Meta-Auto-Decoder for Solving Parametric Partial Differential Equations

Xiang Huang[1] Zhanhong Ye[2] Hongsheng Liu[3]

Beiji Shi[3] Zidong Wang[3] Kang Yang[3] Yang Li[3] Min Wang[3]

Haotian Chu[3] Fan Yu[3] Bei Hua[1] Lei Chen[4] Bin Dong[2]

1 University of Science and Technology of China 1958

2 Peking University 1898

3 HUAWEI

4 HKUST

Definition of Parametric PDEs

General form of parametric PDEs

$$\mathcal{L}_{\tilde{x}}^{\gamma_1} u = 0, \quad \tilde{x} \in \Omega \subset \mathbb{R}^d,$$
$$\mathcal{B}_{\tilde{x}}^{\gamma_2} u = 0, \quad \tilde{x} \in \partial\Omega.$$

Given $u \in \mathcal{U}$ and $\eta \in \mathcal{A}$, solving parametric PDEs requires to learn an infinite-dimensional operator

$$G: \mathcal{A} \to \mathcal{U}$$

that map any PDE parameter η to its corresponding solution u^η (i.e., the solution mapping).

Symbol	Definition
Ω	Solution regions
$\partial\Omega$	The boundaries of Ω
$\mathcal{L}^{\gamma_1}$	Partial differential operators corresponding to governing equations parametrized by γ_1
$\mathcal{B}^{\gamma_2}$	Partial differential operators corresponding to boundary conditions parametrized by γ_2
$\tilde{x}$	Independent variable
u	Solution of the PDEs
$\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^d)$	The function space of the solution of PDEs (i.e., $u \in \mathcal{U}$)
$\eta = (\gamma_1, \gamma_2, \Omega)$	The variable parameter of the PDEs
$\mathcal{A}$	The space of PDE parameters (i.e., $\eta \in \mathcal{A}$)
u^η	Solution u specific to the PDE parameter η

Classification of Learning-Based PDE Solvers

- **NN as a new ansatz of solution** Approximating the solution of the PDEs with a neural network. PDEs or their variant forms are used as loss terms for training the neural network.
  - Physics-Informed Neural Networks (PINNs) M. Raissi et al., JCP, 378:686-707, 2019.
  - Deep Galerkin Method (DGM) Sirignano and Spiliopoulos, JCP, 375:1339-1364, 2018.
  - Deep Ritz Method (DRM) W. E and B. Yu, CMS, 6(1), 1-12, 2018.
  - Weak Adversarial Network (WAN) Y. Zang et al., JCP, 411:109409, 2020.

- **NN as a new ansatz of solution mapping** Using neural networks to learn the solution mapping between two infinite-dimensional function spaces.
  - PDE-Net Z. Long et al., ICML 2018.
  - Deep Operator network (DeepONet) L. Lu, P. Jin, G. E. Karniadakis, arXiv:1910.03193.
  - Fourier Neural Operator (FNO) Z. Li et al., arXiv:2010.08895.

Classification of Learning-Based PDE Solvers

Category	Method	Label-Free	Mesh-Free	Without Retraining
NN as a new ansatz of solution	PINNs	✓	✓	✗
	DGM	✓	✓	✗
	DRM	✓	✓	✗
	WAN	✓	✓	✗
NN as a new ansatz of solution mapping	PDE-Net	✗	✗	✓
	DeepONet	✗	✓	✓
	FNO	✗	✗	✓

- **Label-Free:** Working in an unsupervised manner without generating labeled data from traditional computational methods or collecting data from the real world.
- **Mesh-Free**: Predefined mesh is not required. Training and inference can be performed on random coordinate points.
- **Without Retraining**: For a new PDE parameter η, inference can be performed directly without retraining.

Dilemma of Methods like PINNs

General form of parametric PDEs

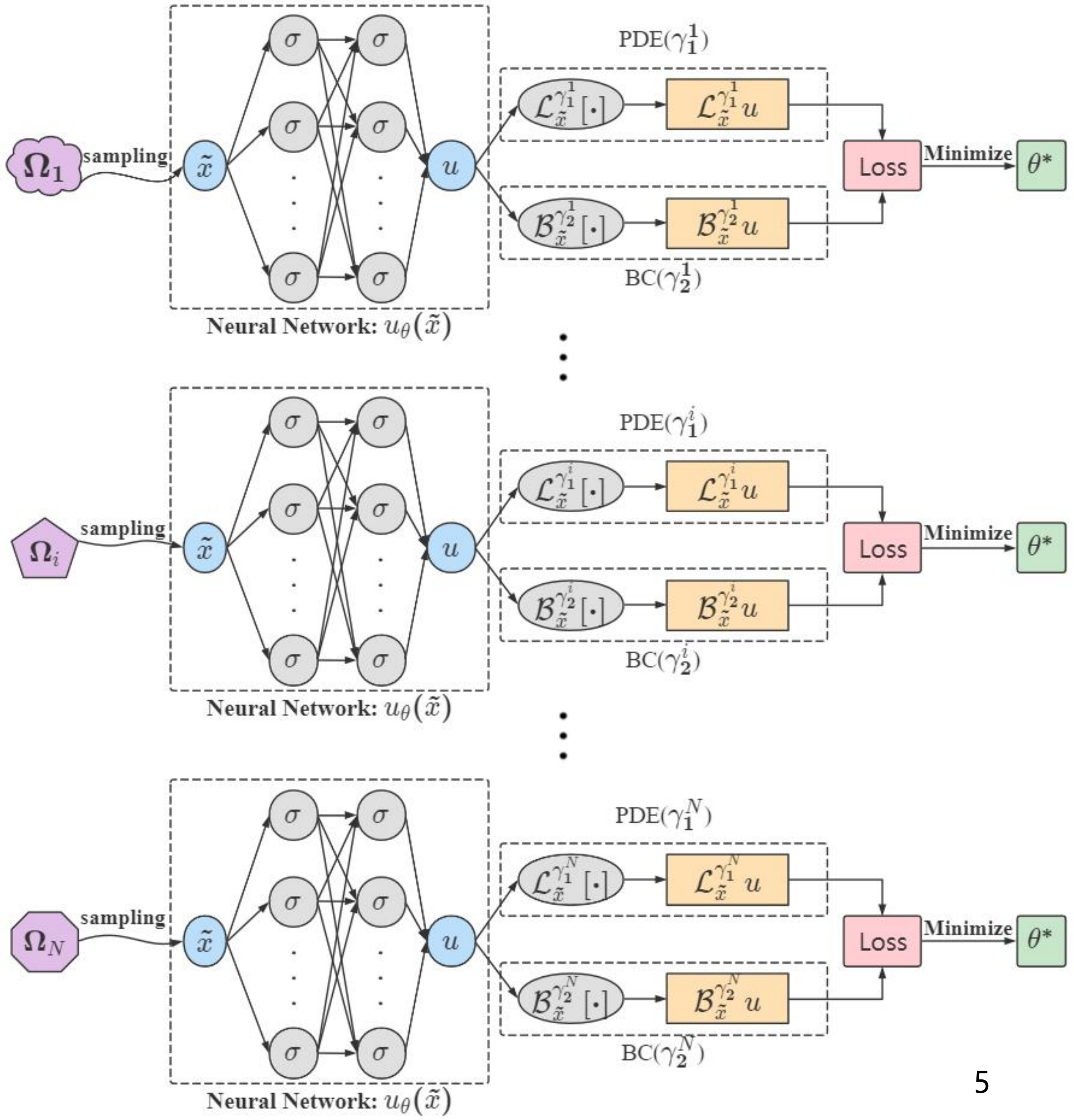
$$\mathcal{L}_{\tilde{x}}^{\gamma_1} u = 0, \quad \tilde{x} \in \Omega \subset \mathbb{R}^d,$$
$$\mathcal{B}_{\tilde{x}}^{\gamma_2} u = 0, \quad \tilde{x} \in \partial\Omega.$$

Physics-informed loss

$$L^{\eta_i}[u] = \left\| \mathcal{L}_{\tilde{x}}^{\gamma_1^i} u \right\|_{L^2(\Omega_i)}^2 + \lambda_{bc} \left\| \mathcal{B}_{\tilde{x}}^{\gamma_2^i} u \right\|_{L^2(\partial\Omega_i)}^2$$

$$\hat{L}^{\eta_i}[u] = \frac{1}{M_r} \sum_{j=1}^{M_r} \left\| \mathcal{L}_{\tilde{x}}^{\gamma_1^i} u(\tilde{x}_j^r) \right\|_2^2 + \frac{\lambda_{bc}}{M_{bc}} \sum_{j=1}^{M_{bc}} \left\| \mathcal{B}_{\tilde{x}}^{\gamma_2^i} u(\tilde{x}_j^{bc}) \right\|_2^2$$

The model weight θ^* need to be trained from scratch separately using loss $\hat{L}^{\eta_i}[u]$ for each PDE parameter $\eta_i = (\gamma_1^i, \gamma_2^i, \Omega_i)$.



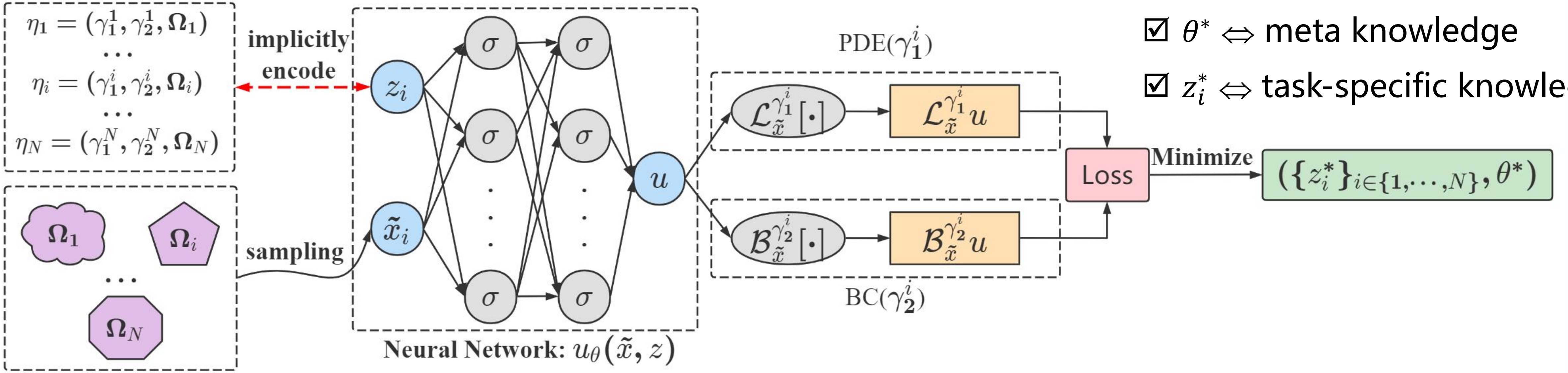
Pre-training of Meta-Auto-Decoder (MAD)

Key Points:

☑ $\eta_i \Leftrightarrow$ one task $\Leftrightarrow z_i$

☑ $\theta^* \Leftrightarrow$ meta knowledge

☑ $z_i^* \Leftrightarrow$ task-specific knowledge



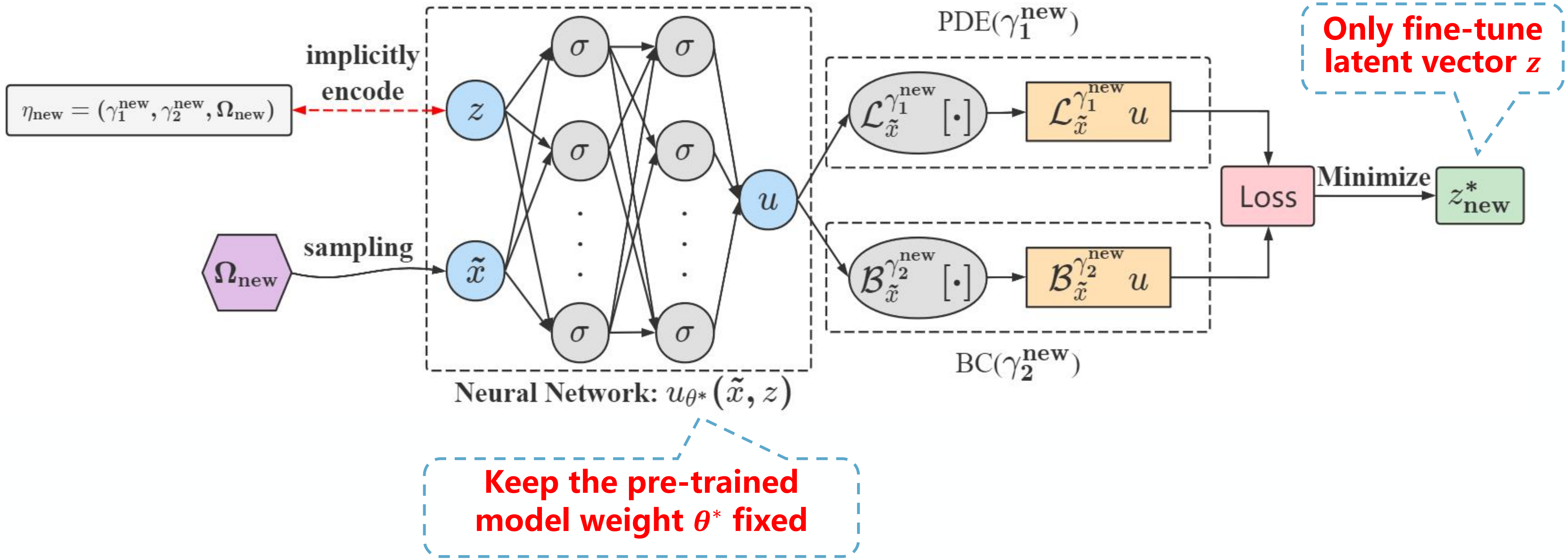
- **Pre-training Stage** Given N randomly generated PDE parameters $\eta_1, \cdots, \eta_N \in \mathcal{A}$, through solving the following optimization problem, a pre-trained model parametrized by θ^* is learned for all tasks and each task is paired with its own decoded latent vector z_i^*.

$$(\{z_i^*\}_{i\in\{1,\cdots,N\}}, \theta^*) = \underset{\theta,\{z_i\}_{i\in\{1,\cdots,N\}}}{\arg\min} \sum_{i=1}^{N} \left(\hat{L}^{\eta_i}[u_\theta(\cdot\,, z_i)] + \frac{1}{\sigma^2}\|z_i\|^2\right)$$

Physical-informed loss corresponding to each PDE parameter η_i

For training stability

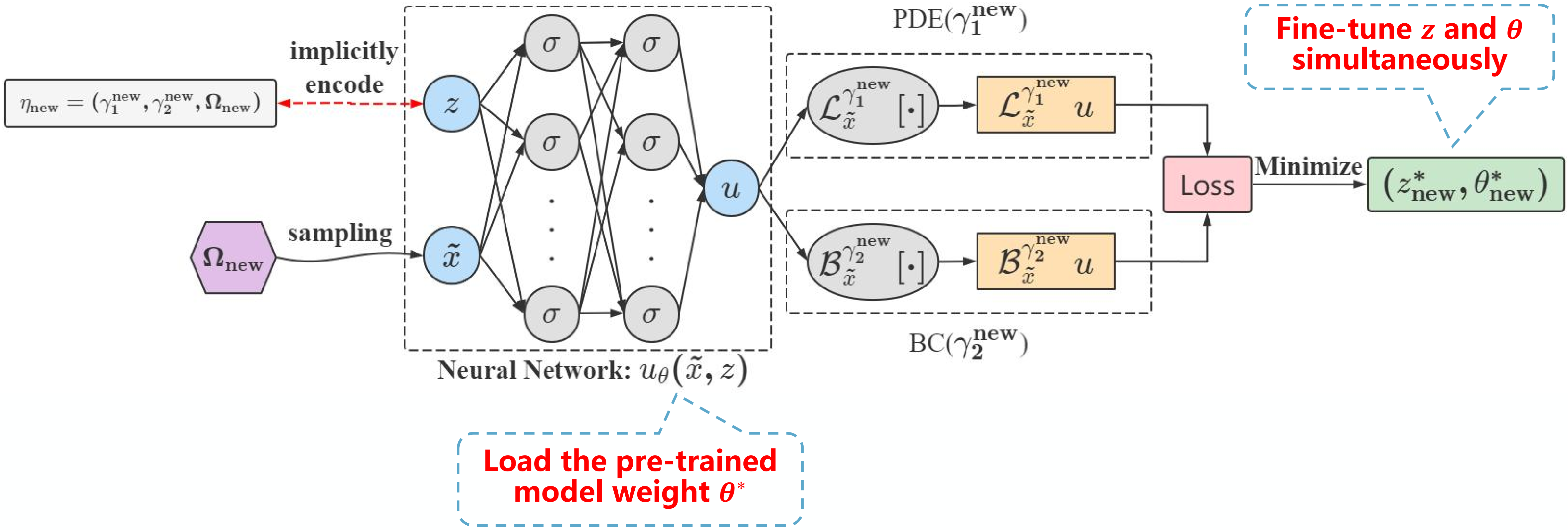
First Fine-tuning Strategy: MAD-L



- **Fine-tuning Stage (MAD-L)** Given a new PDE parameter η_{new}, MAD-L keeps the pre-trained weight θ^* fixed, and minimizes the following loss function to get obtain the optimal latent vector z^*_{new}. Then, $u_{\theta^*}(\cdot, z^*_{new})$ is the approximate solution of PDEs with parameter η_{new}.

$$z^*_{new} = \arg\min_{z} \hat{L}^{\eta_{new}}[u_{\theta^*}(\cdot, z)] + \frac{1}{\sigma^2}\|z\|^2$$

Second Fine-tuning Strategy: MAD-LM

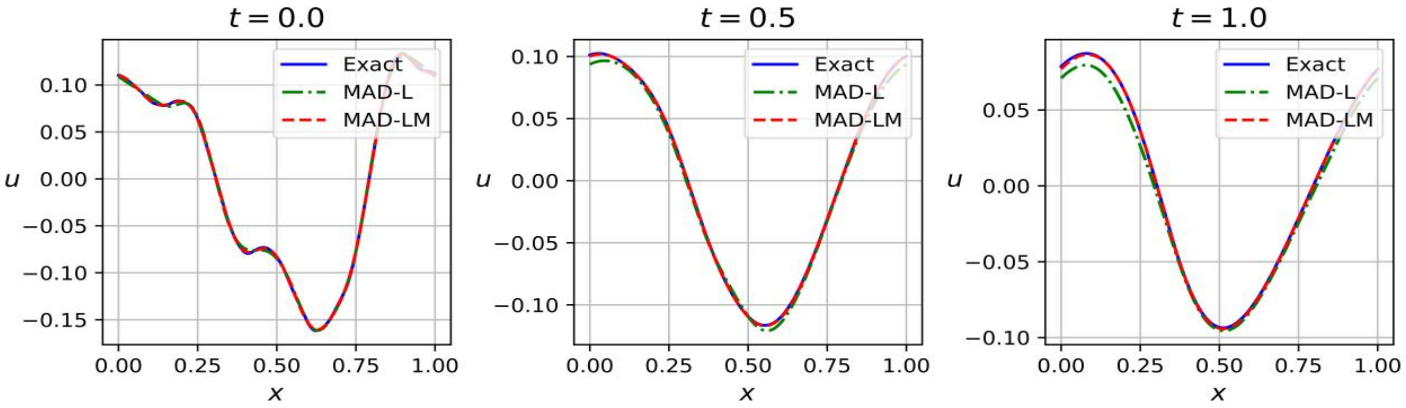


- **Fine-tuning Stage (MAD-LM)** Given a new PDE parameter η_{new}, MAD-LM fine-tunes the model weight θ with the latent vector z simultaneously, and solves the following optimization problem with initial model weight θ^*.

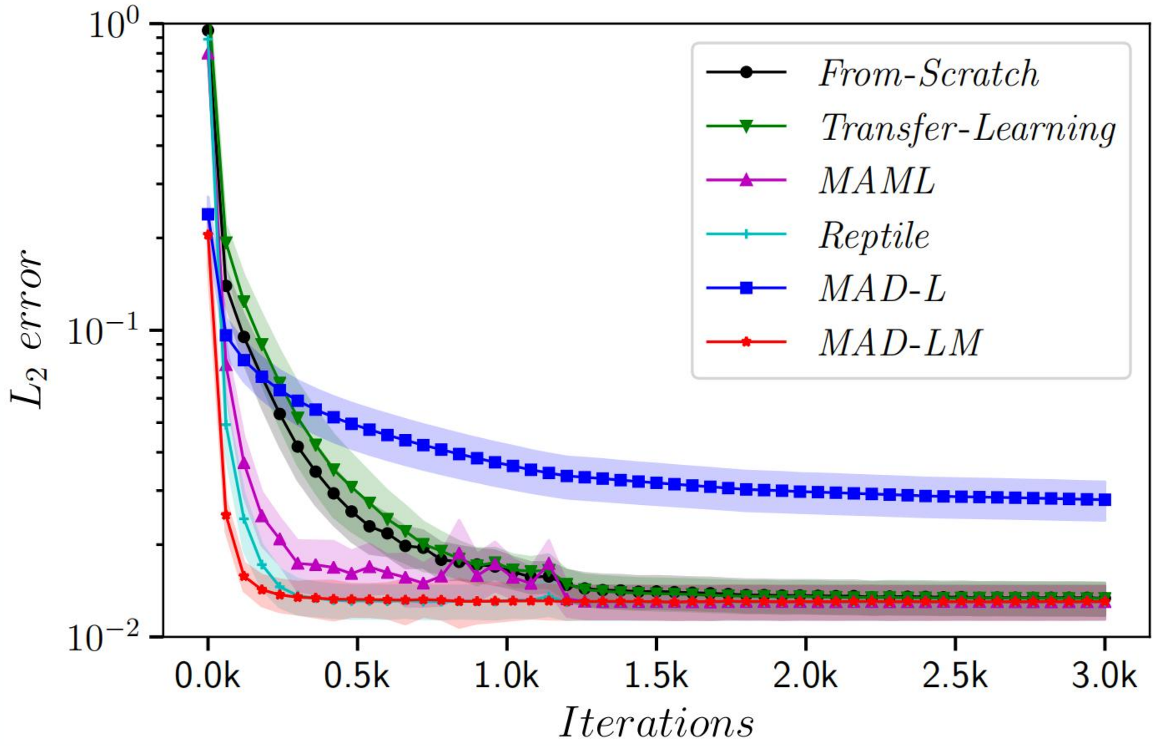
$$(z^*_{new}, \theta^*_{new}) = \arg\min_{z,\theta} \hat{L}^{\eta_{new}}[u_\theta(\cdot, z)] + \frac{1}{\sigma^2}\|z\|^2$$

Burgers' Equation with Variable Initial Conditions

$$\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}=\nu\frac{\partial^2 u}{\partial x^2},\quad x\in(0,1), t\in(0,1],$$
$$u(x,0)=u_0(x),\quad x\in(0,1).$$



Reference solutions vs. model predictions



Method	Meaning
From-Scratch	Train the model from scratch case-by-case.
Transfer-Learning	Randomly select a PDE parameter for pre-training stage and load the pre-trained weight in fine-tuning stage.
MAML	Meta-train the model based on MAML algorithm, and then load the pre-trained weight in meta-testing stage.
Reptile	Similar to MAML, except that the model weight is updated using the Reptile algorithm in meta-training stage.
MAD-L	Only fine-tune the latent vector z.
MAD-LM	Fine-tune the model weight θ and the latent vector z simultaneously.

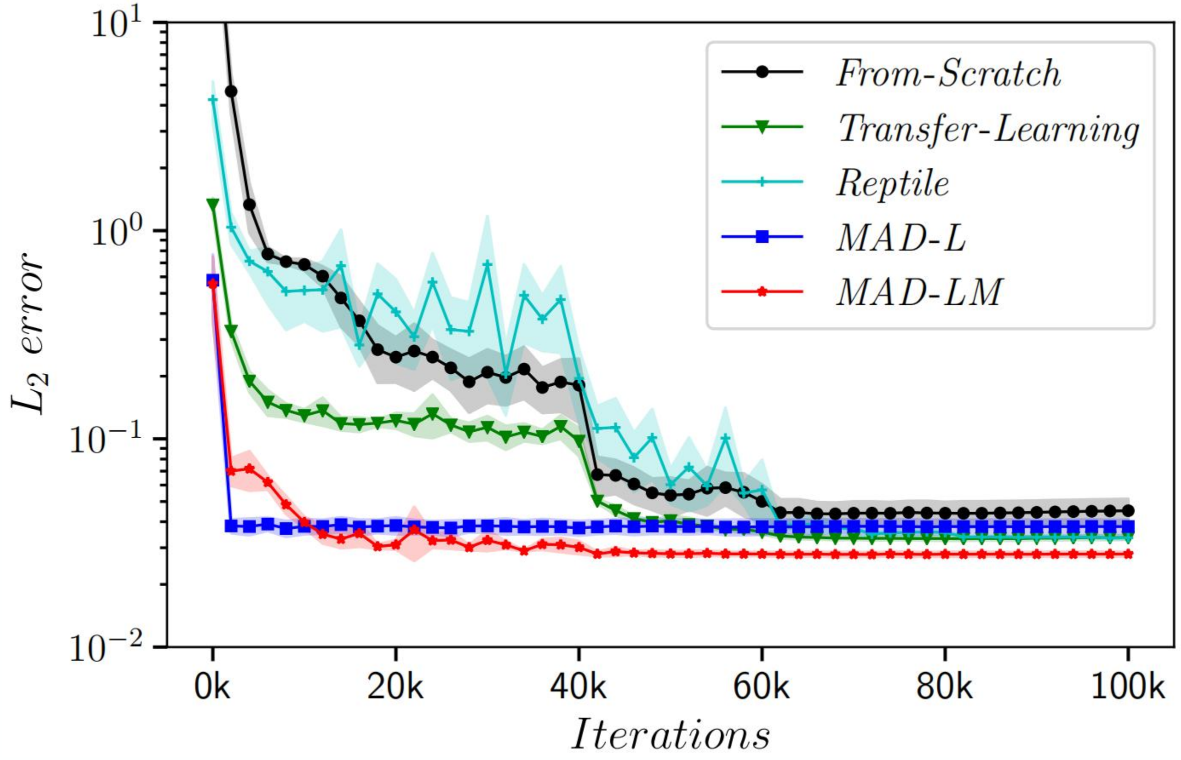
Maxwell's Equations with Variable Equation Coefficients

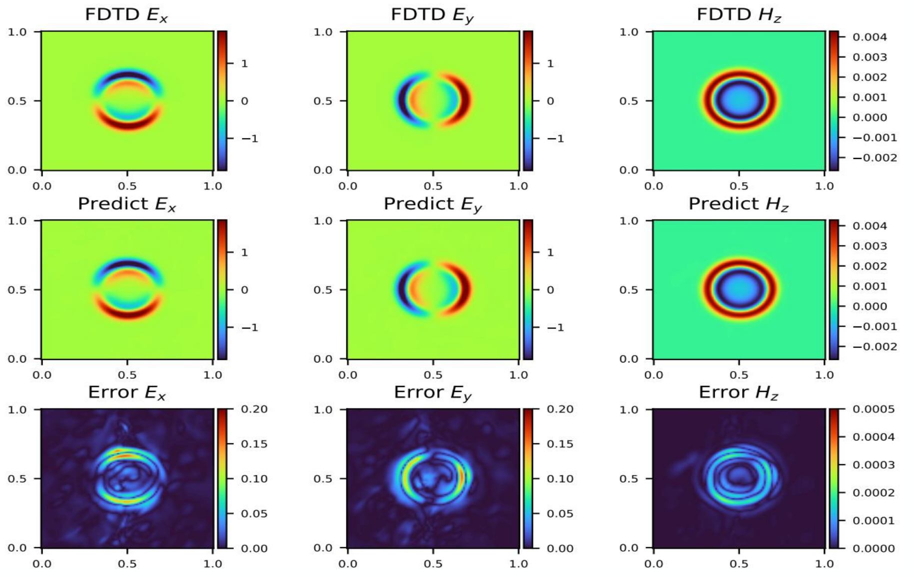
$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_0 \varepsilon_r} \frac{\partial H_z}{\partial y}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_0 \varepsilon_r} \frac{\partial H_z}{\partial x}$$

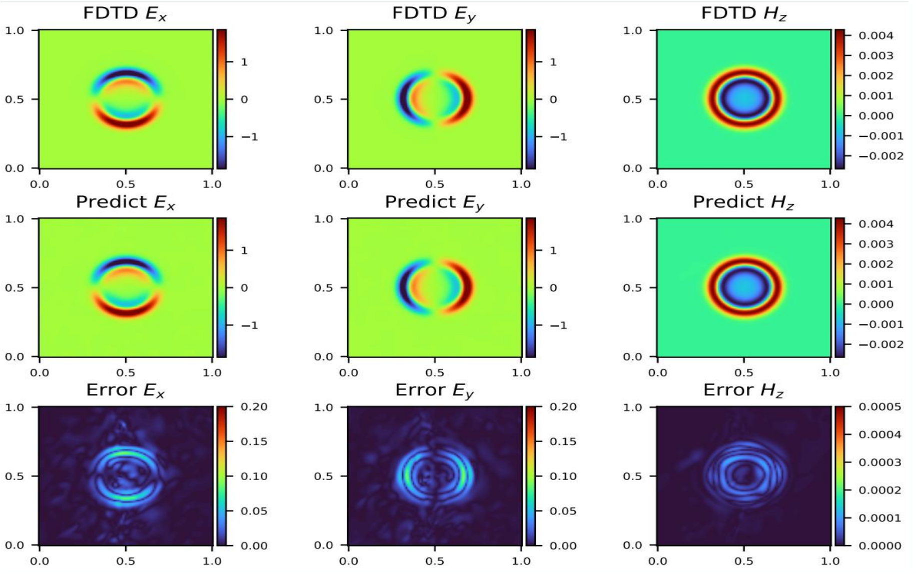
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu_0 \mu_r}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + J\right)$$

$$J = e^{-\left(\frac{t-d}{\tau}\right)^2} \delta(x - x_0)\delta(y - y_0)$$





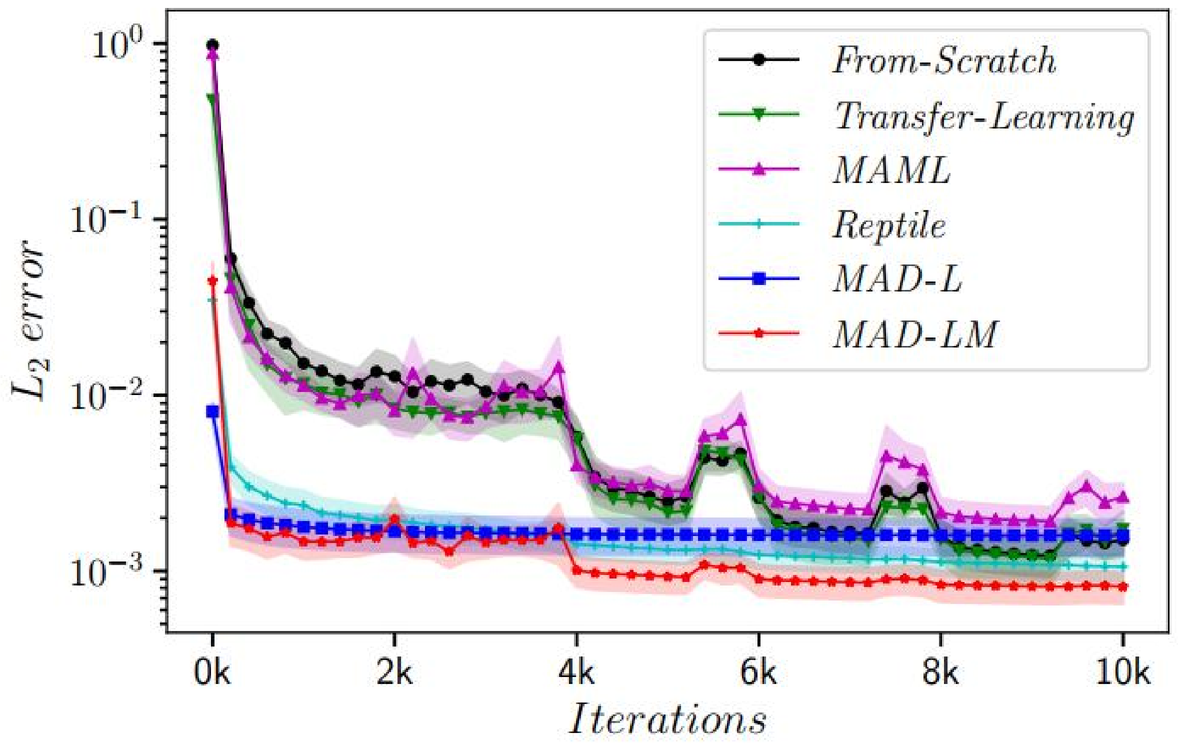
MAD-L

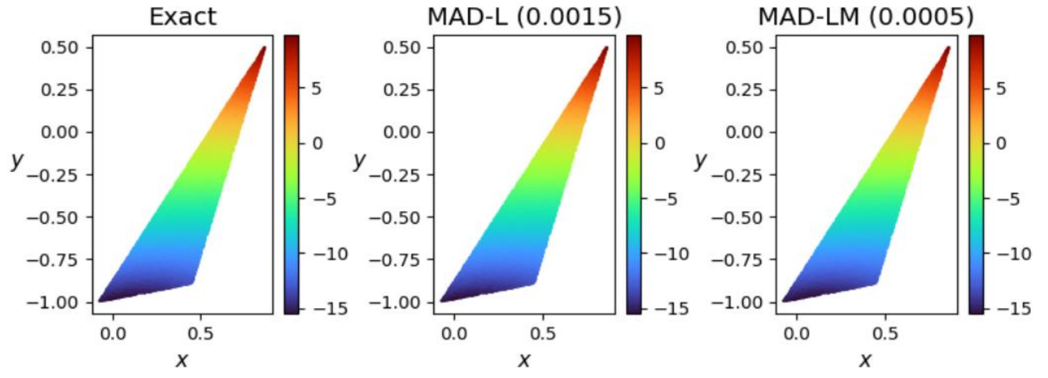


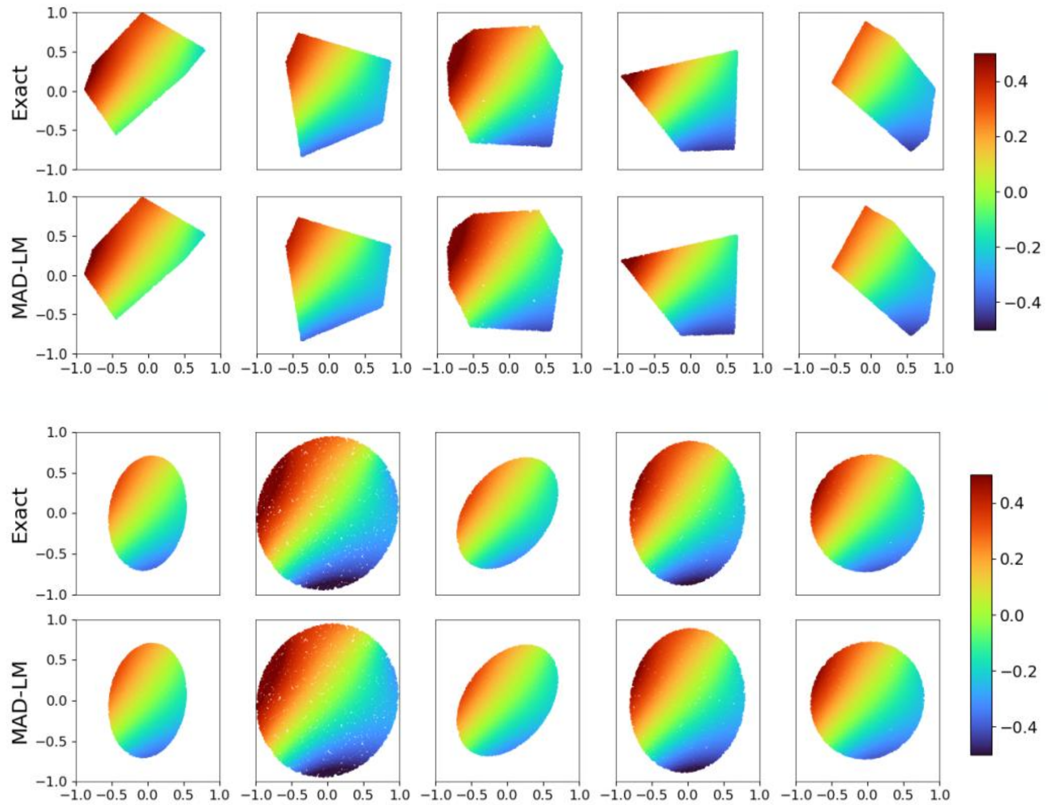
MAD-LM

Laplace’s Equation with Variable Solution Domains and Boundary Conditions

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad (x, y) \in \Omega,$$
$$u(x, y) = g(x, y), \quad (x, y) \in \partial\Omega$$







Thanks！ Q&A

- ☑ Source Code:
  https://gitee.com/mindspore/mindscience/tree/master/MindElec/
- ☑ Our code is implemented by MindSpore.
- ☑ For more questions, please send email to sahx@mail.ustc.edu.cn.

MindSpore