Paper:

OpInf-LLM: Parametric PDE Solving with LLMs via Operator Inference

Code:

Zhuoyuan Wang, Hanjiang Hu, Xiyu Deng, Saviz Mowlavi, Yorie Nakahira

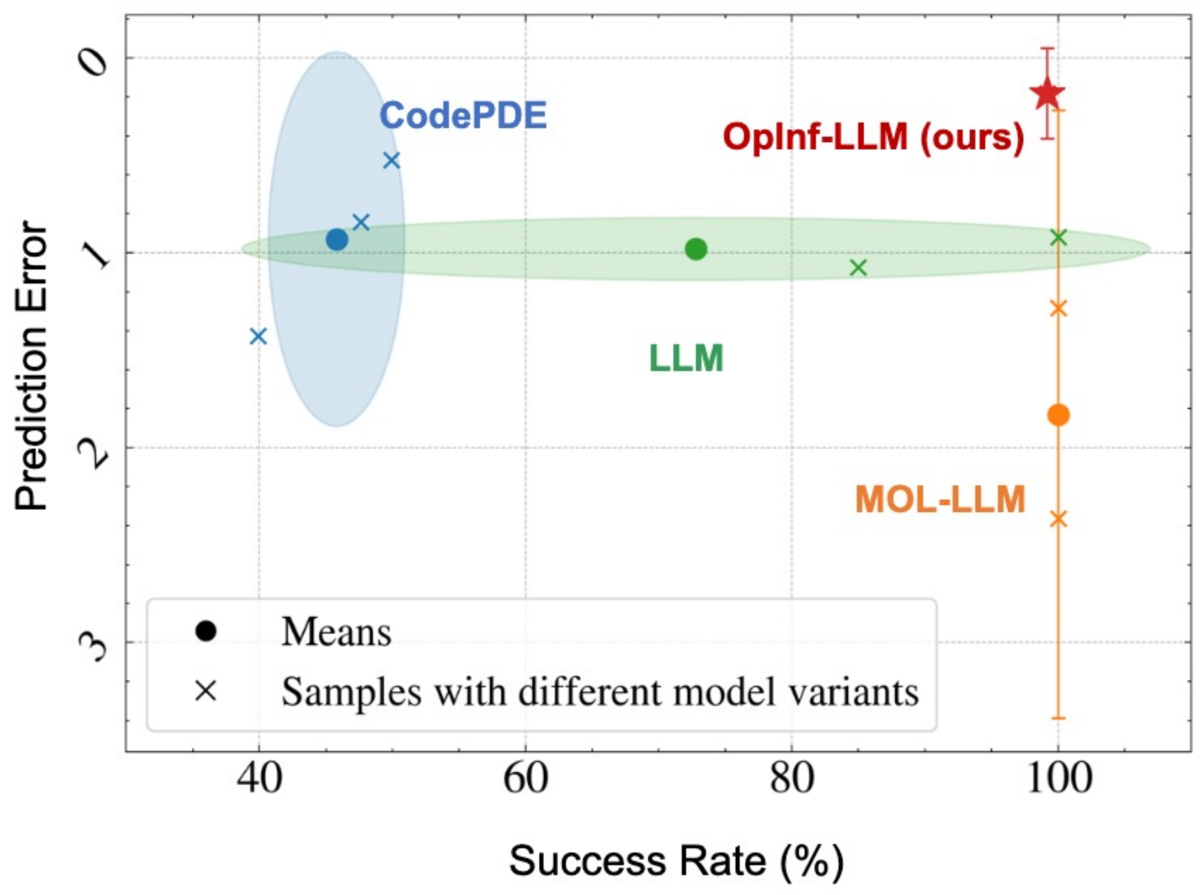
Carnegie Mellon University

MITSUBISHI ELECTRIC RESEARCH LABORATORIES

Motivation

LLM-based partial differential equation (PDE) solving is difficult due to the following challenges:

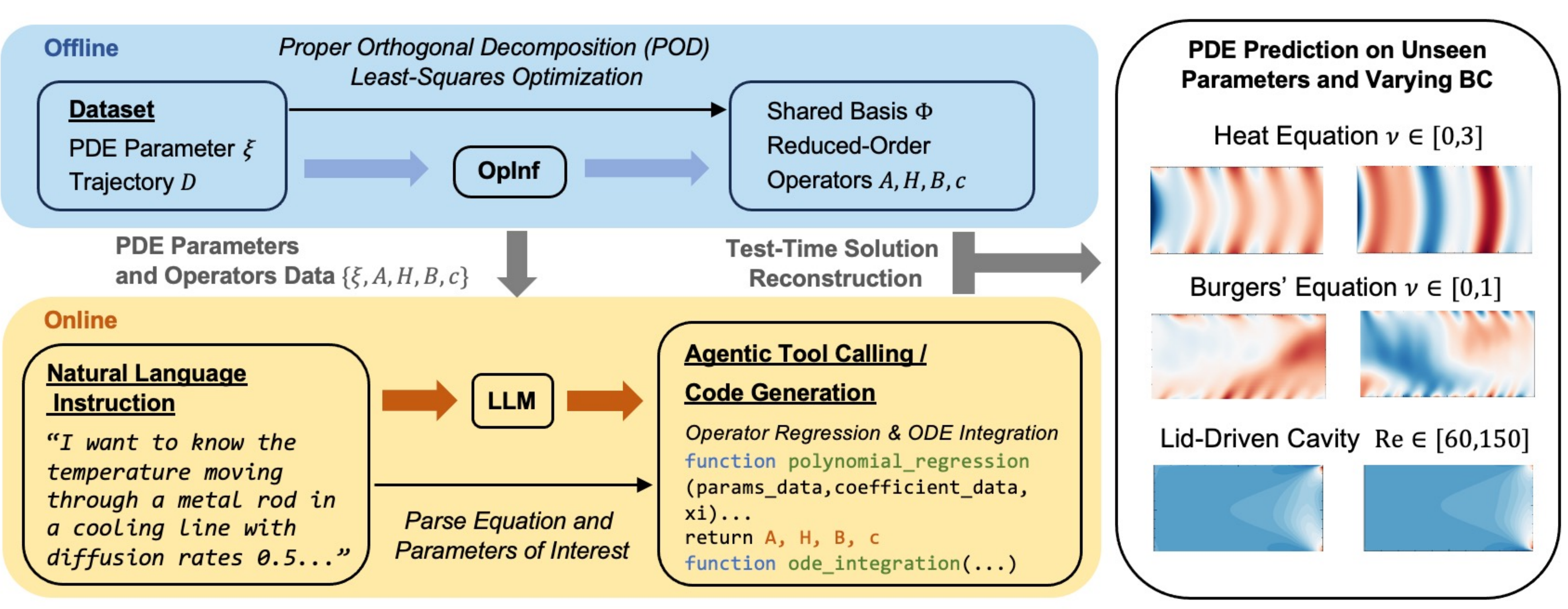
- LLM is not able to predict diverse PDE solutions directly from human instruction.
- LLM-based solver generation suffers from code execution failures and does not incur consistently high accuracy.
- Foundation model (FM) based methods require large dataset and hard to generalize to settings beyond training regime without fine-tuning.



Method

Key insights:

- leverage **operator inference** insight, to enable **LLM** generate reliable parametric **PDE solving** pipeline



Parametric operator inference (OpInf):

- Convert **parametric PDE solving** of diverse boundary conditions to **regression on operator space** and ODE integration

Consider a general class of nonlinear parametric PDEs

$$\frac{\partial y}{\partial t} = \mathcal{F}(\frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, \cdots, s, \xi), \quad x \in \Omega, \quad t \in [0, T],$$

with initial and boundary conditions

$$\mathcal{I}[y](x, 0) = g(x), \quad x \in \Omega, \qquad \mathcal{B}[y](x, t) = u(t), \quad x \in \partial\Omega,$$

OpInf approximates the PDE dynamics within a finite subspace spanned by a set of basis functions $\phi_1(x), \ldots, \phi_r(x)$

$$y(x, t, \xi) = \sum_{i=1}^{r} a_i(t, \xi)\phi_i(x),$$

where the modal coefficients $a_i(t, \xi)$ inherit reduced dynamics given ξ

$$\frac{da}{dt} = A(\xi)a + H(\xi)(a \otimes a) + B(\xi)u + c(\xi) \qquad (1)$$

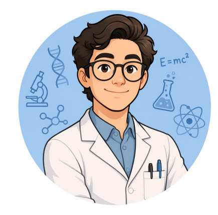
The operators A, H, B, c can be obtained through the following optimization

$$\min_{A,H,B,c} \sum_{k=1}^{K} \|[Aa + H(a \otimes a) + Bu + c - \dot{a}](t_k)\|_F^2 + \lambda \left(\|A\|_F^2 + \|H\|_F^2 + \|B\|_F^2 + \|c\|_2^2\right) \qquad (2)$$

where a and $\dot{a}$ can be obtained from the snapshot data at available ξ, and λ is the hyperparameter for regularization intensity

LLM online PDE solving:

- Parse diverse human instruction, including **non-technical** description



I want to study how temperature moves through a metal rod in a cooling line, with diffusion rates 0.1 and 0.5.

Heat equation with $\nu = 0.1, 0.5$

Method	Exact Match	Field Accuracy
Static Parser	52% (26/50)	75.7% (109/144)
OpInf-LLM	**100% (50/50)**	**100% (144/144)**

- **Reason** over correct operator dependence according to **physical law** without explicit prompts
  - Obtain correct regression parameterization without explicit prompting

Method	**Test** Re	**Distribution**	**Feature**	**Error**
OpInf	110	In-distribution	Re	3.57e−2
OpInf-LLM	110	In-distribution	1/Re	2.57e−2
OpInf	200	OOD	Re	instability
OpInf-LLM	200	OOD	1/Re	7.47e−2

- **Construct solver** based on OpInf for diverse PDE solving, through either **tool-calling** or **code generation**
  - Use regression on available A, H, B, c from optimization (2) to obtain operators for PDE parameters ξ of interest
  - Solve ODE (1) to obtain PDE solution of diverse BC

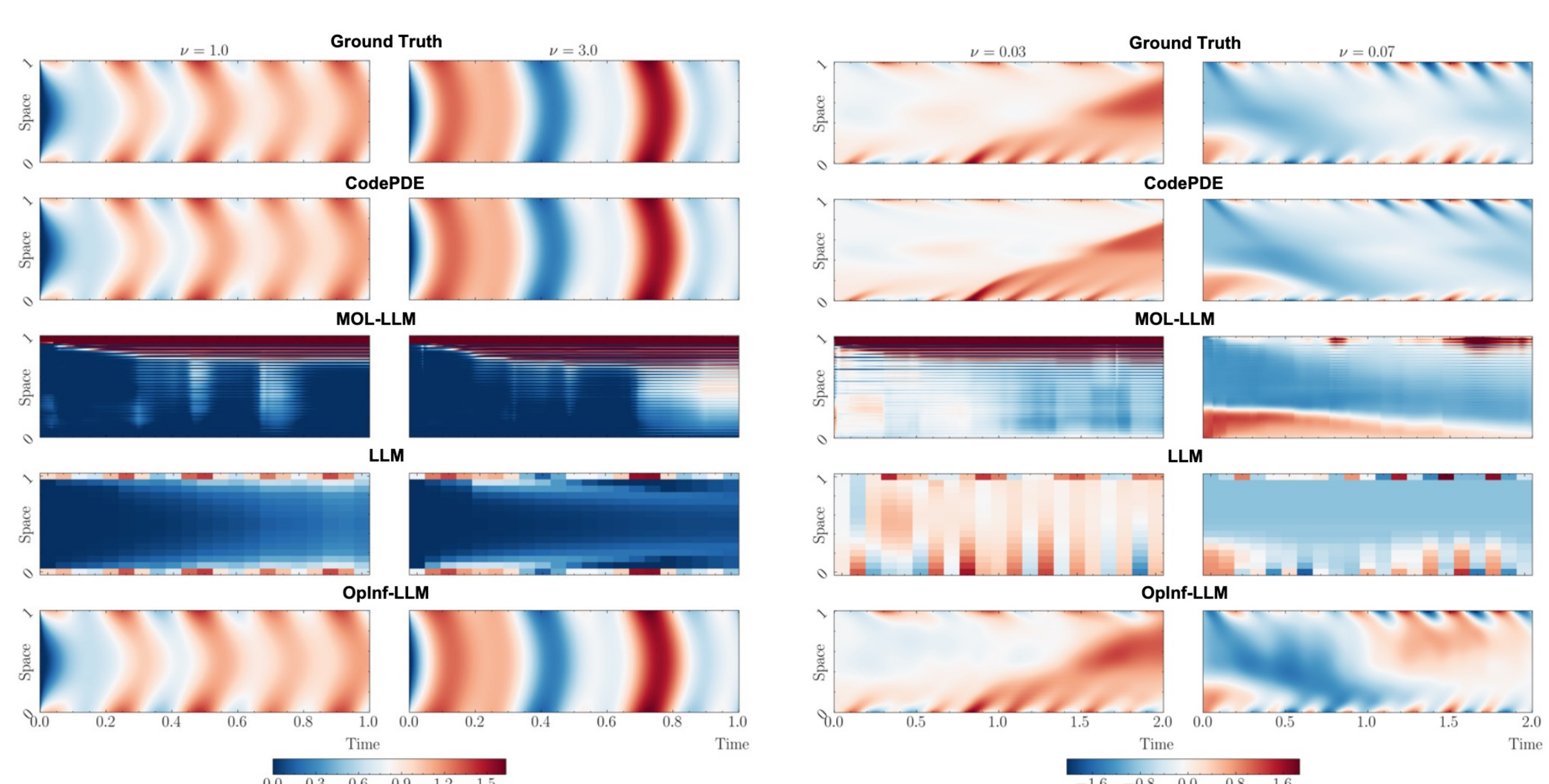
Metric	**Value**
Bug-free rate	100%
Success rate	99.2% (1 ROM instability)
Avg. code attempts	1.00 (tool-calling) 1.89 (code generation)

Experiments

- Require only **small amount** of **data**
- **High execution success rate** for both tool-calling and code generation
- Accurately generalize to **unseen parameters** and **BCs**, and to **longer time horizon**

Table 1. Results summary. Average relative L_2 error is reported for each equation. **Bold** denotes the best result, and underline denotes the second best. ↑ higher is better, ↓ lower is better.

Method	Data	Train time (s)	Success rate ↑	Heat ↓	Burgers ↓	Cavity ↓
CodePDE (GPT-4.1)	–	–	49.9%	1.59e−2	**1.50e−1**	1.41e0
CodePDE (GPT-4o)	–	–	39.9%	1.60e−1	1.23e0	2.89e0
CodePDE (Gemini-2.0-flash)	–	–	47.6%	1.74e−2	9.56e−1	1.55e0
MOL-LLM	449	14306	**100.0%**	1.69e0	1.44e0	7.24e−1
MOL-LLM (large dataset)	15000	35915	**100.0%**	4.87e0	1.58e0	6.44e−1
LLM (GPT-4.1)	–	–	**100.0%**	7.79e−1	9.82e−1	1.00e0
LLM (GPT-4o)	–	–	85.0%	9.24e−1	1.30e0	1.00e0
LLM (Gemini-2.0-flash)	–	–	33.3%	9.02e−1	failed	failed
OpInf-LLM (GPT-4.1)	449	30	99.2%	**1.29e−2**	4.91e−1	**4.63e−2**
OpInf-LLM (GPT-4o)	449	30	99.2%	**1.29e−2**	4.91e−1	**4.63e−2**
OpInf-LLM (Gemini-2.0-flash)	449	30	99.2%	**1.29e−2**	4.91e−1	**4.63e−2**



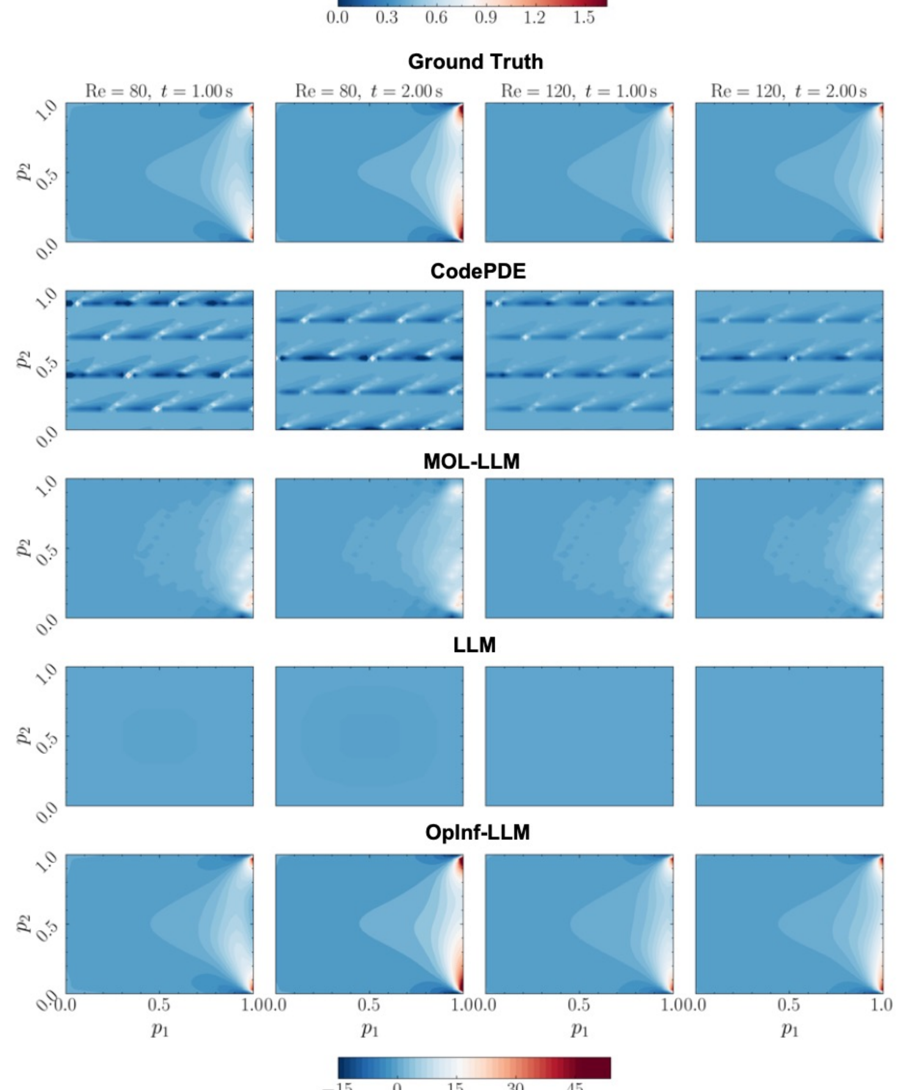


Table 2. Prediction error of OpInf-LLM with extrapolation in time.

LLM model	Equation	Error $[0, T]$	Error $[T, 2T]$
GPT-4o	Heat	1.29e−2	3.60e−2
	Burgers	4.91e−1	6.36e−1
	Cavity	4.63e−2	4.86e−2
Gemini-2.0 flash	Heat	1.29e−2	3.60e−2
	Burgers	4.91e−1	6.36e−1
	Cavity	4.63e−2	4.86e−2

Table 3. POD basis ablation for OpInf-LLM.

Equation	POD	Energy (%)	Error $[0, T]$	Error $[T, 2T]$
Burgers	4	88.01	3.93e−1	4.99e−1
Burgers	5	92.26	3.44e−1	4.48e−1
Burgers	6	95.63	4.13e−1	9.63e−1
Burgers	7	97.26	3.52e−1	5.41e−1
Burgers	8	98.31	3.64e−1	4.90e−1
Burgers	9	98.85	4.70e−1	6.46e−1
Burgers	10	99.20	4.91e−1	6.36e−1