Scientific Equation Discovery via Evolutionary Search with Large Language Models

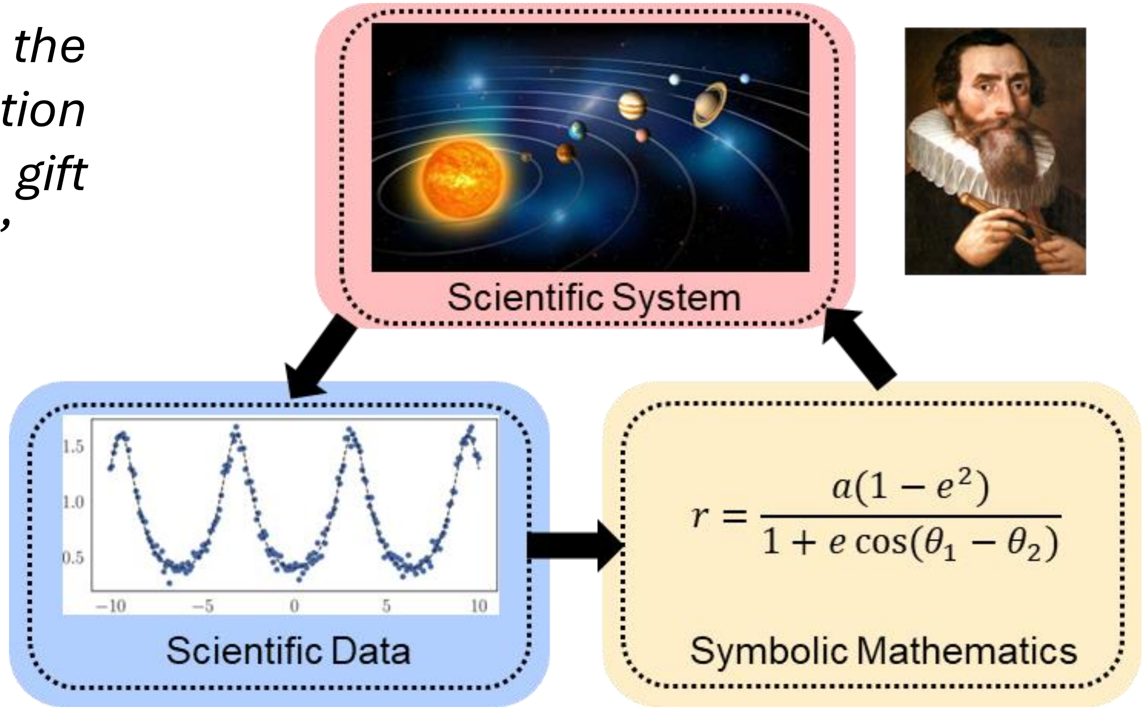
*Parshin Shojaee *Kazem Meidani Shashank Gupta Amir Barati Farimani Chandan K. Reddy

ICLR 2025 Oral Presentation

Unreasonable Effectiveness of Mathematics

“The miracle of the appropriateness of the ***language of mathematics*** *for the formulation of the* ***laws of physics*** *is a wonderful gift which we neither understand nor deserve.”*

Eugene Wigner -- The ***Unreasonable Effectiveness of Mathematics in the Natural Sciences***



Scientific System

Scientific Data

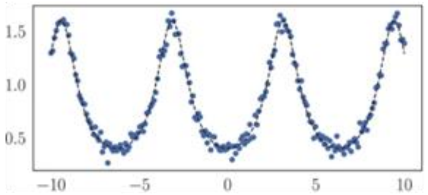
$$r = \frac{a(1-e^2)}{1+e\cos(\theta_1-\theta_2)}$$

Symbolic Mathematics

* Plot borrowed from the presentation of AI Feynman (2020)

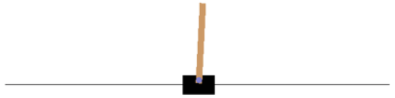
Scientific Discovery

Mathematical Equations help us *understand*, *build upon*, *predict*, and *control* scientific systems.



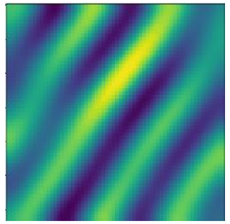
$$r = \frac{a(1-e^2)}{1 + e\cos(\theta_1 - \theta_2)}$$

Functional Relations



$$\frac{d\dot{\theta}}{dt} = \frac{Mg\sin\theta + mg\sin\theta - ml\dot{\theta}\sin\theta\cos\theta}{ML + ml\sin^2\theta}$$
$$\frac{d\dot{s}}{dt} = \frac{mL\dot{\theta}^2\sin\theta - mg\sin\theta\cos\theta}{M + m\sin^2\theta}$$

Dynamical Systems
(ODEs)



$$\frac{\partial u}{\partial t} = \nabla.(D\nabla u) - \nabla.(vu) + R$$

Partial Differential Equations
(PDEs)

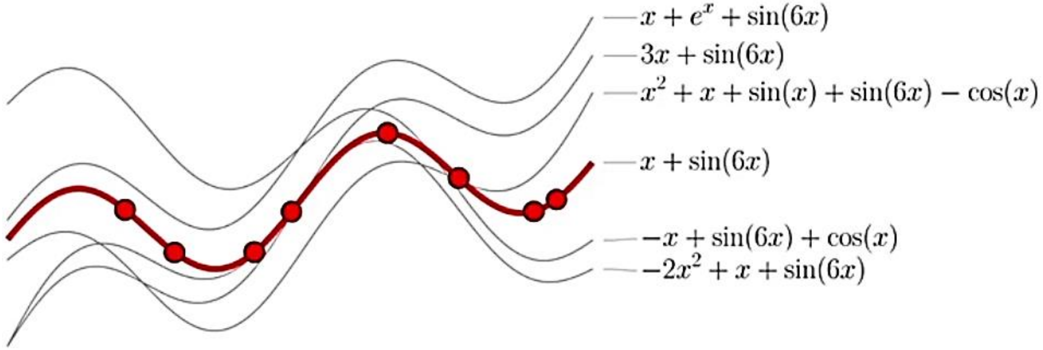
Equation Discovery | Definition

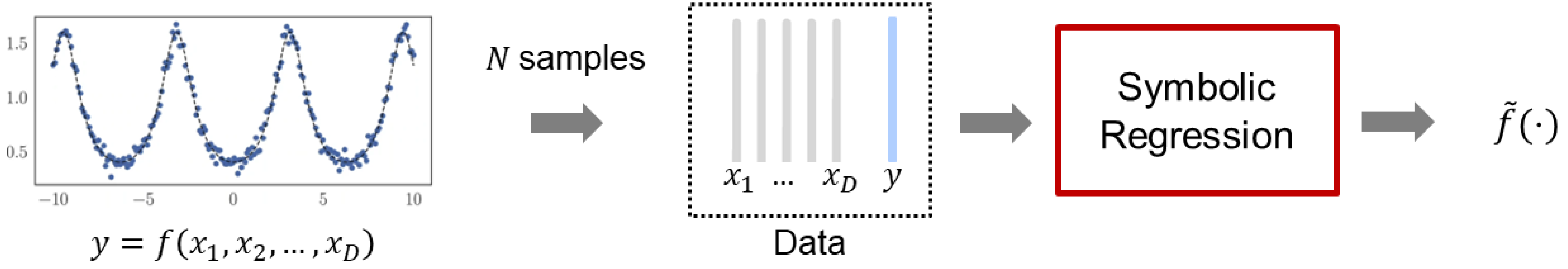
Given a dataset $(\boldsymbol{X_i}, \boldsymbol{y_i})_{\boldsymbol{i \leq N}}$, where each point $\boldsymbol{X_i} \in \mathbb{R}^{\boldsymbol{d}}$ and $\boldsymbol{y_i} \in \mathbb{R}$, find a **mathematical expression** $\tilde{\boldsymbol{f}}: \mathbb{R}^{\boldsymbol{d}} \rightarrow \mathbb{R}$ such that $\tilde{\boldsymbol{f}}(\boldsymbol{X_i}) \approx \boldsymbol{y_i}$

Accuracy

Interpretability

Generalization





$y = f(x_1, x_2, \ldots, x_D)$

N samples

x_1 … x_D y

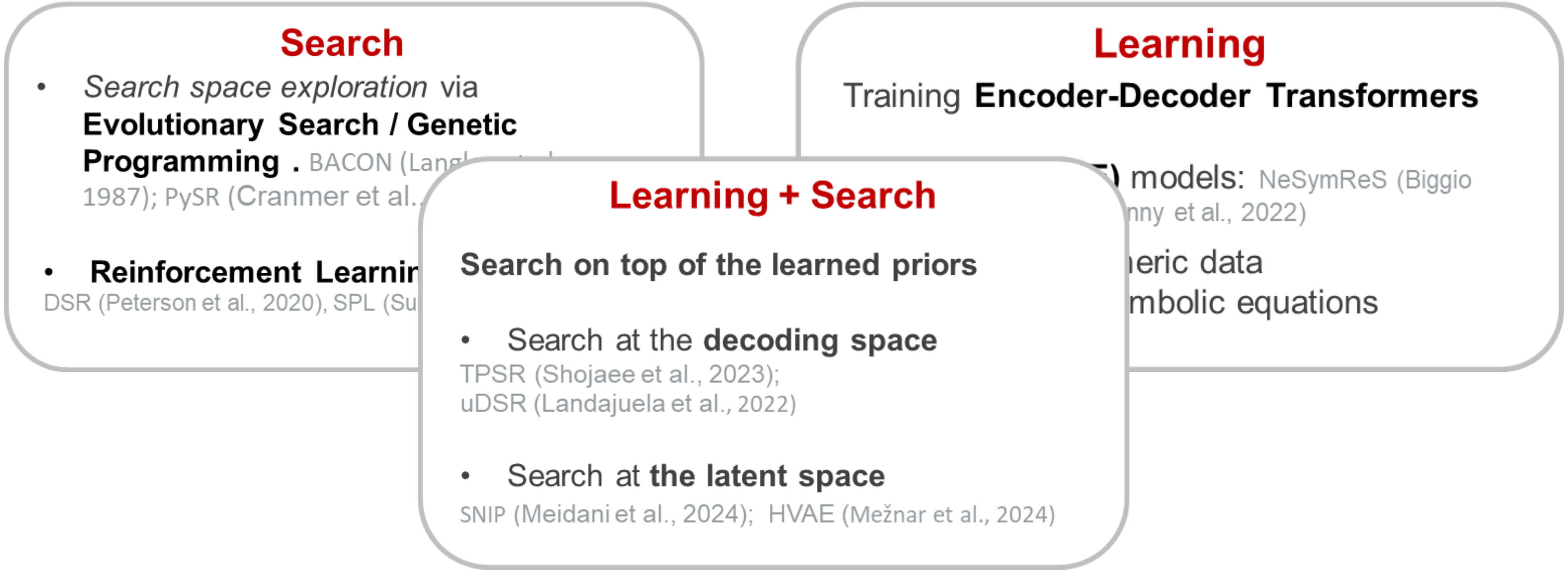
Data

Symbolic Regression

$\tilde{f}(\cdot)$

* Plot borrowed from presentation of Unified Deep Symbolic Regression (Landajuela et al., 2022)

Current Methods of Equation Discovery



Search

- *Search space exploration* via **Evolutionary Search / Genetic Programming .** BACON (Langl
1987); PySR (Cranmer et al.,

- **Reinforcement Learnin**
DSR (Peterson et al., 2020), SPL (Su

Learning

Training **Encoder-Decoder Transformers**

) models: NeSymReS (Biggio
nny et al., 2022)

ieric data
mbolic equations

Learning + Search

Search on top of the learned priors

- Search at the **decoding space**

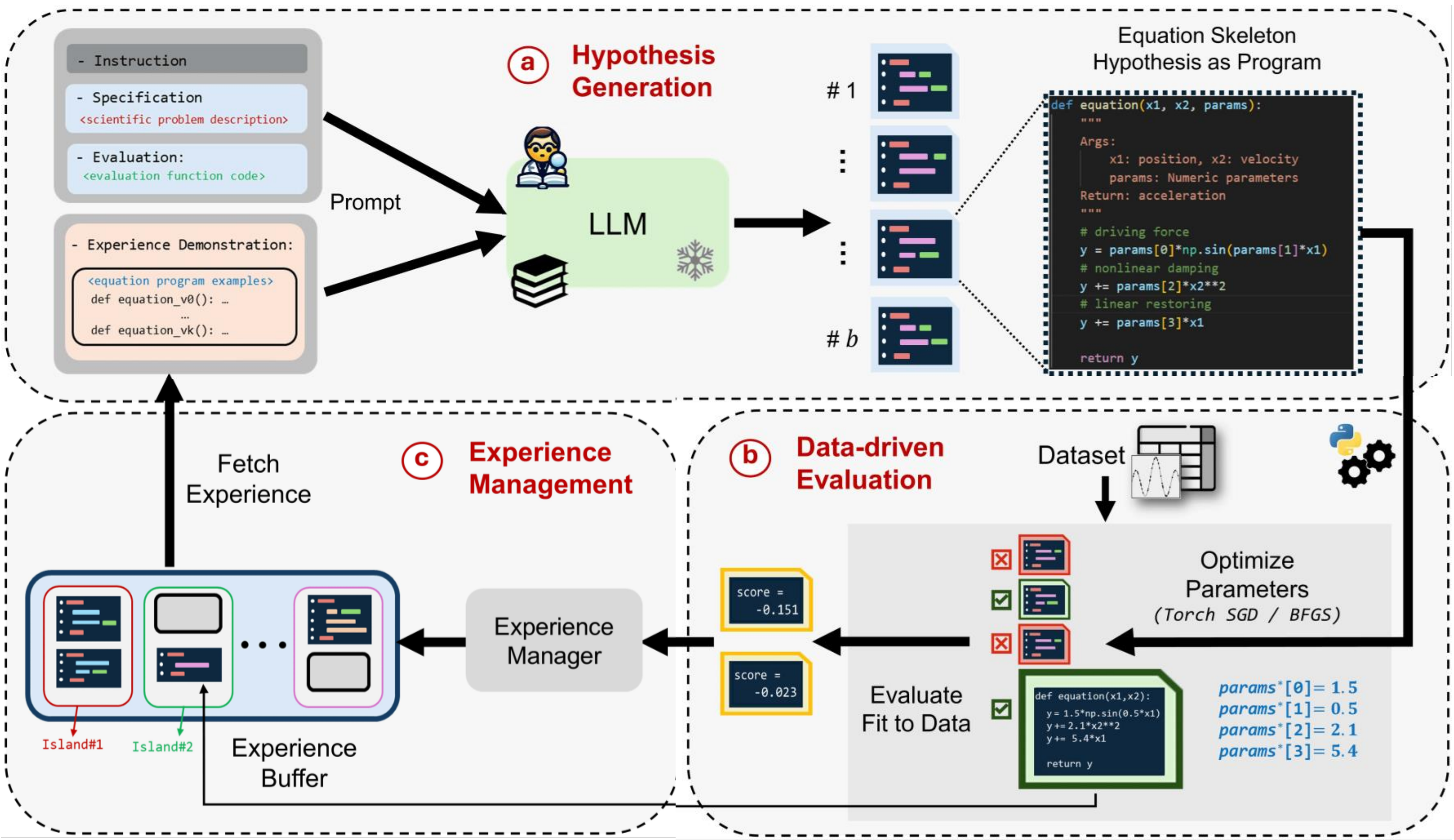
TPSR (Shojaee et al., 2023);
uDSR (Landajuela et al., 2022)

- Search at **the latent space**

SNIP (Meidani et al., 2024); HVAE (Mežnar et al., 2024)

- However, these techniques do not benefit from the context or domain knowledge for scientific problems.
- Can we incorporate such **scientific domain knowledge** into the process of equation discovery?

LLM-SR: Scientific Equation Discovery via Programming with LLMs



```
- Instruction
- Specification
  <scientific problem description>
- Evaluation:
  <evaluation function code>
```

```
- Experience Demonstration:
  <equation program examples>
  def equation_v0(): …
  …
  def equation_vk(): …
```

```
def equation(x1, x2, params):
    """
    Args:
        x1: position, x2: velocity
        params: Numeric parameters
    Return: acceleration
    """
    # driving force
    y = params[0]*np.sin(params[1]*x1)
    # nonlinear damping
    y += params[2]*x2**2
    # linear restoring
    y += params[3]*x1

    return y
```

☒ ☑ ☒ ☑

```
def equation(x1,x2):
  y = 1.5*np.sin(0.5*x1)
  y+= 2.1*x2**2
  y+= 5.4*x1

  return y
```

$params^*[0] = 1.5$
$params^*[1] = 0.5$
$params^*[2] = 2.1$
$params^*[3] = 5.4$

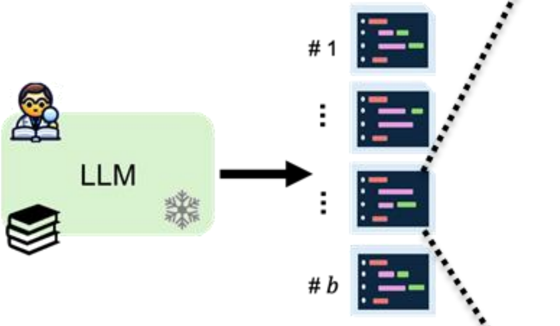
```
score =
  -0.151
```

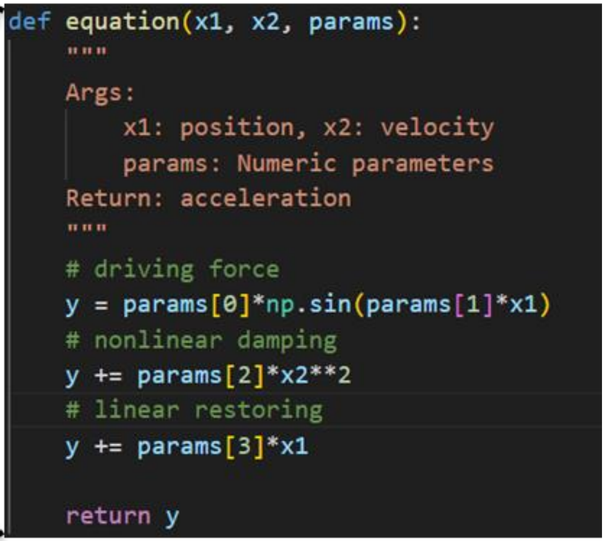
```
score =
  -0.023
```

LLM-SR | Hypothesis Generation

Equation as Program

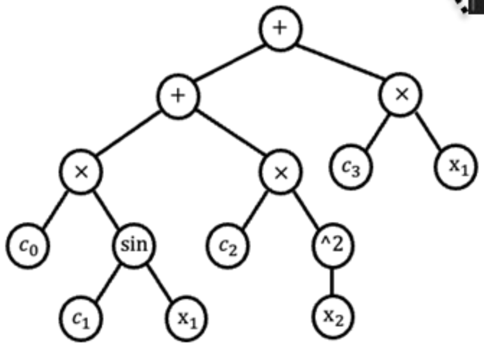
- Example: A Python function taking in **input variables** and **parameters** (equation coefficients) and **returning target variable**
- More **flexible** representation:
  - Piece-wise, Conditional (if-else), etc.
  - No need for defining a limited set of operators
- More **interpretable** representation:
  - Separate different components
  - Comments
- **Differentiable** programming





```
def equation(x1, x2, params):
    """
    Args:
        x1: position, x2: velocity
        params: Numeric parameters
    Return: acceleration
    """
    # driving force
    y = params[0]*np.sin(params[1]*x1)
    # nonlinear damping
    y += params[2]*x2**2
    # linear restoring
    y += params[3]*x1

    return y
```



$$c_0 \sin(c_1 x_1) + c_2 x_2^2 + c_3 x_1$$

LLM-SR | Data-Driven Evaluation

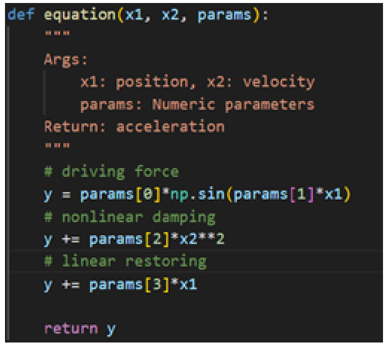
- Evaluating the generated hypothesis (Equation Skeleton)

Step 1) **Optimizing the parameters**

- Nonlinear Optimization:
    - NumPy Operators with SciPy BFGS
- Differentiable programming with gradient descent:
    - PyTorch Operators with SGD/Adam (gradient descent)

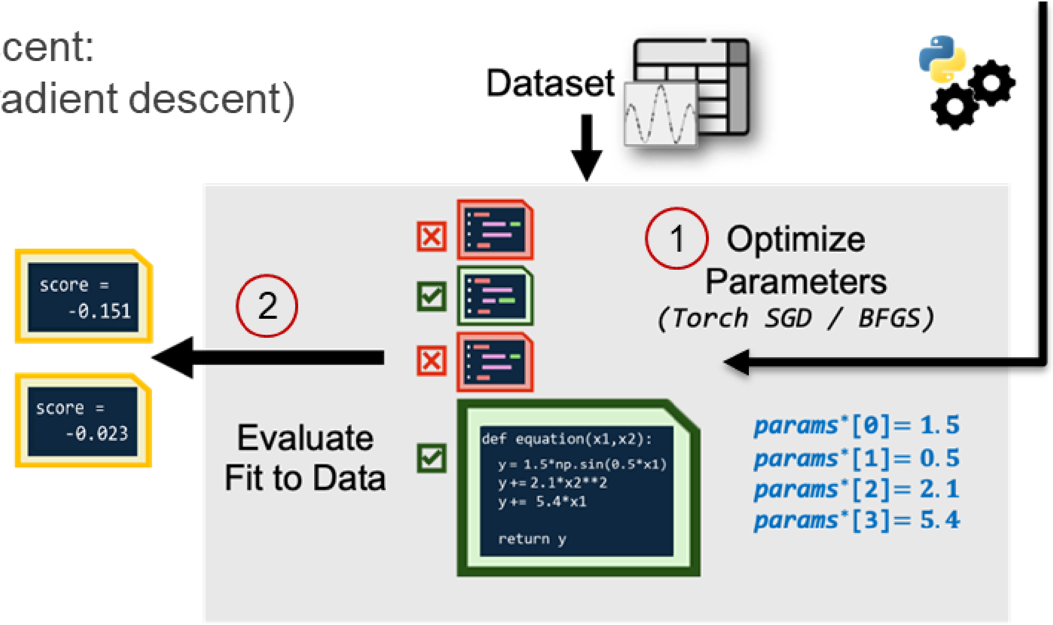
Step 2) **Evaluation Score**

- Discard infeasible equations
- Assign the score based on **fitness to data** with optimized parameters



```python
def equation(x1, x2, params):
    """
    Args:
        x1: position, x2: velocity
        params: Numeric parameters
    Return: acceleration
    """
    # driving force
    y = params[0]*np.sin(params[1]*x1)
    # nonlinear damping
    y += params[2]*x2**2
    # linear restoring
    y += params[3]*x1

    return y
```



```
score =
  -0.151
```

```
score =
  -0.023
```

```python
def equation(x1,x2):
  y= 1.5*np.sin(0.5*x1)
  y+= 2.1*x2**2
  y+=  5.4*x1

  return y
```

LLM-SR | Experience Management

- To better navigate the landscape of hypotheses and avoid local optima.

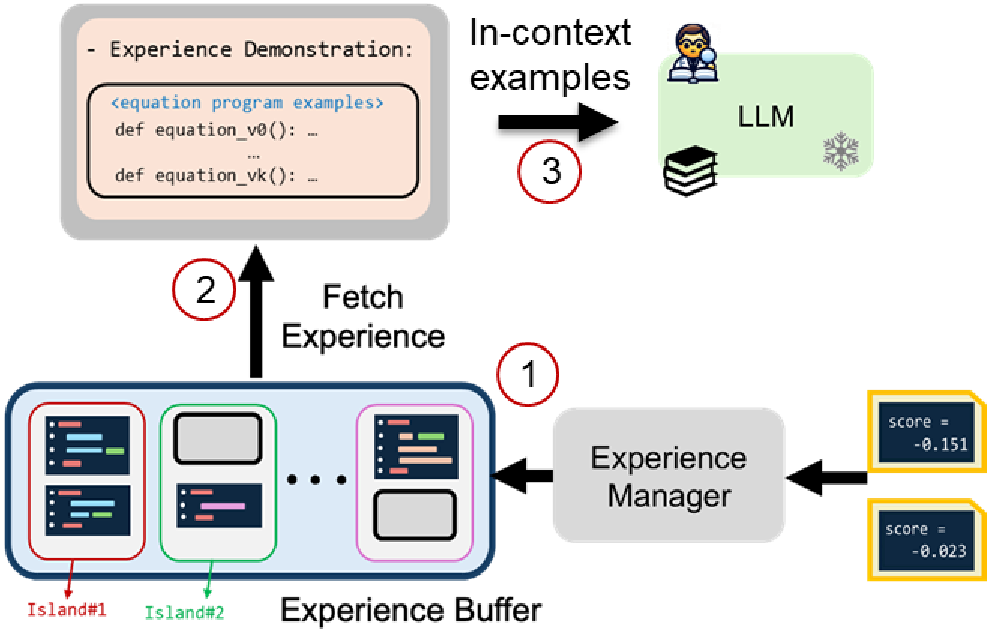
Step 1) **Store hypotheses in an experience buffer**:

- Multi-population (Islands) model to maintain diverse equations

Step 2) **Sample multiple examples** favoring higher scores and lower complexities from the buffer

Step 3) Provide these hypotheses as **in-context examples** to LLM.

- LLM performs as mutation and cross-over operators



```
- Experience Demonstration:
    <equation program examples>
    def equation_v0(): …
                  …
    def equation_vk(): …
```

```
score =
  -0.151
```

```
score =
  -0.023
```

Experiments | Datasets

Benchmarks

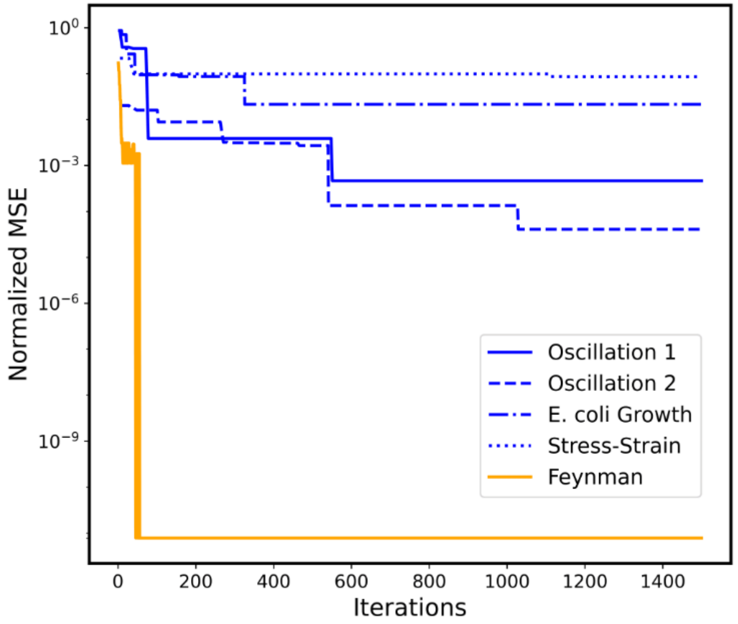
- LLM-SR rapidly finds well-known equations in benchmarks such as *Feynman* physics equations, suggesting that LLMs have likely **memorized** these prevalent equations.

- Therefore, we **introduce new benchmark problems** across different scientific domains, designed to simulate the conditions for scientific discovery:

  1) **Nonlinear damped oscillators:**
     - design arbitrary yet feasible nonlinear terms

  2) **Bacterial (E. coli) growth rate:**
     - custom models for temperature and pH dependency

  3) **Material stress behavior (Stress-Strain):**
     - experimental data covering tensile tests on an Aluminum alloy

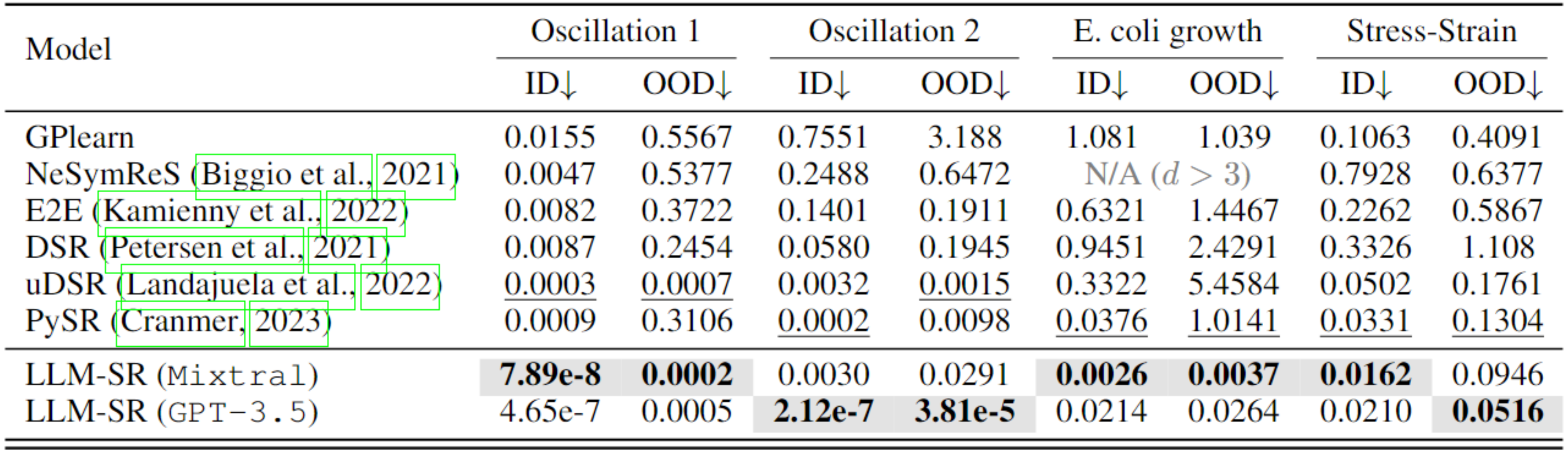


Findings | Accuracy

- LLM-SR (with both LLM backbones of 'Mixtral 8x7b-instruct' and 'GPT-3.5-Turbo') **consistently outperforms** state-of-the-art SR baselines without domain-specific knowledge.

- Reporting **Normalized Mean Squared Error (NMSE)** on:
  - **In-domain (ID):** test set coming from the same distribution as training data
  - **Out-of-domain (OOD):** test set coming from domains beyond the training data

- The performance gap between LLM-SR and baselines is more pronounced in OOD test settings compared to ID settings.

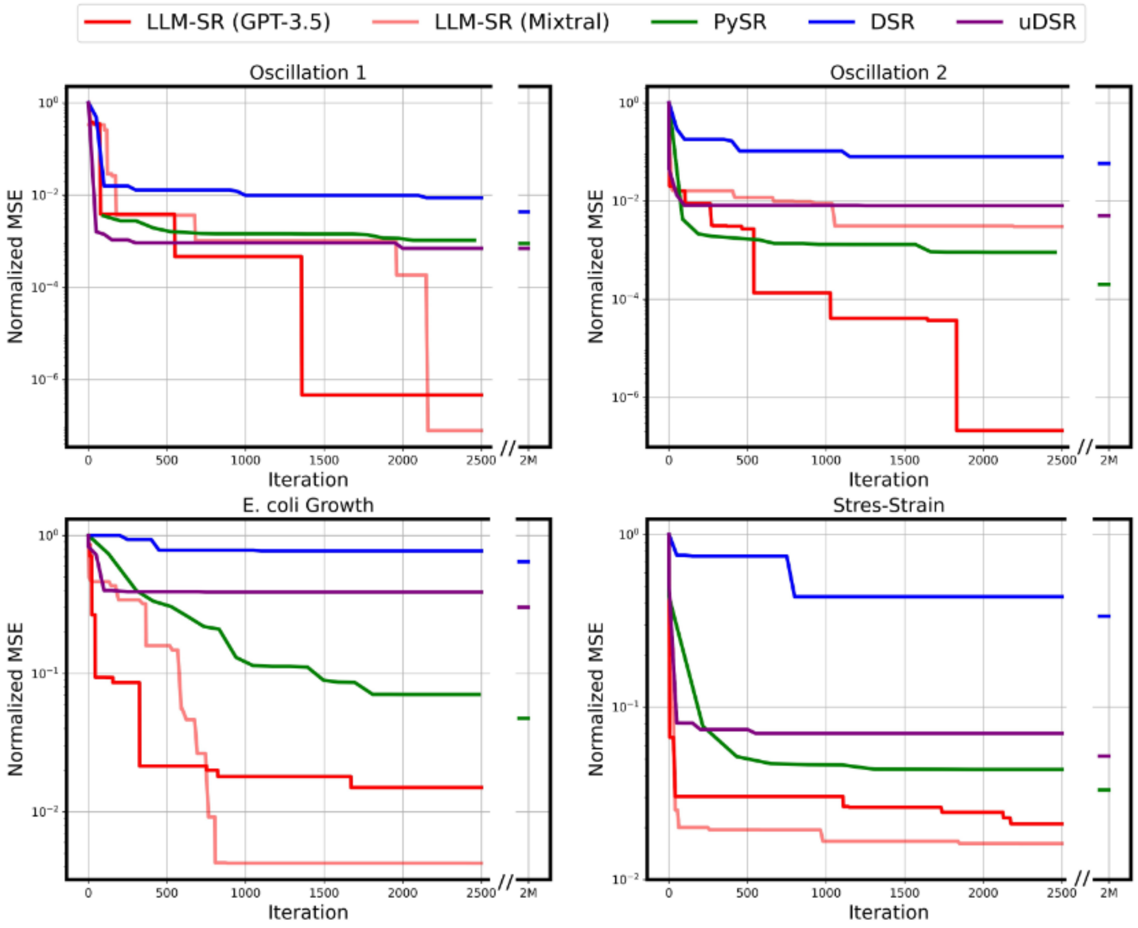


Model	Oscillation 1		Oscillation 2		E. coli growth		Stress-Strain	
	ID↓	OOD↓	ID↓	OOD↓	ID↓	OOD↓	ID↓	OOD↓
GPlearn	0.0155	0.5567	0.7551	3.188	1.081	1.039	0.1063	0.4091
NeSymReS (Biggio et al., 2021)	0.0047	0.5377	0.2488	0.6472	N/A ($d > 3$)		0.7928	0.6377
E2E (Kamienny et al., 2022)	0.0082	0.3722	0.1401	0.1911	0.6321	1.4467	0.2262	0.5867
DSR (Petersen et al., 2021)	0.0087	0.2454	0.0580	0.1945	0.9451	2.4291	0.3326	1.108
uDSR (Landajuela et al., 2022)	0.0003	0.0007	0.0032	0.0015	0.3322	5.4584	0.0502	0.1761
PySR (Cranmer, 2023)	0.0009	0.3106	0.0002	0.0098	0.0376	1.0141	0.0331	0.1304
LLM-SR (Mixtral)	**7.89e-8**	**0.0002**	0.0030	0.0291	**0.0026**	**0.0037**	**0.0162**	0.0946
LLM-SR (GPT-3.5)	4.65e-7	0.0005	**2.12e-7**	**3.81e-5**	0.0214	0.0264	0.0210	**0.0516**

Findings | Efficiency

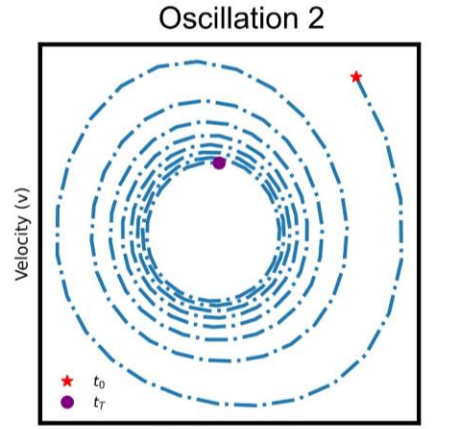
Efficiency

- LLM-SR achieves high fitting equations *faster* due to domain-specific knowledge.
- The *improvement gap widens* over iterations due to effective LLM modifications (mutation/crossover)



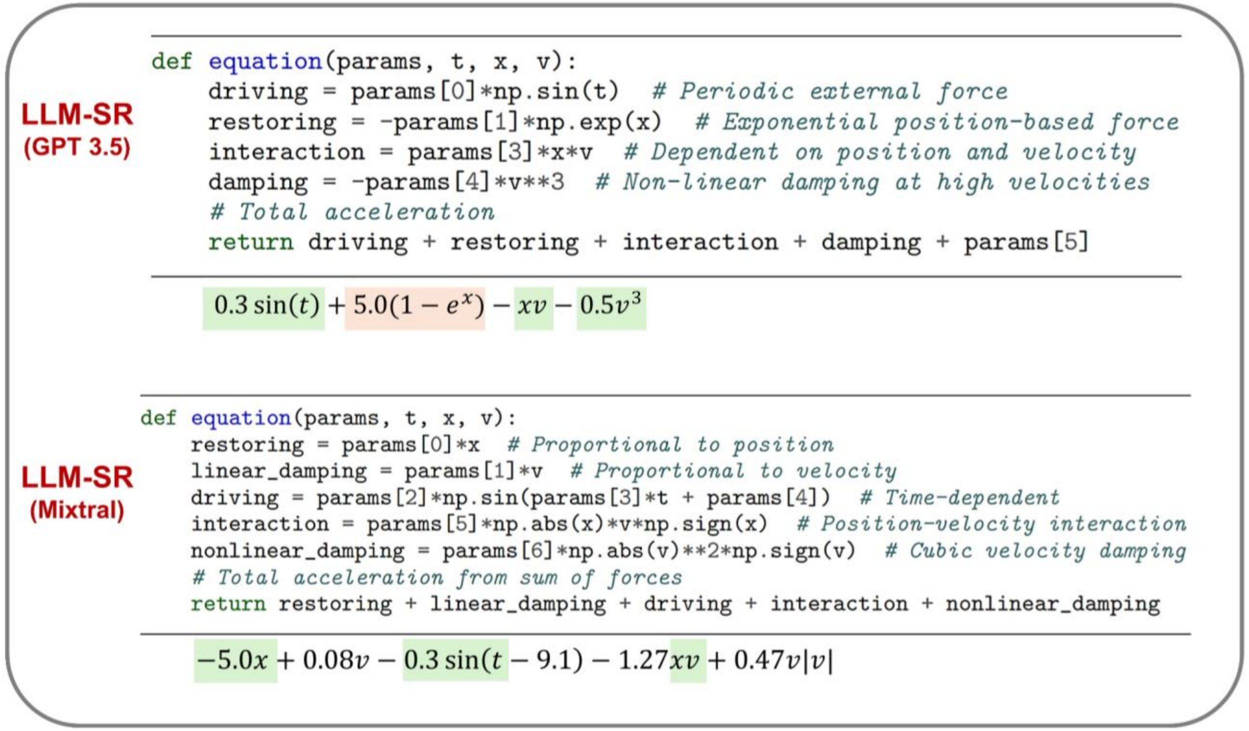
Findings | Interpretability

More interpretable Equations
(Nonlinear oscillation problem)



$$\dot{v} = 0.3\sin(t) - 0.5v^3 - xv - 5x\exp(0.5x)$$

(a) Ground Truth



LLM-SR
(GPT 3.5)

```python
def equation(params, t, x, v):
    driving = params[0]*np.sin(t)  # Periodic external force
    restoring = -params[1]*np.exp(x)  # Exponential position-based force
    interaction = params[3]*x*v  # Dependent on position and velocity
    damping = -params[4]*v**3  # Non-linear damping at high velocities
    # Total acceleration
    return driving + restoring + interaction + damping + params[5]
```

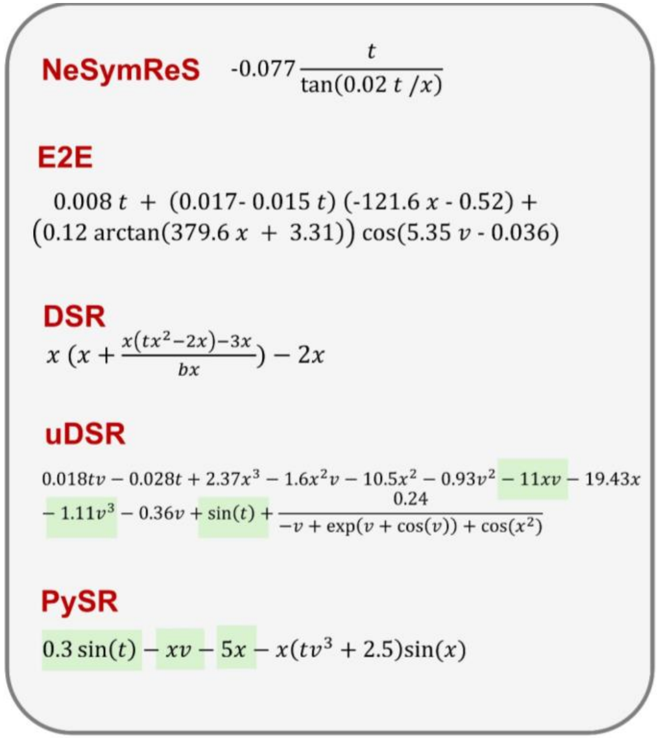
$$0.3\sin(t) + 5.0(1 - e^x) - xv - 0.5v^3$$

LLM-SR
(Mixtral)

```python
def equation(params, t, x, v):
    restoring = params[0]*x  # Proportional to position
    linear_damping = params[1]*v  # Proportional to velocity
    driving = params[2]*np.sin(params[3]*t + params[4])  # Time-dependent
    interaction = params[5]*np.abs(x)*v*np.sign(x)  # Position-velocity interaction
    nonlinear_damping = params[6]*np.abs(v)**2*np.sign(v)  # Cubic velocity damping
    # Total acceleration from sum of forces
    return restoring + linear_damping + driving + interaction + nonlinear_damping
```

$$-5.0x + 0.08v - 0.3\sin(t - 9.1) - 1.27xv + 0.47v|v|$$

(b) LLM-SR Equations



NeSymReS $-0.077\dfrac{t}{\tan(0.02\,t\,/x)}$

E2E

$$0.008\,t + (0.017 - 0.015\,t)(-121.6\,x - 0.52) + (0.12\arctan(379.6\,x + 3.31))\cos(5.35\,v - 0.036)$$

DSR

$$x\left(x + \frac{x(tx^2 - 2x) - 3x}{bx}\right) - 2x$$

uDSR

$$0.018tv - 0.028t + 2.37x^3 - 1.6x^2v - 10.5x^2 - 0.93v^2 - 11xv - 19.43x - 1.11v^3 - 0.36v + \sin(t) + \frac{0.24}{-v + \exp(v + \cos(v)) + \cos(x^2)}$$

PySR

$$0.3\sin(t) - xv - 5x - x(tv^3 + 2.5)\sin(x)$$

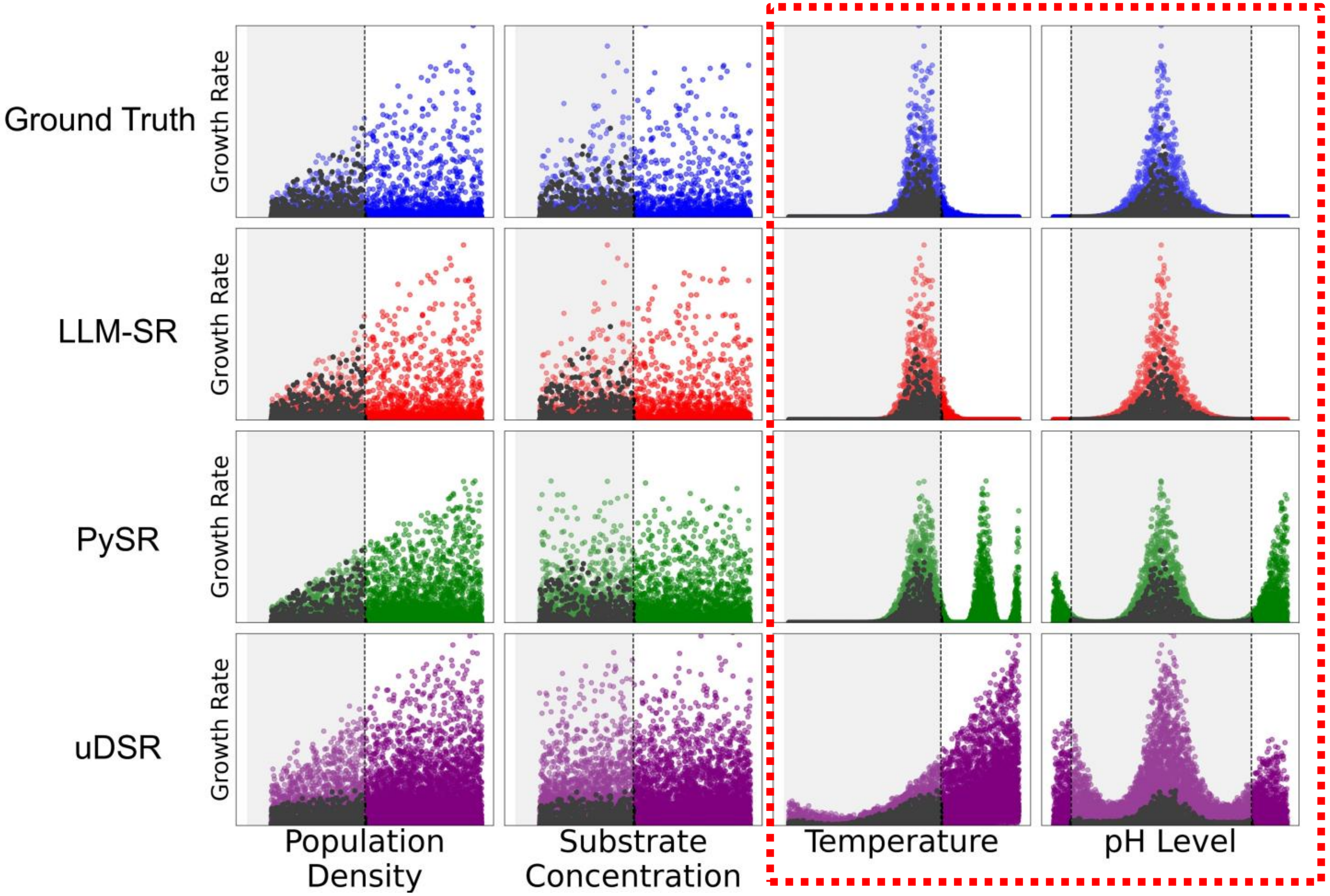
(c) Baseline SR Models

Qualitative Analysis

Better Out-of-domain (OOD) Generalization

(Bacterial growth rate problem)

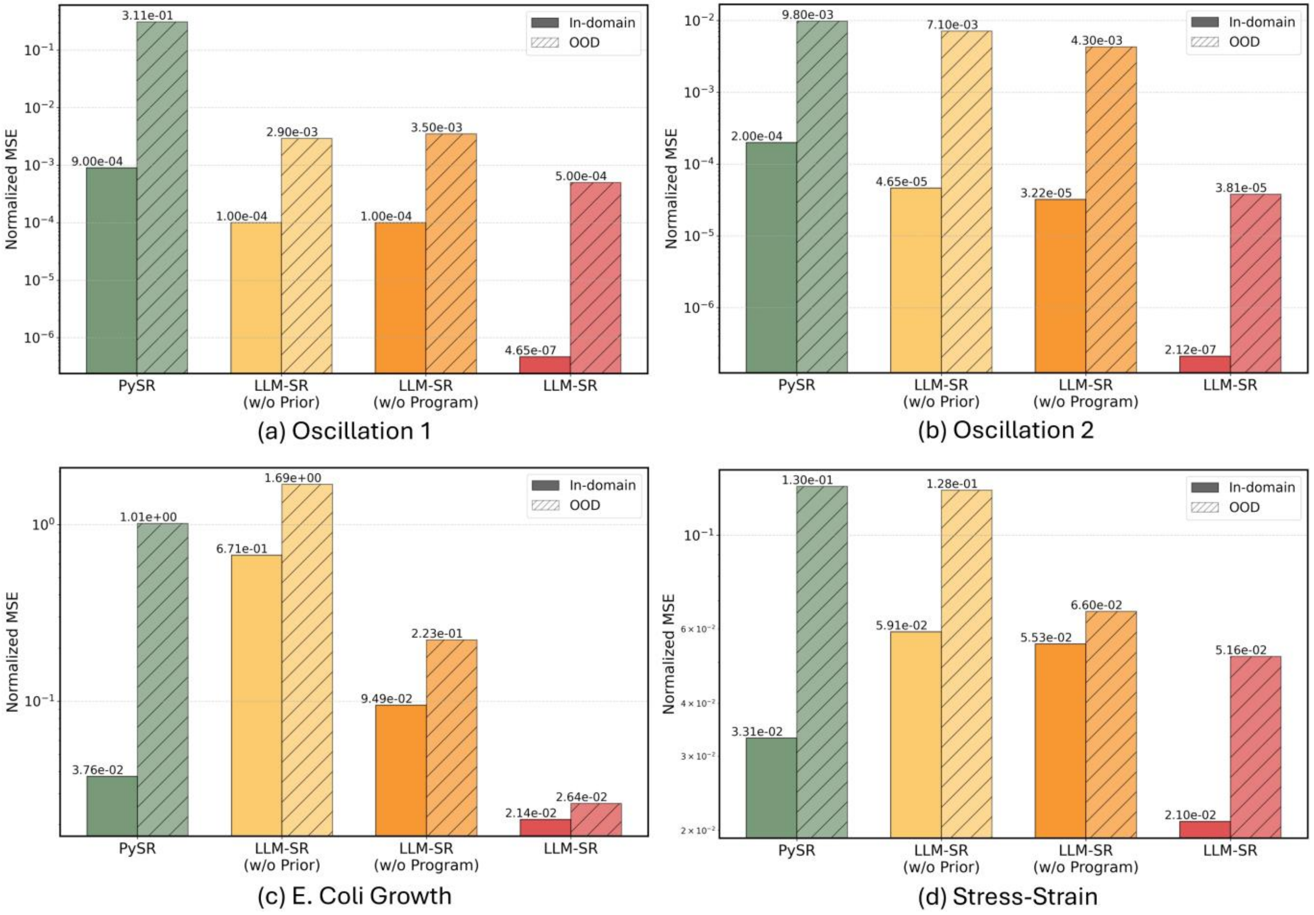
- Shaded regions and black points indicate training data samples



Ablation Study

Ablation Study of Key Components:

What's the impact of LLMs' **(1) Scientific Prior Knowledge** and **(2) Code Generation** capabilities in **equation discovery?**



(a) Oscillation 1

(b) Oscillation 2

(c) E. Coli Growth

(d) Stress-Strain

Thank You! Q&A

Contact: parshinshojaee@vt.edu

Code on GitHub:

SCAN ME