MDBench: Benchmarking Data-Driven Methods for Model Discovery

Amirmohammad Ziaei Bideh[1], **Aleksandra Georgievska**[2], **Jonathan Gryak**[1,2]

[1]Department of Computer Science, Graduate Center, CUNY, New York, NY, USA
[2]Department of Computer Science, Queens College, CUNY, New York, NY, USA
aziaeibideh@gradcenter.cuny.edu, aleksgeorgi@gmail.com, jonathan.gryak@qc.cuny.edu

Abstract

Model discovery aims to uncover governing differential equations of dynamical systems directly from experimental data. Benchmarking such methods is essential for tracking progress and understanding trade-offs in the field. While prior efforts have focused mostly on identifying single equations, typically framed as symbolic regression, there remains a lack of comprehensive benchmarks for discovering dynamical models. To address this, we introduce MDBench, an open-source benchmarking framework for evaluating model discovery methods on dynamical systems. MDBench assesses 12 algorithms on 14 partial differential equations (PDEs) and 63 ordinary differential equations (ODEs) under varying levels of noise. Evaluation metrics include derivative prediction accuracy, model complexity, and equation fidelity. We also introduce seven challenging PDE systems from fluid dynamics and thermodynamics, revealing key limitations in current methods. Our findings illustrate that linear methods and genetic programming methods achieve the lowest prediction error for PDEs and ODEs, respectively. Moreover, linear models are in general more robust against noise. MDBench accelerates the advancement of model discovery methods by offering a rigorous, extensible benchmarking framework and a rich, diverse collection of dynamical system datasets, enabling systematic evaluation, comparison, and improvement of equation accuracy and robustness.

Code — https://github.com/gryaklab/mdbench
Datasets — https://zenodo.org/records/17611099
Extended version — https://arxiv.org/abs/2509.20529

1 Introduction

Data-driven equation discovery—rather than relying solely on conservation laws or physical principles—has historically played a pivotal role in scientific breakthroughs. Johannes Kepler discovered the third law of planetary motion by leveraging geometrical intuition and searching for patterns in empirically gathered data (Cranmer 2023). Similarly, Edwin Hubble identified the relationship between redshift and distance through empirical observations that, at the time, lacked theoretical explanation (Kragh 2021).

A dynamical system is a mathematical framework that describes how the state of a system evolves over time through differential equations. Many complex real-world systems arising in finance, medicine, and engineering can be modeled as nonlinear dynamical systems, e.g., in physics, they model fluid dynamics (Kleinstreuer 2018); in biology, they describe neural activity (Breakspear 2017); and in engineering, they underpin control theory (Walker 2013).

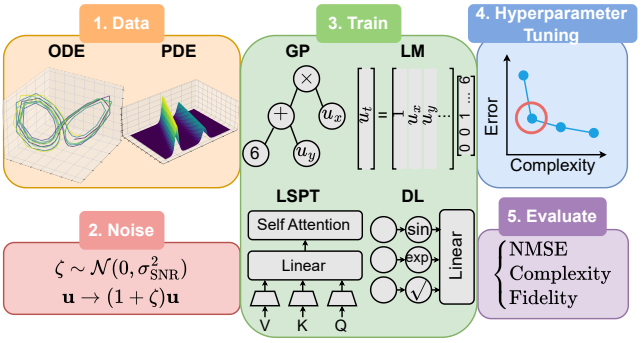


Figure 1: A schematic overview of MDBench pipeline.

Relying solely on expert domain knowledge to derive governing equations is increasingly impractical, especially in the era of high-dimensional and large-scale experimental data. This has motivated the development of automated tools for scientific discovery. While machine learning (ML) models have demonstrated remarkable predictive performance, their black-box nature often impedes interpretability and insight into the underlying system dynamics. Data-driven model discovery (MD) aims to bridge this gap by using measurement data and ML algorithms to infer interpretable dynamical system models.

Unlike traditional black-box ML methods, MD involves optimizing both the parameters and the structure of the model, resulting in models that are inherently interpretable. However, generic MD techniques are not directly applicable to the discovery of differential equations. In this work, we present an extension of MD tailored specifically to the discovery of dynamical system models. In the literature, symbolic regression (SR) typically refers to the discovery of a single equation rather than a system. Here, we adopt the term model discovery to reflect the broader scope of our work.

Despite recent progress, the MD field lacks a standardized benchmark for evaluating algorithms on dynamical systems.

To address this, we introduce **MDBench**, an open-source and extensible benchmarking framework, that includes a suite of ODE and PDE datasets (Figure 1). Establishing such a benchmark is essential for characterizing algorithmic strengths and weaknesses, facilitating fair comparison, and tracking the evolution of the field over time. We encourage researchers to contribute their methods and datasets to MDBench, enabling more robust and reproducible performance evaluations in terms of predictive accuracy, equation complexity, and robustness to noise. Our main contributions are as follows:

1. We introduce the first comprehensive testbed for model discovery, covering a diverse set of methods, including linear models; genetic programming; deep learning; and large scale pretraining approaches, and spanning both ODE and PDE systems.
2. We contribute new datasets simulating real-world PDE systems with up to six state variables, providing significantly more challenging benchmarks than existing MD datasets.
3. We conduct systematic evaluations of state-of-the-art MD methods on these datasets, assessing their performance in equation re-discovery, predictive accuracy, and model complexity under varying noise conditions.

The remainder of this paper is organized as follows. Section 2 reviews existing MD algorithms and related benchmarks. Section 3 details the structure of MDBench, including datasets, training pipeline, metrics, and evaluated methods. Section 4 presents experimental results and highlights current limitations. Finally, Section 5 concludes with a summary of our work.

2 Related Work

This section reviews prior benchmarking efforts in symbolic and scientific model discovery, identifies their limitations, and categorizes the landscape of existing MD methods to four classes.

2.1 Overview of Benchmarks

Several efforts have been made to benchmark MD algorithms. One of the earliest large-scale initiatives is SRBench (La Cava et al. 2021), which introduced a publicly available benchmark evaluating 14 SR methods on a variety of regression problems. A later extension, SRBench 2.0 (Aldeia et al. 2025), expanded the number of evaluated algorithms to 25 and introduced additional metrics such as energy consumption. Notably, the authors concluded that no single SR method consistently outperforms others across all metrics, and they proposed several best practices for future method development such as minimizing hyperparameters and standardizing evaluation procedures.

Despite these contributions, these benchmarks are limited to time-invariant regression problems involving a single mathematical equation and do not account for dynamical systems governed by differential equations. Model discovery for dynamical systems presents additional challenges such as handling systems of coupled equations, high dimensionality, and numerical differentiation.

PDEBench (Takamoto et al. 2022) is a more recent benchmark focused on PDE systems that evaluated 11 physical systems using four machine learning surrogates: U-Net (Ronneberger, Fischer, and Brox 2015), Fourier Neural Operator (Li et al. 2021), and Physics-Informed Neural Networks (PINNs) (Raissi, Perdikaris, and Karniadakis 2019). While these methods are effective for solving PDEs, they are black-box models and cannot generate symbolic equations, limiting their utility for interpretable model discovery.

Gilpin et al. (Gilpin 2021) benchmarked 131 chaotic ODE systems, framing them as time series forecasting tasks. Although their study includes symbolic regression methods, only four MD methods are tested, and no evaluation is provided on the complexity or interpretability of the discovered models.

Other benchmarks have narrower or domain-specific scopes. cp3-bench (Thing and Koksbang 2025) evaluated 12 MD algorithms on 28 datasets from cosmology and astroparticle physics. CFDBench (Luo, Chen, and Zhang 2023) focuses on computational fluid dynamics (CFD) simulations and evaluated neural operators under varying conditions, without considering symbolic recovery of governing equations. The benchmark by Otness et al. (Otness et al. 2021) includes four PDE systems, but evaluated only black-box predictive models such as convolutional neural networks and k-nearest neighbors.

Table 1 summarizes key characteristics of these benchmarks, highlighting MDBench's broader coverage across ODEs, PDEs, and model discovery algorithms.

Benchmark	#ODE	#PDE	#MD	Noise
SRBench 1.0	7	0	14	✓
SRBench 2.0	0	0	25	✓
PDEBench	0	11	0	
(Gilpin 2021)	131	0	4	
cp3-bench	7	0	12	✓
CFDBench	0	4	0	
(Otness et al. 2021)	0	4	0	
MDBench (ours)	63	14	12	✓

Table 1: Comparison of existing benchmarks for scientific MD. **#ODE** and **#PDE** refer to the number of ODE and PDE problems, **#MD** refers to the number of model discovery algorithms evaluated, and **Noise** indicates whether robustness to noise is studied.

2.2 Overview of Model Discovery Methods

Since Koza's early work on genetic programming (GP) for symbolic regression (Koza 1994), numerous methods have emerged for data-driven model discovery. We categorize them into four main classes.

Genetic Programming (GP) GP-based methods evolve expression tree of equations using evolutionary operators to search for equations that best fit the data. Modern and efficient implementations, such as PySR (La Cava et al. 2021)

and Operon (Burlacu, Kronberger, and Kommenda 2020), are widely used in the literature.

GP methods often struggle with convergence due to their large, unstructured search spaces. They also do not naturally learn parameters from data. Neural-guided approaches like those in (Mundhenk et al. 2021) mitigate this by incorporating learned search heuristics. Recent work further accelerates GP convergence by using large language models (LLMs) as crossover and mutation operators (Shojaee et al. 2025; Meyerson et al. 2024).

Linear Models (LM) Linear model-based methods represent equations as sparse linear combinations of candidate basis functions ("library"). A typical form is $f = \sum_{i=1}^{k} \beta_i \phi_i$, where ϕ_i are predefined basis terms and β_i are scalar coefficients. SINDy (Brunton, Proctor, and Kutz 2016) pioneered this approach using LASSO for sparsity. PDEFIND (Rudy et al. 2017) extended it to PDEs, and WSINDy (Messenger and Bortz 2021) further improved robustness by formulating equations in weak form, avoiding direct differentiation. ESINDy (Fasel et al. 2022) introduced ensemble learning for robustness in low-data or high-noise regimes. DeepMoD (Both et al. 2021) combines neural networks with sparse regression by learning spatial features via automatic differentiation.

Bayesian variants (Yuan et al. 2023; North, Wikle, and Schliep 2025; More et al. 2023) introduce uncertainty modeling and noise-aware inference. However, all LM methods are constrained by the assumption of linearity with respect to basis functions and often lack principled ways to construct these function libraries for unknown systems.

Large-Scale Pretraining (LSPT) LSPT methods pretrain a single model—often a Transformer architecture (Vaswani et al. 2017)—on a large corpus of symbolic regression problems. These models learn to map input–output pairs to symbolic equations, enabling rapid inference on new data once training is complete.

Neural Symbolic Regression that Scales (NeSymReS) (Biggio et al. 2021) exemplifies this approach. It uses a Set Transformer (Lee et al. 2019) as an encoder to learn latent representations of input–output mappings from synthetic equations. A decoder then generates symbolic expressions autoregressively using beam search. After generating symbolic forms, constant placeholders are refined using post-hoc optimization with methods such as BFGS (Fletcher 2000).

To overcome limitations of this two-stage design, Kamienny et al. (Kamienny et al. 2022) introduced an end-to-end approach where the Transformer directly generates complete mathematical expressions, including constants. ODEFormer (d'Ascoli et al. 2024) further extends this framework to dynamical systems, generating full differential equations from a single observed trajectory.

The main advantage of LSPT methods is their inference speed. Once trained, they can quickly generate symbolic equations for new datasets without the need for retraining or optimization, which makes them suitable for real-time applications. However, they may perform poorly on real-world systems whose dynamics differ from the synthetic equations seen during pretraining (Kamienny et al. 2022).

Deep Learning (DL) DL-based methods use neural networks to learn symbolic equations from data. One example is Equation Learner Networks (EQL) (Martius and Lampert 2016; Sahoo, Lampert, and Martius 2018), which are fully differentiable feed-forward networks that incorporate symbolic operators (e.g., sine, cosine, multiplication) as activation functions. This design enables the integration of symbolic structure into neural models and allows EQL to be combined with other architectures for scientific discovery tasks (Kim et al. 2020).

Another class of DL methods uses recurrent neural networks (RNNs) to learn a probability distribution over symbolic expressions conditioned on data. Deep Symbolic Regression (DSR) (Petersen et al. 2021) follows this approach. It trains an RNN using a risk-seeking policy gradient algorithm, where the reward is based on the predictive accuracy of sampled equations. However, the learning signal in this setup comes only from the scalar reward based on the (X, y) fit, which can be weak and indirect.

To overcome this limitation, uDSR (Landajuela et al. 2022) extends DSR by integrating optimization techniques from GP, LM, and LSPT approaches as inner loops. This hybrid approach achieves state-of-the-art performance on the SRBench benchmark (La Cava et al. 2021).

3 MDBench

MDBench is a unified benchmarking framework for evaluating data-driven MD algorithms on both ODE and PDE systems (Figure 1). It includes a diverse suite of 63 ODEs and 14 PDEs, ranging from simple linear dynamics to high-dimensional physical systems. MDBench standardizes data formats, provides symbolic preprocessing for PDEs, incorporates realistic noise modeling, and supports automated hyperparameter tuning across methods.

3.1 Datasets

Dynamical systems are commonly modeled using either ODEs or PDEs, depending on whether the variables of interest evolve only over time or over both time and space. ODE-based systems involve temporal dynamics, while PDE-based systems also incorporate spatial variation.

ODE Systems. ODEs describe the evolution of a system's state variables over time. Formally, a dynamical system with d state variables $\mathbf{u} = (u_1, \ldots, u_d) \in \mathbb{R}^d$ is governed by equations of the form:

$$\frac{du_i}{dt} = f_i(t, u_1, u_2, \ldots, u_d), \quad i = 1, \ldots, d,$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$. Each dataset consists of observed trajectories $\mathbf{U} \in \mathbb{R}^{N_t \times d}$, where N_t is the number of samples across time.

We adopt the **ODEBench** dataset (d'Ascoli et al. 2024), which includes 63 systems from a textbook by Strogatz (Strogatz 2024) and several sourced from Wikipedia. These systems span one to four state variables and cover a range of real-world phenomena. For all methods, the input consists of state variables $\mathbf{u}$, and the targets are their time derivatives $\dot{\mathbf{u}}$.

PDE Systems. PDEs describe spatiotemporal systems where each state variable depends on both space and time. A PDE system with d state variables $\mathbf{u} = (u_1, \ldots, u_d) \in \mathbb{R}^d$ is represented on a D-dimensional spatial grid $\mathbf{X} \in \mathbb{R}^{N_{x_1} \times \cdots \times N_{x_D}}$. The spatial grid is constructed from uniformly-spaced intervals along each spatial coordinate, with spacing that may differ between them (i.e., $\Delta x_1 \neq \cdots \neq \Delta x_D$). The time domain $t \in [0, T]$ is uniformly spaced.

We focus on PDEs of the form:

$$\frac{\partial u_i}{\partial t} = f_i(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x_1}, \frac{\partial^2 \mathbf{u}}{\partial x_1^2}, ..., \frac{\partial \mathbf{u}}{\partial x_2}, \frac{\partial^2 \mathbf{u}}{\partial x_2^2}, ...); \quad i = 1, ..., d, \tag{1}$$

where f_i depends nonlinearly on the state variables and their spatial derivatives.

Our benchmark includes both widely-studied PDE systems in the MD literature and seven systems simulating complex physical processes in fluid dynamics. Previously established datasets include *Advection* and *Burgers* (Takamoto et al. 2022); *Korteweg-de Vries (KdV)*, *Kuramoto-Sivishinky (KS)*, *Nonlinear Schrödinger (NLS)* , and *Reaction-Diffusion (RD)* (Brunton, Proctor, and Kutz 2016); *Advection-Diffusion (AD)* (Both et al. 2021).

We observe that commonly used PDE systems in the MD literature often fail to capture the complexity of real-world phenomena, which may involve space-dependent or piecewise forcing functions, or exhibit high dimensionality. Therefore, we include the following PDE systems in our testbed: *Heat (Laser)* (Abali 2016); *Heat (Solar)*, *Navier-Stokes (Channel)*, *Navier-Stokes (Cylinder)*, and *Reaction-Diffusion (Cylinder)* (Langtangen and Logg 2017).

A summary of the dataset properties (Table A1) along with full equations and implementation details are included in the supplementary material.

3.2 Experimental Setup

Noise Setup. To assess robustness, we simulate measurement noise by corrupting clean state variables with Gaussian noise at signal-to-noise ratios (SNRs) of 40 dB, 30 dB, 20 dB, and 10 dB. We apply multiplicative noise as:

$$\mathbf{u} \to (1 + \xi)\mathbf{u}, \quad \xi \sim \mathcal{N}(0, \sigma), \quad \sigma = 10^{-\text{SNR}_{\text{dB}}/20}.$$

This impacts both the right-hand side and the estimated time derivatives. We compute noisy derivatives using finite differences and evaluate prediction error against the clean derivatives.

Metrics. In order to assess the performance of the discovered equations, we investigate two metrics: 1) **NMSE** refers to the accuracy of the predicted time derivatives defined as the Normalized Mean Square Error of the derivatives, $\text{NMSE}(\mathbf{u}_t, \hat{\mathbf{u}}_t) = \frac{\sum_{i=1}^{N} \left(u_t^{(i)} - \hat{u}_t^{(i)}\right)^2}{\sum_{i=1}^{N} \left(u_t^{(i)}\right)^2 + \epsilon}$, where ϵ is a small regularization constant set to 10^{-10}. NMSE measures trajectory-level predictive fidelity. A lower NMSE indicates better agreement between the model and data, and serves as a proxy for how well the discovered dynamics reproduce observed behavior. 2) **Complexity** refers to the total number of nodes, constant terms, and operations in the expression tree of the equations. This metric encourages parsimony and penalizes unnecessary long or redundant terms. The `SymPy` package is used to parse the discovered equations and compute their complexities (Meurer et al. 2017).

Hyperparameter Tuning. Since MD methods are sensitive to hyperparameters, we adopt a unified and automated tuning protocol. For each method-dataset pair, we evaluate all combinations from a predefined hyperparameter grid and select the best configuration using a composite fitness score that balances between the complexity and accuracy of the equations. Similar to (Merler et al. 2024; Shojaee et al. 2023), the fitness function is defined as

$$s(f|\mathbf{u}) = \frac{1}{1 + \text{NMSE}(\mathbf{u}_t, f(\mathbf{u}))} + \lambda \exp\left(-\frac{l(f)}{L}\right), \tag{2}$$

where the hyperparameter λ weights the relative importance of accuracy and complexity, $l(.)$ computes the complexity of the equations, and L denotes the maximum equation length. The set of defined hyperparameters for each algorithm is provided in the supplementary materials. Throughout the experiments, we set $\lambda = 1$ and $L = 200$.

Extending Generic Methods to PDEs. Most generic MD methods were originally designed for standard supervised learning tasks, where both input features and target outputs are explicitly available in the datasets. However, in the context of PDE-based dynamical systems, the right-hand side of the equations, which typically involve spatial derivatives, must be constructed from the observed data. To enable the application of these methods to PDE discovery, we preprocess each dataset to compute the required symbolic features.

Specifically, for each PDE system with d state variables, we generate a symbol set that includes the state variables themselves along with their spatial derivatives up to fourth order. These are computed using second-order accurate finite difference approximations via the `findiff` package (Baer 2018). The resulting symbolic feature set is $\{u_1, u_2, ..., u_d\} \cup \left\{ \frac{\partial^j u_i}{\partial x_k^j} \mid i = 1, \ldots, d; j = 1, \ldots, 4; k = 1, \ldots, D \right\}$. The target variables, i.e., the time derivatives $\partial u_i / \partial t$, are also computed using the same finite difference scheme.

Training and Inference. We split each dataset along the time dimension into three parts: the first 60% is used for training, the next 20% for validation (used in hyperparameter tuning), and the final 20% for testing. After selecting the optimal hyperparameters based on the validation set, each model is retrained on the combined training and validation sets. The final evaluation is performed on the test set using the metrics described earlier.

For GP-based methods, we follow the authors' recommended default settings and train the models on the combined training and validation data. From the set of candidate expressions generated during evolution, we select the best equation using the fitness function defined in Equation 2.

To ensure fairness and feasibility, we impose a time limit of 12 hours per experiment, covering both training and hy-

perparameter tuning. All time-bounded experiments are conducted on an Ubuntu 22.04.3 server equipped with a 24-core Intel Core Ultra 9 285K @ 1.44 GHz CPU and 96 GB of RAM. For methods that require GPU acceleration (EQL, uDSR, ODEFormer, End2end, and DeepMoD), we use a single NVIDIA RTX 4000 Ada Generation GPU with 20 GB of memory.

3.3 Methods

From the algorithm categories presented in Section 2, we selected a representative set of model discovery methods to benchmark. Selection criteria included strong reported performance in prior work and the availability of well-documented, easily adoptable implementations. The studied method include PDEFIND (Rudy et al. 2017), WSINDy (Messenger and Bortz 2021), ESINDy/EWSINDy (Fasel et al. 2022), Bayesian (More et al. 2023), DeepMoD (Both et al. 2021), SINDy (Brunton, Proctor, and Kutz 2016), EQL (Sahoo, Lampert, and Martius 2018), uDSR (Landajuela et al. 2022), PySR (Cranmer 2023), Operon (Burlacu, Kronberger, and Kommenda 2020), ODEFormer (d'Ascoli et al. 2024), and End2end (Kamienny et al. 2022). Table 2 provides an overview of the selected algorithms, including their methodological type, the systems they are applied to, and a brief description.

Due to memory or runtime limitations, the methods DeepMoD, uDSR, and End2end are unable to handle large datasets efficiently. For these methods, we perform training on a subsampled version of the data, limited to 10,000 points.

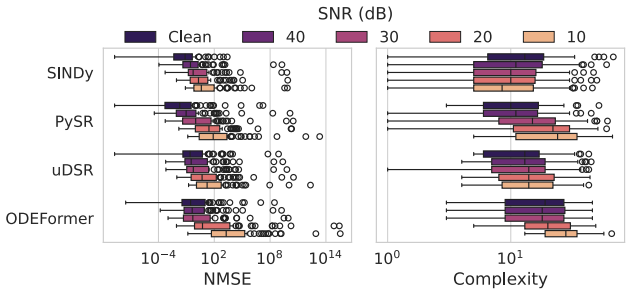
4 Results and Discussion

4.1 Performance on ODE Data

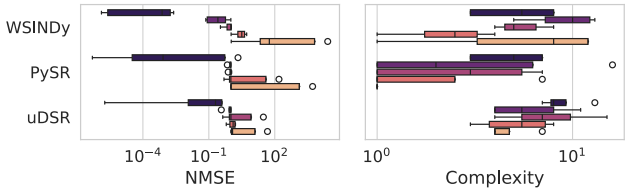
Figure 2a compares the performance of a representative subset of model discovery methods on the ODEBench dataset (see Figure A10a for additional methods). Most methods were able to recover valid equations under noiseless conditions, but their robustness to increasing noise varied considerably. Tables A5 and A6 in supplementary materials provide a more detailed account of the performance metrics on ODE datasets.

Equations generated by ODEFormer have good predictive accuracy on time derivatives in low-noise settings and clean data (SNR $\geq$ 30 dB), indicating effective generalization. However, its performance degrades rapidly under higher noise levels (SNR $\leq$ 20 dB), suggesting overfitting likely due to its large number of trainable parameters. This is also reflected in the increasing complexity of the discovered equations as noise increases.

In contrast, linear models SINDy and ESINDy generated simpler equations under noisy conditions. The Wilcoxon signed-rank test showed a significant difference between the complexity of SINDy equations and that of EQL (next simplest equations) for noisy data $W = 6951.5, p < 10^{-8}$. The included sparsity-promoting regularization acts as an implicit denoising method, suppressing uncertain or spurious terms introduced by noise. This behavior, while leading to less complex equations, may also result in the omission of relevant nonlinear dynamics, especially at low SNR.



(a) ODE datasets.



(b) PDE datasets.

Figure 2: Performance box plot for representative MD methods on ODE and PDE datasets.

GP-based methods generally achieved lower error in predicted time derivatives compared to other methods in low-noise settings and clean data (SNR $\geq$ 30 dB). For instance, PySR has a significantly lower NMSE than ESINDy based on Wilcoxon signed-rank test (log-scaled, $W = 9774, p \leq 0.0001$). GP algorithms showed a tendency to overfit in noisy settings, producing highly complex and inaccurate expressions. This might suggest that GP-based methods lacking explicit regularization or those imposing minimal assumptions on the equation skeleton in the search space are more sensitive to noisy inputs.

End2end exhibited the highest predictive errors among the evaluated methods, indicating that it struggled to capture the underlying system dynamics (Wilcoxon signed-rank test with EQL, log scale NMSE, $W = 12795, p < 10^{-10}$). This may be due to the fact that its pretrained model was not exposed to any dynamical systems during training, limiting its ability to generalize to such tasks.

As system dimensionality increases, most methods exhibit a corresponding rise in training time. Among the evaluated approaches, SINDy and Operon are the most computationally efficient, whereas EQL and uDSR are significantly slower, differing by several orders of magnitude. Figure A11 in supplementary materials shows the training times of various MD methods on ODE datasets.

4.2 Performance on PDE Data

Compared to ODEs, PDE systems present a more challenging testbed due to the presence of spatial derivatives and higher-order terms, which tend to amplify noise. Figure 2b shows the aggregate NMSE and complexity of representative methods over a selected subset of PDE datasets for

Method	Type	Systems	Description
PDEFIND	LM	PDE	Sparse regression on a library of candidate terms for discovering PDEs.
SINDy	LM	ODE	Sparse regression on a library of candidate terms for discovering ODEs.
WSINDy	LM	PDE	Weak formulation of PDEFIND for robust identification from noisy or sparse data.
E(W)SINDy	LM	ODE/PDE	Ensemble-SINDy uses ensembles to quantify the inclusion probability of library terms.
Bayesian	LM	PDE	An extension of PDEFIND with variational Bayes to handle noisy data.
DeepMoD	LM	PDE	Joint optimization of function approximation and sparse structure.
EQL	DL	PDE/ODE	Shallow neural network with symbolic operators as activation functions.
uDSR	DL	PDE/ODE	Unified GP, LM, and LSPT with a recurrent neural network.
PySR	GP	PDE/ODE	Evolutionary algorithm with parallel multi-population support.
Operon	GP	PDE/ODE	Efficient C++-based GP framework with fine-grained control.
ODEFormer	LSPT	ODE	Transformer-based equation discovery from ODE trajectory data.
End2end	LSPT	ODE/PDE	End-to-end transformer model for discovering equations from input-output data.

Table 2: Overview of model discovery methods benchmarked in MDBench.

which all methods completed successfully (refer to Figure A10b for all methods). The selected datasets include *Advection*, *Burgers*, *KdV*, *KS*, *Heat (Solar) 1D*, and *AD*. The primary reasons for a method's failure to discover an underlying model include: a lack of support for higher-dimensional data in the published software, failing to finish training before the timeout threshold was reached, and runtime errors within the published codebases. EQL and End2end are excluded from the PDE evaluations due to repeated failures, mainly caused by implementation issues or limitations in handling high-dimensional inputs. In particular, the pretrained transformer model for the End2end method has been trained on synthetic equations with up to 10 features (Kamienny et al. 2022), which prohibits its usage on high-dimensional systems or systems with more than two state variables.

Figure 2b shows that there is a sharp error gap between the NMSE for clean and low-noise systems (SNR = 40 dB) among the selected PDE datasets. The logarithm of the error ratio between the noisy and clean settings for PDEs is 3.13 ± 2.09, compared to 0.88 ± 2.12 for the ODE datasets. This illustrates that the accuracy of the predicted time derivatives estimated by model discovery methods for PDE systems is more sensitive to noise than for ODE systems. One reason is that while ODE systems involve only a single time derivative per state variable, PDE systems include higher-order and spatial derivatives. These high-order derivatives amplify noise in measured data, leading to greater accumulation of the errors.

Most LM methods, except for DeepMoD, achieved lower NMSE and better robustness in low-noise settings (SNR $\geq$ 30 dB) and noise-free data compared to GP methods. For instance, WSINDy obtained significantly lower error compared to PySR ($W = 455.0, p < 10^{-4}$) and Operon ($W = 466.0, p < 10^{-4}$) in the mentioned noise settings. PySR and Operon, while producing competitive results in clean settings, suffer from reduced predictive fidelity as noise increases. Despite simpler equations, uDSR tends to generate the most inaccurate equations in the noise-free setting.

Table 3 summarizes the NMSE and expression complexity for each method across all PDE datasets in the clean (noise-free) setting. Almost all methods achieve low prediction error on the *Heat (Solar) 1D* dataset. However, as the spatial dimension of the *Heat (Solar)* system increases (from 1D to 3D), the NMSE of predicted derivatives decreases. This is because higher dimensions roughly double the size of the function library for sparse-regression methods and the number of symbols for GP-based methods, thereby enlarging the search space and making model discovery more challenging. These results highlight the need for developing scalable algorithms capable of handling high-dimensional data.

The datasets *Heat (Laser)* and *RD (Cylinder)* are more challenging than the rest of the PDEs in the benchmark due to their use of custom forcing functions. The *Heat (Laser)* system contains a spatially-dependent forcing function, while the *RD (Cylinder)* system not only includes six state variables but also has piecewise linear forcing functions that vary across the spatial domain. Most of the LM methods fail at generating equations for these datasets likely because of the huge library size and sparse regression problem. The results of PySR, Operon, and uDSR methods on these datasets illustrate that existing model discovery methods perform poorly on such datasets, suggesting the need for more generalizable and robust algorithms for discovering real-world PDE systems.

Runtimes vary widely by method and PDE dataset, ranging from a few seconds to several hours (Table 3). Overall, PDEFind, WSINDy, PySR, and uDSR demonstrate the fastest runtimes. While GP methods, particularly PySR, tend to be slower than linear methods on 1D problems, they scale more favorably in higher dimensions. For example, on the *Heat (Solar)* dataset, PDEFind is approximately $40\times$ and $465\times$ slower in the 2D and 3D cases, respectively, compared to the 1D case. In contrast, PySR slows down by about $33\times$ in 2D and only $12\times$ in 3D. This disparity likely stems from the growing size of sparse regression systems encountered by linear model-based methods. This suggests that in general, in time-constrained environments, LM methods are a better choice for low-dimensional datasets, while GP methods and uDSR scale more robustly to higher-dimensional data. More granular runtime data can be found in Table A7.

Dataset	PDEFIND	Bayesian	WSINDy	EWSINDy	DeepMoD	PySR	Operon	uDSR
Advection	✓	$10^{-6}(21)$	✓	✓	$10^{1}(12)$	✓	✓	$10^{-6}(8)$
Burgers	✓	$\underline{10^{-6}(9)}$	✓	✓	✓	✓	✓	$10^{-1}(8)$
KdV	✓	$\underline{10^{-3}(9)}$	✓	✓	✓	✓	$10^{-1}(7)$	$10^{-1}(7)$
KS	✓	$\underline{10^{-3}(12)}$	✓	✓	$10^{1}(3)$	✓	$\underline{10^{-3}(23)}$	$10^{0}(4)$
AD	$\underline{10^{-5}(17)}$	$\underline{10^{-5}(12)}$	✓	✓	$10^{1}(3)$	✓	✓	$10^{0}(15)$
Heat (Solar) 1D	✓	$10^{-3}(8)$	✓	✓	$10^{6}(3)$	✓	✓	$10^{-2}(13)$
Heat (Solar) 2D	$10^{-2}(30)$	-	$10^{-3}(17)$	$10^{-3}(9)$	-	$\underline{10^{-3}(3)}$	$10^{-3}(7)$	$10^{-2}(12)$
Heat (Solar) 3D	$10^{1}(23)$	-	-	-	-	$10^{0}(10)$	$10^{0}(15)$	$10^{0}(3)$
Heat (Laser)	$10^{0}(14)$	-	-	-	-	$10^{0}(11)$	$10^{0}(15)$	$10^{0}(6)$
NLS	$10^{-1}(16)$	-	$10^{-1}(16)$	-	-	$10^{0}(29)$	$10^{-1}(53)$	$10^{2}(22)$
RD	$10^{-3}(6)$	-	$10^{-3}(6)$	-	-	$10^{-2}(5)$	$10^{-2}(14)$	$10^{-2}(12)$
NS (Channel)	$10^{2}(82)$	-	-	-	-	$10^{-3}(4)$	$10^{-3}(15)$	$10^{-3}(22)$
NS (Cylinder)	$10^{-2}(42)$	-	$10^{-2}(95)$	-	-	$10^{-1}(25)$	$10^{-1}(30)$	$10^{-1}(38)$
RD (Cylinder)	-	-	$10^{-2}(3253)$	$10^{-1}(1603)$	-	$10^{-1}(27)$	$10^{-2}(51)$	$10^{-1}(50)$
Runtime (min)	26 ± 53	102 ± 205	4 ± 6	94 ± 87	108 ± 131	8 ± 13	54 ± 104	9 ± 11

Table 3: Performance of MD methods on PDE datasets, measured by the NMSE of the time derivatives on the test set. Equation complexity is given in parenthesis. Reported NMSEs are rounded to the nearest power of 10. A check mark (✓) indicates successful identification of the full equation, with all coefficients being in $\pm 5\%$ of ground truth. Underlined entries denote partially correct equations, where only one term is missing or extra, and the remaining terms have coefficients that are within $\pm 5\%$ of ground truth. Empty entries (-) show method failure or timeout. The table includes the average and standard deviation of method runtimes on clean data (in minutes).

4.3 Limitations

Despite recent progress in data-driven model discovery, we highlight several key limitations.

First, many algorithms implicitly assume uniform physical parameters (e.g., diffusivity, conductivity) across space and time. This assumption simplifies inference but fails to reflect heterogeneities in real-world systems such as *Heat (Laser)* and *Reaction-Diffusion (Cylinder)* systems (Table 3). Models built under this assumption may generalize poorly to such heterogeneous environments.

Second, current methods show degraded performance on systems with many state variables or high-dimensional spaces. As shown in Table 3, systems with one state variable have a much higher successful discovery rate and lower errors on average compared to systems with more than one state variable. In the case of *Heat (Solar)* systems, the error increases for all methods as the spatial dimension grows. LM approaches suffer from the exponential growth of function libraries, while GP-based methods face intractable search spaces. This scalability bottleneck limits applicability to realistic systems such as multi-component dynamical systems or high-resolution simulations.

Third, most discovery methods operate on numerical approximations of derivatives, making them highly sensitive to noise. In PDE systems, the amplification of noise through higher-order derivatives further compromises model fidelity as can be seen from Figure A10b in supplementary materials. Without robust denoising mechanisms or noise-aware formulations, the discovered equations may be structurally incorrect despite yielding a low NMSE on predicted time derivatives.

Fourth, there is no established metric for quantifying equation fidelity. Standard metrics such as NMSE and symbolic complexity do not always reflect the symbolic correctness of discovered equations. For example, while GP methods achieved a relatively low NMSE on *NS (Channel)* dataset, the discovered equations do not capture the true dynamics. Without a robust metric that captures functional equivalence, e.g., using a discretized version of the Sobolev semi-norm for PDE systems (Schaback 2015), benchmarking results may obscure deeper algorithmic failures.

Finally, several methods fail on specific datasets due to implementation limitations (e.g., lack of support for multi-dimensional systems, unstable solvers, or uninformative error messages). This undermines reproducibility and suggests the need for compatible implementations that can be used by other researchers to apply the methods on their datasets.

5 Conclusion

In this paper, we introduced MDBench, a comprehensive and extensible benchmark for evaluating 12 model discovery methods across 77 dynamical systems. We evaluated the quality of the discovered dynamics across four methodological classes, assessing predictive accuracy of time derivatives, equation complexity, and fidelity to ground-truth dynamics. We release MDBench as an open-source, extensible benchmarking framework as well as a representative collection of datasets, and believe it provides a foundation for the evaluation and development of novel model discovery algorithms for dynamical systems. We hope MDBench serves as a foundation for future research in interpretable, robust model discovery across complex dynamical systems.

References

Abali, B. E. 2016. *Computational reality*. Springer.

Aldeia, G. S. I.; Zhang, H.; Bomarito, G.; Cranmer, M.; Fonseca, A.; Burlacu, B.; La Cava, W. G.; and de França, F. O. 2025. Call for Action: towards the next generation of symbolic regression benchmark. *arXiv preprint arXiv:2505.03977*.

Alnæs, M.; Blechta, J.; Hake, J.; Johansson, A.; Kehlet, B.; Logg, A.; Richardson, C.; Ring, J.; Rognes, M. E.; and Wells, G. N. 2015. The FEniCS project version 1.5. *Archive of numerical software*, 3(100).

Baer, M. 2018. findiff Software Package. https://github.com/maroba/findiff.

Biggio, L.; Bendinelli, T.; Neitz, A.; Lucchi, A.; and Parascandolo, G. 2021. Neural symbolic regression that scales. In *International Conference on Machine Learning*, 936–945. Pmlr.

Both, G.-J.; Choudhury, S.; Sens, P.; and Kusters, R. 2021. DeepMoD: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428: 109985.

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

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

Burlacu, B.; Kronberger, G.; and Kommenda, M. 2020. Operon C++ an efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 1562–1570.

Cranmer, M. 2023. Interpretable machine learning for science with PySR and SymbolicRegression. jl. *arXiv preprint arXiv:2305.01582*.

d'Ascoli, S.; Becker, S.; Schwaller, P.; Mathis, A.; and Kilbertus, N. 2024. ODEFormer: Symbolic Regression of Dynamical Systems with Transformers. In *The Twelfth International Conference on Learning Representations*.

Fasel, U.; Kutz, J. N.; Brunton, B. W.; and Brunton, S. L. 2022. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260): 20210904.

Fletcher, R. 2000. *Practical methods of optimization*. John Wiley & Sons.

Gilpin, W. 2021. Chaos as an interpretable benchmark for forecasting and data-driven modelling. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Kamienny, P.-A.; d'Ascoli, S.; Lample, G.; and Charton, F. 2022. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35: 10269–10281.

Kim, S.; Lu, P. Y.; Mukherjee, S.; Gilbert, M.; Jing, L.; Čeperić, V.; and Soljačić, M. 2020. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9): 4166–4177.

Kleinstreuer, C. 2018. *Modern fluid dynamics*. Springer.

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

Kragh, H. 2021. *Cosmology and Controversy : The Historical Development of Two Theories of the Universe*. New Jersey :: Princeton University Press. ISBN 9780691227719.

La Cava, W.; Burlacu, B.; Virgolin, M.; Kommenda, M.; Orzechowski, P.; de França, F. O.; Jin, Y.; and Moore, J. H. 2021. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, 2021(DB1): 1.

Landajuela, M.; Lee, C. S.; Yang, J.; Glatt, R.; Santiago, C. P.; Aravena, I.; Mundhenk, T.; Mulcahy, G.; and Petersen, B. K. 2022. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems*, 35: 33985–33998.

Langtangen, H. P.; and Logg, A. 2017. *Solving PDEs in Python: The FEniCS Tutorial I*. Springer Nature.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 3744–3753. PMLR.

Li, Z.; Kovachki, N. B.; Azizzadenesheli, K.; liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*.

Logg, A.; Mardal, K.-A.; and Wells, G. 2012. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media.

Luo, Y.; Chen, Y.; and Zhang, Z. 2023. CFDBench: A Large-Scale Benchmark for Machine Learning Methods in Fluid Dynamics. *arXiv preprint arXiv:2310.05963*.

Martius, G.; and Lampert, C. H. 2016. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*.

Merler, M.; Haitsiukevich, K.; Dainese, N.; and Marttinen, P. 2024. In-Context Symbolic Regression: Leveraging Large Language Models for Function Discovery. In Fu, X.; and Fleisig, E., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, 427–444. Bangkok, Thailand: Association for Computational Linguistics. ISBN 979-8-89176-097-4.

Messenger, D. A.; and Bortz, D. M. 2021. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443: 110525.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore,

J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

Meyerson, E.; Nelson, M. J.; Bradley, H.; Gaier, A.; Moradi, A.; Hoover, A. K.; and Lehman, J. 2024. Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning*, 4(4): 1–40.

More, K. S.; Tripura, T.; Nayek, R.; and Chakraborty, S. 2023. A Bayesian framework for learning governing partial differential equation from data. *Physica D: Nonlinear Phenomena*, 456: 133927.

Mundhenk, T.; Landajuela, M.; Glatt, R.; Santiago, C. P.; Petersen, B. K.; et al. 2021. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. *Advances in Neural Information Processing Systems*, 34: 24912–24923.

North, J. S.; Wikle, C. K.; and Schliep, E. M. 2025. A bayesian approach for spatio-temporal data-driven dynamic equation discovery. *Bayesian Analysis*, 20(2): 375–404.

Otness, K.; Gjoka, A.; Bruna, J.; Panozzo, D.; Peherstorfer, B.; Schneider, T.; and Zorin, D. 2021. An Extensible Benchmark Suite for Learning to Simulate Physical Systems. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Petersen, B. K.; Larma, M. L.; Mundhenk, T. N.; Santiago, C. P.; Kim, S. K.; and Kim, J. T. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234–241. Springer.

Rudy, S. H.; Brunton, S. L.; Proctor, J. L.; and Kutz, J. N. 2017. Data-driven discovery of partial differential equations. *Science advances*, 3(4): e1602614.

Sahoo, S.; Lampert, C.; and Martius, G. 2018. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, 4442–4450. Pmlr.

Schaback, R. 2015. A computational tool for comparing all linear PDE solvers: Error-optimal methods are meshless. *Advances in Computational Mathematics*, 41(2): 333–355.

Shojaee, P.; Meidani, K.; Barati Farimani, A.; and Reddy, C. 2023. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36: 45907–45919.

Shojaee, P.; Meidani, K.; Gupta, S.; Farimani, A. B.; and Reddy, C. K. 2025. LLM-SR: Scientific Equation Discovery via Programming with Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

Strogatz, S. H. 2024. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Chapman and Hall/CRC.

Takamoto, M.; Praditia, T.; Leiteritz, R.; MacKinlay, D.; Alesiani, F.; Pflüger, D.; and Niepert, M. 2022. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611.

Thing, M.; and Koksbang, S. 2025. cp3-bench: a tool for benchmarking symbolic regression algorithms demonstrated with cosmology. *Journal of Cosmology and Astroparticle Physics*, 2025(01): 040.

Turek, S.; and Becker, C. 1998. FEATFLOW DFG Benchmark 2D-2. https://wwwold.mathematik.tu-dortmund.de/~featflow/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark2_re100.html. Accessed 2025-07-14.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Walker, J. A. 2013. *Dynamical systems and evolution equations: theory and applications*, volume 20. Springer Science & Business Media.

Yuan, Y.; Li, X.; Li, L.; Jiang, F. J.; Tang, X.; Zhang, F.; Goncalves, J.; Voss, H. U.; Ding, H.; and Kurths, J. 2023. Machine discovery of partial differential equations from spatiotemporal data: A sparse Bayesian learning framework. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(11).