LLM-ACES: Closed-Loop Discovery of Dynamical Systems with LLM-Guided Adaptive Search

Nikhil Abhyankar **Sha Li** **Sanchit Kabra**
Naren Ramakrishnan **Yulia Gel** **Chandan K. Reddy**

Virginia Tech
nikhilsa@vt.edu, sanchit23@vt.edu

Abstract

Recovering governing ordinary differential equations (ODEs) from data is a central challenge in modeling dynamical systems across scientific domains. Existing approaches cast discovery as a static inference problem over fixed datasets, assuming that the observed trajectories are sufficiently informative. However, dynamical systems evolve over large state spaces, and limited data can make multiple equations observationally indistinguishable, leading to identifiability gaps and the recovery of incorrect governing equations. To address this, we introduce LLM-ACES, or **LLM**-guided **A**ctive **C**losed-loop **E**quation **S**earch, a closed-loop framework that jointly optimizes symbolic hypothesis construction and adaptive data acquisition. In LLM-ACES, a large language model (LLM) proposes operator priors that define constrained subspaces of the symbolic search space, within which candidate equations are fit to the observed data. The disagreement among these candidates guides the acquisition of informative trajectories, creating a feedback loop that iteratively refines both the hypothesis space and the discovered dynamics. On 122 ODE systems spanning ODEBench and ODEBase, LLM-ACES achieves the lowest median NMSE, outperforming state-of-the-art baselines while reaching the highest symbolic accuracy up to 70.8% on ODEBench and 69.5% on ODEBase across different LLM backbones. Our analysis further shows that LLM-ACES is sample-efficient, achieving better performance with less data. Under observation noise and sparse sampling, LLM-ACES keeps the lowest reconstruction error without inflating expression complexity, whereas baselines tend to add spurious terms that fit the data locally but obscure the governing relationships.

Code : https://github.com/scientific-discovery/LLM-ACES

1 Introduction

Discovering governing equations from experimental data is a fundamental challenge in dynamical-systems modeling and a key driver of scientific progress [30, 14]. In scientific domains, ordinary differential equations (ODEs) provide compact and interpretable representations of continuous-time dynamics through vector fields [7, 29, 57]. Although such equations have traditionally been derived from first principles, the availability of high-fidelity data has made data-driven system identification increasingly viable [5], thus motivating automated discovery methods based on genetic programming, sparse regression, symbolic regression, and deep learning [23, 8, 51, 42]. Recently, large language models (LLMs) have emerged as powerful tools for scientific hypothesis generation, using pre-trained scientific and mathematical knowledge to propose structured candidate laws [40, 18, 47]. However, dynamical equation discovery is not simply a problem of fitting equations to observed trajectories. It is a closed-loop process that also requires acquiring informative observations across diverse conditions to identify the underlying governing dynamics correctly.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Comparison of symbolic ODE discovery methods.

Method	Hypothesis Refinement	Closed Loop Discovery	LLM-induced Prior	Data Acquisition	Search Paradigm
SINDy [8]	✗	✗	✗	Passive	Sparse Regression
PySR [14]	✓	✗	✗	Passive	Evolutionary
ODEFormer [15]	✗	✗	✗	Passive	Transformer-based
LLM-ODE [5]	✓	✗	✓	Passive	Evolutionary
APPS-ODE [26]	✓	✓	✗	**Active**	Grammar-based RL + Adaptive Sampling
Active-SINDy [25]	✓	✓	✗	**Active**	Sparse Regression + Adaptive Sampling
LLM-ACES (ours)	✓	✓	✓	**Active**	**Evolutionary + Adaptive Sampling**

Trajectories initialized from a limited region of the state space can render multiple structurally distinct equations observationally indistinguishable. A candidate equation may fit the observed data well yet predict entirely different behavior under new initial conditions or over longer time horizons (see Figure 12 in Appendix D.1). This risk is especially pronounced in nonlinear systems, where small changes in initial conditions lead to dramatically divergent trajectories [49], producing an equation with low training error but incorrect governing structure. Thus, *the central challenge is not only to search over candidate equations, but also to acquire data that actively distinguishes among competing dynamical hypotheses.* This setting arises naturally in self-driving laboratories and Bayesian experimental design, where an agent can adaptively query experimentally feasible conditions under a limited acquisition budget [12, 52]. Existing methods address parts of this challenge, but not the full closed-loop discovery problem. Classical regression methods search for interpretable equations from fixed datasets but rely on manually specified, static operator spaces and passive observations [8, 14]. LLM-guided approaches improve hypothesis generation by injecting scientific priors into symbolic search, treating discovery as a passive regression task [47, 5]. Active symbolic discovery methods adaptively acquire new trajectories, but typically do so without using structured symbolic hypothesis spaces as the objects that guide experimentation [22, 21, 26]. This leaves open a fundamental question: *can symbolic hypotheses themselves guide adaptive data acquisition toward more data-efficient and identifiable dynamical-system discovery?*

To address this question, we propose **LLM-ACES** (**LLM**-guided **A**ctive **C**losed-loop **E**quation **S**earch), a closed-loop framework that jointly performs symbolic hypothesis construction and adaptive trajectory acquisition. Unlike prior methods that separate equation search from data collection, LLM-ACES formulates *dynamical-system discovery as an active inference process in which the symbolic hypothesis space and the acquired dataset co-evolve*. As illustrated in Figure 1, LLM-ACES uses LLMs to induce operator-level priors that constrain the symbolic search spaces, instead of relying on LLMs to output final equations directly (Figure 1 (A)). Candidate equations to fit the observed data are then instantiated and optimized within these constrained spaces. The resulting population of fitted equations serves as a predictive-disagreement set over the unknown dynamics (Figure 1 (B)). LLM-ACES then selects experimentally feasible initial conditions by identifying regions of the state space where candidate equations produce maximally divergent rollouts (Figure 1 (C)). The newly acquired trajectories are fed back into the discovery loop, thereby refining both the candidate equations and the subsequent LLM-guided construction of the hypothesis space (Figure 1 (D)). In this way, LLM-ACES uses candidate hypotheses to resolve identifiability gaps, rather than treating them as passive regression outputs. Table 1 compares dynamical-system discovery methods across hypothesis refinement, closed-loop discovery, LLM-induced priors, data acquisition, and search paradigm. Existing methods largely address these dimensions in isolation. Classical approaches [8, 17] rely on fixed symbolic spaces and passive datasets, limiting generalization across dynamical regimes. `PySR` [14], `LLM-SR` [47], and `LLM-ODE` [5] improve symbolic search and hypothesis exploration, but still operate on static datasets. `APPS-ODE` [26] introduces active acquisition for ODE discovery, but does not couple acquisition with LLM-induced symbolic hypothesis-space refinement. In contrast, LLM-ACES unifies LLM-guided operator-prior induction, iterative symbolic refinement, and predictive-divergence-driven trajectory acquisition within a single closed-loop framework. We instantiate LLM-ACES with LLM backbones `GPT-4o-mini` and `Qwen3-32B` and evaluate it on ODEBench and ODEBase datasets. Our results show that LLM-ACES improves equation recovery and sample efficiency compared to prior methods. Further analyses demonstrate the importance of LLM-induced hypothesis spaces, feedback-driven refinement, and disagreement-based data acquisition in resolving identifiability gaps. We summarize our contributions as follows:

- **Closed-loop equation discovery.** We propose LLM-ACES, a unified framework that tightly couples LLM-guided symbolic hypothesis construction with adaptive trajectory acquisition, enabling the hypothesis space and observed dataset to co-evolve through iterative feedback.

- **LLM-induced symbolic search spaces.** We use LLMs to construct domain-informed operator priors that constrain symbolic regression, decoupling hypothesis-space design from equation fitting.
- **Divergence-driven data acquisition.** We introduce a trajectory acquisition strategy that selects initial conditions by maximizing predictive divergence among candidate equations, directly targeting regions where competing dynamical hypotheses disagree most.
- **Empirical improvements.** On ODEBench and ODEBase, LLM-ACES outperforms strong passive and active baselines across reconstruction, generalization, and extrapolation settings, achieving the highest symbolic accuracy while remaining sample-efficient, noise-robust, and interpretable.

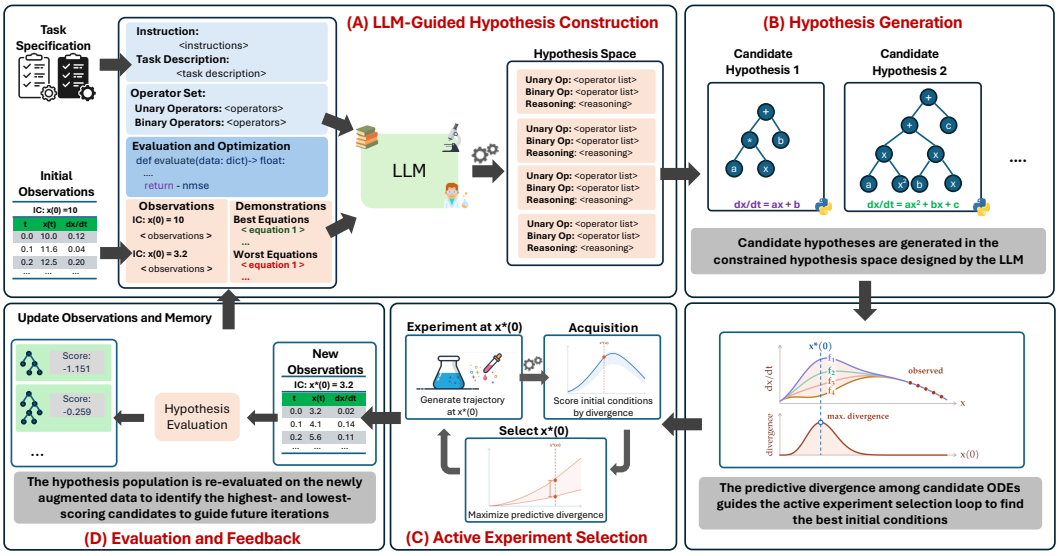


Figure 1: **Overview of LLM-ACES**. LLM-ACES couples LLM-guided hypothesis-space construction with active trajectory acquisition. (A) The LLM induces constrained symbolic search spaces from task information and prior feedback. (B) Candidate equations are generated and optimized within these spaces. (C) New initial conditions are selected by maximizing predictive disagreement among candidate rollouts. (D) Acquired trajectories update the candidate equations and refine the next hypothesis space.

2 LLM-ACES

2.1 Problem Formulation

Dynamical symbolic discovery. We consider an autonomous dynamical system $\dot{\mathbf{u}}(t) = \mathbf{f}^\star(\mathbf{u}(t))$, where $\mathbf{u}(t) \in \mathbb{R}^d$ is the state and $\mathbf{f}^\star : \mathbb{R}^d \to \mathbb{R}^d$ is the unknown governing vector field. The goal is to recover an interpretable symbolic approximation $\hat{\mathbf{f}}$ from a hypothesis space $\mathcal{H}$ constructed from an operator vocabulary $\mathcal{O}$. Given $\mathcal{D} = \{(\mathbf{u}_i, \dot{\mathbf{u}}_i)\}_{i=1}^N$ with N observations, or derivatives estimated from sampled trajectories, discovery seeks a parsimonious model that best fits the data:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f} \in \mathcal{H}} \mathcal{L}(\mathbf{f}; \mathcal{D}), \tag{1}$$

where $\mathcal{L}$ measures prediction error; parsimony is handled by the symbolic-regression backend's complexity-aware model selection (Appendix B). Two equations $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{H}$ are *observationally indistinguishable* on dataset $\mathcal{D}$ if $\mathcal{L}(\mathbf{f}_1; \mathcal{D}) \approx \mathcal{L}(\mathbf{f}_2; \mathcal{D})$ yet $\mathbf{f}_1 \neq \mathbf{f}_2$ structurally. We refer to this as the *identifiability gap* in which distinct candidate dynamics fit the available observations comparably well and cannot be reliably distinguished at the available observation precision.

Active data acquisition. The key insight is that dynamical systems, unlike static datasets, can be *queried* given access to a simulator or experimental oracle Ω. Ideally, the next initial condition $\mathbf{u}_0$ yields a trajectory $\tau(\mathbf{u}_0)$ that contradicts at least one competing candidate. Because $\tau(\mathbf{u}_0)$ is unknown before the query, we use the candidates' predicted disagreement as a proxy. Let $\hat{\tau}_{\mathbf{f}}(\mathbf{u}_0)$ denote the rollout of candidate $\mathbf{f}$ from $\mathbf{u}_0$. For any trajectory $\tau^\star$, the triangle inequality gives

$$\max\big(\|\hat{\tau}_{\mathbf{f}_1}(\mathbf{u}_0) - \tau^\star\|,\ \|\hat{\tau}_{\mathbf{f}_2}(\mathbf{u}_0) - \tau^\star\|\big) \geqslant \tfrac{1}{2}\,\|\hat{\tau}_{\mathbf{f}_1}(\mathbf{u}_0) - \hat{\tau}_{\mathbf{f}_2}(\mathbf{u}_0)\|.$$

Thus, a large separation between candidate rollouts implies that at least one candidate must incur a substantial trajectory error, motivating predictive disagreement as a heuristic for selecting informative queries.

2.2 LLM-guided Hypothesis Generation

LLM-ACES begins with an LLM-guided hypothesis generation stage. Instead of asking the LLM to directly output final equations, we use an LLM π_θ to generate operator priors that define structured subspaces of the symbolic hypothesis space. Candidate equations are then fitted within these constrained subspaces using a symbolic regression backend. This design separates LLM-based hypothesis-space construction from numerical equation fitting and evaluation.

Hypothesis-space exploration. At each iteration t, the LLM receives a prompt p_t containing (i) task-specific information, (ii) the available operator vocabulary $\mathcal{O}$, (iii) the evaluation objective, and (iv) in-context demonstrations derived from prior iterations. It generates a set of K operator priors

$$\mathcal{C}_t = \{c_1^{(t)}, \ldots, c_K^{(t)}\}, \qquad c_i^{(t)} \sim \pi_\theta(\cdot \mid p_t). \tag{2}$$

Each prior $c_i^{(t)}$ specifies a subset of unary operators, such as $\sin$, $\cos$, $\exp$, and $\log$, and binary operators, such as $+$, $-$, $\times$, and $\div$. These priors define constrained symbolic subspaces $\mathcal{H}(c_i^{(t)}) \subseteq \mathcal{H}$ that encode dynamically plausible functional forms. One prior is generated to exploit high-performing operator patterns stored in the experience buffer $\mathcal{E}_t$, while the remaining $K-1$ priors encourage structurally distinct operator compositions conditioned on previously explored operators. This induces a structured exploration-exploitation tradeoff over symbolic hypothesis spaces.

Candidate equation generation. Given the operator prior set $\mathcal{C}_t$, LLM-ACES fits candidate equations within the corresponding constrained symbolic subspaces. For each prior $c_i^{(t)}$, a symbolic regression backend solves $\mathbf{f}_i^{(t)} = \arg\min_{\mathbf{f} \in \mathcal{H}(c_i^{(t)})} \mathcal{L}(\mathbf{f}; \mathcal{D}_t^{\mathrm{tr}})$. This produces a candidate population $\mathcal{P}_t^{\mathrm{new}} = \{\mathbf{f}_1^{(t)}, \ldots, \mathbf{f}_K^{(t)}\}$, spanning multiple LLM-induced symbolic subspaces. Each candidate is evaluated on a held-out validation set $\mathcal{D}_t^{\mathrm{val}}$ for every dimension using:

$$s_i^{(t)} = -\mathcal{L}(\mathbf{f}_i^{(t)}; \mathcal{D}_t^{\mathrm{val}}), \tag{3}$$

so that higher scores correspond to lower validation errors. These scores are used to update the experience buffer $\mathcal{E}_t$, which conditions future operator-prior generation. Implementation details are provided in Appendix B.3.

2.3 Hypothesis-Driven Data Acquisition

We formulate active equation discovery as a coupled optimization process over the candidate hypothesis population and the acquired dataset. Candidate equations identify where additional data (trajectories) would be most informative, while newly acquired trajectories refine both the candidate equations and subsequent hypothesis-space construction.

Predictive-divergence acquisition. The acquisition committee at round t is $\mathcal{Q}_t = \mathcal{P}_t^{\mathrm{new}}$, the candidate systems fitted in that round (at most K). For a candidate initial condition $\mathbf{u}_0$, each $\mathbf{f} \in \mathcal{Q}_t$ is rolled out with forward Euler, $\mathbf{u}_{\ell+1} = \mathbf{u}_\ell + \Delta t\, \mathbf{f}(\mathbf{u}_\ell)$, for L steps, and its states are stacked into $\hat{\tau}_{\mathbf{f}}(\mathbf{u}_0) \in \mathbb{R}^{Ld}$. Non-finite rollouts are discarded, leaving $\tilde{\mathcal{Q}}_t(\mathbf{u}_0)$. With $\bar{\tau}$ the mean of the retained rollouts,

$$D_{ij}(\mathbf{u}_0) = \frac{\|\hat{\tau}_{\mathbf{f}_i}(\mathbf{u}_0) - \hat{\tau}_{\mathbf{f}_j}(\mathbf{u}_0)\|_2^2}{Ld\,\|\bar{\tau}\|_2^2 + \varepsilon}, \qquad A(\mathbf{u}_0; \mathcal{Q}_t) = \operatorname*{median}_{i<j} D_{ij}(\mathbf{u}_0), \tag{4}$$

with $A = 0$ if fewer than two rollouts are retained. Normalizing by the committee mean makes D_{ij} symmetric and measures disagreement relative to the magnitude of the predicted trajectories. We approximately maximize A over the feasible box $\mathcal{U}$ with a Gaussian-process UCB search (Matérn-5/2, $\kappa = 2$) over 256 uniformly sampled initial conditions: 3 are scored at random, 7 more are chosen by UCB, and the highest-scoring of the 10 is queried. This GP models the divergence score A; it is distinct from the BO baseline (Section 3.1), whose GP models PySR's validation error. A is a heuristic: rollouts use forward Euler over $t \in [0, 0.9]$ while the oracle uses LSODA, non-finite rollouts are discarded rather than counted as disagreement, and with $K = 3$ the median is taken over only three pairs, so one outlying candidate (which enters two of them) still sets the score.

Scoring and experience management. LLM-ACES keeps three candidate sets. (i) The acquisition committee $\mathcal{P}_t^{\mathrm{new}}$ contains only the candidates fitted in iteration t. The committee is rebuilt from candidates fitted on the current training data, rather than retaining candidates from previous rounds. If fewer than two candidates are fitted in an iteration, no data is acquired in that iteration. (ii) The

memory $\mathcal{M}_t^{(k)}$ stores, for each state dimension k, every component equation fitted so far. After each acquisition step, all entries are re-scored on the updated validation set. The experience buffer $\mathcal{E}_t$ takes the top-B and bottom-B components per dimension ($B = 2$) and is inserted into the next prompt together with all previously proposed operator sets. High-scoring components reinforce effective operator compositions, while low-scoring ones discourage spurious structures. (iii) The final system is assembled per dimension from $\mathcal{M}_T^{(k)}$ (Appendix B.3).

2.4 Implementation Details

Algorithm 1 summarizes the proposed closed-loop discovery framework. Following [5], we omit semantic and physical descriptions of the system from the prompts and label state variables generically ($x_0, x_1, \ldots$), so the LLM relies on *reasoning* and *data-driven feedback* rather than memorized equations. The initial dataset $\mathcal{D}_{\text{init}}$ contains 20 oracle samples from the training initial condition, split by alternating indices into 10 training samples ($\mathcal{D}_{\text{tr}}^0$) and 10 validation samples ($\mathcal{D}_{\text{val}}^0$), with every successful acquisition adding 10 samples to each split. At each iteration t, the LLM π_θ receives task metadata m, the experience buffer $\mathcal{E}_t$, and all previous operator priors, and proposes up to $K = 3$ new priors: one exploiting the best operator patterns in $\mathcal{E}_t$, the others exploring operator families not yet represented. Each prior selects, per state dimension, a subset of an operator vocabulary $\mathcal{O}$ of unary (e.g., $\sin$, $\cos$, $\exp$, $\log$) and binary ($+$, $-$, $\times$, $\div$, exponentiation) operators. For each prior, `PySR` [14] regresses each derivative component on the full state within that subspace, yielding one candidate system; these form $\mathcal{P}_t^{\text{new}}$, and their component equations are added to the per-dimension memory $\mathcal{M}_t^{(k)}$. ACQUIREDATA$(\Omega, \mathcal{P}_t^{\text{new}})$ then selects the next initial condition via Eq. (4) (skipped if fewer than two candidates were fitted), and the oracle Ω, implemented with SciPy's [55] `solve_ivp` (LSODA), returns a 20-sample trajectory that is split by alternating indices and appended to $\mathcal{D}_{\text{tr}}^t$ and $\mathcal{D}_{\text{val}}^t$. All components in memory are then re-scored on the updated validation set, and $\mathcal{E}_{t+1}$ is rebuilt from the two best and two worst components per dimension. After T rounds, the final system is assembled from the best component in each dimension. Further details, prompts, and regression settings are in Appendix B.3.

Algorithm 1 LLM-ACES

Require: Initial data $\mathcal{D}_{\text{init}}$, oracle Ω, rounds T, priors K, LLM π_θ, metadata m, operators $\mathcal{O}$

▶ Initialize

$\mathcal{D}_{\text{tr}}^0, \mathcal{D}_{\text{val}}^0 \leftarrow \mathcal{D}_{\text{init}}[0::2], \mathcal{D}_{\text{init}}[1::2]$

$\{\mathcal{M}_0^{(k)}\}_k, \mathcal{E}_0 \leftarrow$ `Init()`

for $t = 0, \ldots, T-1$ **do**

▶ Induce priors

$\mathcal{C}_t \leftarrow \{c_i^{(t)} \sim \pi_\theta(m, \mathcal{O}, \mathcal{E}_t)\}_{i=1}^K$

▶ Fit hypotheses

$\mathcal{P}_t^{\text{new}} \leftarrow \varnothing$

for all $c_i^{(t)} \in \mathcal{C}_t$ **do**

$f_i^{(t)} \leftarrow$ `FitData`$(\mathcal{D}_{\text{tr}}^t, c_i^{(t)})$

$\mathcal{P}_t^{\text{new}} \leftarrow \mathcal{P}_t^{\text{new}} \cup \{f_i^{(t)}\}$

end for

$\mathcal{M}_{t+1}^{(k)} \leftarrow \mathcal{M}_t^{(k)} \cup \{f_{i,k}^{(t)}\}_i$ for all k

▶ Acquire data

$\tau^{(t)} \leftarrow$ `AcquireData`$(\Omega, \mathcal{P}_t^{\text{new}})$

$\mathcal{D}_{\text{tr}}^{t+1}, \mathcal{D}_{\text{val}}^{t+1} \leftarrow$ `Update`$(\mathcal{D}_{\text{tr}}^t, \mathcal{D}_{\text{val}}^t, \tau^{(t)})$

▶ Score and update

$s^{(t+1)} \leftarrow$ `Score`$(\{\mathcal{M}_{t+1}^{(k)}\}_k, \mathcal{D}_{\text{val}}^{t+1})$

$\mathcal{E}_{t+1} \leftarrow$ `UpdateMemory`$(\{\mathcal{M}_{t+1}^{(k)}\}_k, s^{(t+1)})$

end for

$\hat{\mathbf{f}} \leftarrow \left(\arg\max_{f \in \mathcal{M}_T^{(k)}} s_f^{(T)}\right)_{k=1}^d$

return $\hat{\mathbf{f}}$

3 Experiments

3.1 Experimental Setup

Evaluation Metrics. Following prior work on equation discovery [48, 4], we evaluate discovered equations along three complementary axes. (i) **Data fidelity** quantifies how accurately the learned dynamics reproduce observed trajectories and generalize across regimes and is measured using normalized mean squared error (NMSE). (ii) **Expression complexity** captures the interpretability and parsimony of symbolic representations, favoring compact and human-interpretable equations. (iii) **Symbolic accuracy** (SA) measures the mean per-dimension recovery of the underlying functional form and assesses whether the discovered equation is mathematically equivalent to the ground-truth dynamics after replacing numerical coefficients with generic constants. Together, these metrics provide a holistic evaluation, since equations with similar symbolic forms may differ numerically and vice versa. Appendix A.2 contains more details on the metrics.

Table 2: **Performance across ODEBench and ODEBase datasets.** We report median NMSE (lower is better) for reconstruction, generalization, and extrapolation evaluation, mean expression complexity, and mean symbolic accuracy (higher is better). Best results are in **bold** (for complexity, indicates proximity to the benchmark's mean complexity of 19.3 for ODEBench and 35.2 for ODEBase), and second-best results are underlined.

	ODEBench					ODEBase				
Method	Recon. ↓	Gen. ↓	Ext. ↓	Comp.	SA (%) ↑	Recon. ↓	Gen. ↓	Ext. ↓	Comp.	SA (%) ↑
Passive Symbolic Discovery										
SINDy	5.07e-04	5.09e-01	1.41e+00	11.7	20.1	1.06e-03	1.12e+00	3.47e-01	12.5	10.2
Operon	7.95e-05	2.57e-01	1.84e+00	14.0	4.0	1.49e-05	1.46e+00	1.41e+00	19.0	5.4
PySR	2.84e-03	8.82e-01	1.43e+00	6.6	18.7	1.37e-03	1.08e+00	1.35e+00	8.8	5.9
E2E	3.69e-01	1.49e+00	2.19e+00	52.6	0.0	1.04e+00	1.19e+01	1.98e+01	83.6	0.0
ODEFormer	4.61e-03	3.83e-01	2.04e+00	15.9	17.1	1.64e-02	1.15e+01	4.09e+00	18.5	4.8
LLM-guided Symbolic Discovery										
LLM-only	2.20e-08	3.45e+00	1.79e+00	38.9	0.0	4.52e-08	1.06e+01	1.11e+00	54.9	0.0
LLM-ODE	4.12e-05	4.72e-03	2.43e-02	25.2	9.5	7.46e-05	8.51e-01	6.59e-04	32.1	3.4
Active Symbolic Discovery										
Query-by-Committee (QBC)	1.81e-09	5.07e-09	5.69e-08	33.7	22.6	4.97e-09	7.47e-06	6.38e-06	45.2	21.5
Bayesian Optimization (BO)	1.06e-14	9.35e-13	3.47e-10	22.6	62.3	8.55e-10	4.61e-10	5.43e-08	31.3	65.0
APPS-ODE	7.52e-01	8.13e-01	1.02e+00	13.4	2.6	5.56e-01	9.12e-01	1.01e+00	10.9	15.2
LLM-ACES (GPT)	**1.33e-17**	**8.28e-17**	**2.46e-16**	17.1	**70.8**	2.54e-14	8.50e-10	3.05e-12	**33.1**	66.1
LLM-ACES (Qwen)	6.30e-16	4.18e-15	1.89e-15	**18.2**	69.2	**3.70e-15**	**4.09e-15**	**3.44e-13**	31.2	**69.5**

Baselines. We compare LLM-ACES against a diverse set of state-of-the-art dynamical system discovery methods spanning *symbolic regression, transformer-based equation generation, LLM-guided discovery*, and *active trajectory acquisition*. Our symbolic regression baselines include `SINDy` [8] and evolutionary approaches such as `PySR` [14] and `Operon` [10]. We also consider transformer-based methods, including `End2End` (`E2E`) [28] and `ODEFormer` [15]. Among LLM-based approaches, we evaluate `LLM-ODE` [5] as well as an `LLM-only` iterative refinement setting following prior work [62, 27]. For active trajectory acquisition, we compare against `APPS-ODE` [26], as well as `Bayesian optimization (BO)` and `Query-by-Committee (QBC)` acquisition strategies [20, 21] built on top of `PySR`. Unlike standard `PySR`, these active variants iteratively acquire additional trajectories under the same acquisition budget as LLM-ACES. To mitigate dataset recall, all LLM-based methods, including LLM-ACES, are evaluated using anonymized dataset versions (see Section 4.3). All LLM-based baselines use `GPT-4o-mini` to generate 1000 candidate equations. LLM-ACES uses 10 iterations with up to 3 priors per round (30 LLM calls per dataset) using `GPT-4o-mini` as well as `Qwen3-32B` to assess robustness across different LLM backbones. Appendices B.2 and B.3 contain implementation details for all the baselines and LLM-ACES.

Evaluation Protocol. We evaluate all methods on **ODEBench** [15] and **ODEBase** [37], comprising a collection of dynamical systems spanning 1D, 2D, 3D, and 4D state spaces from physics, mathematics, and biology. Following the evaluation protocols of ODEFormer [15] and MDBench [4], we assess performance across three trajectory-level settings: reconstruction, generalization, and extrapolation. We report one run per system to limit the computational cost and summarize performance using medians and distributional analyses across benchmark systems. In *reconstruction*, the model is evaluated on trajectories from the training initial condition over $t \in [0, 1]$ with 100 uniformly sampled time steps. In *generalization*, the model trained on the reconstruction dataset is evaluated on the held-out initial condition over the time interval $[0, 1]$ with 100 samples. In *extrapolation* evaluation, we test extrapolation over an extended time range $t \in (1, 10]$ with 150 samples from the training initial condition. These settings jointly evaluate the ability to fit observed dynamics, generalize across initial conditions, and extrapolate beyond the training regime. Additional details on datasets, including dataset names, equations, and stratified subsets used for analyses, are provided in Appendix A.1.

3.2 Main Results

Table 2 summarizes results on ODEBench and ODEBase, containing 63 and 59 dynamical systems, respectively. Reconstruction and extrapolation performance are evaluated using trajectories from a single initial condition, while generalization is assessed on trajectories from a distinct initial condition.

ODEBench. Across all settings, LLM-ACES achieves the strongest overall performance. With the `GPT-4o-mini` backbone, LLM-ACES attains median reconstruction, generalization, and extrapolation NMSEs of 1.33×10^{-17}, 8.28×10^{-17}, and 2.46×10^{-16}, respectively, outperforming passive and LLM-guided baselines by more than ten orders of magnitude on generalization and extrapolation,

and the strongest active baseline, BO, by about six orders on extrapolation (3.47×10^{-10}). Among the baselines, BO is the strongest competitor, followed by QBC. However, these methods recover the correct symbolic structure less reliably, achieving symbolic accuracies of 62.3% and 22.6%, respectively. In contrast, LLM-ACES achieves the best symbolic accuracy among all methods (70.8% with GPT-4o-mini and 69.2% with Qwen3-32B). The average ground-truth complexity on ODEBench is 19.3, while LLM-ACES obtains complexities of 17.1 and 18.2 with GPT-4o-mini and Qwen3-32B, respectively. Thus, LLM-ACES recovers equations that are not merely low-error fits but are also close in structural complexity to the underlying governing equations.

ODEBase. Consistent with ODEBench, LLM-ACES achieves the strongest overall performance. With the Qwen3-32B backbone, it obtains the best median reconstruction, generalization, and extrapolation NMSEs. Passive symbolic discovery methods often achieve reasonable reconstruction error, but their performance deteriorates substantially on generalization and extrapolation trajectories, suggesting that they fit observed data without reliably identifying the governing structure. Among active baselines, BO is again strongest, followed by QBC, but both lag behind LLM-ACES (Qwen) in predictive accuracy and symbolic recovery. LLM-ACES achieves the best symbolic accuracy, reaching 69.5% with Qwen3-32B and 66.1% with GPT-4o-mini, compared with 65.0% for BO. The average ground-truth complexity is 35.2, and LLM-ACES produces equations with complexities of 31.2 and 33.1 using Qwen3-32B and GPT-4o-mini, respectively. In contrast, PySR produces simpler but less faithful expressions, oversimplifying the dynamics (see Section 4.2), while QBC produces more complex expressions without matching LLM-ACES's accuracy. Overall, LLM-ACES provides the strongest balance between predictive accuracy, symbolic fidelity, and ground-truth-aligned expression complexity. Appendix C includes distributional analysis, robustness to noise and sampling, hyperparameter selection, system recovery, and the role of LLMs in search.

4 Analysis

4.1 Ablation Study

We ablate LLM-ACES on 30 ODEBench systems subset (10 each of 1D, 2D, and 3D), with all variants sharing the same query budget and settings (Figure 2). LLM-ACES attains near machine-precision median NMSE ($\sim 10^{-32}$) on all three splits and recovers the true equation structure for $26/30$ systems. **'*w/o* LLM'** replaces the LLM with a UCB bandit over operator subsets; it degrades the most, with the largest outliers and the most failures (7 systems with extrapolation NMSE $> 10^{-3}$ vs. 4 for the full method), showing that the LLM's semantic priors contribute beyond operator restriction alone. In **'*w/o* Memory'**, the LLM proposes priors without seeing past equations or tried operator sets; 24% of its priors repeat earlier ones, and accuracy drops on 3D systems. **'*w/o* Predictive Divergence'** replaces the acquisition in Eq. 4 with uniform sampling from $\mathcal{U}$; it yields the highest median errors on every split and is the least stable, losing 6 systems after first solving them (3 by the final round), as randomly placed queries fail to discriminate between competing hypotheses. Each of these three variants recovers only $23/30$ systems. **'*w/o* Diversity'** requests all priors in a single call without asking for distinct ones or filtering repeats; it reaches the same final accuracy ($26/30$), but 20% of its priors repeat previously tried operator sets, so it explores fewer distinct hypothesis spaces than LLM-ACES (23.8 vs. 27.3 per run) despite fitting more candidates. Overall, each component targets a distinct failure mode: LLM priors steer the search toward plausible operator families, memory and diversity prevent re-exploring tried hypothesis spaces, and predictive-divergence acquisition gathers the data that separates competing hypotheses; removing any of them either loses systems or wastes search budget. We further analyze memorization and sample efficiency in Sections 4.3 and 4.4.

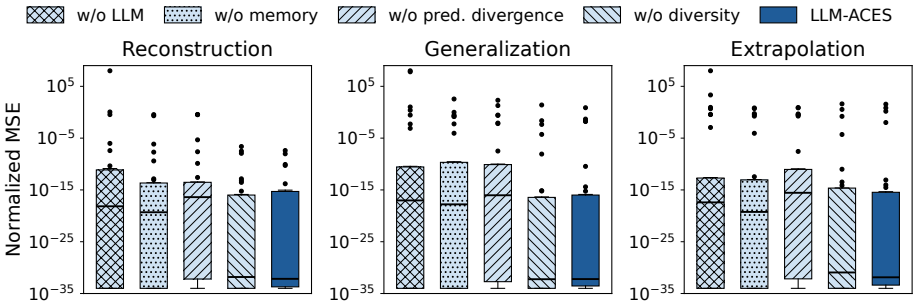


Figure 2: **Ablation on** 30 **ODEBench systems.** Normalized MSE distributions across (i) Reconstruction, (ii) Generalization, and (iii) Extrapolation evaluation settings.

4.2 Qualitative Analysis

Figure 3 compares the final equations discovered by LLM-ACES and representative baselines on the Schnackenberg and Maxwell-Bloch systems. Passive discovery methods that rely solely on fixed datasets often fail to recover the correct governing structure, even when they achieve low error on the observed trajectories. This reflects an identifiability gap: when the available trajectories are not sufficiently informative, structurally distinct equations can remain observationally indistinguishable over the sampled region of state space. We further study this failure mode in Appendix D.1. `PySR` and `SINDy` tend to oversimplify the dynamics, returning sparse or nearly linearized expressions that omit essential nonlinear interaction terms. `Operon` and `ODEFormer` search over richer hypothesis classes, but frequently introduce spurious polynomial or oscillatory terms that can improve local interpolation while obscuring the underlying mechanism. LLM-guided methods such as `LLM-ODE` recover some meaningful components but still miss key coupled interactions. Although `APPS-ODE` performs active data acquisition, its recovered equations are oversimplified, showing that data acquisition alone does not guarantee identifiability, as the queried trajectories must distinguish among competing equations that achieve similar observed-data error. In contrast, LLM-ACES selects new trajectories using predictive divergence among candidate equations, directly targeting regions where plausible symbolic hypotheses disagree. Combined with LLM-guided operator priors, this enables LLM-ACES to recover compact and correct equations that preserve the underlying dynamics, including the reaction term in Schnackenberg dynamics and the bilinear interaction terms in Maxwell–Bloch dynamics. Appendix D.3 provides additional trajectory-level comparisons between the dynamics generated by LLM-ACES and the corresponding ground-truth systems for both ODEBench and ODEBase datasets.

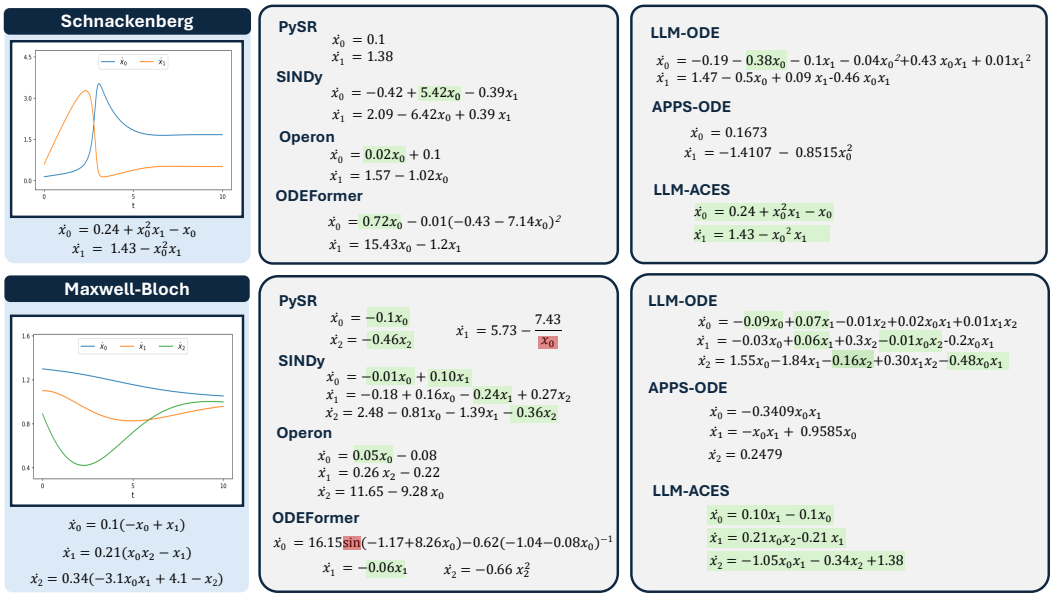


Figure 3: **Discovered equations for the Schnackenberg (top) and Maxwell-Bloch (bottom) systems.** (*Left*): ground-truth governing equations and corresponding phase trajectories. (*Middle*): equations recovered by symbolic and neural ODE discovery baselines. (*Right*): equations recovered by LLM-guided and active learning-based baselines. Green indicates the correctly recovered symbolic components from ground truth equations.

4.3 Memorization Analysis

Many benchmark systems correspond to canonical equations that are likely to have appeared during pretraining, leading to outputs that reproduce verbatim content [11, 19]. To disentangle direct dataset recall from genuine discovery, we anonymize ODEBench and ODEBase by replacing all system names, time derivatives, and semantic identifiers with generic labels ($x_0, x_1, \ldots \dot{x}_0, \dot{x}_1, \ldots$), removing any domain-specific terminology from prompts. This eliminates variable-name leakage while preserving the numerical structure of the observations, providing a more faithful test of whether the LLM is reasoning from data or retrieving memorized equations. We study this effect using `Qwen3-32B` in an iterative refinement setting. At each iteration, the model is provided with the previous best equation along with its corresponding NMSE and asked to generate candidate equations for 100 iterations.

Figure 4 compares equation discovery performance across Feynman [53], ODEBench, and an anonymized version of ODEBench. Both Feynman and ODEBench rapidly achieve extremely low NMSE, reaching near-perfect numerical fits after only a few iterations. In contrast, the anonymized version of ODEBench saturates around an NMSE value of 10^{-4}, indicating that removing semantic cues significantly increases the difficulty of the task. Furthermore, we exactly recovered 31 of 120 equations (25.8%) for the Feynman datasets and 17 out of 63 for ODEBench, suggesting a high degree of equation recall. In contrast, anonymizing ODEBench reduces the number of exact recoveries to only 2 out of 63 systems. These results indicate that anonymized benchmarks more faithfully assess equation-discovery capability by reducing memorization-based shortcuts and requiring models to infer the underlying functional structure from the available evidence rather than relying on familiarity with known equations.

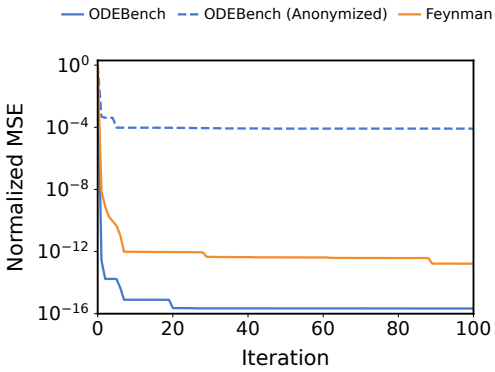


Figure 4: Error analysis comparing `Qwen3-32B` on 120 Feynman problems versus 63 ODEBench datasets.

4.4 Efficiency Analysis

We evaluate the sample efficiency of LLM-ACES on a set of 15 selected datasets, described in Appendix A.1, containing five systems from each of the 1D, 2D, and 3D sets. We compare LLM-ACES against a diverse set of baselines, including symbolic regression (`PySR`), LLM-based symbolic regression (`LLM-ODE`), and the Bayesian Optimization variant of `PySR` (`BO`), across increasing observation budgets of 100, 200, 500, and 1000 samples from the respective framework's selected IC(s). Results are aggregated using the geometric mean of NMSE across datasets, which gives each dataset equal weight on the log scale and reduces the influence of large errors relative to arithmetic averaging. As shown in Figure 5, additional observations substantially improve the performance of `PySR` between 100 and 200 samples, but its performance quickly saturates around 10^{-14} to 10^{-15} NMSE thereafter. LLM-ODE exhibits little sensitivity to the number of observations, with reconstruction errors remaining around 10^{-4} to 10^{-5} across all sample budgets. `BO` benefits from additional samples, reaching 2.01×10^{-22} at 1000 observations. Crucially, LLM-ACES outperforms all baselines at *every* observation budget from 100 samples to 1000 samples, maintaining the lowest reconstruction error across all evaluated observation budgets beating `BO` at each level. Even when competing methods are given a significantly larger number of observations ($5-10\times$ data), they cannot surpass LLM-ACES's performance. This shows that LLM-ACES's advantage stems from using observations more effectively by querying the *right data*. Appendix C contains additional results and analyses.

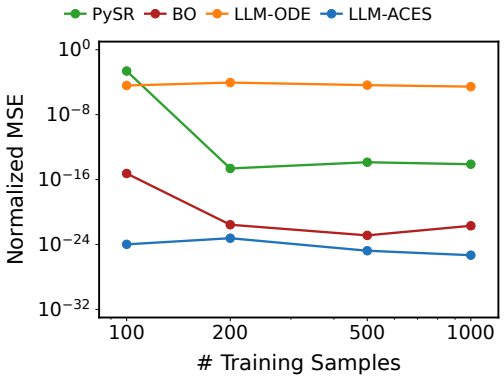


Figure 5: Geometric-mean reconstruction NMSE across the 15-system subset at training budgets of $100, 200, 500$, and 1000 observations. LLM-ACES achieves the lowest error at each evaluated budget.

5 Related Work

Discovery in Dynamical Systems. Dynamical symbolic regression (SR) seeks governing equations f from trajectories $(t, \mathbf{x}(t))$, often by learning relationships over $(\mathbf{x}(t), \dot{\mathbf{x}}(t))$ pairs despite unavailable or noisy derivatives. Neural and hybrid approaches [13, 53, 59] provide flexible dynamics models but often sacrifice interpretability or require strong inductive biases. Symbolic methods, including genetic programming [14, 10] and sparse regression [8, 41], instead seek concise interpretable expressions but typically rely on fixed datasets and manually specified operator spaces. Recent neural and transformer-based methods [54, 6, 34, 46, 28] improve scalability by framing equation discovery as sequence prediction, but largely remain passive and do not adaptively acquire data for discovery.

LLMs for Scientific Discovery. LLMs have recently been used to guide scientific hypothesis search [2, 43], often by proposing candidates evaluated through external oracles in evolutionary or iterative refinement loops [33, 31, 35, 44]. In equation discovery, LLM-SR [47] uses LLMs to guide symbolic regression from problem descriptions, while LLM-ODE [5] extends this idea to dynamical systems using observed trajectories. LLM-guided search has also been applied to program synthesis, molecular and materials discovery, and scientific optimization [38, 36, 58, 1]. Beyond direct candidate generation, [60] uses LLMs to orchestrate tools and configure downstream models, further illustrating their role in structuring scientific search. These approaches, however, primarily use LLMs to organize or explore the hypothesis space defined by the available evidence; the resulting hypotheses do not themselves determine which observations should be acquired next. LLM-ACES closes this loop by using LLMs to induce structured symbolic search spaces whose fitted hypotheses directly guide subsequent trajectory acquisition.

Active Learning for Equation Discovery. Active learning improves data efficiency by adaptively querying informative samples, including uncertain and diverse examples in deep learning [45, 3]. For dynamical systems, related work studies active input design, adaptive trajectory sampling, and exploration strategies for more efficient system identification [56, 39, 61, 50, 20, 22]. Related symbolic-regression and ODE-discovery methods use active acquisition to select useful observations or phase-space regions [9, 26]. Recent work couples LLM-generated mechanisms with hypothesis-guided experiment design [27], while related active and PDE discovery methods combine adaptive acquisition or structural search with predefined model classes [25, 16, 24]. LLM-ACES instead couples acquisition with LLM-guided symbolic search-space construction: the LLM proposes operator-level priors, symbolic regression instantiates candidate equations, and their disagreement determines subsequent trajectory acquisition.

6 Conclusion

In this work, we introduced LLM-ACES, a closed-loop framework for dynamical equation discovery that couples LLM-guided symbolic hypothesis construction with active trajectory acquisition. Rather than treating equation discovery as passive inference over a fixed dataset, LLM-ACES uses disagreement among competing symbolic hypotheses to guide the acquisition of informative trajectories, allowing data collection and equation discovery to refine one another iteratively. Across the ODEBench and ODEBase benchmarks, LLM-ACES achieves the best performance among the evaluated methods under reconstruction, generalization, and extrapolation evaluation. In particular, LLM-ACES achieves the lowest median NMSE across all settings on ODEBench and ODEBase with the `GPT-4o-mini` and `Qwen3-32B` backbones, respectively. LLM-ACES also attains the highest symbolic accuracy among all methods, reaching 70.8% on ODEBench with `GPT-4o-mini` and 69.5% on ODEBase with `Qwen3-32B`. These improvements are obtained while producing expressions whose complexity is closely aligned with the ground-truth equations, rather than merely minimizing complexity. Ablation studies show that operator-prior induction, memory, and disagreement-based acquisition each help prevent discovery failures, while diversity enforcement improves search efficiency. More broadly, our results suggest that equation discovery should be viewed not as a static regression problem, but as an iterative process in which hypotheses and observations continually inform one another. Symbolic models should serve not only as explanations of existing data, but also as tools for deciding which evidence should be collected next. We hope this perspective will inspire future AI-for-science systems that integrate foundation models, structured priors, and adaptive experimentation to enable more interpretable, data-efficient, and reliable scientific discovery.

Limitations. LLM-ACES focuses on autonomous ODE systems with query access to an oracle and may not directly generalize to PDEs, stochastic dynamics, or heavily noisy settings without changes to its components. Disagreement-based acquisition weights all candidates equally; implausible or numerically unstable candidates can dominate the score and steer queries toward uninformative regions. The dependence on LLM-induced operator priors and a symbolic-regression backend makes performance sensitive to models, prompts, and search budget.

Acknowledgements

This research was partially supported by the U.S. National Science Foundation (NSF) under Grant No. 2416728 and Autodesk Research. We thank Modal and Amazon Web Services for providing the computational resources that supported the hosting and implementation of the LLMs in this study.

References

[1] Nikhil Abhyankar, Sanchit Kabra, Saaketh Desai, and Chandan K. Reddy. LLEMA: Evolutionary search with LLMs for multi-objective materials discovery. In *The Fourteenth International Conference on Learning Representations*, 2026.

[2] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

[3] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[4] Amirmohammad Ziaei Bideh, Aleksandra Georgievska, and Jonathan Gryak. MDBench: Benchmarking data-driven methods for model discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 19746–19754, 2026.

[5] Amirmohammad Ziaei Bideh and Jonathan Gryak. LLM-ODE: Data-driven discovery of dynamical systems with large language models. *arXiv preprint arXiv:2603.20910*, 2026.

[6] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International conference on machine learning*, pages 936–945. Pmlr, 2021.

[7] Michael Breakspear. Dynamic models of large-scale brain activity. *Nature neuroscience*, 20(3):340–352, 2017.

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

[9] Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. In *International conference on intelligent data engineering and automated learning*, pages 209–218. Springer, 2007.

[10] 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*, pages 1562–1570, 2020.

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

[12] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical science*, pages 273–304, 1995.

[13] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[14] Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

[15] Stéphane d'Ascoli, Sören Becker, Philippe Schwaller, Alexander Mathis, and Niki Kilbertus. ODEFormer: Symbolic regression of dynamical systems with transformers. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Jinyang Du, Hao Ma, Xiaohu Shi, Bo Yang, Yanchun Liang, Heow Pueh Lee, and Chunguo Wu. LLM-PDESR: Robust PDE discovery via subdomain weighted residuals and LLM-guided symbolic hypothesis generation. *arXiv preprint arXiv:2607.10546*, 2026.

[17] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings. Mathematical, Physical, and Engineering Sciences*, 478(2260):20210904, 2022.

[18] 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.

[19] Valentin Hartmann, Anshuman Suri, Vincent Bindschaedler, David Evans, Shruti Tople, and Robert West. Sok: Memorization in general-purpose large language models. *arXiv preprint arXiv:2310.18362*, 2023.

[20] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. Active learning improves performance on symbolic regression tasks in StackGP. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 550–553, 2022.

[21] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. Active learning in genetic programming: Guiding efficient data collection for symbolic regression. *IEEE Transactions on Evolutionary Computation*, 29(4):1100–1111, 2024.

[22] Nathan Haut, Bill Punch, and Wolfgang Banzhaf. Active learning informs symbolic regression model development in genetic programming. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 587–590, 2023.

[23] Baihe He, Qiang Lu, Qingyun Yang, Jake Luo, and Zhiguang Wang. Taylor genetic programming for symbolic regression. In *Proceedings of the genetic and evolutionary computation conference*, pages 946–954, 2022.

[24] YuJie Huang, WenWu He, ZhuoEr Lin, Congcong Liu, Dong Liang, and Zhuo-Xu Cui. Hypothesize, evaluate, refine: A scientific agent for PDE discovery with unknown spatial coefficient fields. *arXiv preprint arXiv:2608.27475*, 2026.

[25] Ana Larranaga Janeiro, Urban Fasel, and Steven L Brunton. Active-SINDy: Intelligent sampling for model discovery in the ultra-low-data limit. In *78th Annual Meeting of the Division of Fluid Dynamics*. APS, 2025.

[26] Nan Jiang, Md Nasim, and Yexiang Xue. Active symbolic discovery of ordinary differential equations via phase portrait sketching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17626–17634, 2025.

[27] Sanchit Kabra, Nikhil Abhyankar, Saaketh Desai, Prasad Iyer, and Chandan K Reddy. LLM-AutoSciLab: Closed-loop scientific discovery via active experimentation with llms. *arXiv preprint arXiv:2605.24043*, 2026.

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

[29] Clement Kleinstreuer. *Modern fluid dynamics*. Springer, 2018.

[30] Helge Kragh. Cosmology and controversy: The historical development of two theories of the universe. 2021.

[31] Robert Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 579–582, 2024.

[32] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Melanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic acids research*, 34(suppl_1):D689–D691, 2006.

[33] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pages 331–366. Springer, 2023.

[34] Wenqiang Li, Weijun Li, Linjun Sun, Min Wu, Lina Yu, Jingyi Liu, Yanjie Li, and Songsong Tian. Transformer-based model for symbolic regression via joint supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[35] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2024.

[36] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[37] Christoph Lüders, Thomas Sturm, and Ovidiu Radulescu. ODEBase: a repository of ODE systems for systems biology. *Bioinformatics Advances*, 2(1):vbac027, 2022.

[38] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[39] Horia Mania, Michael I. Jordan, and Benjamin Recht. Active learning for nonlinear system identification with guarantees. *Journal of Machine Learning Research*, 23(32):1–30, 2022.

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

[41] Daniel A Messenger and David M Bortz. Weak SINDy: Galerkin-based data-driven model selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, 2021.

[42] Zhaozhi Qian, Krzysztof Kacprzyk, and Mihaela van der Schaar. D-CODE: Discovering closed-form ODEs from observed trajectories. In *International Conference on Learning Representations*, 2022.

[43] Chandan K Reddy and Parshin Shojaee. Towards scientific discovery with generative AI: Progress, opportunities, and challenges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28601–28609, 2025.

[44] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[45] Burr Settles. Active learning literature survey. 2009.

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

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

[48] 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. In *Forty-second International Conference on Machine Learning*, 2025.

[49] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (studies in nonlinearity)*, volume 1. Westview press, 2001.

[50] Bhavya Sukhija, Lenart Treven, Cansu Sancaktar, Sebastian Blaes, Stelian Coros, and Andreas Krause. Optimistic active exploration of dynamical systems. *Advances in Neural Information Processing Systems*, 36:38122–38153, 2023.

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

[52] Gary Tom, Stefan P Schmid, Sterling G Baird, Yang Cao, Kourosh Darvish, Han Hao, Stanley Lo, Sergio Pablo-García, Ella M Rajaonson, Marta Skreta, et al. Self-driving laboratories for chemistry and materials science. *Chemical Reviews*, 124(16):9633–9732, 2024.

[53] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science advances*, 6(16):eaay2631, 2020.

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

[55] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. SciPy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[56] Andrew Wagenmaker and Kevin Jamieson. Active learning for identification of linear dynamical systems. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3487–3582. PMLR, 09–12 Jul 2020.

[57] John A Walker. *Dynamical systems and evolution equations: theory and applications*. Springer Science & Business Media, 2013.

[58] Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[59] Juliane Weilbach, Sebastian Gerwinn, Christian Weilbach, and Melih Kandemir. Inferring the structure of ordinary differential equations. *arXiv preprint arXiv:2107.07345*, 2021.

[60] Jianke Yang, Ohm Venkatachalam, Mohammad Kianezhad, Sharvaree Vadgama, and Rose Yu. Think like a scientist: Physics-guided LLM agent for equation discovery. *arXiv preprint arXiv:2602.12259*, 2026.

[61] Zichen Zhao and Qianxiao Li. Adaptive sampling methods for learning dynamical systems. In Bin Dong, Qianxiao Li, Lei Wang, and Zhi-Qin John Xu, editors, *Proceedings of Mathematical and Scientific Machine Learning*, volume 190 of *Proceedings of Machine Learning Research*, pages 335–350. PMLR, 15–17 Aug 2022.

[62] Tianshi Zheng, Kelvin Kiu Wai Tam, Newt Nguyen Kim Hue Nam, Baixuan Xu, Zhaowei Wang, Cheng Jiayang, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Wong, and Simon See. NewtonBench: Benchmarking generalizable scientific law discovery in LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

Impact Statement

The paper presents LLM-ACES, a closed-loop framework for discovering governing equations of dynamical systems through iterative hypothesis generation, active data acquisition, and feedback. By combining symbolic regression, large language models, and active learning, LLM-ACES aims to accelerate scientific discovery in domains where governing equations are unknown or difficult to derive. The framework could support model discovery in biology, physics, climate science, and engineering, particularly when experiments are costly. However, LLM-ACES relies on LLMs, which may introduce biased or incorrect priors, and the discovered equations may not generalize beyond the observed data regime. The framework also assumes access to a queryable simulator or experimental interface, which may not always be available. We therefore recommend using LLM-ACES as a hypothesis-generation tool with expert oversight and rigorous validation.

Ethics Statement

This work uses established dynamical-system benchmarks and simulator-generated trajectories and does not involve human subjects, personal data, or interventions on physical systems. LLM-ACES is intended as a research framework rather than an autonomous system for safety-critical deployment; real-world use should incorporate domain-specific feasibility constraints, safety checks, uncertainty assessment, and human oversight. We also mitigate potential benchmark memorization by evaluating LLM-based methods on anonymized versions of the datasets and explicitly analyze its effects.

Reproducibility Statement

We provide the code repository and detailed specifications of the experimental pipeline. Section 2 and Algorithm 1 describe LLM-ACES, Appendix A documents datasets and evaluation metrics, and Appendix B reports computational resources, baseline configurations, prompts, operator constraints, symbolic-regression settings, LLM sampling parameters, and the acquisition procedure. Section 3.1 specifies evaluation protocols and budgets, while Section 4.3 describes the benchmark anonymization procedure. As hosted LLM APIs are stochastic and may change over time, exact generations may vary, but we report the model choices, sampling settings, and computational budgets for reproducibility.

Appendix

A Experiment Setup

A.1 Datasets

We evaluate all methods on two complementary ODE discovery benchmarks: **ODEBench** [15], which contains predominantly physics-inspired dynamical systems, and **ODEBase** [37], which consists of biologically grounded systems curated from experimental studies. Together, these benchmarks provide a broad evaluation suite spanning nonlinear dynamics and real-world scientific modeling.

ODEBench. ODEBench [15] contains a collection of dynamical systems derived from Steven Strogatz's textbook on nonlinear dynamics [49], together with additional systems sourced from scientific reference materials. The benchmark contains 63 ODE systems covering a diverse range of physical and mathematical phenomena, including population dynamics, oscillatory systems, epidemiological models, chemical reaction networks, chaotic attractors, and classical mechanics. Specifically, it comprises 23 one-dimensional systems, 28 two-dimensional systems, 10 three-dimensional systems, and 2 four-dimensional systems. The benchmark spans a wide range of functional forms, including polynomial, rational, exponential, logarithmic, and trigonometric dynamics, making it a challenging testbed for symbolic equation discovery. The complete list of systems is provided in Tables 3–5. To limit cost, ablations use two fixed ODEBench subsets split by dimension (4D excluded): ODEBench-30 (10 systems each in 1D, 2D and 3D) for the component ablation, and ODEBench-15 (the hardest 5 each from ODEBench-30) for the additional sweeps. Each result is compared with default LLM-ACES on the same systems.

ODEBase. ODEBase [37] is a repository of ordinary differential equation systems constructed from curated models in the BioModels database [32]. The repository was originally developed to facilitate benchmarking and evaluation of symbolic computation methods in systems biology by providing standardized ODE representations derived from SBML models. ODEBase contains mechanistic models originating from a wide range of biological domains, including cancer biology, immunology, virology, pharmacokinetics, metabolic regulation, cell-cycle dynamics, and signaling pathways. From this repository, we select 59 systems for evaluation, comprising 23 two-dimensional and 36 three-dimensional ODEs. Tables 6, 7, and 8 report the ODEBase systems.

Table 3: 1-Dimensional ODEBench datasets.

ID	System	Equation
1	RC-circuit (charging capacitor)	$\dot{x}_0 = \frac{0.7 - \frac{x_0}{1.2}}{2.31}$
2	Population growth (naive)	$\dot{x}_0 = 0.23x_0$
3	Population growth with carrying capacity	$\dot{x}_0 = 0.79x_0\left(1 - \frac{x_0}{74.3}\right)$
4	RC-circuit with nonlinear resistor	$\dot{x}_0 = -0.5 + \frac{1}{e^{0.5 - x_0/0.96} + 1}$
5	Falling object with air resistance	$\dot{x}_0 = 9.81 - 0.0021175x_0^2$
6	Autocatalysis	$\dot{x}_0 = 2.1x_0 - 0.5x_0^2$
7	Gompertz law	$\dot{x}_0 = 0.032x_0 \log(2.29x_0)$
8	Logistic with Allee effect	$\dot{x}_0 = 0.14x_0\left(-1 + \frac{x_0}{4.4}\right)\left(1 - \frac{x_0}{130}\right)$
9	Language death model	$\dot{x}_0 = 0.32(1 - x_0) - 0.28x_0$
10	Refined language model	$\dot{x}_0 = 0.2x_0^{1.2}(1 - x_0) - 0.8x_0(1 - x_0)^{1.2}$
11	Critical slowing down	$\dot{x}_0 = -x_0^3$
12	Photons in laser	$\dot{x}_0 = 1.8x_0 - 0.1107x_0^2$
13	Rotating hoop	$\dot{x}_0 = 0.0981(9.7\cos x_0 - 1)\sin x_0$
14	Budworm model	$\dot{x}_0 = 0.78x_0\left(1 - \frac{x_0}{81}\right) - \frac{0.9x_0^2}{21.2^2 + x_0^2}$
15	Budworm (dimensionless)	$\dot{x}_0 = 0.4x_0\left(1 - \frac{x_0}{95}\right) - \frac{x_0^2}{x_0^2 + 1}$
16	Landau equation	$\dot{x}_0 = 0.1x_0 - 0.04x_0^3 + 0.001x_0^5$
17	Logistic + harvesting	$\dot{x}_0 = 0.4x_0\left(1 - \frac{x_0}{100}\right) - 0.3$
18	Improved harvesting	$\dot{x}_0 = 0.4x_0\left(1 - \frac{x_0}{100}\right) - \frac{0.24x_0}{50 + x_0}$
19	Logistic (dimensionless)	$\dot{x}_0 = -\frac{0.08x_0}{0.8 + x_0} + x_0(1 - x_0)$
20	Gene switching	$\dot{x}_0 = 0.1 - 0.55x_0 + \frac{x_0^2}{x_0^2 + 1}$
21	Reduced SIR	$\dot{x}_0 = 1.2 - 0.2x_0 - e^{-x_0}$
22	Protein activation	$\dot{x}_0 = 1.4 + \frac{0.4x_0^5}{123 + x_0^5} - 0.89x_0$
23	Driven pendulum	$\dot{x}_0 = 0.21 - \sin(x_0)$

A.2 Evaluation Metrics

Data Fidelity. We evaluate data fidelity using Normalized Mean Squared Error (NMSE), which measures the relative discrepancy between predicted and ground-truth dynamics, normalized by each system's scale. Given predictions $\hat{y}$ and ground-truth values y, we compute:

$$\mathrm{NMSE}(\hat{y}, y) = \frac{\|\hat{y} - y\|_2^2}{\|y\|_2^2 + \varepsilon}, \tag{5}$$

where $\varepsilon = 10^{-10}$ is used for numerical stability. NMSE is nonnegative and ranges from 0 to ∞, with $\mathrm{NMSE} = 0$ indicating exact agreement between the predicted and true dynamics. Values closer to zero indicate higher predictive fidelity, while larger values indicate increasing deviation from the ground-truth dynamics. We report NMSE across three evaluation regimes that probe progressively stronger forms of distribution shift. In the reconstruction setting, the discovered model is numerically integrated from the training initial condition IC_0 using the LSODA solver and compared with the ground-truth trajectory over $t \in [0, 1]$ sampled at 100 uniformly spaced points. In the generalization setting, the same learned dynamics are evaluated from a held-out initial condition IC_1 over the same time horizon, isolating robustness to unseen initial conditions. In the extrapolation setting, the model is rolled out from IC_0 over the longer horizon $t \in (1, 10]$ using 150 sampled points to test long-horizon stability beyond the training window. Together, these regimes assess whether the discovered equations not only fit the observed trajectory but also reproduce the underlying dynamics under new initial conditions and extended temporal extrapolation.

Table 4: 2-Dimensional ODEBench datasets.

ID	System	Equations
24	Harmonic oscillator	$\dot{x}_0 = x_1$ $\dot{x}_1 = -2.1x_0$
25	Damped oscillator	$\dot{x}_0 = x_1$ $\dot{x}_1 = -4.5x_0 - 0.43x_1$
26	Lotka–Volterra competition	$\dot{x}_0 = x_0(3 - 2x_1 - x_0)$ $\dot{x}_1 = x_1(2 - x_0 - x_1)$
27	Lotka–Volterra	$\dot{x}_0 = x_0(1.84 - 1.45x_1)$ $\dot{x}_1 = -x_1(3 - 1.62x_0)$
28	Pendulum	$\dot{x}_0 = x_1$ $\dot{x}_1 = -0.9\sin(x_0)$
29	Dipole system	$\dot{x}_0 = 0.65x_0x_1$ $\dot{x}_1 = -x_0^2 + x_1^2$
30	RNA catalysis	$\dot{x}_0 = x_0(-1.61x_0x_1 + x_1)$ $\dot{x}_1 = x_1(-1.61x_0x_1 + x_0)$
31	SIR infection	$\dot{x}_0 = -0.4x_0x_1$ $\dot{x}_1 = 0.4x_0x_1 - 0.314x_1$
32	Double well oscillator	$\dot{x}_0 = x_1$ $\dot{x}_1 = -0.18x_1 - x_0^3 + x_0$
33	Glider	$\dot{x}_0 = -0.08x_0^2 - \sin(x_1)$ $\dot{x}_1 = x_0 - \frac{\cos(x_1)}{x_0}$
34	Rotating hoop	$\dot{x}_0 = x_1$ $\dot{x}_1 = (-0.93 + \cos(x_0))\sin(x_0)$
35	Shear flow dynamics	$\dot{x}_0 = \cos(x_0)\cot(x_1)$ $\dot{x}_1 = (4.2\sin^2(x_1) + \cos^2(x_1))\sin(x_0)$
36	Nonlinear damped pendulum	$\dot{x}_0 = x_1$ $\dot{x}_1 = -0.07x_1\cos(x_0) - x_1 - \sin(x_0)$
37	Van der Pol	$\dot{x}_0 = x_1$ $\dot{x}_1 = -0.43x_1(x_0^2 - 1) - x_0$
38	Van der Pol (Strogatz)	$\dot{x}_0 = 3.37\left(-\frac{x_0^3}{3} + x_0 + x_1\right)$ $\dot{x}_1 = -\frac{x_0}{3.37}$
39	Glycolytic oscillator	$\dot{x}_0 = 2.4x_1 + x_0^2x_1 - x_0$ $\dot{x}_1 = -2.4x_0 + 0.07 - x_0^2x_1$
40	Duffing	$\dot{x}_0 = x_1$ $\dot{x}_1 = 0.886x_1(1 - x_0^2) - x_0$
41	Cell cycle (Tyson)	$\dot{x}_0 = 15.3(0.001 + x_0^2)(-x_0 + x_1) - x_0$ $\dot{x}_1 = 0.3 - x_0$
42	Chemical reaction model	$\dot{x}_0 = 8.9 - \frac{4.0x_0x_1}{x_0^2+1} - x_0$ $\dot{x}_1 = 1.4x_0\left(-\frac{x_1}{x_0^2+1} + 1\right)$
43	Driven pendulum	$\dot{x}_0 = x_1$ $\dot{x}_1 = 1.67 - 0.64x_1 - \sin(x_0)$
44	Quadratic damping	$\dot{x}_0 = x_1$ $\dot{x}_1 = 1.67 - 0.64x_1\lvert x_1\rvert - \sin(x_0)$
45	Gray–Scott	$\dot{x}_0 = 0.5(1 - x_0) - x_0x_1^2$ $\dot{x}_1 = -0.02x_1 + x_0x_1^2$
46	Bar magnets	$\dot{x}_0 = 0.33\sin(x_0 - x_1) - \sin(x_0)$ $\dot{x}_1 = -0.33\sin(x_0 - x_1) - \sin(x_1)$
47	Binocular rivalry	$\dot{x}_0 = -x_0 + \frac{1}{e^{4.89x_1 - 1.4} + 1}$ $\dot{x}_1 = -x_1 + \frac{1}{e^{4.89x_0 - 1.4} + 1}$
48	Bacterial respiration	$\dot{x}_0 = 18.3 - \frac{x_0x_1}{0.48x_0^2+1} - x_0$ $\dot{x}_1 = 11.23 - \frac{x_0x_1}{0.48x_0^2+1}$
49	Brusselator	$\dot{x}_0 = 3.1x_0^2x_1 - 4.03x_0 + 1$ $\dot{x}_1 = 3.03x_0 - 3.1x_0^2x_1$
50	Schnackenberg	$\dot{x}_0 = 0.24 + x_0^2x_1 - x_0$ $\dot{x}_1 = 1.43 - x_0^2x_1$
51	Oscillator death	$\dot{x}_0 = 1.432 + \sin(x_1)\cos(x_0)$ $\dot{x}_1 = 0.972 + \sin(x_1)\cos(x_0)$

Table 5: 3-Dimensional and 4-Dimensional ODEBench datasets.

ID	System	Equations
52	Maxwell–Bloch	$\dot{x}_0 = 0.1(-x_0 + x_1)$ $\dot{x}_1 = 0.21(x_0 x_2 - x_1)$ $\dot{x}_2 = 0.34(-3.1 x_0 x_1 + 4.1 - x_2)$
53	Apoptosis model	$\dot{x}_0 = 0.1 - 0.05 x_0 - \frac{0.4 x_0 x_1}{x_0 + 0.1}$ $\dot{x}_1 = 0.4 x_0 x_1 - 0.5 x_1 x_2 - 0.1 x_1$ $\dot{x}_2 = 0.5 x_1 x_2 - 0.5 x_2$
54	Lorenz periodic	$\dot{x}_0 = 5.1(-x_0 + x_1)$ $\dot{x}_1 = 12 x_0 - x_0 x_2 - x_1$ $\dot{x}_2 = -1.67 x_2 + x_0 x_1$
55	Lorenz complex	$\dot{x}_0 = 10(-x_0 + x_1)$ $\dot{x}_1 = 99.96 x_0 - x_0 x_2 - x_1$ $\dot{x}_2 = -(8/3) x_2 + x_0 x_1$
56	Lorenz chaotic	$\dot{x}_0 = 10(-x_0 + x_1)$ $\dot{x}_1 = 28 x_0 - x_0 x_2 - x_1$ $\dot{x}_2 = -(8/3) x_2 + x_0 x_1$
57	Rössler stable	$\dot{x}_0 = 5(-x_1 - x_2)$ $\dot{x}_1 = 5(-0.2 x_1 + x_0)$ $\dot{x}_2 = 5(0.2 + x_2(-5.7 + x_0))$
58	Rössler periodic	$\dot{x}_0 = 0.1(-x_1 - x_2)$ $\dot{x}_1 = 0.1(-0.2 x_1 + x_0)$ $\dot{x}_2 = 0.1(0.2 + x_2(-5.7 + x_0))$
59	Rössler chaotic	$\dot{x}_0 = 0.2(-x_1 - x_2)$ $\dot{x}_1 = 0.2(-0.2 x_1 + x_0)$ $\dot{x}_2 = 0.2(0.2 + x_2(-5.7 + x_0))$
60	Aizawa attractor	$\dot{x}_0 = -0.65 x_1 + x_0(-0.7 + x_2)$ $\dot{x}_1 = 0.65 x_0 + x_1(-0.7 + x_2)$ $\dot{x}_2 = 0.6 + 0.95 x_2 - \frac{x_2^3}{3} - x_0^2 - x_1^2 + 0.25 x_2 x_0^3$
61	Chen–Lee attractor	$\dot{x}_0 = 5 x_0 - x_1 x_2$ $\dot{x}_1 = -10 x_1 + x_0 x_2$ $\dot{x}_2 = -3.8 x_2 + \frac{x_0 x_1}{3}$
62	Binocular rivalry (4D)	$\dot{x}_0 = -x_0 + \frac{1}{e^{0.89 x_2 + 0.4 x_1 - 1.4} + 1}$ $\dot{x}_1 = x_0 - x_1$ $\dot{x}_2 = -x_2 + \frac{1}{e^{0.89 x_0 + 0.4 x_3 - 1.4} + 1}$ $\dot{x}_3 = x_2 - x_3$
63	SEIR model	$\dot{x}_0 = -0.28 x_0 x_2$ $\dot{x}_1 = -0.47 x_1 + 0.28 x_0 x_2$ $\dot{x}_2 = 0.47 x_1 - 0.3 x_2$ $\dot{x}_3 = 0.3 x_2$

Table 6: 2-Dimensional ODEBase datasets. *Systems 69 and 79 come from different BioModels entries that reduce to the same ODEs in ODEBase.

ID	System	Equations
64	MPF and Cyclin Oscillations	$\dot{x}_0 = 1.0x_0^2x_1 - 10.0x_0/(x_0 + 1.0) + 3.466x_1$ $\dot{x}_1 = 1.2 - 1.0x_0$
65	Cancer–Immune System Competition	$\dot{x}_0 = -0.03125x_0^2 - 0.125x_0x_1 + 0.0625x_0$ $\dot{x}_1 = -0.08594x_0x_1 - 0.03125x_1^2 + 0.03125x_1$
66	FitzHugh–Nagumo Nerve Membrane	$\dot{x}_0 = -1.0x_0^3 + 3.0x_0 + 3.0x_1 - 1.2$ $\dot{x}_1 = -0.3333x_0 - 0.2667x_1 + 0.2333$
67	One-hit Neuronal Cell Death	$\dot{x}_0 = -0.278x_0$ $\dot{x}_1 = -0.223x_1$
68	Stem Cell Simple Model	$\dot{x}_0 = 0.004x_0 + 0.004x_1/(0.01x_0^{1.0} + 1.0)$ $\dot{x}_1 = 0.006x_0 - 0.003x_1 - 0.004x_1/(0.01x_0^{1.0} + 1.0)$
69	Alzheimer Acetylcholine Positive Feedback*	$\dot{x}_0 = -0.007x_0x_1$ $\dot{x}_1 = -0.004x_0 - 0.01x_1 + 0.33$
70	Bistable Schlögl Model	$\dot{x}_0 = -0.00096x_0^3 + 0.1229x_0^2 - 3.072x_0 + 12.5$ $\dot{x}_1 = 0.00096x_0^3 - 0.1229x_0^2 + 3.072x_0 - 12.5$
71	Lotka–Volterra CAR-T / Tumour	$\dot{x}_0 = 0.002x_0x_1 - 0.16x_0$ $\dot{x}_1 = 0.15x_1$
72	Cytokine Inflammation (Rheumatoid Arthritis)	$\dot{x}_0 = -x_0 + 3.5x_1^2/(x_1^2 + 0.25)$ $\dot{x}_1 = 1.0x_1^2/(x_0^2x_1^2 + 1.0x_0^2 + 1.0x_1^2 + 1.0) - 1.25x_1 + 0.025/(x_0^2 + 1.0)$
73	Calcium Oscillations	$\dot{x}_0 = -5.0x_0x_1^{4.0}/(x_1^{4.0} + 81.0) - 0.01x_0 + 2.0x_1$ $\dot{x}_1 = 5.0x_0x_1^{4.0}/(x_1^{4.0} + 81.0) + 0.01x_0 - 3.0x_1 + 1.0$
74	Acute Myeloid Leukaemia	$\dot{x}_0 = -0.1x_0 + 0.3x_0/(0.5x_0 + 0.5x_1 + 1.0)$ $\dot{x}_1 = -0.1x_1 + 0.3x_1/(0.5x_0 + 0.5x_1 + 1.0)$
75	Oncolytic M1 Virus–Sensitive/Normal (SN)	$\dot{x}_0 = -0.2x_0x_1 - 0.02x_0 + 0.02$ $\dot{x}_1 = 0.16x_0x_1 - 0.03x_1$
76	Bone Marrow Invasion (Absolute)	$\dot{x}_0 = -0.2x_0^2 + 0.1x_0$ $\dot{x}_1 = -1.0x_0x_1 - 0.8x_1^2 + 0.7x_1$
77	Reversible Isomerization	$\dot{x}_0 = -0.12x_0 + 1.0x_1$ $\dot{x}_1 = 0.12x_0 - 1.0x_1$
78	Tumour Under Nonstationary Therapy	$\dot{x}_0 = -1.0x_0x_1 + 2.0x_0$ $\dot{x}_1 = 1.0x_0x_1 - 0.2x_0 - 0.5x_1 + 0.25$
79	Alzheimer Choline-Leakage Hypothesis*	$\dot{x}_0 = -0.007x_0x_1$ $\dot{x}_1 = -0.004x_0 - 0.01x_1 + 0.33$
80	Birth–Death Process	$\dot{x}_0 = 1.0 - 0.025x_0$ $\dot{x}_1 = 0.025x_0 - 1.0$
81	HIV Latency / Immune Response	$\dot{x}_0 = -0.029x_0x_1 + 0.134x_0/(x_0 + 380.0) + 0.001$ $\dot{x}_1 = -0.927x_0x_1 + 0.07$
82	Bone Marrow Invasion (Relative)	$\dot{x}_0 = -0.8x_0^2 - 0.9x_0x_1 + 0.7x_0$ $\dot{x}_1 = -0.1x_0x_1 - 0.2x_1^2 + 0.1x_1$
83	Bioterrorism Panic–Protection	$\dot{x}_0 = -0.6x_0^2 - 2.8x_0x_1 + 6.0x_0$ $\dot{x}_1 = 1.0x_0x_1$
84	Tumour Immunotherapy	$\dot{x}_0 = 0.004x_0 - 4.0x_1$ $\dot{x}_1 = 0.09x_0x_1 - 0.1x_1$
85	Cancer–Immune Cell Count	$\dot{x}_0 = 0.514x_0$ $\dot{x}_1 = 10.0 - 0.02x_1$
86	Tumour–Immune Interaction (Base)	$\dot{x}_0 = -0.006544x_0^2 - 1.0x_0x_1 + 1.636x_0$ $\dot{x}_1 = -0.003x_0x_1 + 1.131x_0x_1/(x_0 + 20.19) - 2.0x_1 + 0.318$

Table 7: 3-Dimensional ODEBase datasets.

ID	System	Equations
87	Human/Mosquito ELP Epidemics	$\dot{x}_0 = 600.0 - 0.411x_0$ $\dot{x}_1 = 0.361x_0 - 0.184x_1$ $\dot{x}_2 = 0.134x_1 - 0.345x_2$
88	Cancer Virotherapy (Phase I)	$\dot{x}_0 = -1.0x_0x_2$ $\dot{x}_1 = 1.0x_0x_2 - 1.0x_1$ $\dot{x}_2 = -0.02x_0x_2 + 1.0x_1 - 0.15x_2$
89	Oncogenesis with Genetic Instability	$\dot{x}_0 = 0.01 - 0.01x_0$ $\dot{x}_1 = 0.03x_1$ $\dot{x}_2 = -0.5x_2^2 + 0.034x_2$
90	p53–Mdm2 Oscillations (Model 5)	$\dot{x}_0 = -3.7x_0x_1 + 2.0x_0$ $\dot{x}_1 = -0.9x_1 + 1.1x_2$ $\dot{x}_2 = 1.5x_0 - 1.1x_2$
91	Insulin Kinetics Model A	$\dot{x}_0 = -0.1x_0x_2 + 0.2x_1x_2 + 0.1x_2$ $\dot{x}_1 = -0.01x_0 + 0.01 + 0.01/x_2$ $\dot{x}_2 = -0.1x_1x_2 + 0.257x_1 - 0.1x_2^2 + 0.331x_2 - 0.3187$
92	p53–Mdm2 Oscillations (Model 1)	$\dot{x}_0 = -3.2x_0x_1 + 0.3$ $\dot{x}_1 = -0.1x_1 + 0.1x_2$ $\dot{x}_2 = 0.4x_0 - 0.1x_2$
93	Colon Crypt Cell Cycle (v1)	$\dot{x}_0 = -0.002207x_0^2 - 0.002207x_0x_1 - 0.002207x_0x_2 + 0.1648x_0$ $\dot{x}_1 = -0.01312x_0^2 - 0.0216x_0x_1 - 0.01312x_0x_2 + 1.574x_0 - 0.008477x_1^2 - 0.008477x_1x_2 + 0.5972x_1$ $\dot{x}_2 = -0.04052x_0x_1 - 0.04052x_1^2 - 0.04052x_1x_2 + 4.863x_1 - 1.101x_2$
94	Prophage Induction	$\dot{x}_0 = -0.99x_0^2/(x_0 + x_1) - 1.0x_0x_1/(x_0 + x_1) + 0.99x_0$ $\dot{x}_1 = -0.99x_0x_1/(x_0 + x_1) - 1.0x_1^2/(x_0 + x_1) + 1.0x_1$ $\dot{x}_2 = -0.001x_2$
95	Tumour–Immune with IL-2	$\dot{x}_0 = -1.0x_0x_1/(x_0 + 1.0) + 0.18x_0$ $\dot{x}_1 = 0.05x_0 + 0.124x_1x_2/(x_2 + 20.0) - 0.03x_1$ $\dot{x}_2 = 5.0x_0x_1/(x_0 + 10.0) - 10.0x_2$
96	Cytotoxic/Helper T Cell–Tumour Interaction	$\dot{x}_0 = -10.0x_0^2 - 2.075x_0x_2 + 10.0x_0$ $\dot{x}_1 = 0.19x_0x_1/(x_0^2 + 0.0016) - 1.0x_1 + 0.5$ $\dot{x}_2 = -2.075x_0x_2 + 1.0x_1x_2 - 1.0x_2 + 2.0$
97	Zombie (SIZRC) Epidemic	$\dot{x}_0 = -0.009x_0x_1 + 0.05$ $\dot{x}_1 = 0.004x_0x_1$ $\dot{x}_2 = 0.005x_0x_1$
98	Circadian Oscillations / NF-κB Signalling	$\dot{x}_0 = -954.5x_0x_2/(x_0 + 0.029) - 0.007x_0/x_2 + 0.007/x_2$ $\dot{x}_1 = 1.0x_0^2 - 1.0x_1$ $\dot{x}_2 = 0.035x_0 + 1.0x_1 - 0.035$
99	Glioma–Immune Interaction	$\dot{x}_0 = -0.482x_0^2 - 0.07x_0x_1/(x_0 + 0.903) - 2.745x_0x_2/(x_0 + 0.903) + 0.482x_0$ $\dot{x}_1 = -0.019x_0x_1/(x_0 + 0.031) - 0.331x_1^2 + 0.331x_1$ $\dot{x}_2 = 0.124x_0x_2/(x_0 + 2.874) - 0.017x_0x_2/(x_0 + 0.379) - 0.007x_2$
100	Hepatitis C Infection Dynamics	$\dot{x}_0 = -0.002x_0 + 1.065e{+}4x_0/(x_0 + x_1) + 0.118x_1$ $\dot{x}_1 = -0.118x_1 + 342.5x_1/(x_0 + x_1)$ $\dot{x}_2 = 204.0x_1 - 17.91x_2$
101	Tumour–Normal Cell Progression	$\dot{x}_0 = -0.931x_0x_1 - 0.138x_0x_2 + 0.431x_0$ $\dot{x}_1 = 1.189x_0x_1 - 0.1772x_1^2 - 0.147x_1x_2 + 0.443x_1$ $\dot{x}_2 = -0.813x_0x_2 + 0.271x_0x_2/(x_0 + 0.813) - 0.363x_1x_2 + 0.783x_1x_2/(x_1 + 0.862) - 0.57x_2 + 0.7$
102	L-Dopa Pharmacokinetics	$\dot{x}_0 = -2.11x_0$ $\dot{x}_1 = 0.889x_0 - 1.659x_1$ $\dot{x}_2 = 0.4199x_1 - 0.06122x_2$
103	p53–Mdm2 Oscillations (Model 4)	$\dot{x}_0 = 0.9 - 1.7x_1$ $\dot{x}_1 = -0.8x_1 + 0.8x_2$ $\dot{x}_2 = 1.1x_0 - 0.8x_2$
104	Tumour Dormancy Equilibrium	$\dot{x}_0 = -1.125x_0^2 - 0.3x_0x_1 + 0.9x_0 + 10.0$ $\dot{x}_1 = 0.1x_1x_2 - 0.02x_1$ $\dot{x}_2 = -0.1x_1x_2 - 1.143x_2^2 + 0.77x_2$

Table 8: 3-Dimensional ODEBase datasets (contd.).

ID	System	Equations
105	Tumour–Immune Noise-assisted Interactions	$\dot{x}_0 = -1.0x_0^2 - 1.0x_0x_1 + 1.748x_0 + 2.73x_2$ $\dot{x}_1 = 1.0x_1x_2 - 0.05x_1$ $\dot{x}_2 = 1.126x_0 - 15.89x_2$
106	Goldbeter Embryonic Cell Cycle	$\dot{x}_0 = -0.25x_0 - 0.25x_2 + 0.25$ $\dot{x}_1 = -6.0x_0x_1/(-2.0x_0x_1 + 2.002x_0 - 1.0x_1 + 1.001) + 6.0x_0/(-2.0x_0x_1 + 2.002x_0 - 1.0x_1 + 1.001) - 1.5x_1/(x_1 + 0.001)$ $\dot{x}_2 = -1.0x_1x_2/(1.001 - x_2) + 1.0x_1/(1.001 - x_2) - 0.7x_2/(x_2 + 0.001)$
107	Helper T Cells in Tumour Immune System	$\dot{x}_0 = -0.003272x_0^2 - 1.0x_0x_1 + 1.636x_0$ $\dot{x}_1 = 0.04x_0x_1 + 0.01x_1x_2 - 0.374x_1$ $\dot{x}_2 = 0.002x_0x_2 - 0.055x_2 + 0.38$
108	Cholesterol Biosynthesis (SREBP2)	$\dot{x}_0 = -0.001x_0$ $\dot{x}_1 = 1.0x_0 - 0.002x_1$ $\dot{x}_2 = 0.462x_1 - 0.004x_2$
109	HIV/CD4 T-cell Interaction	$\dot{x}_0 = -0.1x_0^2x_2 - 0.1x_0x_1x_2 + 0.8x_0x_2 - 0.1x_0$ $\dot{x}_1 = -0.1x_0x_1x_2 + 0.2x_0x_2 - 0.1x_1^2x_2 + 1.0x_1x_2 - 0.2x_1$ $\dot{x}_2 = 1.0x_1 - 0.5x_2$
110	Cyclin-dependent Kinase Oscillations	$\dot{x}_0 = -0.25x_0x_2/(x_0 + 0.001) - 0.046x_0 + 0.06$ $\dot{x}_1 = -4.0x_0x_1/(-x_0x_1 + 1.002x_0 - 0.5x_1 + 0.501) + 4.0x_0/(-x_0x_1 + 1.002x_0 - 0.5x_1 + 0.501) - 2.0x_1/(x_1 + 0.002)$ $\dot{x}_2 = -1.0x_1x_2/(1.01 - x_2) + 1.0x_1/(1.01 - x_2) - 0.7x_2/(x_2 + 0.01)$
111	Tumour–Normal–Vitamins (TNVM)	$\dot{x}_0 = -0.982x_0x_1 + 0.222x_0x_2 + 0.431x_0$ $\dot{x}_1 = 0.229x_0x_1 - 0.1772x_1^2 - 0.497x_1x_2 + 0.443x_1$ $\dot{x}_2 = 0.898 - 0.961x_2$
112	Colon Crypt Cell Cycle (v0)	$\dot{x}_0 = 0$ $\dot{x}_1 = 1.0x_0^2/(x_0 + 2.924) + 0.218x_0 - 0.024x_1$ $\dot{x}_2 = 1.0x_1^2/(x_1 + 29.24) + 0.547x_1 - 1.83x_2$
113	Circadian Rhythms (Neurospora)	$\dot{x}_0 = -0.505x_0/(x_0 + 0.5) + 1.6/(x_2^{4.0} + 1.0)$ $\dot{x}_1 = 0.5x_0 - 0.5x_1 - 1.4x_1/(x_1 + 0.13) + 0.6x_2$ $\dot{x}_2 = 0.5x_1 - 0.6x_2$
114	Toxicant–Immune System Dynamics	$\dot{x}_0 = -0.2x_0^2 - 0.05x_0x_1 + 0.9x_0$ $\dot{x}_1 = 0.295x_0x_1 - 0.8x_1 + 0.04$ $\dot{x}_2 = 2.4x_0 - 0.1x_2$
115	Tumour–CD4+–Cytokine Interactions	$\dot{x}_0 = -3.0e{-}5x_0^2 - 0.1x_0x_2/(x_0 + 1.0) + 0.03x_0$ $\dot{x}_1 = 0.02x_0x_1/(x_0 + 10.0) - 0.02x_1 + 10.0$ $\dot{x}_2 = 0.1x_0x_1/(x_0 + 0.1) - 47.0x_2$
116	Proteasome Dynamics (Parkinson's)	$\dot{x}_0 = -1.0x_0x_1 + 25.0/(x_1 + 1.0)$ $\dot{x}_1 = -1.0x_0x_1 - x_1 + 1.0x_2 + 1.0$ $\dot{x}_2 = 1.0x_0x_1 - 1.0x_2$
117	Weight Cycling Dynamics	$\dot{x}_0 = -0.1x_0/(x_0 + 0.2) + 0.1x_1$ $\dot{x}_1 = -1.5x_1x_2/(x_1 + 0.01) - 1.0x_1/(1.01 - x_1) + 1.0/(1.01 - x_1)$ $\dot{x}_2 = -6.0x_0x_2/(1.01 - x_2) + 6.0x_0/(1.01 - x_2) - 2.5x_2/(x_2 + 0.01)$
118	Stem Cell + Transit Amplifying Cells	$\dot{x}_0 = 0.004x_0 + 0.0096x_1/(0.01x_0^{1.0} + 1.0)$ $\dot{x}_1 = 0.006x_0 - 0.004x_1$ $\dot{x}_2 = 0.024x_1 - 0.0096x_1/(0.01x_0^{1.0} + 1.0) - 0.003x_2$
119	Tumour–Immune Immunotherapy	$\dot{x}_0 = 0.044x_0x_2/(x_2 + 0.02) - 0.038x_0 + 1.009x_1$ $\dot{x}_1 = -0.018x_0 - 0.123x_1^2 + 0.123x_1$ $\dot{x}_2 = 0.9x_0 - 1.8x_2$
120	Tumour Growth Model	$\dot{x}_0 = -17.86x_0x_2^2 + 0.05x_0x_2/(x_0 + x_1 + x_2 + 1.0) - 0.1x_0 + 0.625x_2 + 0.01$ $\dot{x}_1 = -x_1 + 2.0x_1/(x_0 + x_1 + x_2 + 1.0)$ $\dot{x}_2 = -25.0x_0x_2^2 - x_2 + 4.0x_2/(x_0 + x_1 + x_2 + 1.0)$
121	Oncolytic M1 Virus–SNT Model	$\dot{x}_0 = -0.2x_0x_1 - 0.5x_0x_2 - 0.02x_0 + 0.02$ $\dot{x}_1 = 0.16x_0x_1 - 0.03x_1$ $\dot{x}_2 = 0.4x_0x_2 - 0.028x_2$
122	CAR T-cell Therapy in ALL	$\dot{x}_0 = -0.07143x_0$ $\dot{x}_1 = 0.033x_1$ $\dot{x}_2 = -0.01667x_2$

Expression Complexity. Expression complexity is measured as the size of the symbolic expression's tree representation. Each discovered equation is encoded as a rooted tree, where leaves correspond to variables and constants, and internal nodes correspond to unary or binary operators (e.g., $+, -, \times, \sin, \exp$). The complexity is defined as the total number of nodes in the tree:

$$\text{Complexity}(\mathbf{f}) = |T_{\mathbf{f}}|,$$

where $T_{\mathbf{f}}$ denotes the expression tree of equation $\mathbf{f}$. This quantity is computed recursively by assigning a unit cost to each node in the tree and summing over all sub-expressions. Specifically, each operator contributes one unit to the complexity, and each variable or constant also contributes one unit. Consequently, expressions with greater depth or a larger number of composed operations yield higher complexity values. This provides a structural measure of model size and is commonly used as a parsimony objective in symbolic regression to discourage unnecessarily complex expressions.

```
###
<Role>
You are a strict but fair judge of structural equivalence between the right-hand side of a
ground-truth ordinary differential equation and the right-hand side discovered by an automated
method. Coefficient values do not matter. Algebraic style does not matter. Missing terms, extra
terms and different exponents do matter.

###
<Procedure>
1. Rewrite as mathematics: ^ and ** are powers, exp2(u) = 2^u, cube(u) = u³, x/x = 1, ....
2. Canonical form (both sides): expand products, distribute, collect like terms, combine powers
   of the same base, .... Identities that need a positive argument (log(x·y) = log(x) + log(y),
   sqrt(x²) = x, Abs(x) = x, ....) hold only if the argument is positive on the stated range.
3. Remove constants: replace every numerical coefficient with a generic constant c.
   Do not remove exponents.
4. Compare term by term: the two term sets must match one-to-one, with exponents within ±0.005.

###
<Decision Rules>
- Answer YES when only coefficient values, algebraic arrangement or exact identities differ.
- Answer NO when a term is missing, there is an extra term, an exponent or a function differs,
  an additive constant differs, ....
- Judge only the mathematics. Ignore how accurate the fitted coefficients are.
```

Instruction

```
###
<Inputs>
- Benchmark and system: ODEBench #27 Lotka-Volterra simple
- Equation being judged: dx0/dt, equation 1 of a 2-dimensional system
- State variables and their ranges: x0 in [0.0635, 9.28], x1 in [0.0093, 8.69]
- Ground truth for dx0/dt (c0, c1, ... are free constants; c0 = 1.84, c1 = 1.45, ....):
  x0 * (c0 - c1 * x1)
- Discovered right-hand side for dx0/dt: (1.2689655 - x1)*x0*1.45

###
<Examples>
GT: c0·x0·(1 − x0/c1)  Prediction: x0*(0.79 - 0.0106*x0).  YES: both are c·x0 + c·x0².
GT: c0·x0 − c1·x0²     Prediction: 2.1*x0 - 0.5*x0**2 + 1e-6*x0**3.  NO: extra term x0³.
....
....

###
<Output>
Return only this JSON object:
{"gt_canonical": "<ground-truth term set after steps 1-3>",
 "pred_canonical": "<prediction term set after steps 1-3>",
 "reasoning": "<short step-by-step comparison>",
 "mismatch_type": "none | missing_term | extra_term | ....",
 "answer": "YES or NO"}
```

Equation Specification

Figure 6: Prompt for LLM-judge to calculate the symbolic accuracy.

Symbolic Accuracy. Following LLM-SRBench [48], we use an LLM judge to decide whether each discovered ODE has the same structure as the ground truth. Exact string match and normalized tree edit distance penalize algebraically equivalent rewrites, so we instead use `Claude-Opus-4.6` (temperature 0) as the judge. Each method's output is first mapped to a common $x_0, x_1, \ldots$ variable naming by a method-specific normalizer and given to the judge with its fitted constants intact, together with the ground-truth right-hand side and the range each state variable takes on the benchmark trajectories. The judge follows a fixed procedure: it rewrites both sides in canonical form (expanding products, collecting like terms, and applying identities such as $\log(xy) = \log x + \log y$ only where they hold on the stated range), replaces every numerical coefficient with a generic constant while keeping exponents, and compares the resulting sets of terms. It answers yes only if the terms match one-to-one with exponents agreeing within ± 0.005; a missing or extra term, a different function, or an added or dropped constant term is a mismatch. Each state dimension is judged independently, and a system's score is the fraction of its dimensions recovered (partial credit; e.g., 0.67 if 2 of 3 are correct), with systems that have no saved equation scoring 0. Further, to validate the LLM judge, we compared its verdicts against a manual review, where the LLM judge agreed with manual review on 98.4% of the assessed comparisons.

B Implementation Details

B.1 Computational Resources

All experiments run on a server with two Intel Xeon Gold 5220R processors (48 physical cores) and 502 GB RAM. Non-LLM regression baselines run on this server with a 12-hour timeout per dataset. Furthermore, we used four NVIDIA RTX 8000 GPUs (48 GB each) for setting up the inference for neural network-based or transformer-based methods. For LLM-based methods, we used hosted APIs, for example, GPT through OpenAI and `Qwen3-32B` through Modal and AWS.

B.2 Baselines

We compare our method against established baselines spanning sparse regression, evolutionary symbolic regression, neural sequence modeling, and active data acquisition. For all static methods, we adopt a unified experimental protocol in which each model is trained on the provided set of 200 data points from the default initial condition. For active regression methods, each method follows its own data acquisition process to acquire equivalent total samples and is ultimately evaluated on the same set of data points. We follow MDBench's implementations[1] for all baselines but expand the candidate operator sets beyond those used in the original benchmark. MDBench restricts its operator pools to a subset that appears in its equation catalog, introducing a systematic bias as the search spaces match the benchmark's equations. All baselines except APPS-ODE [26], ODEFormer [15], and Operon [10] use the expanded set:

- **Unary Operators**: $\sin, \cos, \tan, \exp, \log, \sqrt{\cdot}, |\cdot|, \tanh, \sinh, \cosh, \cdot^2, \cdot^3, \cdot^{-1}, -\cdot, \sqrt[3]{\cdot}, \log_2, \log_{10}, 2^{\cdot}$
- **Binary Operators**: $\{+, -, \times, \div, \hat{}\}$

Table 9: Hyperparameter settings for baselines.

Model	Hyperparameter	Values
SINDy	threshold	`np.logspace(-7,0,16)`
	basis functions	[polynomial], [polynomial, sin, cos, tan, exp, log, sqrt, abs, tanh, sinh, cosh, square, cube, inv, neg, cbrt, log2, log10, exp2]
	polynomial order	1, 2, 3, 4
	alpha	$10^{-5}, 10^{-4}$
	optimizer	STLSQ
	max iterations	200
PySR	#iterations, #cycles per iteration	100, 1000
	#populations, population size	20, 100
	max size, max depth	40, 20
	binary operators	[+, -, ×, /, $\hat{}$]
	unary operators	sin, cos, tan, exp, log, sqrt, abs, tanh, sinh cosh, square, cube, inv, neg, cbrt, $\log_2$, $\log_{10}$, $\exp^2$
End2End	max input points	200
	#trees to refine	10
	rescale	True
ODEFormer	beam temperature	0.05, 0.1, 0.2, 0.3, 0.5
	beam size	50
Operon	symbols	add, sub, mul, div, aq, pow, abs, cbrt, cos cosh, exp, log, sin, sinh, sqrt, tan, tanh, square
	brood size	10
	max depth, max length	10, 50
	pool size, population size	1000, 1000
	tournament size	3
	mutation probability	0.25
	optimizer	LM (Levenberg–Marquardt)
APPS-ODE	#epochs	50
	reward threshold	$1/(1 + 10^{-6})$
	grammar max length	10
	top-K size	10
	function set	[add, sub, mul, div, sin, exp, poly, const]

Passive Symbolic Discovery

SINDy [8] uses the `PySINDy` implementation with STLSQ (sequential thresholding least squares) optimizer. The sparsity threshold is swept over a log-uniform grid from 10^{-7} to 1 (16 values), and ℓ_2 regularization strength $\alpha \in \{10^{-5}, 10^{-4}\}$. We extend the basis library beyond MDBench's polynomial-only setting to also search over the full expanded nonlinear library above, combined with polynomials up to degree 4.

[1] `https://github.com/gryaklab/mdbench`

Operon [10] uses the operator set for `operon` on top of implementation given in [4]. Model selection uses minimum description length on the Pareto front.

PySR [14] uses MDBench's implementation of `PySR` with multi-population evolutionary search over expression trees, with an expanded operator set. Nested constraints prevent pathological compositions (e.g., $\exp(\exp(\cdot))$), and the complexity-fitness tradeoff is scored via a Pareto fitness metric.

End2End (E2E) [28] uses the pretrained Transformer from `symbolicregression`, following the implementation in [4]. We cap the input at 200 points, enable input rescaling, and refine the top 10 predicted expression trees using the model's constant-refinement procedure.

ODEFormer [15] is run in inference mode using the pretrained checkpoint, without fine-tuning[2]. As a fixed pretrained model, its internal operator vocabulary cannot be modified.

LLM-guided Symbolic Discovery

LLM-only follows the protocol from NewtonBench[3], prompting `GPT-4o-mini` iteratively with feedback from the previous round to guide the future equations. The LLM is implemented using a temperature 1.0 to maximize generative diversity.

LLM-ODE [5] uses the original implementation from `llm-ode` with LLM-proposed structural templates and numerical parameter optimization. The multi-island evolutionary strategy and dynamic experience buffer are used as in the original implementation.

Active Symbolic Discovery

Query-By-Committee (QBC) [20, 22, 21] adapts `QBC` active learning to ODE initial condition selection, using `PySR` operating on gradient-matched data, where the committee is drawn from the Pareto front. `QBC` also uses the same oracle as LLM-ACES to query new initial conditions.

Bayesian Optimization (BO) combines `PySR` with a Gaussian process surrogate over initial condition space. Each iteration, the GP predicts the expected value of the NMSE of the `PySR` fit from a candidate IC and selects the next query using Expected Improvement (EI) from a pool of 256 uniformly sampled ICs. After querying the oracle, `PySR` is refit on all accumulated data. The GP is updated with the observed NMSE as the reward.

APPS-ODE [26] implementation uses the original grammar-RL pipeline with grammar function set $\{+, -, \times, \div, \sin, \exp, \mathrm{poly}, \mathrm{const}\}$. This grammar-constrained setting reduces the symbolic search space and therefore provides a favorable inductive bias when the target dynamics lie within or near the supported grammar. We train APPS-ODE for 50 policy-gradient epochs, querying initial conditions with 100 observations as given in their official repository[4]. Coefficients are optimized with BFGS; the reward signal is inverse NMSE. For both datasets, trajectories are queried from fresh initial conditions each epoch using the same oracle used for LLM-ACES.

B.3 LLM-ACES

LLM-ACES follows a closed-loop *hypothesis–evaluation–acquisition* framework for discovery of dynamical systems. Instead of directly generating full symbolic equations, the language model operates at the level of *operator priors*, which define structured hypothesis spaces. These priors are instantiated, evaluated, and iteratively refined through feedback and active data acquisition.

Hypothesis Exploration. Figure 7 illustrates how LLM-ACES constructs and iteratively refines the hypothesis space for symbolic ODE discovery. At each iteration, the LLM is conditioned on a structured prompt that defines its role and enforces a standardized interface for prior generation. The prompt (Figure 7) consists of four key components: (i) Specification, which formalizes the task, system dynamics, and evaluation interface (including the objective defined via negative NMSE and the function signature for candidate equations); (ii) Current Best Equations, which provide the current top-performing solutions along with their scores; (iii) Failure Cases, which capture poorly performing hypotheses to discourage ineffective operator patterns; and (iv) Existing Operator Priors, which

[2] `https://github.com/sdascoli/odeformer`

[3] `https://github.com/HKUST-KnowComp/NewtonBench`

[4] `https://github.com/jiangnanhugo/APPS-ODE`

maintain a memory of previously discovered operator priors in an experience buffer. Conditioned on these inputs, the LLM generates new priors, defined as operator-level templates that specify the functional structure of candidate equations. The Output Format enforces that these operator priors remain high-level and structured, requiring explicit operator sets per dimension, constrained cardinality, and standardized naming, while explicitly prohibiting direct equation synthesis. This design ensures that the LLM contributes abstract inductive biases instead of explicit solutions, enabling controlled expansion of the hypothesis space across iterations rather than collapsing to a narrow set of forms. The prompt shown in Figure 8 promotes diversity in hypothesis exploration. The model is explicitly instructed to generate operators from previously unused mathematical families, encouraging exploration of new functional regimes while avoiding redundancy with existing operator priors. Each prior defines a constrained symbolic template over a restricted operator vocabulary, including unary transformations such as trigonometric, exponential, polynomial, and logarithmic functions, as well as standard binary operators. To ensure validity and numerical stability, structural constraints are enforced during parsing. These include requiring the presence of basic arithmetic operators (such as addition and multiplication), restricting exponentiation ranges, and filtering out syntactically invalid or duplicate proposals. Together, these constraints produce structured and tractable symbolic search spaces while preventing degenerate or unstable formulations. Success cases guide exploitation, while failure cases and existing operator sets jointly shape exploration by incorporating both negative feedback and historical priors.

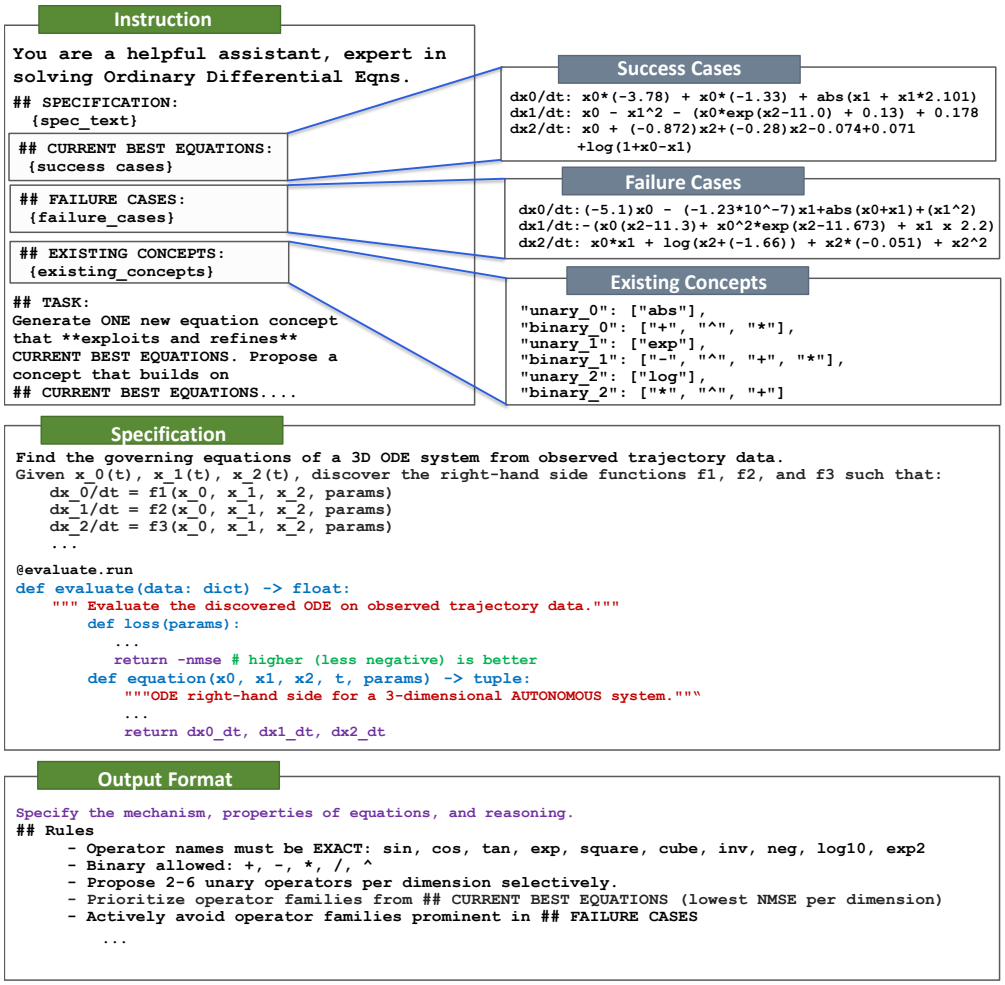


Figure 7: **Prompt example of LLM-ACES for 3D ODE discovery.** The prompt includes instructions, a task specification describing the objective, system characteristics, and data samples, as well as the target function and required structured output format. It also shows intermediate elements such as the current best equations, existing concepts, and failure cases. The prompts refer to operator priors as concepts.

Hypothesis Generation. Given a set of validated operator priors, LLM-ACES instantiates each prior as a constrained symbolic search space and performs dimension-wise regression using PySR [14]. For a d-dimensional system and K operator priors per iteration, this results in $d \times K$ independent regression problems, each restricted to the operator set defined by its corresponding prior. To ensure fair comparison across priors, all regression tasks are executed under a fixed computational budget, providing uniform search capacity. For symbolic regression, each operator prior set is evaluated using `PySR` with 20 iterations and 15 populations. Candidate equations are assessed using normalized mean squared error (NMSE) on validation splits. We select the best equation for each state dimension independently by retaining the one with the best validation performance. This decoupled selection strategy lets LLM-ACES preserve partially correct components of the system, avoiding the failure mode in which errors in other dimensions discard accurate sub-dynamics. LLM-ACES is instantiated with `GPT-4o-mini` and `Qwen3-32B`, with `Qwen3-32B` served via Modal and DeepInfra, and GPT accessed through the OpenAI API. LLM-ACES runs `PySR` for 20 iterations for each operator set generated by the LLM. Each run consists of 10 rounds of LLM calls with up to three operator sets generated per iteration at a sampling temperature of 0.8, balancing exploration of novel functional forms with exploitation of prior structural knowledge.

Hypothesis Exploration

```
You are helping with scientific equation discovery for ordinary differential equations.

## SPECIFICATION:
{spec_text}

## CURRENT BEST EQUATIONS:
{success cases}

## EXISTING CONCEPTS:
{existing_concepts}

## TASK:

Generate ONE new equation concept whose unary operators come from a DIFFERENT mathematical family than
any concept listed above.

Operator families for reference:
- Trigonometric: sin, cos, tan
- Exponential/logarithmic: exp, exp2, log, log2, log10
- Power/polynomial: square, cube, sqrt, cbrt
- Hyperbolic: tanh, sinh, cosh
- Misc: abs, inv, neg

Rules:
- Do NOT reorder or rename operators from an existing concept
- Choose at least 2 operators from a family not yet represented in the existing concepts above.
- If you have no genuinely new concept to offer, respond with exactly: {concept_stop_token}
- Propose EXACTLY ONE concept defining how the right-hand side of each equation
  ({dim_description}) is constructed in terms of:
    - types of operators used
    - how those operators are combined (structure)
    - an abstract description of the resulting dynamics
      ......

### Output Format

{dim_blocks}
mechanism: <string> # 4-8 words describing the formulation
properties: <string> # 4-8 words describing behavior
reasoning: <1-3 sentences explaining why this concept is different and what new functional space it
explores>
{dim_operator_blocks}
```

Figure 8: **Prompt example of LLM-ACES for hypothesis exploration.** The prompt specifies the task specification, current best equations, existing operator set, and instructs the model to generate a new equation operator prior. This prompt enhances the diversity of operators.

Data Acquisition via Active Experiment Selection. To address the fundamental limitation of static datasets, LLM-ACES incorporates an active learning mechanism that expands the dataset in regions where current hypotheses disagree. A pool of candidate initial conditions is sampled from predefined bounds, and for each candidate, the learned equation systems are simulated forward using short-horizon numerical integration (e.g., forward Euler with fixed step size). At each round, the acquisition committee consists of the candidate systems fitted in that round. We sample a pool of 256 feasible initial conditions and evaluate the predictive-divergence score at three randomly selected conditions, followed by seven conditions selected using Gaussian-process UCB. The score is the median normalized pairwise discrepancy among finite candidate rollouts, as defined in Eq. 4. The highest-scoring evaluated initial condition is queried through the oracle. For each selected initial condition, trajectories are generated using the LSODA solver over 20 uniformly spaced time points in the interval $t \in [0, 1]$. The selected initial conditions are then queried through a ground-truth ODE oracle, producing new trajectory data. Newly acquired samples are incorporated into the dataset via interleaved splitting, where even-indexed samples are assigned to the training set and odd-indexed samples to the validation set. Starting with the initial 20 data points, each of the ten acquisition steps contributes 10 training and 10 validation samples, ensuring balanced and incremental data augmentation.

Feedback-Driven Refinement. LLM-ACES maintains a feedback-driven memory that aggregates information from prior iterations. Candidate equations are ranked per dimension based on validation performance. High-performing equations are abstracted into positive exemplars (top 2 candidates), while the worst-performing two candidates are recorded as failure cases. These summaries are injected into subsequent prompts, guiding the language model toward effective structural motifs and away from previously unproductive operator combinations. The exploration–exploitation balance is further controlled by varying prompts across iterations, alternating between refining promising operator families and exploring novel structures.

C Additional Results

C.1 Distributional Analysis Across Benchmark Systems

Figures 9 and 10 illustrate the distribution of reconstruction, generalization, and extrapolation NMSE, together with expression complexity, across all systems in ODEBench and ODEBase. Across both benchmarks, LLM-ACES exhibits error distributions that are consistently shifted toward lower values, demonstrating that its improvements are not driven by a few favorable systems but are observed broadly across the benchmarks. In particular, the first and third quartiles (Q1–Q3) of LLM-ACES are substantially lower than those of competing approaches across reconstruction, generalization, and extrapolation settings, indicating that the majority of systems benefit from the proposed framework. Among the baselines, Bayesian optimization is the strongest competitor, although its interquartile ranges are generally wider and shifted toward higher errors, especially under distribution shift. Passive symbolic regression methods exhibit even larger quartiles centered around higher NMSE values, reflecting limited robustness beyond the training trajectories. From the perspective of expression complexity, LLM-ACES maintains relatively compact equations with a comparatively narrow interquartile range, whereas `QBC` and `E2E` often produce considerably more complex expressions and exhibit much greater variability. The consistency of these trends across both ODEBench and ODEBase highlights the robustness of LLM-ACES across datasets, system dimensions, and evaluation settings.

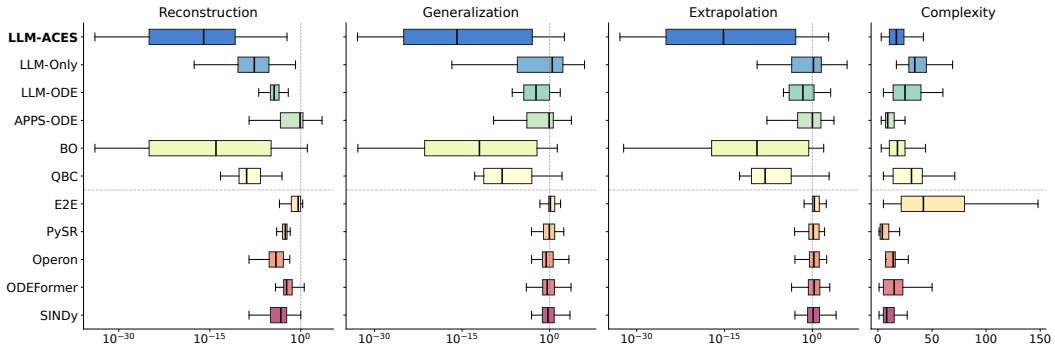


Figure 9: Distribution of reconstruction, generalization, extrapolation NMSE, and expression complexity across methods for ODEBench datasets.

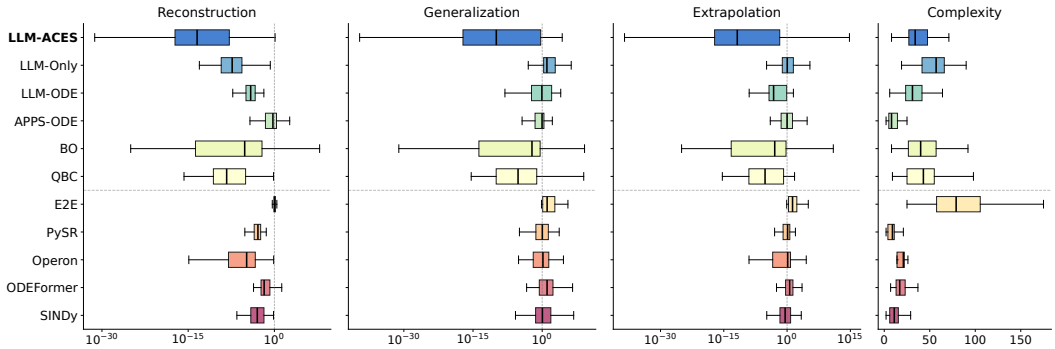


Figure 10: Distribution of reconstruction, generalization, extrapolation NMSE, and expression complexity across methods for ODEBase datasets.

C.2 Role of LLMs in Search-Space Design

At each iteration, LLM-ACES restricts the symbolic regression backend to an operator set proposed by the LLM. We evaluate whether the resulting gains stem from LLM-guided, context-dependent search-space design rather than from search-space restriction alone. Across three controlled variants, we hold the remainder of the pipeline fixed, including candidate fitting, initial-condition acquisition, re-scoring, and the overall query budget, and vary only the mechanism used to construct the operator set. The controls isolate the contribution of LLM guidance from the effects of simple restriction.

Effect of operator-space restriction. We constructed a privileged oracle-informed operator library from the union of ground-truth operator families appearing across the benchmark: `polynomial`, `sin`, `cos`, `exp`, and `log`. This setting provides substantially stronger prior information than would be available in a realistic discovery scenario and isolates the effect of restricting the symbolic search space. Table 10 compares standard symbolic-regression baselines under their default operator sets and the oracle-informed library. The effect of restriction is limited and inconsistent across methods: PySR improves by approximately 18%, Operon remains nearly unchanged, and SINDy degrades. Despite access to the ground-truth operator families, all three baselines remain more than ten orders of magnitude above LLM-ACES in reconstruction NMSE. These results indicate that static restriction of the operator space is insufficient to account for the performance of LLM-ACES.

Table 10: Median reconstruction NMSE (lower is better) under the default operator set versus an oracle-informed library containing the ground-truth operator families.

Method	Full set	Oracle-informed set
SINDy	5.07×10^{-4}	6.23×10^{-4}
PySR	2.84×10^{-3}	2.34×10^{-3}
Operon	7.95×10^{-5}	7.73×10^{-5}
LLM-ACES (ours)	$\mathbf{1.33 \times 10^{-17}}$	

Oracle-informed selection within LLM-ACES. We further isolated operator selection within the LLM-ACES pipeline by replacing the LLM with the same oracle-informed operator library while leaving all remaining components unchanged, evaluated on a fixed subset of 15 systems, comprising five systems each from the 1D, 2D, and 3D benchmarks (same as Section 4.4). The oracle-informed variant is consistently outperformed by LLM-ACES across reconstruction, generalization, and extrapolation prediction (Table 11). Access to the correct operator families in aggregate therefore does not reproduce the benefit of LLM guidance. The relevant distinction is the selection of an operator subspace conditioned on the current system and search state, rather than the availability of a globally correct operator library.

Table 11: LLM-ACES versus an oracle-informed-library variant of the same pipeline, on the 15-system subset. We report the Median NMSE values.

Variant	Recon. NMSE	Gen. NMSE	Ext. NMSE
Oracle-informed LLM-ACES	2.53×10^{-18}	2.12×10^{-17}	1.02×10^{-17}
LLM-ACES (ours)	$\mathbf{4.01 \times 10^{-19}}$	$\mathbf{7.39 \times 10^{-18}}$	$\mathbf{2.29 \times 10^{-18}}$

Adaptive non-LLM operator selection. We designed an Operator Bandit (OB) using independent UCB1 bandits over the unary and binary operators available to LLM-ACES. At each iteration, the operator set is formed from operators with the highest UCB scores, balancing empirical reward and exploration. PySR performs symbolic search within the selected operator set, and the resulting validation score is propagated to the selected operators through semi-bandit credit assignment. The reward is defined as $1/(1+\mathrm{NMSE}_{\mathrm{val}})$, with zero reward assigned to failed searches. The downstream search procedure and computational budget remain matched to LLM-ACES. Table 12 shows that LLM-ACES outperforms both baselines across all three metrics on the subset of 15 datasets. OB also underperforms the fixed oracle-informed variant, indicating that generic adaptation based on scalar validation feedback does not recover the gains of LLM-guided selection. The distinction is therefore

not adaptivity alone, but adaptation conditioned on the current candidate equations and accumulated search context.

Table 12: LLM-guided operator selection versus a non-LLM adaptive selector and the fixed oracle-informed library, on the 15-system subset under a matched downstream pipeline and budget. We report the Median NMSE values.

Variant	Recon. NMSE	Gen. NMSE	Ext. NMSE
Oracle-informed (fixed, non-adaptive)	2.53×10^{-18}	2.12×10^{-17}	1.02×10^{-17}
Operator Bandit (adaptive, non-LLM)	7.67×10^{-17}	2.70×10^{-17}	6.40×10^{-17}
LLM-ACES (ours)	$\mathbf{4.01 \times 10^{-19}}$	$\mathbf{7.39 \times 10^{-18}}$	$\mathbf{2.29 \times 10^{-18}}$

C.3 Hyperparameter Sensitivity

Number of acquisition rounds (T). The acquisition budget determines the number of trajectories that can be queried from the oracle while maintaining the overall number of samples (see Table 13). Since the intended setting assumes costly experiments, performance at small T is particularly relevant. We evaluate $T \in \{1, 2, 5, 10\}$ (each setting is trained on the same number of data points) on the fixed 15-system subset, with $T{=}10$ used as the default. Performance improves substantially as the acquisition budget increases, with reconstruction NMSE decreasing monotonically across the sweep. Importantly, strong performance is already obtained with a single acquisition round. At $T{=}1$, LLM-ACES achieves errors substantially below the full-budget baselines reported in the main evaluation, demonstrating that its gains persist in the low-query regime. Additional acquisitions further refine the candidate hypotheses, yielding the strongest reconstruction, generalization, and extrapolation performance at the default budget of $T{=}10$.

Table 13: Performance of LLM-ACES as a function of the acquisition budget T. We report the Median NMSE values.

T	Recon. NMSE	Gen. NMSE	Ext. NMSE
1	1.79×10^{-16}	8.07×10^{-15}	8.18×10^{-14}
2	3.40×10^{-17}	1.41×10^{-15}	6.84×10^{-16}
5	2.35×10^{-17}	1.69×10^{-16}	6.89×10^{-16}
10	4.01×10^{-19}	7.39×10^{-18}	2.29×10^{-18}

Number of priors per round (K). The parameter K determines the number of operator hypotheses generated and fitted at each round. We evaluate $K \in \{1, 3, 10\}$ with $T{=}10$ fixed. Increasing K consistently improves performance across all three metrics. The largest qualitative change occurs between $K{=}1$ and $K{=}3$: with a single candidate prior, the population cannot provide a meaningful cross-hypothesis disagreement signal for active acquisition, whereas multiple priors support both broader symbolic search and hypothesis-driven experiment selection. Increasing K to 10 yields effectively exact recovery on this subset; however, the computational cost increases significantly as compared to the $K = 3$ setting.

Table 14: Effect of the number of operator priors per round, K. We report the Median NMSE values.

K	Recon. NMSE	Gen. NMSE	Ext. NMSE
1	7.40×10^{-16}	7.88×10^{-13}	1.32×10^{-10}
3	4.01×10^{-19}	7.39×10^{-18}	2.29×10^{-18}
10	7.12×10^{-33}	8.03×10^{-33}	5.54×10^{-33}

Robustness to prompt variation. To study the robustness to prompt variations, we removed each of four instructions independently: `no_required_operators` removes the requirement that $\{+, *\}$ appear in the binary operator set; `no_operator_count_bound` removes the constraint of 2–6 unary operators; `no_family_diversity` removes the operator-family reference list and the cross-round

diversity instruction; and `no_best_equation_study` removes conditioning on the operator families present in the current best equations. Performance remains strong under all four prompt perturbations, with every variant retaining errors in the low-NMSE regime. As Table 15 shows, LLM-ACES does not depend critically on any single prompt instruction. All variants stay below 10^{-17}, indicating a strong performance over passive baselines.

Table 15: Sensitivity to prompt variations. We report the Median NMSE values.

Variant	Recon. NMSE	Gen. NMSE	Ext. NMSE
`no_required_operators`	3.62×10^{-19}	1.44×10^{-20}	3.38×10^{-19}
`no_operator_count_bound`	4.45×10^{-19}	1.64×10^{-19}	7.84×10^{-20}
`no_family_diversity`	1.53×10^{-19}	5.61×10^{-19}	8.75×10^{-19}
`no_best_equation_study`	3.67×10^{-21}	3.56×10^{-21}	4.74×10^{-21}
Full prompt	4.01×10^{-19}	7.39×10^{-18}	2.29×10^{-18}

C.4 Robustness to Noise and Irregular Sampling

To assess robustness under realistic observation noise, we follow the MDBench [4] evaluation protocol by corrupting clean trajectories at multiple SNR levels and evaluating the discovered right-hand side functions using reconstruction NMSE, while also reporting expression complexity and symbolic accuracy. This setup tests whether a method preserves the underlying dynamics rather than merely fitting noisy measurements, since MDBench explicitly evaluates predictive fidelity under varying noise conditions and penalizes overly complex symbolic forms. In addition, we follow the ODEBench [15] sampling protocol by varying the observation subsampling ratio ρ, where a fraction of trajectory points is dropped uniformly at random, thereby measuring performance under both dense and sparse temporal observations. As shown in Figure 11, across SNR levels and sampling regimes, LLM-ACES achieves consistently stronger reconstruction accuracy than `SINDy`, `PySR`, `Operon`, and `ODEFormer`, indicating that its generated hypotheses remain stable even when the observed trajectories are noisy or partially sampled. Importantly, this improvement is obtained without a corresponding explosion in expression complexity, suggesting that LLM-ACES discovers more robust equation structures rather than relying on unnecessarily long symbolic expressions. Overall, these results show that LLM-ACES offers a favorable robustness–parsimony trade-off compared to existing baselines, with particularly clear gains in degraded observation settings where conventional symbolic regression and sparse-regression methods become less reliable.

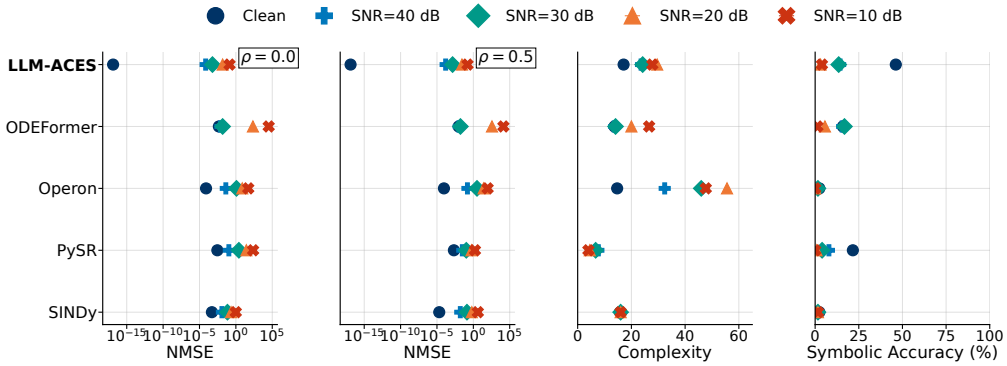


Figure 11: Evaluation under noisy observations. The complexity and symbolic accuracy results are for the $\rho = 0.0$ setting.

C.5 Full-system Recovery

While Symbolic Accuracy (SA) measures recovery at the level of individual component equations, it can assign partial credit to a multi-dimensional system whose dynamics are only partially recovered. We therefore additionally evaluate binary full-system recovery. As shown in Table 16, LLM-ACES recovers 38/63 complete systems, compared with 28/63 for the strongest baseline, `BO`. It leads on 1D (11 vs. 8) and 2D systems (20 vs. 14) and also exceeds `BO` on 3D/4D systems (7 vs. 6). Comparing BO with PySR, which share the same regression backend, shows how much actively acquired data contributes; LLM-ACES adds LLM-induced hypothesis spaces and disagreement-driven acquisition

on top, giving the highest recovery. Because this metric gives no credit to partial recovery, these results show that LLM-ACES's lower NMSE is accompanied by more complete recovery of the governing equations, rather than only more accurate trajectory fitting.

Table 16: **Binary full-system recovery across ODEBench (63 systems)**. A system counts as recovered only when every component equation matches the ground-truth structure under the symbolic-accuracy protocol.

Method	1D	2D	3D/4D	Total
PySR	1	1	0	2
ODEFormer	2	3	0	5
Operon	2	0	0	2
SINDy	3	3	0	6
LLM-ODE	4	2	0	6
BO	8	14	6	28
LLM-ACES (GPT)	**11**	**20**	**7**	**38**

C.6 Robustness to the internal validation protocol.

During closed-loop discovery, LLM-ACES uses a validation split only to score and select among candidate equations; the reconstruction, generalization, and extrapolation test sets are fixed and never enter the loop. By default, each queried trajectory is split by interleaving, with alternate time points assigned to training and validation. To test whether this choice affects which equations are selected, we rerun LLM-ACES on all 63 ODEBench systems under two alternative protocols: (i) *held-out IC validation*, where candidates are validated on oracle trajectories from two fixed initial conditions that are never used for training, and (ii) *temporal validation*, where the first half of each trajectory is used for training and the second half for validation, so that candidates are selected by how well they extrapolate forward in time. Table 17 reports median NMSE, the number of fully recovered systems, and symbolic accuracy (SA) computed as the fraction of correctly recovered component equations within each system, averaged across systems. The results are stable across protocols: median NMSE remains below 10^{-15} on all three test sets, full-system recovery ranges from 37/63 to 43/63, and SA ranges from 66.1% to 73.7%. Held-out IC validation, which is independent of the training trajectories, yields the highest recovery and SA, indicating that the interleaved default does not inflate symbolic recovery. Hence, the equations selected by LLM-ACES perform consistently better than the baselines shown in Table 2 and the resulting NMSE and symbolic-recovery conclusions are not too sensitive to the validation protocol.

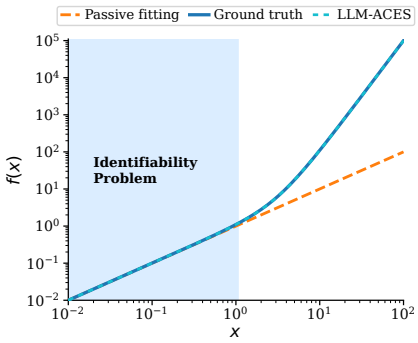
Table 17: LLM-ACES on ODEBench (63 systems) under three internal validation protocols. NMSE is the median over systems on the fixed reconstruction (Recon.), generalization (Gen.), and extrapolation (Ext.) test sets. Recovered counts systems whose every equation is recovered; SA is symbolic accuracy.

	Median NMSE ↓				
Validation protocol	Recon.	Gen.	Ext.	Recovered ↑	SA (%) ↑
Interleaved (default)	1.33×10^{-17}	8.28×10^{-17}	2.46×10^{-16}	38/63	70.8
Held-out IC	1.35×10^{-16}	7.91×10^{-17}	2.74×10^{-16}	43/63	73.7
Temporal (split halves)	1.52×10^{-16}	1.50×10^{-16}	6.27×10^{-16}	37/63	66.1

D Additional Analyses

D.1 Data Acquisition for Dynamical Equation Discovery

Most existing equation discovery methods operate under a static-data paradigm, implicitly assuming that the available observations are sufficient for identifying the underlying dynamics. Yet low fitting error alone provides no guarantee of correct structural recovery, as different equations may explain the same observations equally well. We therefore examine the role of data acquisition in equation discovery and demonstrate that actively selecting experiments leads to both improved predictive performance and more faithful recovery of the true governing equations.



Method	Reconstructed equation	Ground truth
SINDy / E-SINDy	$1.0096x$	$x + 0.1x^3$
PySR	x	
Operon	$1.023x + 0.0038$	
ODEFormer	$1.0053x$	
LLM-ODE	$0.0001x - 0.261$	
LLM-ACES	$x + 0.1x^3$	

Figure 12: **Identifiability gap in passive equation fitting.** (*Left*) Comparison of the ground-truth dynamics, LLM-ACES, and passive fitting baselines. The shaded region denotes the low-x regime where passive fitting remains visually close to the ground truth. *(Right)* Reconstructed equations returned by each method.

Identifiability Gap Analysis. When trajectories explore only a limited region of the state space, structurally distinct equations may produce nearly indistinguishable observations. Figure 12 illustrates this phenomenon using a system with dynamics $\dot{x} = x + 0.1x^3$. Training data collected from a low-x regime render the nonlinear contribution effectively invisible, causing a variety of passive methods to recover approximately linear equations despite most of it achieving negligible fitting error. Although these models agree within the observed region, their predictions diverge dramatically outside it. This demonstrates an *identifiability gap*: the inability to distinguish competing hypotheses due to insufficiently informative data. Importantly, the limitation arises from the observations themselves rather than from deficiencies in the symbolic regression algorithms. Further optimization on the same observations may not reliably distinguish competing structures when their predictions differ only below the available observation precision.

Effect of trajectory diversity. Figure 13 examines the effect of increasing the number of initial conditions available for equation discovery while keeping the total number of observations at 100 samples. We used data from 10 different randomly sampled initial conditions to test the multiple trajectories setting. The evaluation is done on the second initial condition trajectory, which is not a part of the sampled initial conditions (for generalization), whereas extrapolation is on the extended time range for the first initial condition. Replacing a single trajectory with trajectories generated from multiple randomly sampled initial conditions consistently improves both generalization and extrapolation performance across symbolic regression methods. These results suggest that the diversity of trajectories provides more useful information than densely sampling a single trajectory. Nevertheless, the trajectories are obtained passively and do not explicitly target regions that are informative for distinguishing among candidate equations. As a result, improvements are limited by the coverage achieved through random sampling.

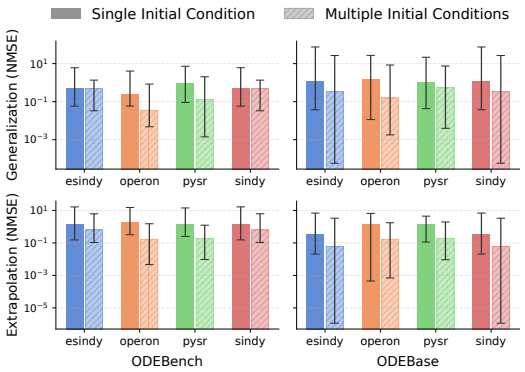


Figure 13: **Comparison of equation recovery from a single trajectory and multiple trajectories.** Trajectory diversity improves performance across methods.

Effect of acquisition strategy. Figure 14 shows that the method used to select trajectories has a much larger impact than simply increasing their number. Single-trajectory datasets provide only local information, whereas data from multiple initial conditions improve coverage without explicitly targeting unresolved uncertainties. Approaches such as Bayesian optimization and query-by-committee (QBC) aim to guide experimentation using uncertainty estimates. Rather than seeking merely uncertain regions, *predictive divergence* actively searches for experiments that discriminate between alternative explanations. Across ODEBench datasets, this strategy consistently achieves the lowest generalization error, compared with both passive baselines and existing active acquisition schemes. These results suggest that the central challenge in equation discovery is not obtaining more data, but obtaining the *right* data. Symbolic hypotheses should therefore guide the collection of new

trajectories, transforming equation discovery from a passive fitting problem into an iterative process of hypothesis generation and experimental design.

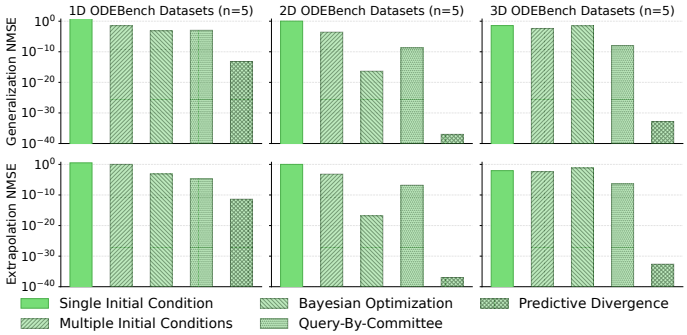


Figure 14: **Comparison of different trajectory acquisition strategies for PySR on 1D, 2D, and 3D ODEBench datasets.** Selecting trajectories based on predictive divergence consistently achieves the lowest NMSE.

D.2 Wall-Clock and Compute Analysis

LLM-ACES incurs additional computation relative to passive symbolic regression because each acquisition round searches multiple LLM-induced hypothesis spaces and evaluates their disagreement before acquiring new evidence. We therefore evaluate whether its performance gains can be explained simply by allocating more computation to symbolic search. We consider the compute-efficient $T{=}1$ configuration of LLM-ACES, which requires 250.5 seconds using 60 PySR iterations and only three LLM calls (per dimension), and increase its symbolic-regression backend, PySR, beyond this wall-clock budget. Table 18 shows that additional passive search compute does not close the performance gap. Increasing PySR from its standard search budget to 500 and 750 iterations raises its wall-clock time to 327.3 and 624.9 seconds (per dimension), respectively, both exceeding the 250.5 seconds required by LLM-ACES ($T{=}1$). Nevertheless, more PySR iterations do not guarantee an improvement in performance, as its reconstruction NMSE remains on the order of 10^{-3}, compared with $1.79{\times}10^{-16}$ for LLM-ACES. Even the 750-iteration PySR run therefore uses approximately $2.5\times$ more wall-clock time while remaining roughly 13 orders of magnitude worse in NMSE. LLM-ACES also requires substantially less wall-clock time than LLM-ODE that runs for 1000 calls to generate 1000 candidates despite achieving considerably lower error.

Table 18: Compute-controlled wall-clock comparison.

Method	NMSE	Time (s)	Configuration
PySR	$2.85{\times}10^{-3}$	327.3	500 iter.
PySR	$2.12{\times}10^{-3}$	624.9	750 iter.
LLM-ODE	$3.82{\times}10^{-5}$	1683.0	$\sim$1K LLM calls
LLM-ACES ($T{=}1$)	$\mathbf{1.79{\times}10^{-16}}$	**250.5**	60 iter. + 3 LLM calls

D.3 Qualitative Analysis

We provide qualitative visualizations for a representative trajectory from each ODE in Figure 15 for ODEBench and Figure 16 for ODEBase. For each system, we overlay the trajectory generated by LLM-ACES's predicted equation onto the corresponding ground-truth trajectory for comparison.

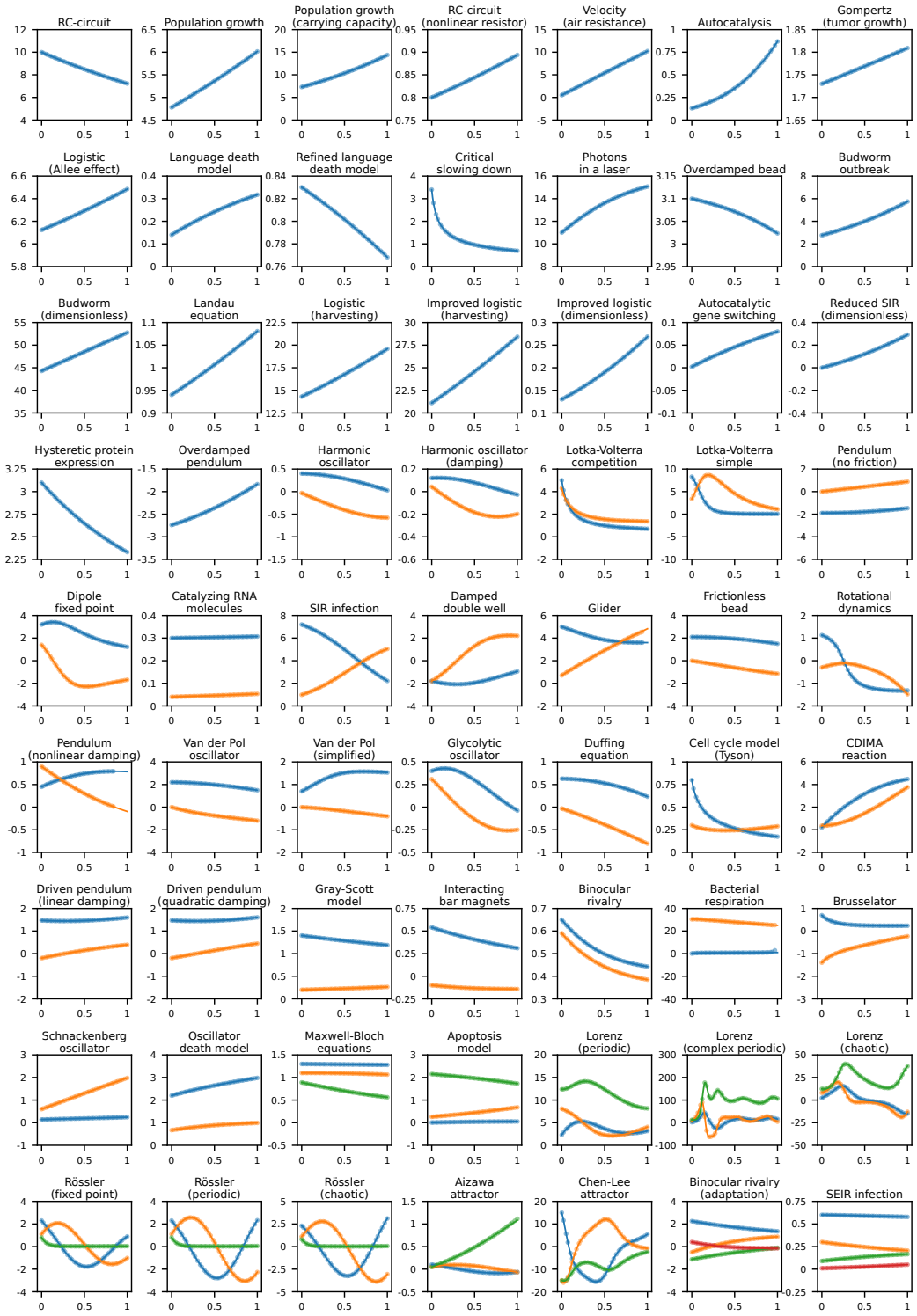


Figure 15: Predictions of LLM-ACES (indicated in bold) and the ground-truth equations for all systems in ODEBench for the first set of initial conditions.

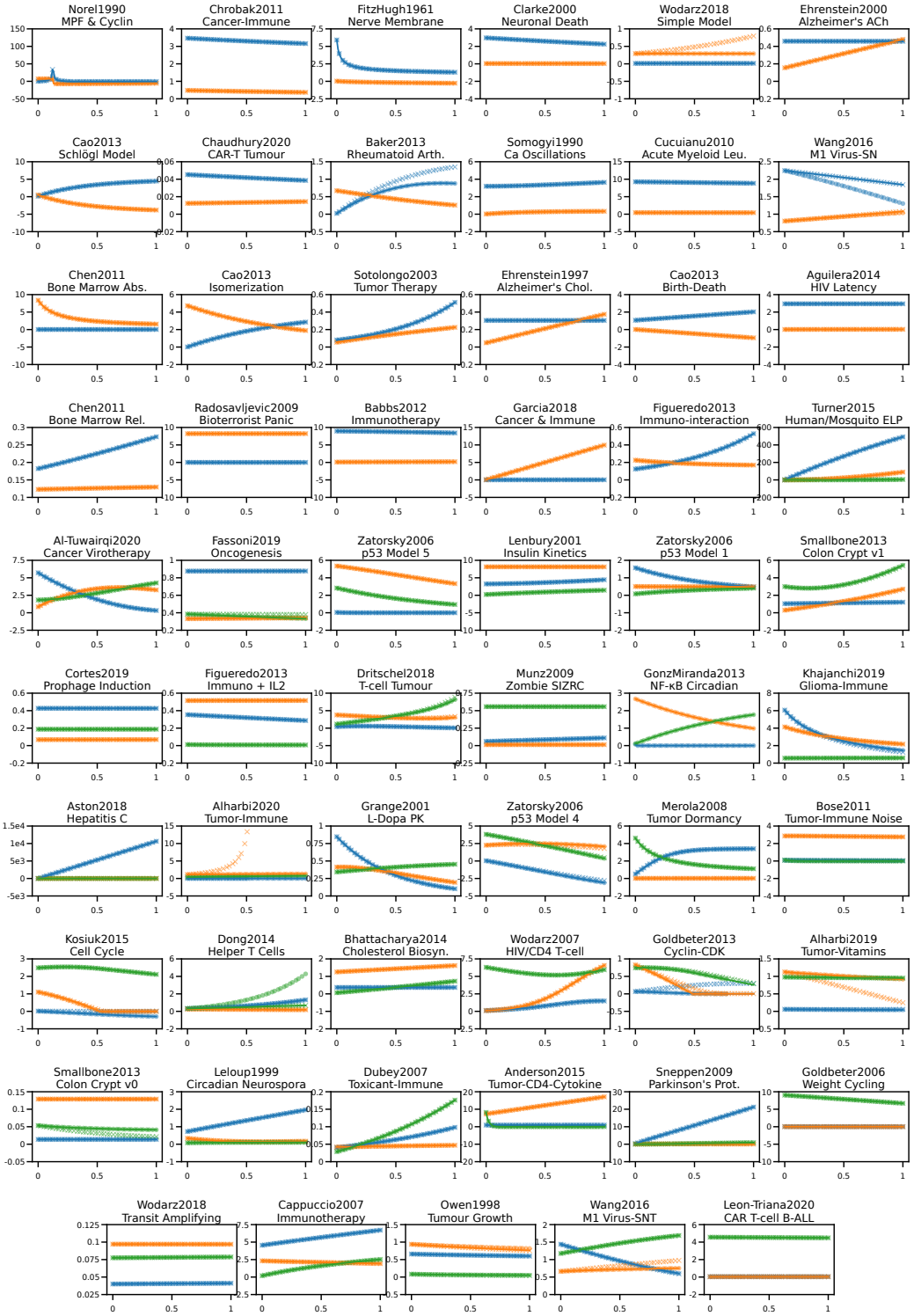


Figure 16: Predictions of LLM-ACES (indicated in bold) and the ground-truth equations for all ODEBase systems for the first set of initial conditions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction outline the key problem addressed, the proposed method, contributions, and a summary of the experimental findings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss the limitations of LLM-ACES following the Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not introduce formal theoretical results, theorems, or convergence guarantees requiring proofs. Its contributions are algorithmic and are evaluated empirically through benchmark experiments, ablations, sensitivity analyses, and robustness studies.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper specifies the complete LLM-ACES procedure, acquisition objective, evaluation protocol, benchmark setup, baseline configurations, and principal hyperparameters in the main text and Appendix B. We additionally provide the implementation needed to reproduce the reported experiments through the code repository linked in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in the supplemental material?

   Answer: [Yes]

   Justification: All experiments use publicly available ODEBench and ODEBase benchmarks, and the implementation of LLM-ACES is released through the repository linked in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer:[Yes]

   Justification: We provide detailed descriptions of the experimental setup and hyperparameter settings in Section 3.1 and Appendix B to support reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the substantial computational cost of symbolic and LLM-based discovery, we perform one run per dynamical system and therefore do not report seed-level confidence intervals or formal significance tests. Instead, we report medians and distributional statistics, including quartiles across benchmark systems, in Appendix C to characterize the consistency and variability of performance.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports the computational environment used for the experiments, including CPU and GPU hardware, available memory, per-dataset computational limits, and the hosted services used for LLM inference. Additionally, Appendix D.2 reports wall-clock time for a representative configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The experiments use established scientific benchmarks and computational or simulated dynamical systems, and do not involve private personal data, human-subject experimentation, or other sensitive data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide an impact statement discussing the societal impacts of the work in detail.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have appropriately cited all relevant prior work, including their code repositories, and properly credited the creators of any code, data, and models used.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the LLM-ACES code in the linked repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: This paper does not involve any research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper clearly describes the use of LLMs as a core component of the method, specifically for generating structured operator priors and guiding the hypothesis space in the closed-loop discovery process along with using it as a judge for calculating symbolic accuracy.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.