$\log\big(1+x^2+y^2\big)-\tanh(x-y)$
	SDSA3	$e^{-x^2-y^2}\,\sin(2x+y)$
	SDSA4	$\arctan(x)+\arctan(y)$
	SDSA5	$\sin(x)\cos(y)+0.1\,xy$
	SDSA6	$3\,\sin(0.4x)\, e^{-0.05y^2}+0.2\,xy$
	SDSA7	$2\,\sin(x+y)\, e^{-0.5x^2}+y^2$
	SDSA8	$\tanh(xy)+0.5\,\sin(0.5x)\,y$
	SDSA9	$1.5\,x^2\cos(0.2y)+0.3\, e^{-0.1x^2}$
Algebraic Manifold of Higher Degree	AMHD1	$x^3+y^3+z^3-3xyz=0$
	AMHD2	$x^3y+y^3z+z^3x=0$
	AMHD3	$x^5+y^5+z^5-xyz=0$
	AMHD4	$x^4y-z^6+\sin(xz)=1$
	AMHD5	$z^5+x^3y^4-e^y=0$
	AMHD6	$x^6-y^4z^2+\tan(z)=2$
	AMHD7	$z^3+x^4y^3-\cos(x)=-1$
	AMHD8	$x^3y^2-z^5+\sin(yz)=0$
	AMHD9	$x^2y^3z-z^4+\sin(x)=-1$
	AMHD10	$z^3+x^5y-e^z+xy^2=0$
	AMHD11	$x^4-y^2z^5+\tan(z)=2$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
	AMHD12	$z^5 + x^3y^4 - \cos(y) = 1$
	AMHD13	$x^6y^2 - z^3 + e^x = 0$
	AMHD14	$z^4 - x^4y + \sin(xz) = -2$
	AMHD15	$x^3 + y^4z^2 - e^y + \cos(x) = 1$
	AMHD16	$x^5y - z^4 + \sin(yz) = 2$
	AMHD17	$z^3 - x^3y + e^z + 2xy = -1$
	AMHD18	$x^4 + y^5z - \cos(xz) = 0$
	AMHD19	$x^6 - y^3z^2 + \tan(z) = 2$
	AMHD20	$x^2y^2z - z^5 + \sin(xz) = 1$
	AMHD21	$z^3 + x^4y - 2e^z + xy^2 = 0$
	AMHD22	$x^5 - y^2z^3 + \cos(xy) = -1$
	AMHD23	$x^3 + y^4z - \tan(x) + 1 = 0$
	AMHD24	$z^5 + x^3y^2 - 2z^2x + \sin(y) = 1$
Transformed Coordinate Surfaces	TCS1	$(\sin(u^2 + v)\, e^{-v},\ \cos(uv)\ \log(1 + \lvert v\rvert),\ \frac{\sin(uv^2)}{1+u^2})$
	TCS2	$(\cosh(u + v^2),\ \sinh(uv),\ \tanh(u^2 - v)\cos v)$
	TCS3	$(e^{\,u-v^2}\sin u,\ u^2\cos v,\ \log(1 + u^2 + v^2))$
	TCS4	$(\sin(u^2)\, v,\ \cos(v^2)\, u,\ u\, e^{-v})$
	TCS5	$(u^3 - v^2,\ \cos(uv^2),\ \tanh(u - v)\log(1 + u^2))$
	TCS6	$((u^2 + v^2)\sin u,\ (u^2 + v^2)\cos v,\ \sqrt{u^2 + v^2}\ \cos\sqrt{u^2 + v^2})$
	TCS7	$(\log(1 + u^2)\cos v,\ \sin(u + v^2),\ u^2\tanh v)$
	TCS8	$(\tanh(u^2)\sin v,\ u\, e^{-v^2},\ \frac{\cos(uv^2)}{1+u^2})$
	TCS9	$(\cos(u^2 + v)\, e^{u/5},\ \sin(v^2 - u),\ u\ \log(1 + v^2)\tanh u)$
	TCS10	$(u\cos(v^2),\ u\sin v,\ e^{(u-v^2)/5})$
High-Dimensional Parametric Surfaces	HDPS1	$(\sinh(u/5),\ \cosh(uv/10),\ \sin(u+ v)\log(1 + v^2))$
	HDPS2	$(u^2\cos v,\ v^2\sin u,\ \tanh(uv))$

Continued on next page

Table 4 - continued from previous page

Category	Equation	Real-world Domain
	HDPS3	$(e^{(u^2-v)/10}\sin v,\ \cos(uv),\ u^2+v^2)$
	HDPS4	$(\tanh(u+v^2),\ \sin(u^2v),\ \cos(u-v^2)\log(1+u^2))$
	HDPS5	$(u\cos(v^2),\ u\sin v,\ \sin(u^2+v))$
	HDPS6	$(\sinh(uv/5),\ \cos(u-v^2),\ e^{-u^2/10}\tanh v)$
	HDPS7	$(u\sin(v^2),\ v\cos u,\ \log(1+u^2+v^2)\sin u)$
	HDPS8	$(\tanh(u^2+v)\cos v,\ \sin(uv^2),\ u^2e^{-v/5})$
	HDPS9	$(e^{(u-v)/5}\sin(u^2),\ \cos(v^2-u),\ u\tanh(v^2))$
	HDPS10	$(\sin(\frac{u^2v}{10}),\ v\cos u,\ \log(1+u^2)\sinh(v/5))$
Topologically Rich Parametric Surfaces	TRPS1	$(\log(1+u^2)\cos(v^2),\ \sin(u+v),\ u\,e^{v^2/10})$
	TRPS2	$(\frac{u^3\sin v}{100},\ \cos(uv^2),\ \tanh(u-v^2))$
	TRPS3	$(\sin(\frac{u^2}{v^2+1}),\ e^{-v/5}\cos u,$
	TRPS4	$((5+v\cos(u/2))\sin u,\ (5+v\cos(u/2))\cos u,\ v\sin(u/2))$
	TRPS5	$((5+\sin(uv))\cos u\sin v,\ (5+\sin(uv))\sin u\sin v,\ (5+\sin(uv))\cos v)$
	TRPS6	$(\cos u\sin v,\ \sin u\sin v,\ \cos v+\frac{u}{2})$
	TRPS7	$(u,\ v,\ \sin\sqrt{u^2+v^2}+\frac{\cos u\sin v}{5})$
	TRPS8	$(\cos u\,(5+\sin 3v),\ \sin u\,(5+\sin 3v),\ \cos(3v)+u/2)$
	TRPS9	$(\sin(2u)\cos^2 v,\ \cos(2u)\sin v,\ \sin u\cos v)$
	TRPS10	$(\sinh(u/5)\cos v,\ \cosh(u/5)\sin v,\ \tanh(v)\cos$

8.2 GROUND TRUTH V PREDICTED PROGRAM/EQUATION

Predicted: OpenEvolve

```python
def func(x, params):
    """
    Calculates the model output using a linear combination of input variables
    or a constant value if no input variables. Operates on a matrix of samples.
    """
    x0 = x[:, 0]
    x1 = x[:, 1]
    x0_squared = x0 ** 2
    x1_squared = x1 ** 2
    x0_x1 = x0 * x1

    # Combine terms for better readability and potential numerical stability
    x0_x1_squared = x0_x1 * x0_x1
    result = (params[0] +  # Constant term
              params[1] * x0 +
              params[2] * x1 +
              params[3] * x0_squared +
              params[4] * x1_squared +
              params[5] * x0_x1 +
              params[6] * x0_x1 * x0 +  # x^2 * y
              params[7] * x0_x1 * x1 +  # x * y^2
              params[8] * x0_x1_squared * x1 +  # Combined cubic term
              params[9] * x0_x1_squared)  # Keeping y^3
    return result
```

Figure 9