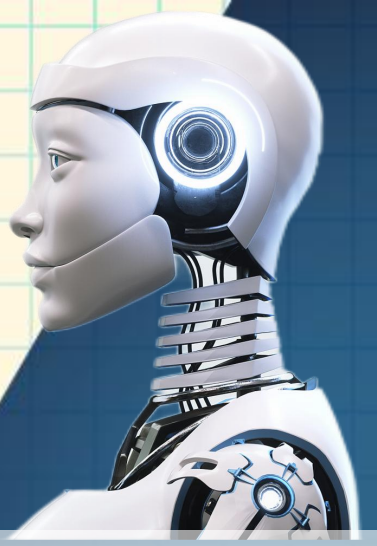


基于物理信息嵌入神经网络的偏微分方程求解方法研究

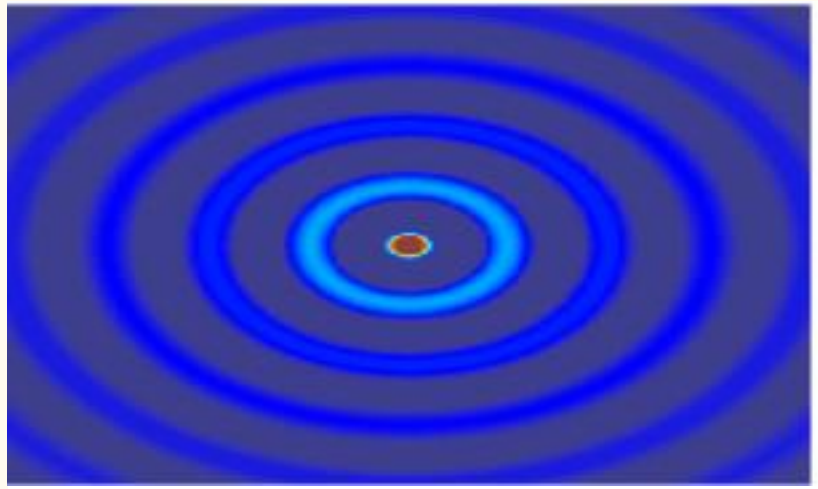
报告人: 黄翔

MindSpore

目录 Contents

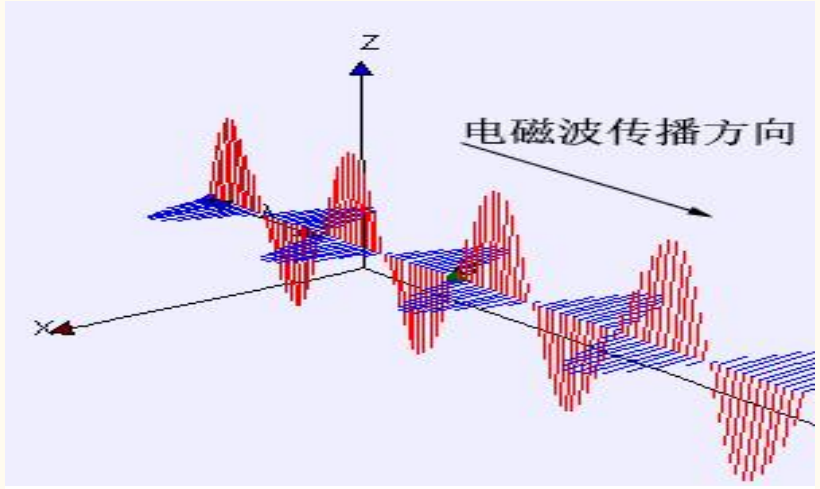
一、研究背景与研究内容

- 偏微分方程 (Partial Differential Equation，PDE) 是描述未知函数及其偏导数之间关系的方程，现实世界中大量的物理现象都可以用偏微分方程来表示。



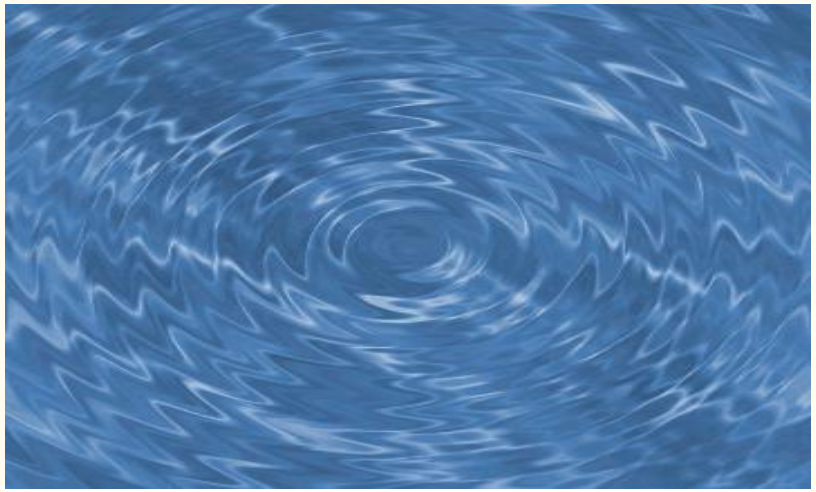
☑ 波动方程

$$u_{tt} - c^2\left(u_{xx} + u_{yy}\right) = f(x, y, t)$$
$$f(x, y, t) = h(t)\delta(x - x_0)\delta(y - y_0)$$



☑ 麦克斯韦方程组

$$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\varepsilon_0}$$
$$\nabla \cdot \boldsymbol{B} = 0$$
$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}$$
$$\nabla \times \boldsymbol{B} = \mu_0(\boldsymbol{J} + \varepsilon_0 \frac{\partial \boldsymbol{E}}{\partial t})$$

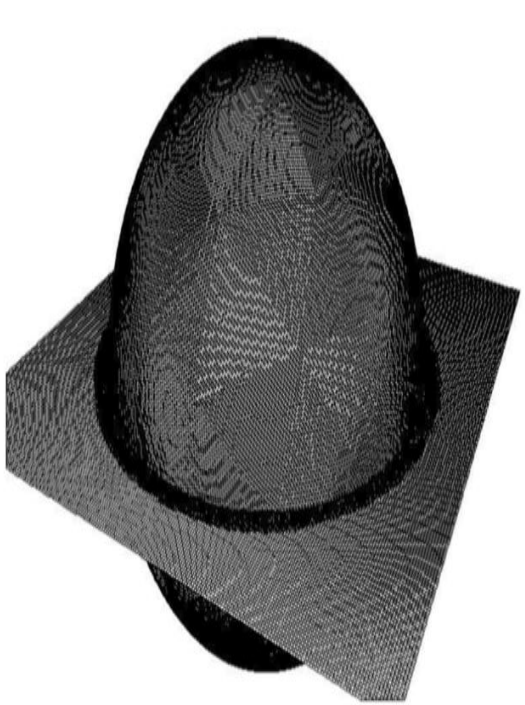
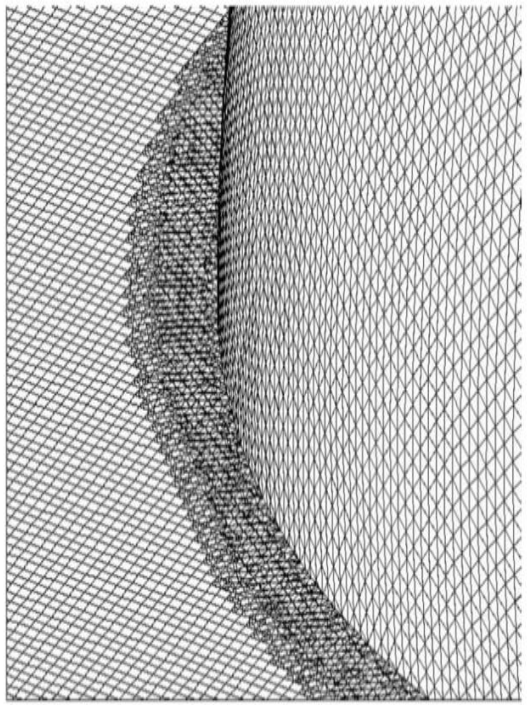


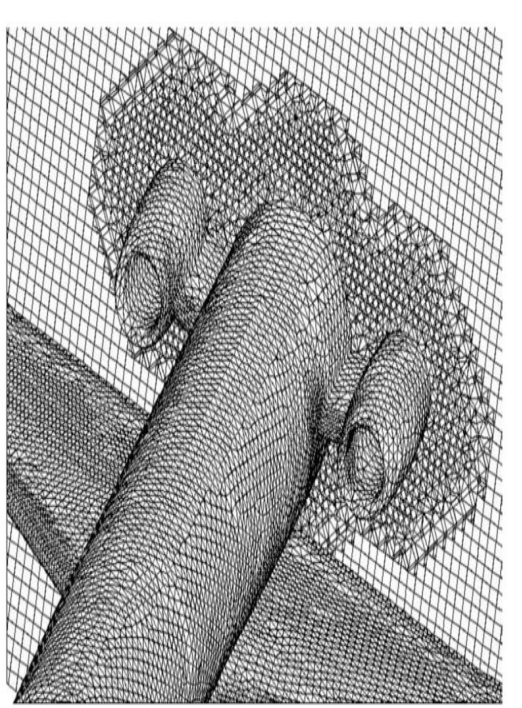
☑ 纳维-斯托克斯方程

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{V} = 0$$
$$\rho\left(\frac{\partial \boldsymbol{V}}{\partial t} + (\boldsymbol{V} \cdot \nabla)\boldsymbol{V}\right) = -\nabla p + \mu\nabla^2\boldsymbol{V} + F$$

一、研究背景与研究内容

- **有代表性的数值求解方法：**
  - ☑ 有限差分方法（Finite Difference Method，FDM）
  - ☑ 有限元方法（Finite Element Method，FEM）



一、研究背景与研究内容

- **有代表性的数值求解方法：**
  - ☑ 有限差分方法（Finite Difference Method，FDM）
  - ☑ 有限元方法（Finite Element Method，FEM）

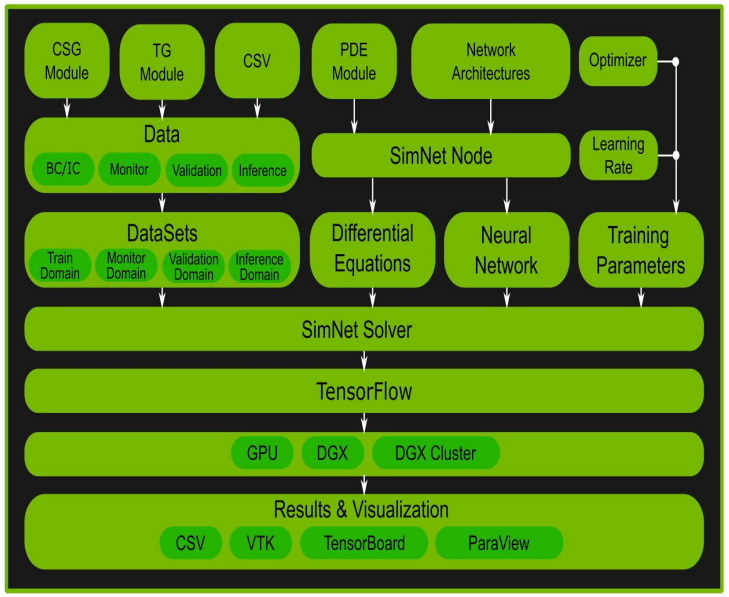
- **传统数值求解方法面临的问题：**
  - ☑ 计算复杂度高：传统数值求解方法的计算复杂度随 PDE 的规模、复杂性和网格精度的提高而急剧增大，特别是计算复杂度随 PDE 的维数呈指数增长，面临维数灾难（Curse of Dimensionality）问题；
  - ☑ 难以求解反问题：传统数值求解方法通常用于求解正问题，即根据已知的输入和参数计算模型的输出。而反问题需要根据已知的输出推断出输入或参数值，由于存在大量的不确定性，传统数值求解方法难以处理这类问题；
  - ☑ 求解参数化 PDE（方程族）的效率很低：求解参数化 PDE 是指求解参数可变的 PDE（将方程的可变参数定义为“PDE参数”）。使用传统数值方法求解参数化 PDE 时，需要针对每个PDE参数对应的方程实例独立求解，耗时非常长。

一、研究背景与研究内容

- **深度学习方法可以求解高维 PDE、反问题、参数化 PDE**
  - ☑ J. Han et al., PNAS 2018. 使用深度神经网络近似解的梯度，并根据离散随机积分和给定的终止条件设计损失函数，成功求解了高维 Black–Scholes 方程、高维 Hamilton–Jacobi–Bellman 方程和 Allen–Cahn 方程；（高维 PDE）
  - ☑ Z. Long et al., ICML 2018. 可以从带噪声的动力学数据中学习出控制方程的具体形式；(反问题)
  - ☑ M. Raissi et al., JCP, 378:686-707, 2019. 能够在标签数据稀缺的情况下，仍能够推断出物理系统的未知参数；(反问题)
  - ☑ Z. Li et al., ICLR 2021. 可以求解初始条件可变的 Burgers 方程和纳维-斯托克斯方程、扩散系数可变的 Darcy 流问题；(参数化 PDE)
- **深度学习方法可以提升方程的求解效率**
  - ☑ Z. Li et al., ICLR 2021. 在 256 × 256 的网格上求解纳维-斯托克斯方程时，相较于传统数值求解方法（伪谱法），深度学习方法可以获得 440 倍的加速；
  - ☑ R. Lam et al., arXiv:2212.12794. 只需单台 Cloud TPU v4 设备，即可在 60 秒内生成 10 天内的天气预报（35GB 数据），并且其预测精度也高于传统数值天气预报模型。

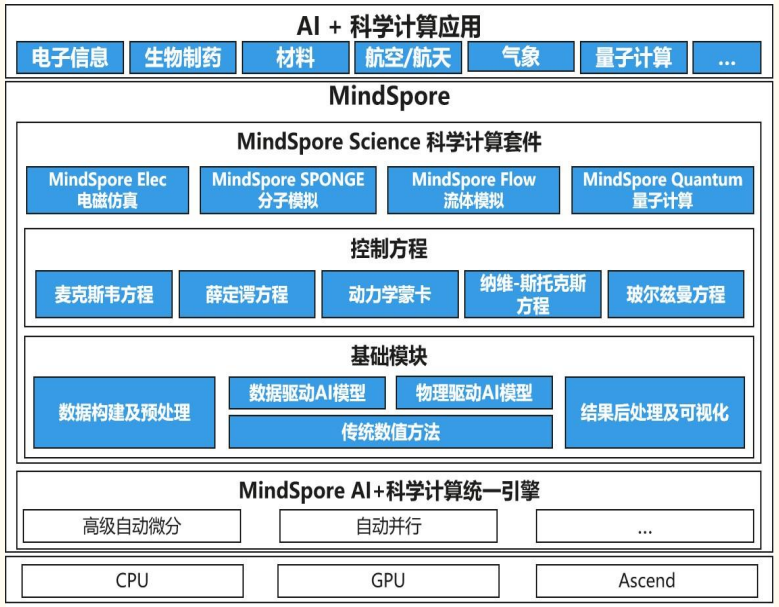
一、研究背景与研究内容

- 各大科技公司投入了大量的资源研发基于深度学习求解PDE的工具包。



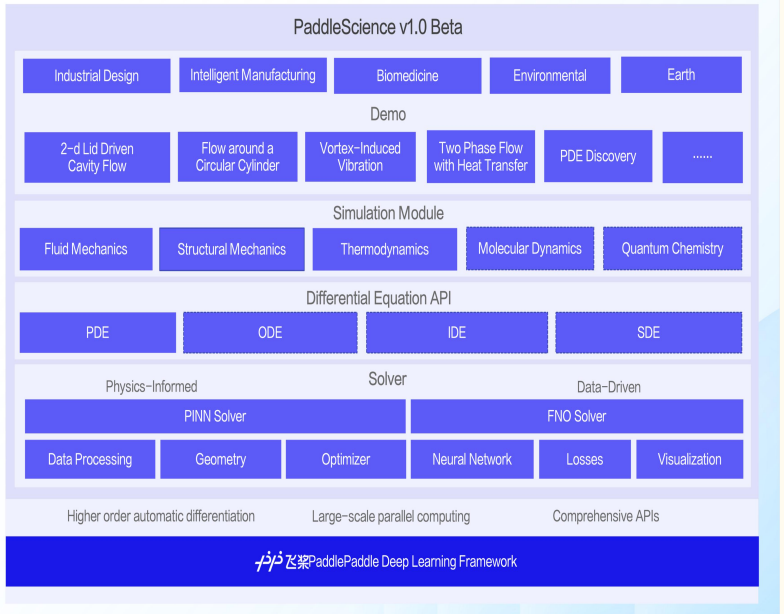
☑ SimNet (后改名为Modulus)

NVIDIA



☑ MindScience

HUAWEI

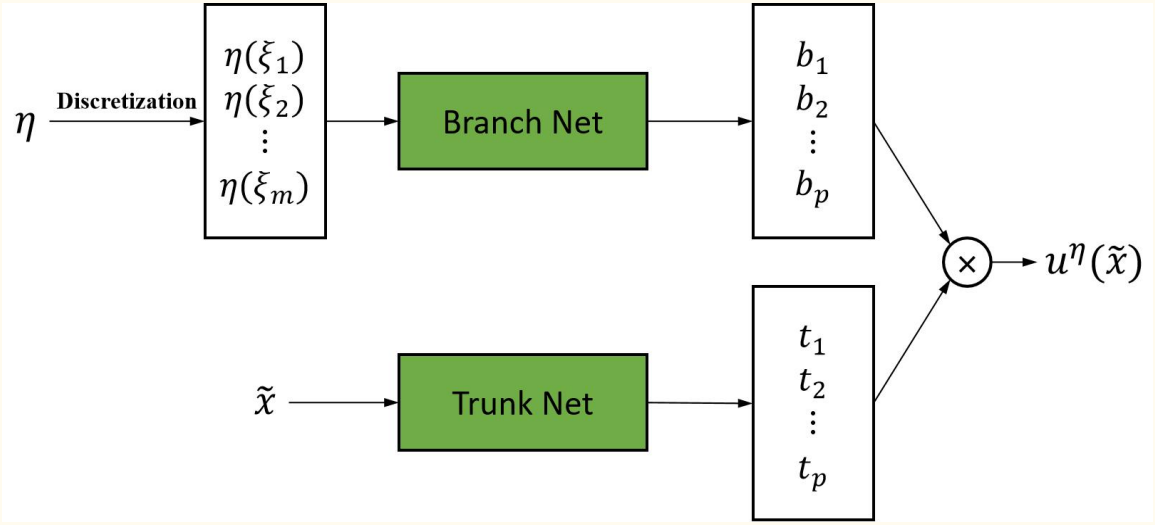


☑ PaddleScience

Baidu 百度

一、研究背景与研究内容

- **数据驱动方法：**
  - ☑ 定义：将神经网络作为算子逼近器来求解参数化 PDE，利用大量标签数据来学习 PDE 参数到方程解的直接映射。
  - ☑ 代表性工作：
    - ➢ 深度算子网络 (Deep Operator Network, DeepONet) L. Lu et al., NAT MACH INTELL, 3(3):218-229, 2021.
    - ➢ 傅里叶神经算子 (Fourier Neural Operator, FNO) Z. Li et al., ICLR 2021.



DeepONet的网络架构

符号	含义
η	PDE参数
$[\eta(\xi_1), \eta(\xi_2), \cdots, \eta(\xi_m)]$	离散化后的PDE参数
$\tilde{x}$	采样点的坐标
$\otimes$	向量内积
$u^\eta(\tilde{x})$	当PDE参数为η时，模型对坐标位置为$\tilde{x}$的方程解的预测

一、研究背景与研究内容

- **数据驱动方法：**
  - ☑ 定义：将神经网络作为算子逼近器来求解参数化 PDE，利用大量标签数据来学习 PDE 参数到方程解的直接映射。
  - ☑ 代表性工作：
    - ➢ 深度算子网络 (Deep Operator Network, DeepONet) L. Lu et al., NAT MACH INTELL, 3(3):218-229, 2021.
    - ➢ 傅里叶神经算子 (Fourier Neural Operator, FNO) Z. Li et al., ICLR 2021.
  - ☑ 存在的问题：数据驱动方法训练出来的模型本质上是黑盒模型，由于黑盒模型的归纳偏置主要来自于数据，当可用的标签数据较少或包含大量噪声时，训练得到的模型精度不高、泛化能力差，也缺乏可解释性。

一、研究背景与研究内容

- **数据驱动方法：**

☑ 定义：将神经网络作为算子逼近器来求解参数化 PDE，利用大量标签数据来学习 PDE 参数到

☑

8-229, 2021.

☑ 置主要来自

化能力差，

在标签数据较少甚至没有标签数据时，如何充分利用物理方程的信息来高精度地求解 PDE？

一、研究背景与研究内容

- **物理信息嵌入的神经网络：**

☑ 物理信息隐式嵌入（物理驱动方法）

- 定义：将物理方程的信息以损失函数的形式隐式地嵌入到神经网络中。
- 代表性工作：
  - ★ 物理信息神经网络 (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.

一、研究背景与研究内容

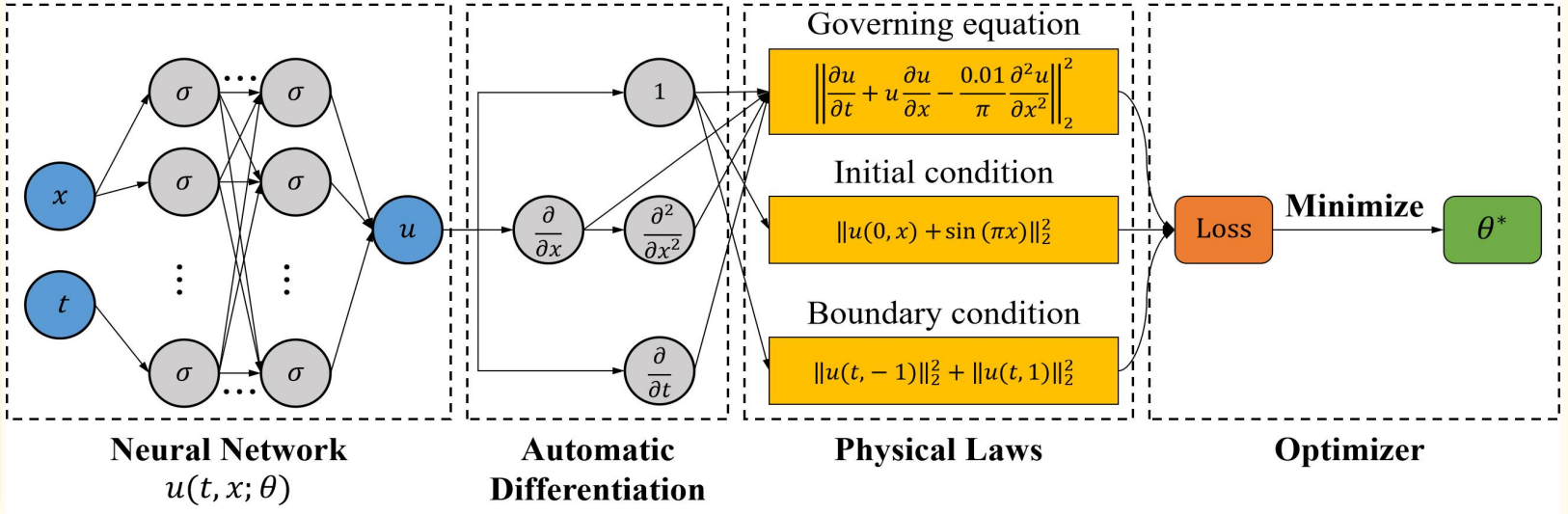
- **物理信息神经网络 PINNs**

求解域为 $[0,1] \times [-1,1]$ 的 Burgers 方程：

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2} = 0, \text{（控制方程）}$$

$$u(0,x) = -\sin(\pi x), \text{（初始条件）}$$

$$u(t,-1) = u(t,1) = 0. \text{（边界条件）}$$



PINNs方法工作原理：

1. 在整个求解区域 $[0,1] \times [-1,1]$ 里面随机采样很多的样本点，并将每个样本点的坐标 (x,t) 作为全连接神经网络的输入；
2. 神经网络的输出是方程的状态变量 u，u 在所有采样点上的值就是方程的解；
3. 可以利用深度学习框架中的自动微分来求得方程中出现的各个偏微分项 $\{\frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\}$；
4. 将这些偏微分项组成控制方程的残差、初始条件的残差、边界条件的残差，将这三项残差对应的 loss 项求和构成总的损失函数；
5. 通过反复在求解区域上进行随机采样和迭代梯度下降算法来最小化总的损失函数，神经网络的可训练参数 θ 会不断更新，并且神经网络的输出 $u(t,x;\theta)$ 会逐渐逼近方程的真解；
6. 训练好了神经网络后，PINNs 方法可以对时空求解域中的任何一个坐标点进行方程解的估计。

一、研究背景与研究内容

- **物理信息嵌入的神经网络：**

☑ 物理信息隐式嵌入（物理驱动方法）

- 定义：将物理方程的信息以损失函数的形式隐式地嵌入到神经网络中。
- 代表性工作：
  - ★ 物理信息神经网络 (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.
- 存在的问题：
  - ★ PINNs 方法目前难以求解带点源的 PDE；
  - ★ 物理驱动方法求解参数化PDE的效率很低。

一、研究背景与研究内容

- **物理信息嵌入的神经网络：**

☑ 物理信息隐式嵌入（物理驱动方法）

- 定义：将物理方程的信息以损失函数的形式隐式地嵌入到神经网络中。
- 代表性工作：……
- 存在的问题：……

☑ 物理信息显式嵌入（数据机理融合方法）

- 定义：将物理方程的信息显式地嵌入到神经网络架构中，并利用少量标签数据即可训练好神经网络。
- 代表性工作：
  - ★ CFD-GCN F. Belbute-Peres et al., ICML 2020.
  - ★ TF-Net R. Wang et al., KDD 2020.
  - ★ JAX-CFD D. Kochkov et al., PNAS, 118(21):e2101784118, 2021.
  - ★ PDE-Net Z. Long et al., ICML 2018.
- 存在的问题：
  - ★ CFD-GCN、TF-Net、JAX-CFD都是针对特定问题而设计的，通用性差；
  - ★ PDE-Net处理复杂非线性项及未知项能力弱。

一、研究背景与研究内容

基于物理信息嵌入神经网络的偏微分方程求解方法研究

	物理信息隐式嵌入 (物理驱动方法)	物理信息显式嵌入 (数据机理融合方法)
存在的问题	PINNs方法难以求解带点源的PDE 物理驱动方法求解参数化PDE效率低	数据机理融合方法通用性差、处理复杂非线性项及未知项能力弱
研究内容	改进PINNs方法求解带点源PDE • 利用对称单峰概率密度函数近似狄拉克函数，消除点源奇异性问题 • 提出下界受限的不确定性加权方法，平衡多个损失函数项 • 提出结合 Sine 函数的多尺度神经网络架构，增强神经网络的表达能力 基于元学习思想高效求解参数化PDE • 提出预训练和微调相结合的参数化PDE求解方法，减小模型重训练的开销 • 从流形学习的角度解释方法的工作原理	预测时空动力学系统演化过程的通用数据机理融合方法 • 结合有限差分方法和黑盒模型，将物理信息嵌入神经网络架构 • 提出新的可训练差分算子，提高长时序列预测的数值稳定性 • 改进PDE-Net，增加其处理复杂非线性项及未知项的能力

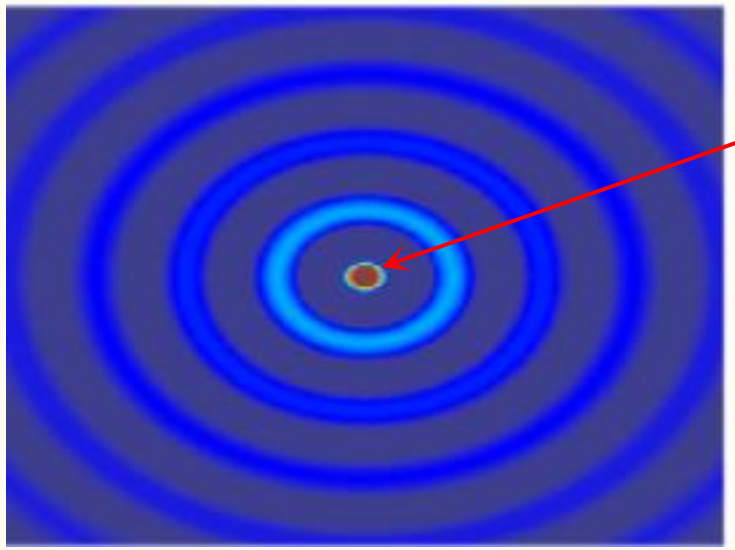
目录 Contents

二、改进 PINNs 方法求解带点源的 PDE

- **带点源的声波方程**

$$u_{tt} - c^2\left(u_{xx} + u_{yy}\right) = f(x, y, t)$$

$$f(x, y, t) = h(t)\delta(x - x_0)\delta(y - y_0)$$



点源

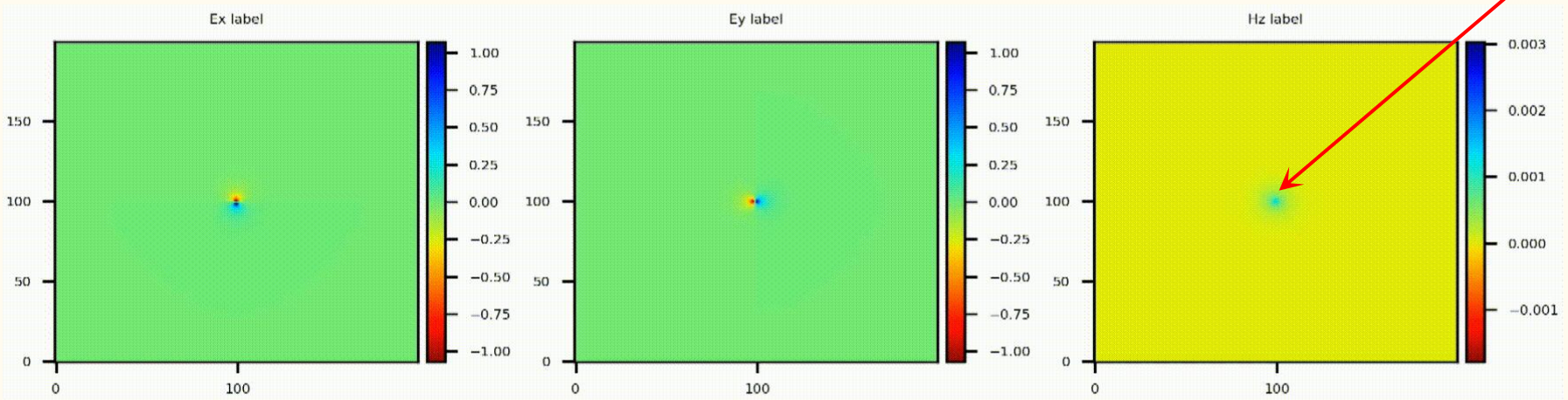
二、改进 PINNs 方法求解带点源的 PDE

- **带点源的麦克斯韦方程组**

$$\frac{\partial E_x}{\partial t}=\frac{1}{\varepsilon}\frac{\partial H_z}{\partial y},\quad \frac{\partial E_y}{\partial t}=-\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x},\quad \frac{\partial H_z}{\partial t}=-\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x}-\frac{\partial E_x}{\partial y}+J\right).$$

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

点源



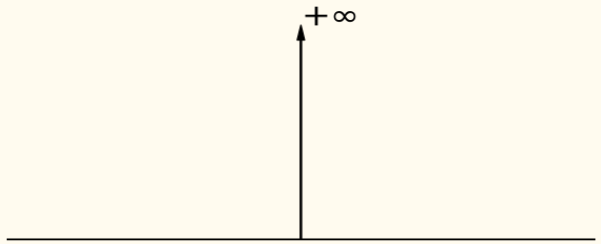
二、改进 PINNs 方法求解带点源的 PDE

- **点源在数学上可以用狄拉克 δ 函数来表示。**

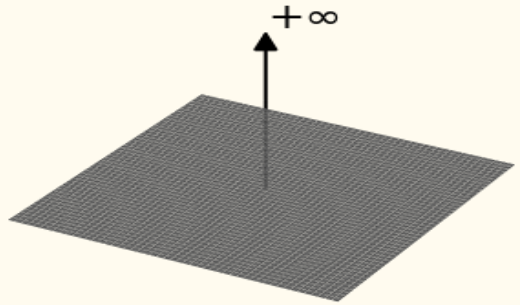
狄拉克 δ 函数的定义：

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases},$$

$$\int_{-\infty}^{+\infty} \delta(x)dx = 1.$$



一维狄拉克 δ 函数



二维狄拉克 δ 函数

二、改进 PINNs 方法求解带点源的 PDE

狄拉克 δ 函数的

$\delta($

因为狄拉克 δ 函数在点源位置处会带来明显的奇异性，所以 PINNs 方法难以求解带点源的 PDE。

二维狄拉克 δ 函数

二、改进 PINNs 方法求解带点源的 PDE

- **求解带点源的 PDE 的深度学习方法的研究现状**

相关工作	论文中针对的 PDE	对方程没有限制	不依赖标签数据
DRM W. E and B. Yu, CMS, 6(1), 1-12, 2018.	泊松方程	✗	✓
NVIDIA SimNet O. Hennigh et al., ICCS 2021.	泊松方程	✗	✓
P. Zhang et al., WMCS, 6:35-40, 2020.	麦克斯韦方程组	✗	✗
B. Moseley et al., arXiv:2006.11894.	波动方程	✗	✗
Yared W. Bekele, arXiv:2010.15426.	Barry&Mercer 问题	✓	✗

注：DRM 论文中并没有求解带点源的泊松方程的实验，我们利用 DRM 的思想做了相应的实验。

二、改进 PINNs 方法求解带点源的 PDE

- **利用连续对称单峰概率密度函数近似狄拉克 δ 函数**

$$\text{当}\alpha \to 0, \eta(x) = \alpha^{-1}\eta\left(\frac{x}{\alpha}\right) \xrightarrow{\text{近似}} \delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}, \quad \int_{-\infty}^{+\infty} \delta(x)dx = 1.$$

☑ 高斯分布：

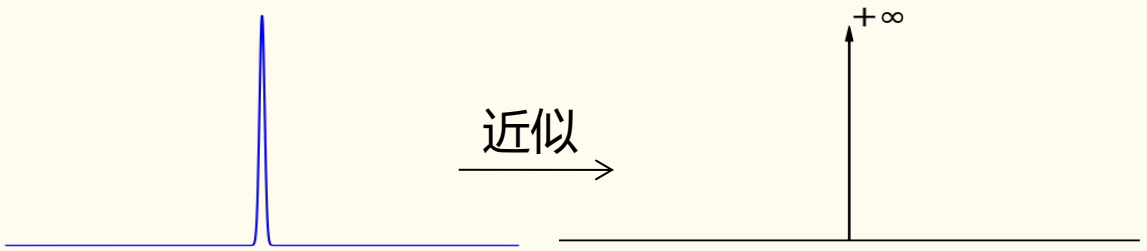
$$\eta_\alpha(x) = \frac{1}{\sqrt{2\pi}\alpha} e^{-\frac{x^2}{2\alpha^2}}$$

☑ 柯西分布：

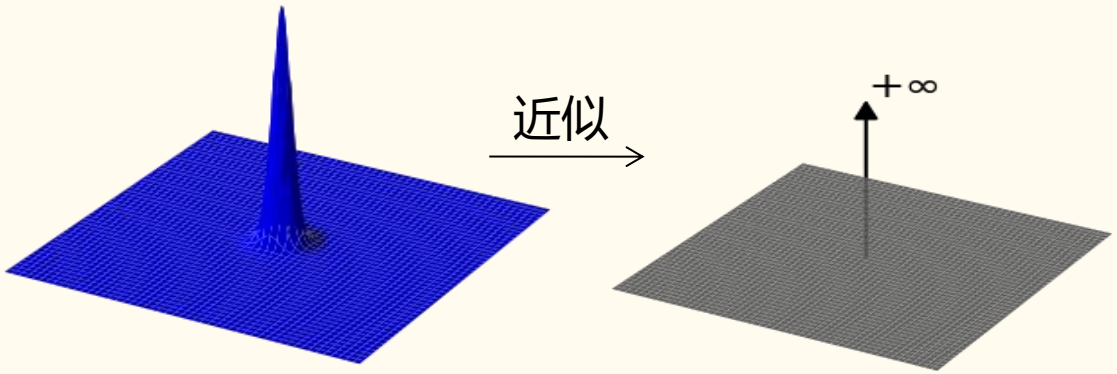
$$\eta_\alpha(x) = \frac{1}{\pi}\frac{\alpha}{x^2 + \alpha^2}$$

☑ 拉普拉斯分布：

$$\eta_\alpha(x) = \frac{1}{2\alpha} e^{-\frac{|x|}{\alpha}}$$



一维高斯分布　　一维狄拉克 δ 函数



二维高斯分布　　二维狄拉克 δ 函数

二、改进 PINNs 方法求解带点源的 PDE

- **区域分解**

☑ **带点源的 PDE 的一般形式：**

控制方程： $\mathcal{N}\big(u(\mathbf{x},t)\big) = f(\mathbf{x},t), \quad (\mathbf{x},t) \in \Omega \times [0,T]$

边界条件： $u(\mathbf{x},t) = g(\mathbf{x},t), \quad (\mathbf{x},t) \in \partial\Omega \times [0,T]$

初始条件： $u(\mathbf{x},0) = h(\mathbf{x}), \quad \mathbf{x} \in \Omega$

其中 $f(\mathbf{x},t)$ 包含 $\delta(\mathbf{x}-\mathbf{x}_0)$， $\mathbf{x}_0$ 是点源位置的坐标。

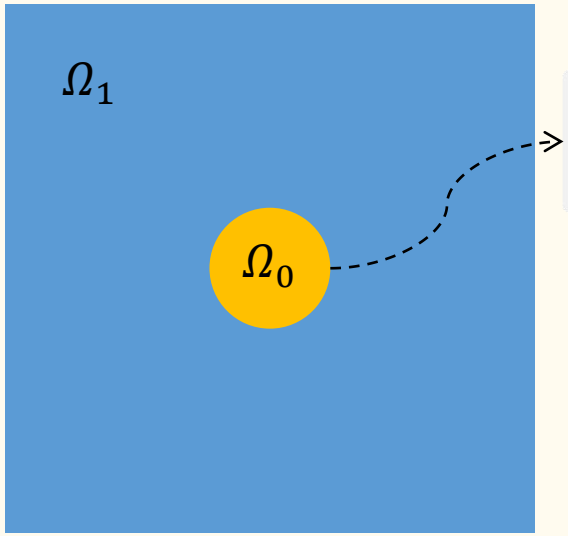
☑ **物理信息损失函数 (Physics-informed Loss)：**

$$L_r(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathcal{N}\big(u(\mathbf{x_i},t_i;\theta)\big) - f(\mathbf{x_i},t_i)\right|^2,$$

$$L_{bc}(\theta) = \frac{1}{N_{bc}}\sum_{i=1}^{N_{bc}}\left|u(\mathbf{x_i},t_i;\theta) - g(\mathbf{x_i},t_i)\right|^2,$$

$$L_{ic}(\theta) = \frac{1}{N_{ic}}\sum_{i=1}^{N_{bc}}\left|u(\mathbf{x_i},0;\theta) - h(\mathbf{x_i})\right|^2,$$

$$L_{total}(\theta) = L_r(\theta) + \lambda_{ic}L_{ic}(\theta) + \lambda_{bc}L_{bc}(\theta).$$



点源所在的子区域

$\Omega = \Omega_0 \cup \Omega_1$

$\Omega_0 \cap \Omega_1 = \emptyset$

☑ **将整个求解域 Ω 分解为 Ω_0 和 Ω_1：**

$$L_r(\theta) = \lambda_{r,0}L_{r,0}(\theta) + \lambda_{r,1}L_{r,1}(\theta),$$

$$L_{total}(\theta) = \lambda_{r,0}L_{r,0}(\theta) + \lambda_{r,1}L_{r,1}(\theta) + \lambda_{ic}L_{ic}(\theta) + \lambda_{bc}L_{bc}(\theta).$$

$$L_{r,0}(\theta) \gg L_{r,1}(\theta), L_{ic}(\theta), L_{bc}(\theta)$$

二、改进 PINNs 方法求解带点源的 PDE

- 区域分解

☑ 带点源的 PDE 的一般形式：

控制方程：

边界条件：

初始条件：

其中 $f(\mathbf{x},t)$ 包

☑ 物理信息损失函

$$L_r(\theta)=\frac{1}{N_r}\sum_{i=1}^{N_r}$$

$$L_{bc}(\theta)=\frac{1}{N_{bc}}\sum_{i=1}^{N_{bc}}|u(\mathbf{x_i},t_i;\theta)-g(\mathbf{x_i},t_i)|^2,$$

$$L_{ic}(\theta)=\frac{1}{N_{ic}}\sum_{i=1}^{N_{bc}}|u(\mathbf{x_i},0;\theta)-h(\mathbf{x_i})|^2,$$

$$L_{total}(\theta)=L_r(\theta)+\lambda_{ic}L_{ic}(\theta)+\lambda_{bc}L_{bc}(\theta).$$

Ω_1

点源所在的子区域

Ω_1

$= \emptyset$

$L_{r,0}$会比其他loss项大很多个数量级，如何为不同的loss项分配不同的权重系数λ？

$$L_r(\theta)=\lambda_{r,0}L_{r,0}(\theta)+\lambda_{r,1}L_{r,1}(\theta),$$

$$L_{total}(\theta)=\lambda_{r,0}L_{r,0}(\theta)+\lambda_{r,1}L_{r,1}(\theta)$$
$$\lambda_{ic}L_{ic}(\theta)+\lambda_{bc}L_{bc}(\theta).$$

$$L_{r,0}(\theta)\gg L_{r,1}(\theta),L_{ic}(\theta),L_{bc}(\theta)$$

二、改进 PINNs 方法求解带点源的 PDE

- **下界受限的不确定性加权方法**

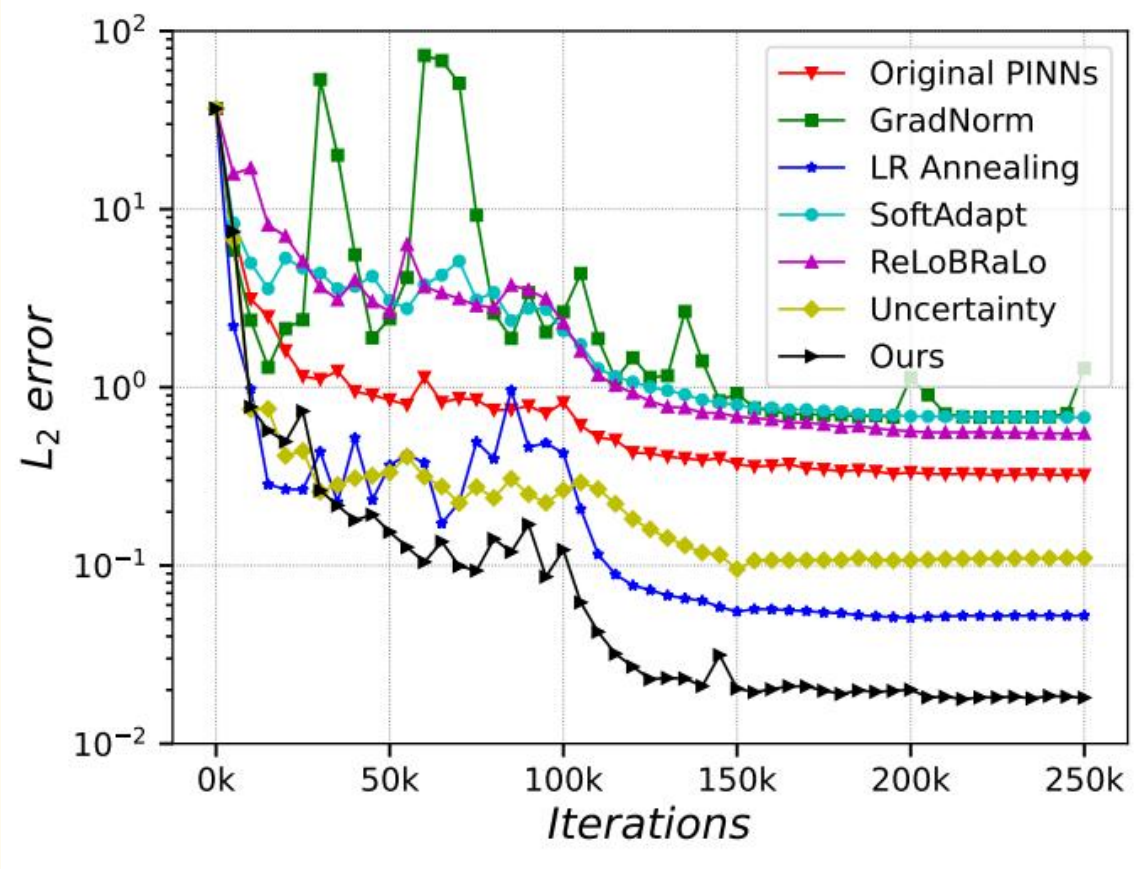
w_i 表示第 i 个loss项的不确定性，是可训练的参数

加权方法	加权公式
手动加权 (Original PINNs)	$L_{total}(\theta) = \sum_{i=1}^{m} \lambda_i L_i(\theta)$
不确定性加权 (Uncertainty A. Kendall et al., CVPR 2017.)	$L_{total}(\theta; \boldsymbol{w}) = \sum_{i=1}^{m} \frac{1}{2w_i^2} L_i(\theta) + log\, w_i$
下界受限的不确定性加权 (Ours)	$L_{total}(\theta; \boldsymbol{w}) = \sum_{i=1}^{m} \frac{1}{2\left(\epsilon^2 + w_i^2\right)} L_i(\theta) + log\left(\epsilon^2 + w_i^2\right)$

不确定性的下界

二、改进 PINNs 方法求解带点源的 PDE

- **下界受限的不确定性加权方法**

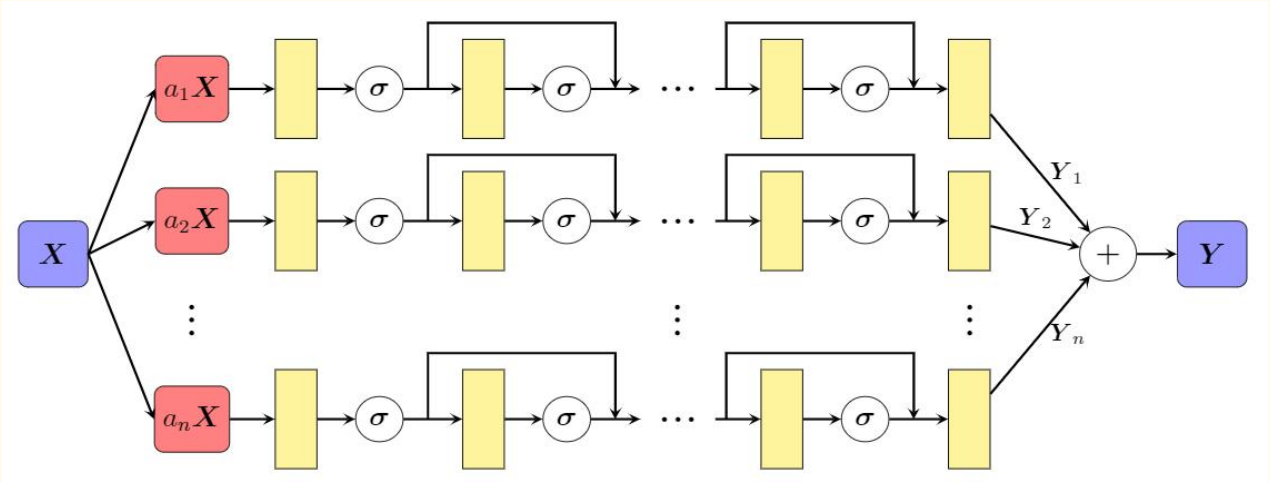


Barry&Mercer 问题：不同的损失函数自适应加权方法的对比。

注：横坐标表示训练时的迭代数目，纵坐标为相对的 L_2 error。

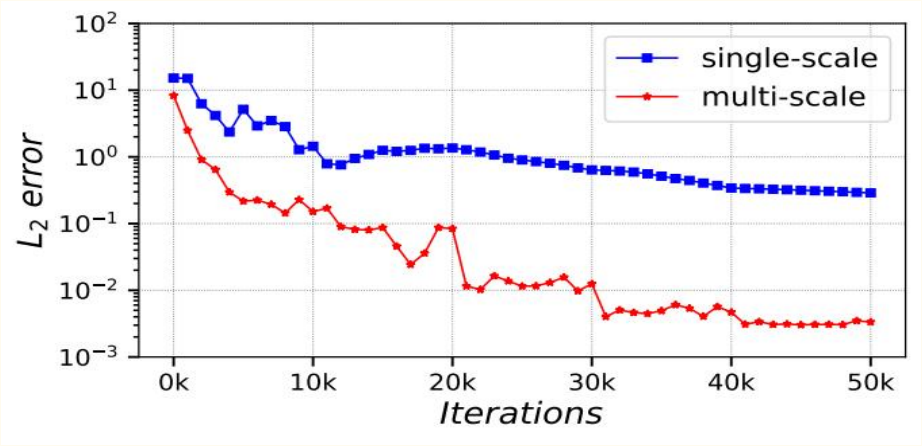
二、改进 PINNs 方法求解带点源的 PDE

- **多尺度神经网络 + Sine 周期性激活函数**

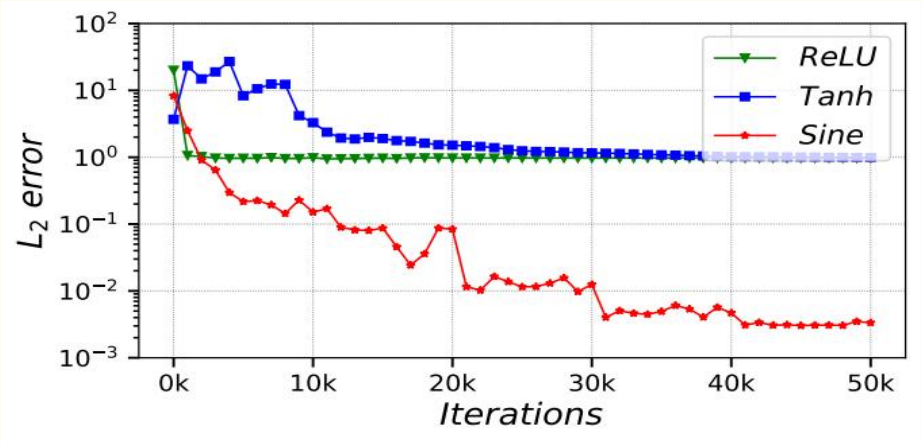


MS-SIREN 网络架构：由 n 个子网构成，不同的子网对应不同的缩放系数。

符号	含义
X	输入
Y	输出
$\{a_1, \cdots, a_n\}$	缩放系数
σ	Sine激活函数
黄色矩形	线性变换层



单尺度 VS. 多尺度



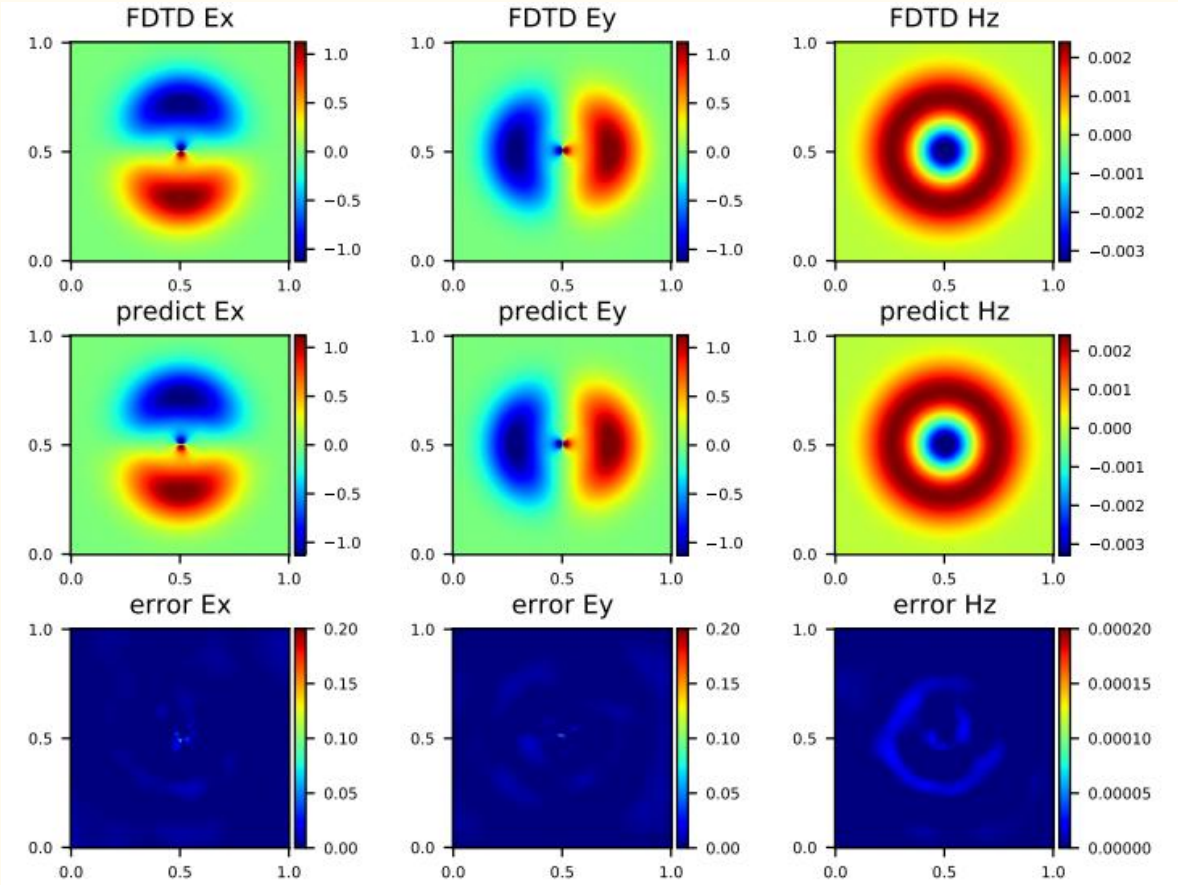
激活函数

二、改进 PINNs 方法求解带点源的 PDE

- **数值实验 —— 带点源的麦克斯韦方程组**

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y},$$
$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x},$$
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + J\right).$$
$$J = e^{-\left(\frac{t-d}{\tau}\right)^2}\,\delta(x-x_0)\delta(y-y_0)$$

Network Architectures			L_2 error			
# subnets	# layers	# neurons	E_x	E_y	H_z	mean
1	7	256	0.192	0.183	0.433	0.269
1	9	256	0.147	0.146	0.070	0.121
2	7	128	0.079	0.074	0.058	0.072
2	9	128	0.054	0.053	0.019	0.027
4	7	64	0.021	0.022	0.001	0.018
4	9	64	0.025	0.022	0.017	0.021



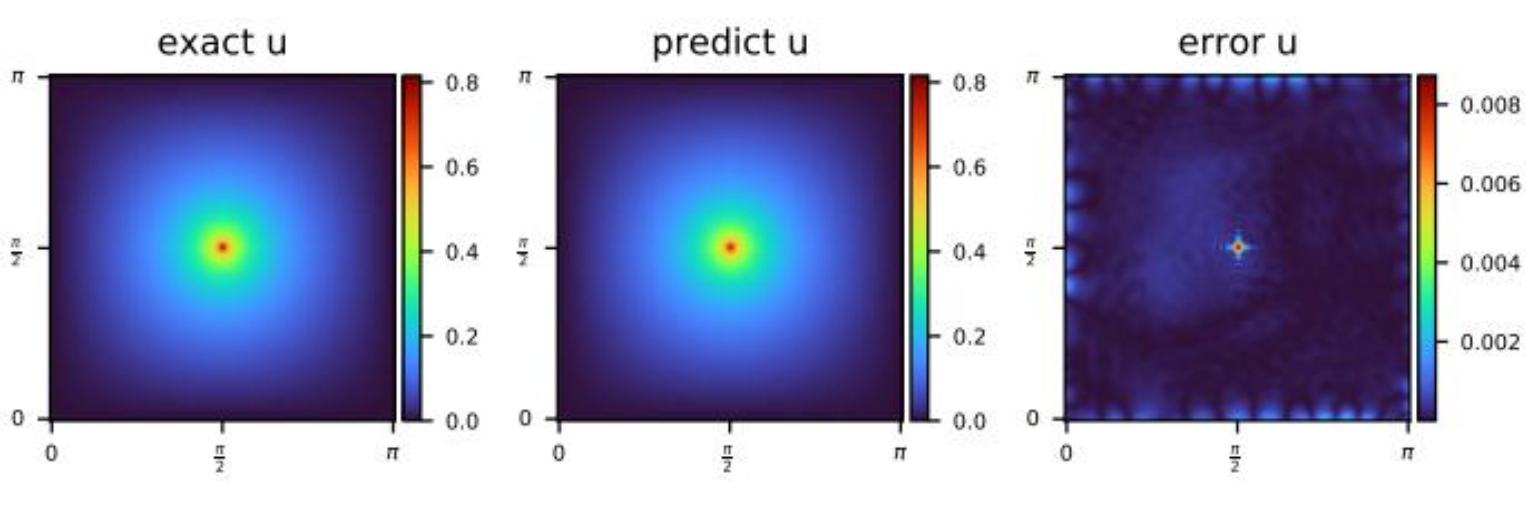
传统数值求解方法FDTD VS. 深度学习方法

二、改进 PINNs 方法求解带点源的 PDE

- **数值实验 —— 带点源的泊松方程**

$$-\Delta u = \delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x}\in\Omega$$
$$u = 0, \quad \mathbf{x}\in\partial\Omega.$$

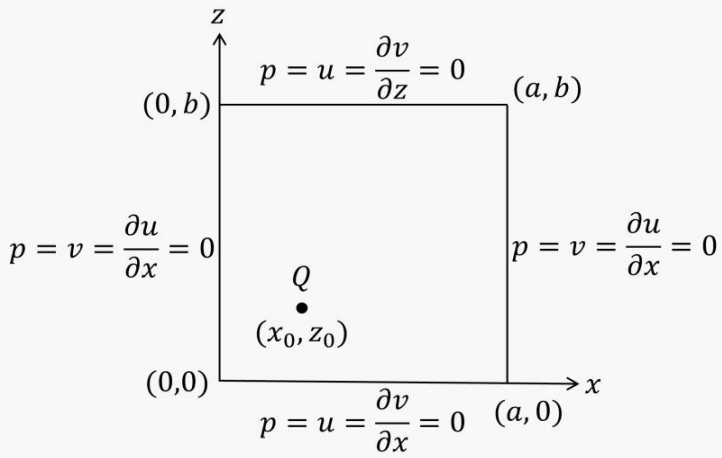
Network Architectures			L_2 error
# subnets	# layers	# neurons	
1	5	256	0.289
1	7	256	1.081
2	5	128	0.003
2	7	128	0.006
4	5	64	0.003
4	7	64	0.002



解析解 VS. 深度学习方法

二、改进 PINNs 方法求解带点源的 PDE

- **数值实验 —— 带点源的 Barry&Mercer 问题**

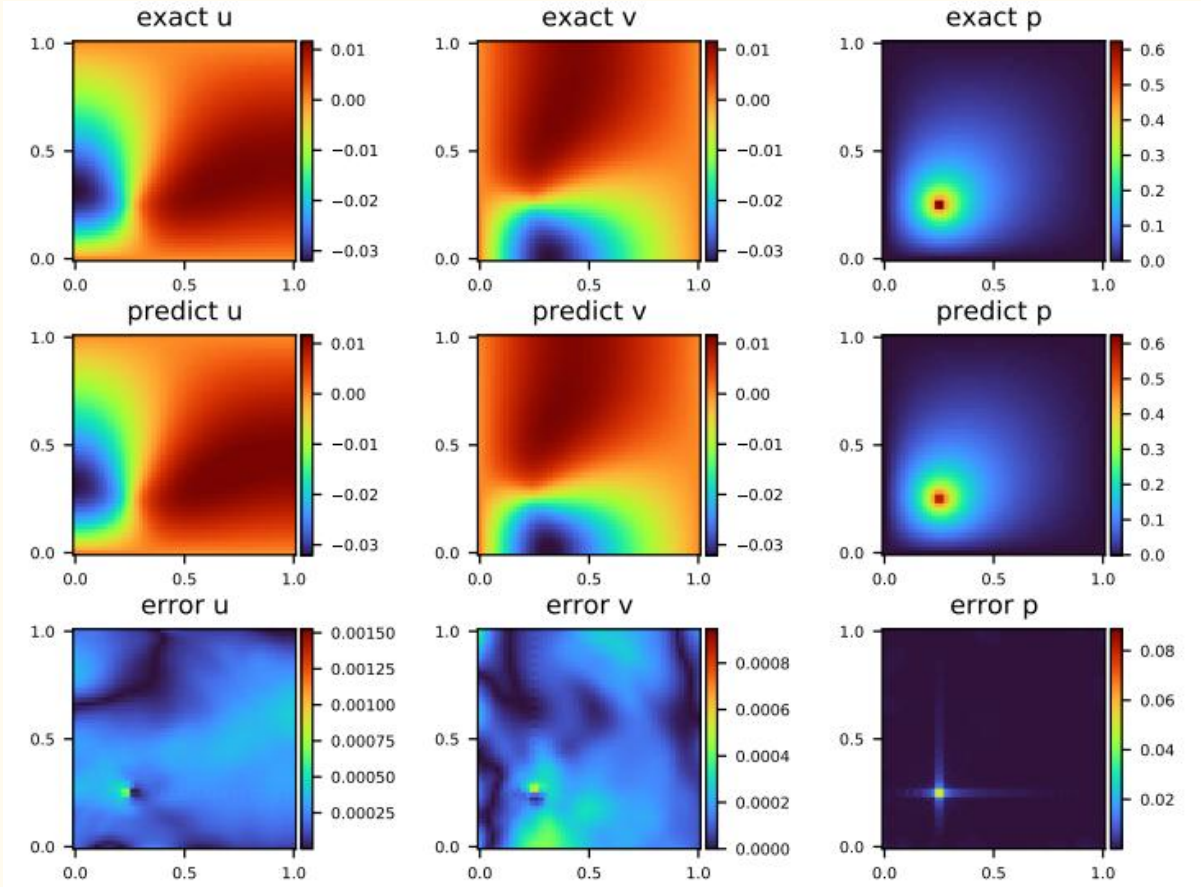


$$\frac{\partial^2 u}{\partial t \partial x} + \frac{\partial^2 v}{\partial t \partial z} - \frac{\partial^2 p}{\partial x^2} - \frac{\partial^2 p}{\partial z^2} - \beta Q = 0,$$

$$(\eta + 1)\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} + \eta \frac{\partial^2 v}{\partial x \partial z} - (\eta + 1)\frac{\partial p}{\partial x} = 0,$$

$$\frac{\partial^2 v}{\partial x^2} + (\eta + 1)\frac{\partial^2 v}{\partial z^2} + \eta \frac{\partial^2 u}{\partial x \partial z} - (\eta + 1)\frac{\partial p}{\partial z} = 0.$$

$$Q(x, z, t) = \delta(x - x_0)\delta(z - z_0)sin(\omega t)$$



解析解 VS. 深度学习方法

目录 Contents

三、基于元学习思想高效求解参数化 PDE

- **参数化 PDE 的定义**

参数化 PDE 的一般形式（将初始条件视作一种特殊边界条件）：

$$\mathcal{L}_{\tilde{x}}^{\gamma_1} u = 0, \quad \tilde{x} \in \Omega \subset \mathbb{R}^d, \quad \text{(控制方程)}$$

$$\mathcal{B}_{\tilde{x}}^{\gamma_2} u = 0, \quad \tilde{x} \in \partial\Omega. \quad \text{(边界条件)}$$

用符号 η 表示 PDE 参数，即 $\eta = (\gamma_1, \gamma_2, \Omega)$；用符号 $\mathcal{A}$ 表示 PDE 参数空间，即 $\eta \in \mathcal{A}$；用符号 $\mathcal{U}$ 表示方程解 u 所在的函数空间，即 $u \in \mathcal{U}$；

- ☑ 不同的 PDE 参数 η 对应不同的方程解 u，一个 PDE 参数对应一个方程实例；
- ☑ 求解参数化 PDE 需要学习一个无穷维的算子：$G: \mathcal{A} \to \mathcal{U}$，该算子可将任意的 PDE 参数 η 映射到其对应的方程解 u^η；
- ☑ 从 η 到 u^η 的映射是 PDE 参数空间 $\mathcal{A}$ 到方程解所在的函数空间 $\mathcal{U}$ 的映射（称为“解映射”）。

符号	含义
Ω	求解域
$\partial\Omega$	求解域 Ω 的边界
$\mathcal{L}$	控制方程对应的偏微分算子
γ_1	控制方程中的可变参数，例如麦克斯韦方程组中的介电系数或磁导率
$\mathcal{B}$	边界条件对应的偏微分算子
γ_2	边界条件中的可变参数，例如Burgers方程中的初始时刻的物理场
$\tilde{x}$	坐标
u	状态变量（方程解）
$\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^d)$	方程解所在的函数空间（即$u \in \mathcal{U}$）
$\eta = (\gamma_1, \gamma_2, \Omega)$	PDE 参数
$\mathcal{A}$	PDE 参数空间 （即 $\eta \in \mathcal{A}$）
u^η	PDE 参数 η 对应的方程解

三、基于元学习思想高效求解参数化 PDE

- **基于深度学习求解参数化 PDE 的研究现状**

分类	方法	Lebel-Free	Mesh-Free	Without Retraining
物理驱动方法	PINNs M. Raissi et al., JCP, 378:686-707, 2019.	✓	✓	✗
	DGM Sirignano and Spiliopoulos, JCP, 375:1339-1364, 2018.	✓	✓	✗
	DRM W. E and B. Yu, CMS, 6(1), 1-12, 2018.	✓	✓	✗
	WAN Y. Zang et al., JCP, 411:109409, 2020.	✓	✓	✗
数据驱动方法	DeepONet L. Lu et al., NAT MACH INTELL, 3(3):218-229, 2021.	✗	✓	✓
	FNO Z. Li et al., ICLR 2021.	✗	✗	✓

☑ **Label-Free:** 不需要标签数据（无监督）；

☑ **Mesh-Free**: 不需要预定义的网格，训练和测试都可以在求解域 Ω 内任意的坐标点上进行；

☑ **Without Retraining**: 对于测试阶段新的 PDE 参数 η_{new}，可以直接进行推理，而不需要重新训练神经网络。

三、基于元学习思想高效求解参数化 PDE

- **PINNs这类物理驱动方法在求解参数化 PDE 时的问题**

☑ 参数化 PDE 的一般形式：

$$\mathcal{L}_{\tilde{x}}^{\gamma_1} u = 0, \quad \tilde{x} \in \Omega \subset \mathbb{R}^d, \quad \text{(控制方程)}$$

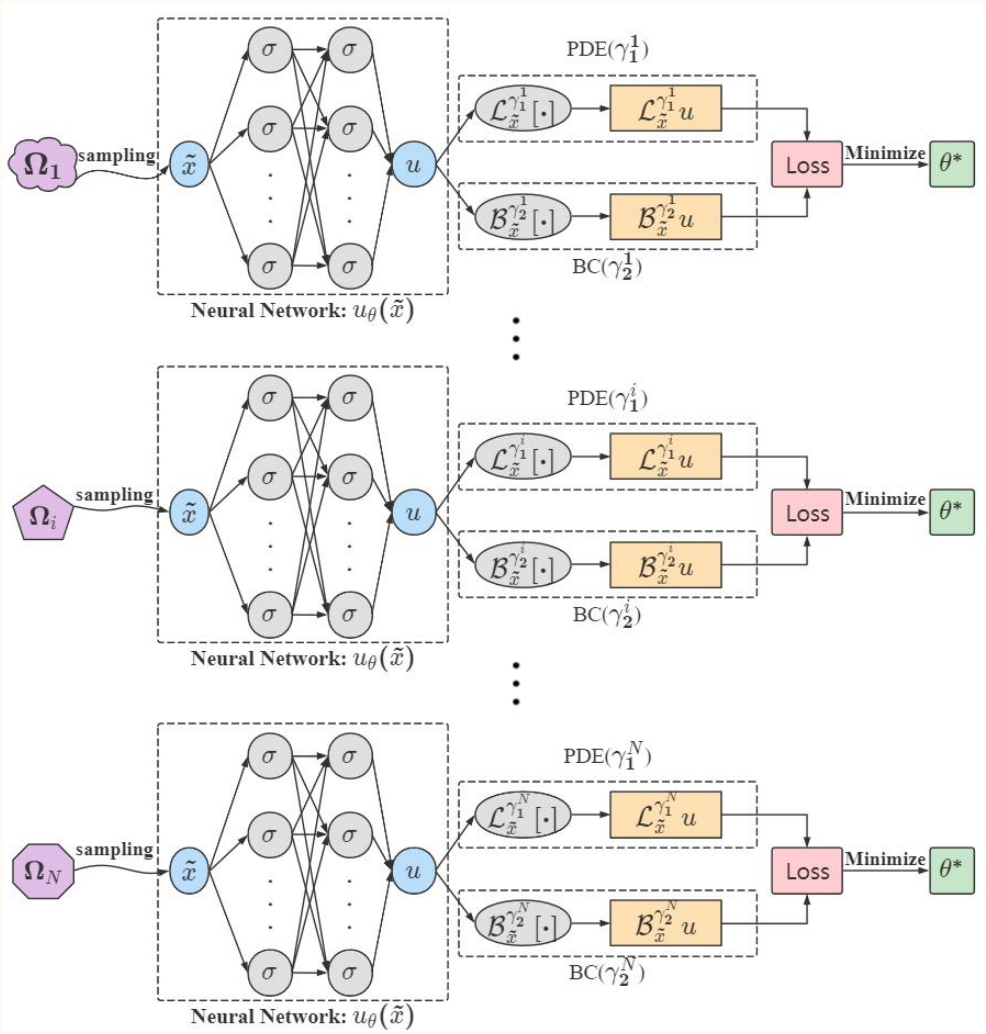
$$\mathcal{B}_{\tilde{x}}^{\gamma_2} u = 0, \quad \tilde{x} \in \partial\Omega. \quad \text{(边界条件)}$$

☑ 物理信息损失函数（Physics-informed Loss）：

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

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

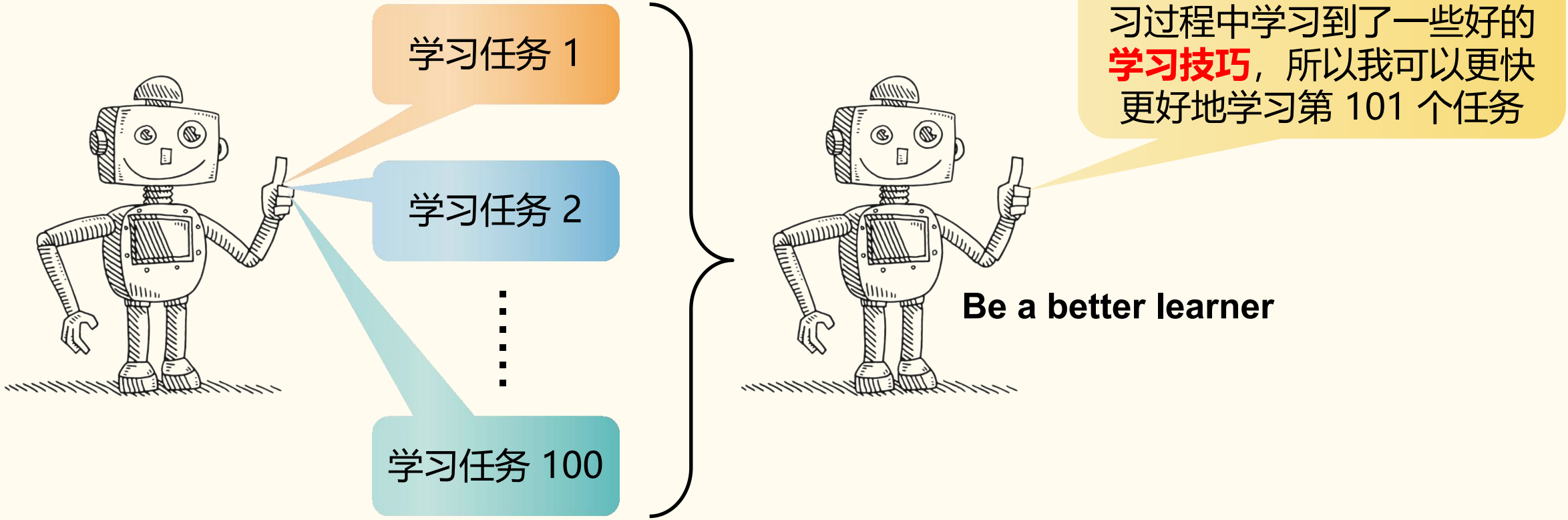
对于每一个 PDE 参数 $\eta_i = (\gamma_1^i, \gamma_2^i, \Omega_i)$，模型权重 θ 需要分别用不同的损失函数 $\hat{L}^{\eta_i}[u]$ 重新训练。



三、基于元学习思想高效求解参数化 PDE

- **元学习（Meta-Learning）**

  ☑ Meta-Learning = Learn to learn



三、基于元学习思想高效求解参数化 PDE

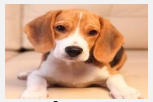
- **元学习（Meta-Learning）**

  ☑ Meta-Learning = Learn to learn

  ☑ 机器学习 VS. 元学习

机器学习

Within-task Training

Train  cat  dog → f_{θ^*}

元学习

Training Tasks

Task 1

Train  apple  orange Test  apple  orange

Task 2

Train  bike  car Test  bike  car

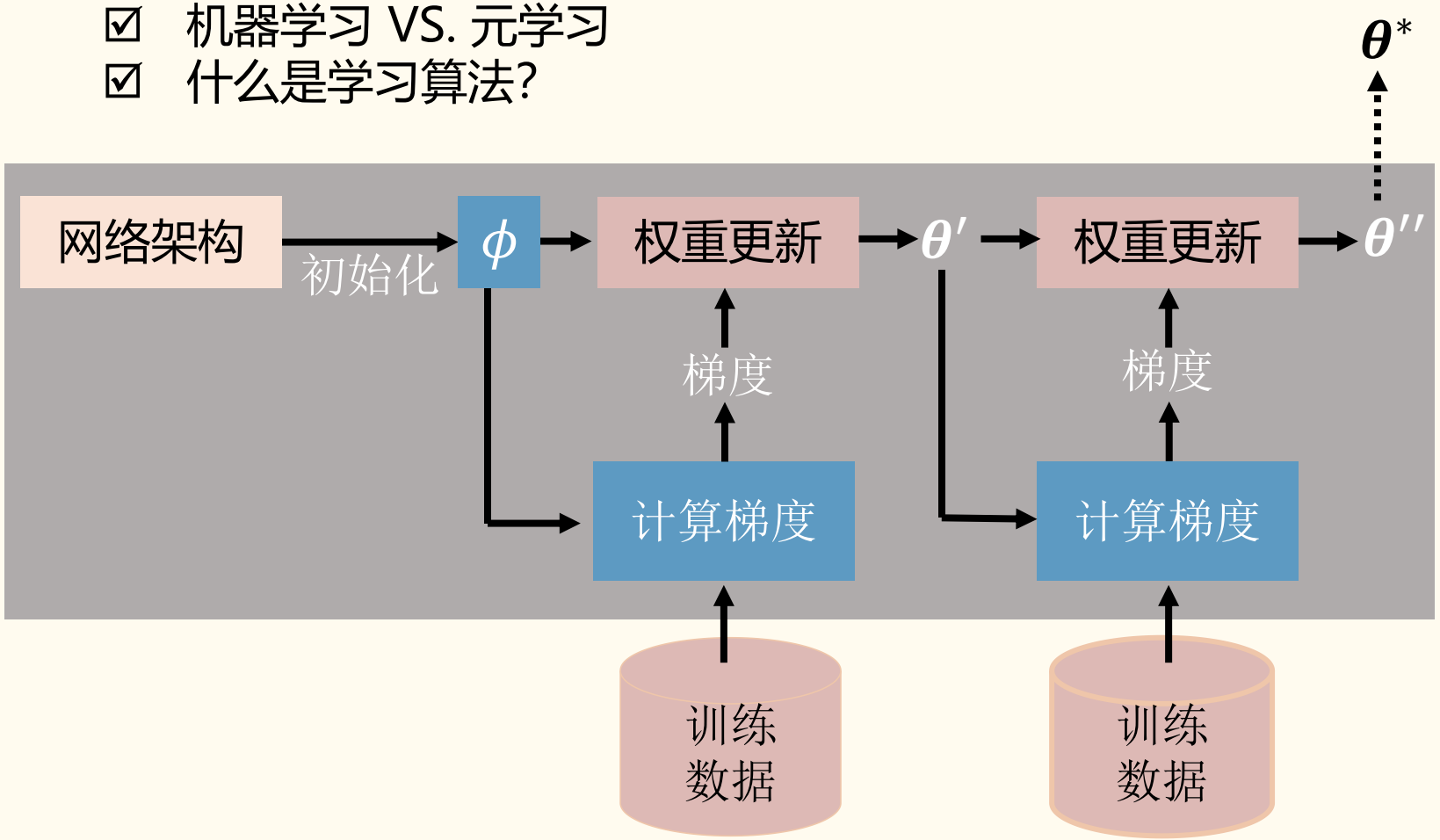
F_{ϕ^*}（学习算法）

Across-task Training

https://speech.ee.ntu.edu.tw/~hylee/ml/2021-spring.php

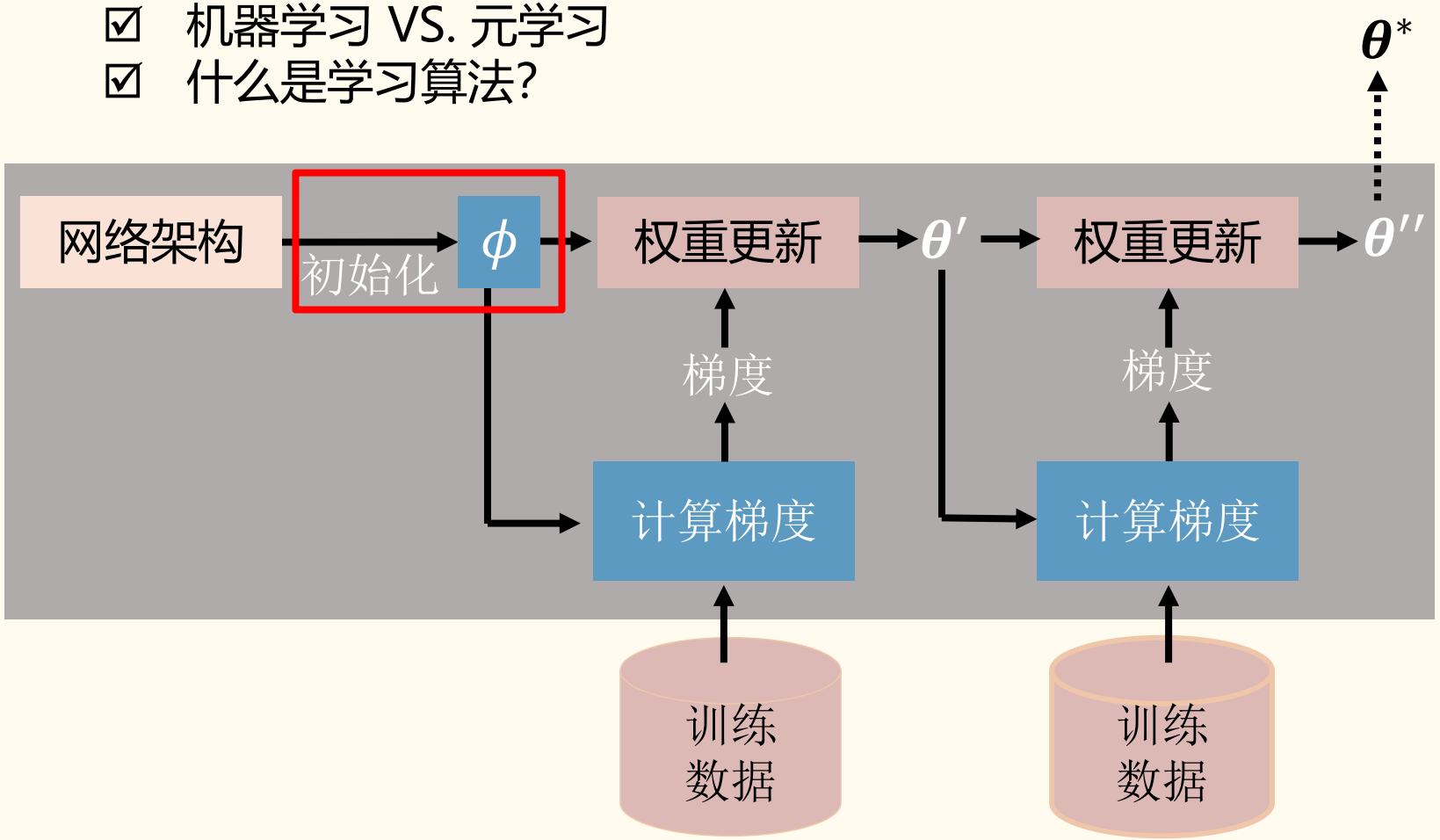
三、基于元学习思想高效求解参数化 PDE

- **元学习（Meta-Learning）**
  - ☑ Meta-Learning = Learn to learn
  - ☑ 机器学习 VS. 元学习
  - ☑ 什么是学习算法?



三、基于元学习思想高效求解参数化 PDE

- **元学习（Meta-Learning）**
  - ☑ Meta-Learning = Learn to learn
  - ☑ 机器学习 VS. 元学习
  - ☑ 什么是学习算法?



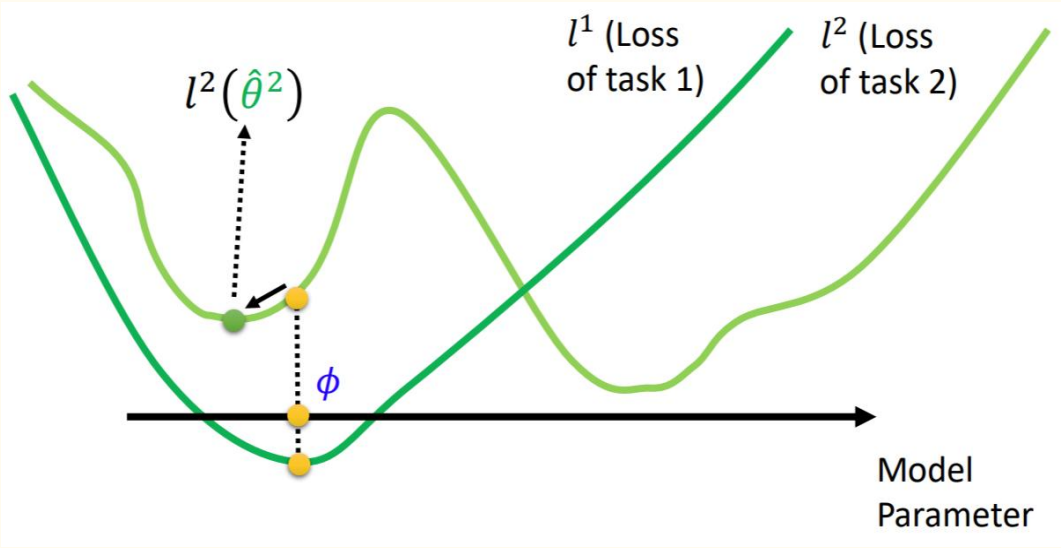
学习算法:

- ☑ 网络架构
- ☑ 模型权重初始化
- ☑ 学习率
- ☑ 优化器
- ☑ 训练数据

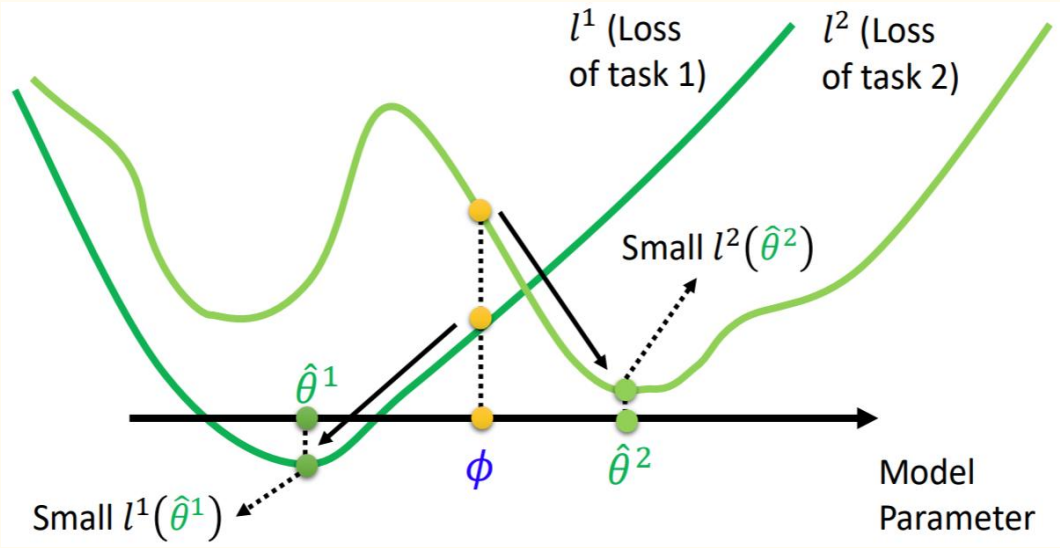
三、基于元学习思想高效求解参数化 PDE

- **元学习（Meta-Learning）**
  - ☑ Meta-Learning = Learn to learn
  - ☑ 机器学习 VS. 元学习
  - ☑ 什么是学习算法?
  - ☑ 迁移学习 VS. 元学习

元学习算法不关心学习到的 ϕ 在当前任务上的表现如何，而是关注 ϕ 在未来学习过程中的潜力。



迁移学习

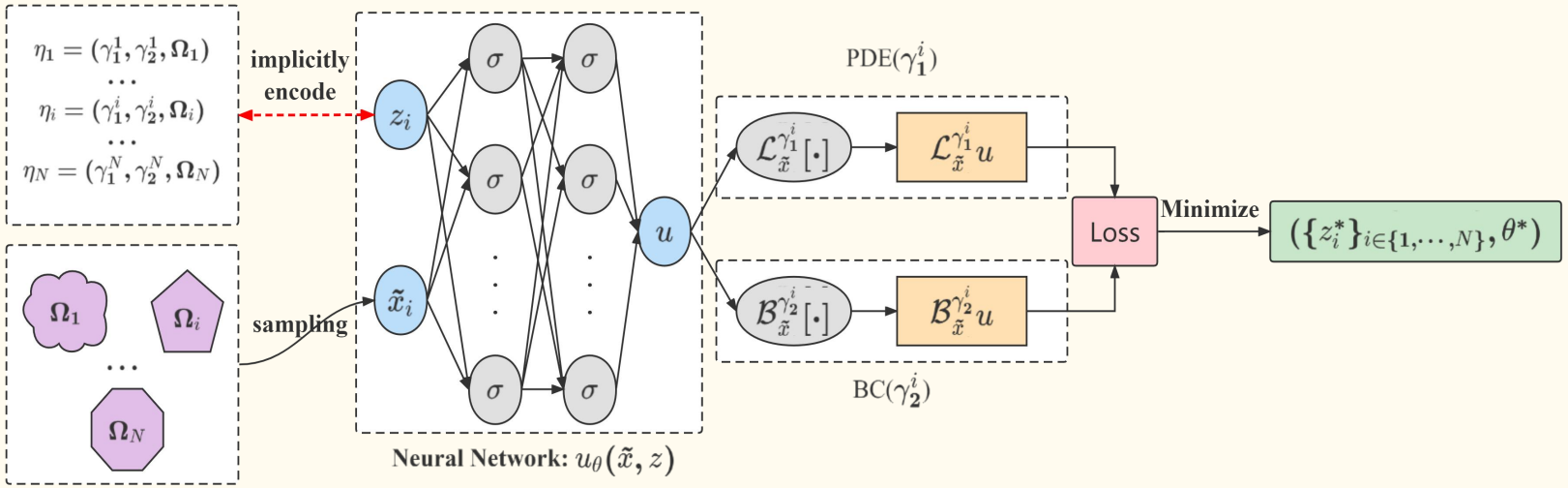


元学习

https://speech.ee.ntu.edu.tw/~hylee/ml/2019-spring.php

三、基于元学习思想高效求解参数化 PDE

- **Meta-Auto-Decoder (MAD) 的预训练**



☑ 给定 N 个 PDE 参数 $\eta_1, \cdots, \eta_N \in \mathcal{A}$，其对应 N 个方程实例求解任务且第 i 个任务对应求解区域 Ω_i，每个 PDE 参数 η_i 会被隐式编码为一个可训练的隐向量 z_i；

☑ 所有任务会共享同一个网络，网络的输入是 $\tilde{x}_i$ 和 z_i 的串联，其中 $\tilde{x}_i$ 表示任务 i 对应的采样点的坐标；

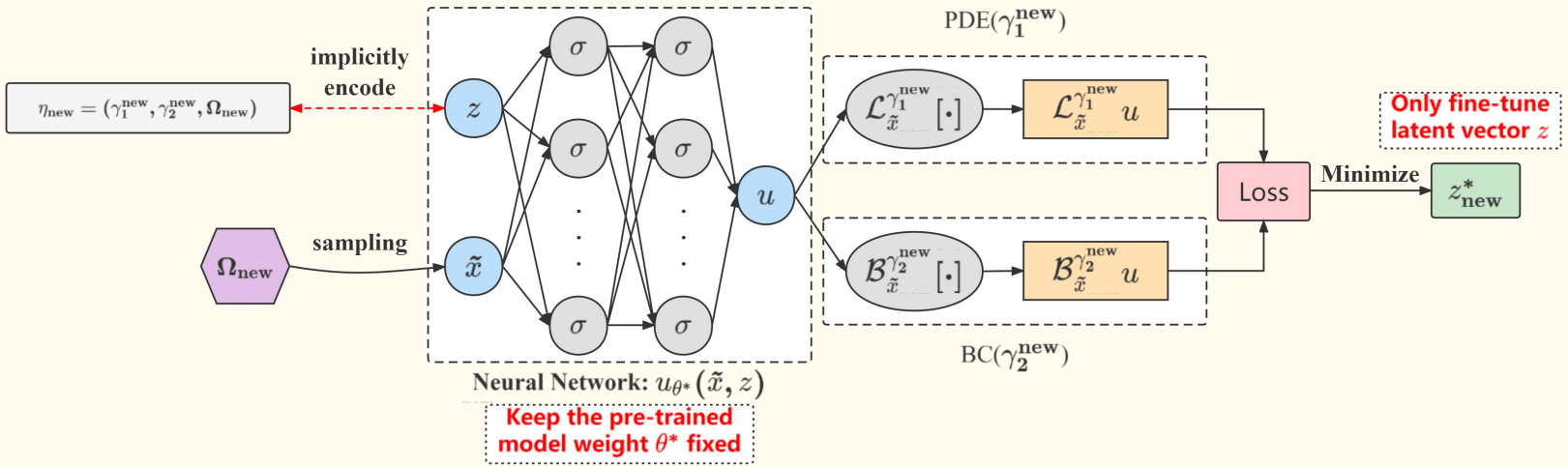
☑ 通过求解以下优化问题，MAD 可以为所有任务学到一个网络权重为 θ^* 的预训练模型以及每个任务特有的隐向量 z_i^*：

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

符号	含义
η_i	第 i 个 PDE 参数 ⇔ 第 i 个方程实例求解任务
Ω_i	第 i 个任务对应的求解域
$\tilde{x}_i$	求解域 Ω_i 中的采样点的坐标
z_i	第 i 个任务对应的可训练的隐向量
z_i^*	预训练结束后第 i 个任务对应的隐向量⇔ 任务特有的知识
θ	模型权重
θ^*	预训练得到的模型权重 ⇔ 元知识
$\hat{L}^{\eta_i}$	第 i 个 PDE 参数对应的物理信息损失函数

三、基于元学习思想高效求解参数化 PDE

- **第一种微调策略 —— MAD-L**



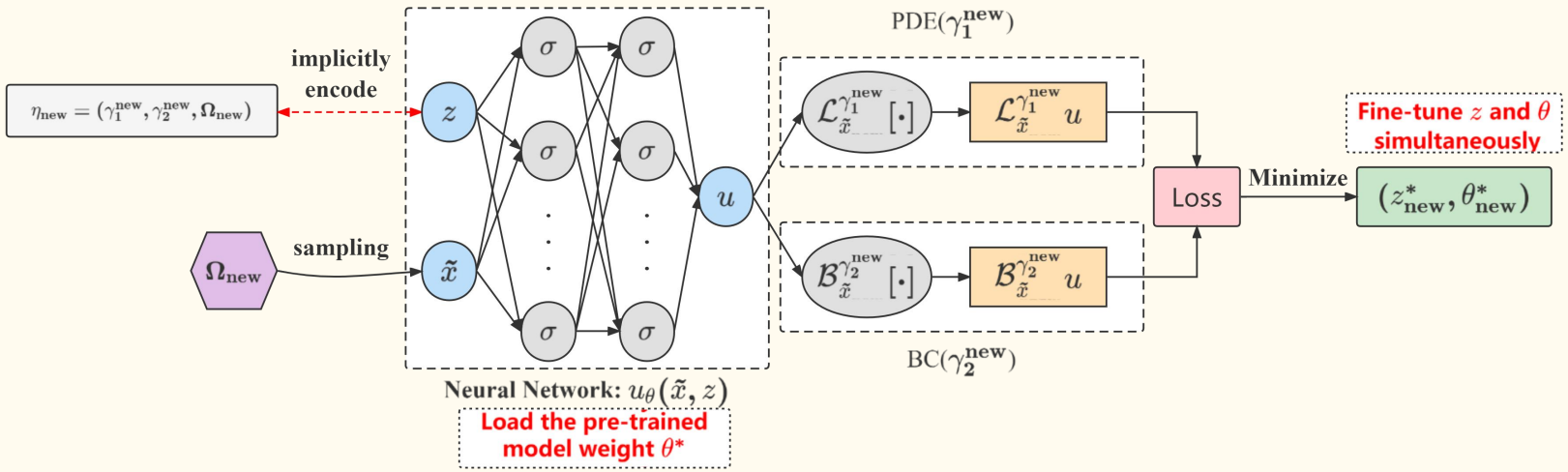
☑ 给定一个新的PDE参数 η_{new}，MAD-L 加载预训练好的网络权重 θ^* 并固定不变，通过求解如下优化问题来获得最优的隐向量 z_{new}^*：

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

符号	含义
η_{new}	微调阶段新的 PDE 参数 ⇔ 新的方程实例求解任务
Ω_{new}	新任务对应的求解域
$\tilde{x}$	求解域 Ω_{new} 中的采样点的坐标
z	新任务对应的可训练的隐向量
z_{new}^*	微调结束后新任务对应的隐向量⇔ 新任务特有的知识
θ^*	预训练得到的模型权重 ⇔ 元知识
$\hat{L}^{\eta_{\text{new}}}$	新的 PDE 参数对应的物理信息损失函数

三、基于元学习思想高效求解参数化 PDE

- **第二种微调策略 —— MAD-LM**



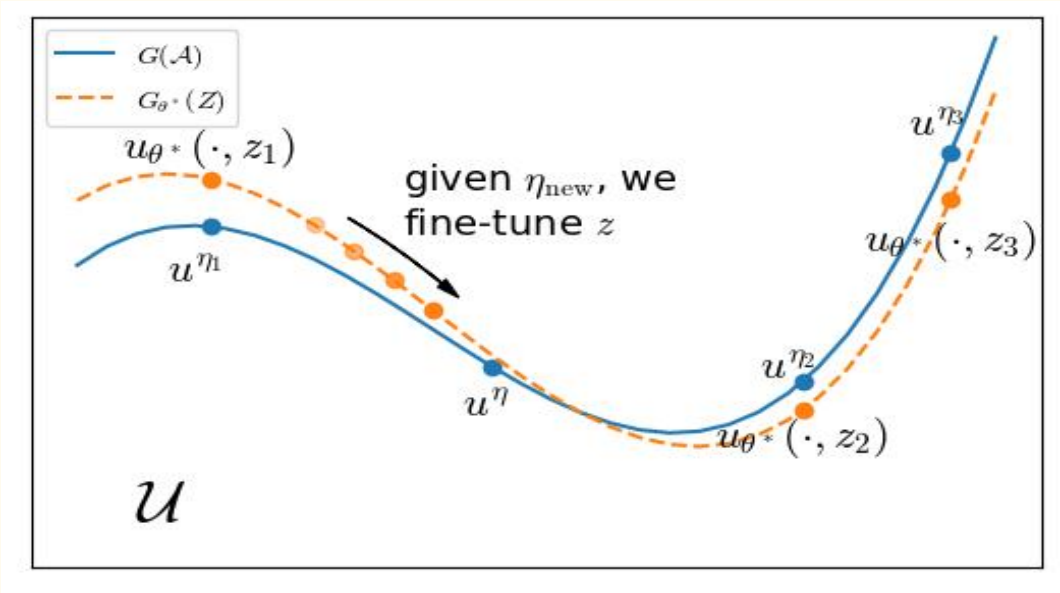
☑ 给定一个新的PDE参数 η_{new}，MAD-LM 首先加载预训练好的网络权重 θ^*，然后通过求解如下优化问题来同时微调网络权重 θ 和隐向量 z：

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

符号	含义
η_{new}	微调阶段新的 PDE 参数 ⇔ 新的方程实例求解任务
Ω_{new}	新任务对应的求解域
$\tilde{x}$	求解域 Ω_{new} 中的采样点的坐标
z	新任务对应的可训练的隐向量
z_{new}^*	微调结束后新任务对应的隐向量⇔ 新任务特有的知识
θ^*	预训练得到的模型权重 ⇔ 元知识
θ_{new}^*	微调结束后得到的模型权重
$\hat{L}^{\eta_{\text{new}}}$	新的 PDE 参数对应的物理信息损失函数

三、基于元学习思想高效求解参数化 PDE

- **从流形学习角度解释 MAD-L 的工作原理**



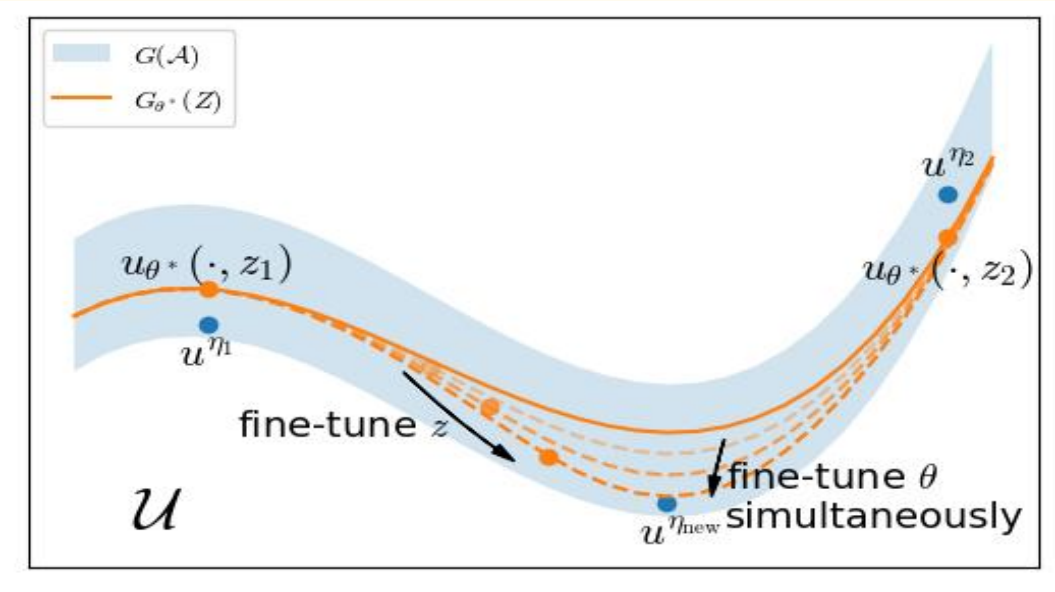
只微调隐向量 z（MAD-L）

符号	含义
$\mathcal{U}$	方程解所在的函数空间（即 $u \in \mathcal{U}$）
$\mathcal{A}$	PDE 参数空间（即 $\eta \in \mathcal{A}$）
$G(\mathcal{A})$	所有 PDE 参数对应的真解组成的集合
u^{η_i}	PDE 参数 η_i 对应的方程解
Z	隐向量所在的函数空间（即 $z \in Z$）
$G_{\theta^*}(Z)$	预训练模型预测得到的方程解组成的集合
$u_{\theta^*}(\cdot, z_i)$	PDE 参数 η_i 对应的神经网络近似解

☑ 将 PDE 的真解所在的函数空间 $\mathcal{U}$ 映射到二维平面上；
☑ 蓝色实线表示所有PDE参数对应的真解 $G(\mathcal{A})$ 所在的流形，而蓝色实线上的每个点代表每一个 PDE 参数对应的真解 u^{η_i}；
☑ 橙色虚线表示预训练模型预测得到的方程解对应的流形 $G_{\theta^*}(Z)$，而橙色虚线上的每个点对应一个隐向量 z_i；
☑ 给定一个新的 PDE 参数 η_{new}，MAD-L 方法不是在整个函数空间 $\mathcal{U}$ 中搜索方程解，而是只在橙色虚线（预训练模型预测得到的方程解对应的流形）上搜索；
☑ 由于搜索空间变小了（从一个二维平面变成了一条曲线），所以微调的速度将非常快。

三、基于元学习思想高效求解参数化 PDE

- **从流形学习角度解释 MAD-LM 的工作原理**



同时微调模型权重 θ 和 隐向量 z（MAD-L)

符号	含义
$\mathcal{U}$	方程解所在的函数空间（即 $u \in \mathcal{U}$)
$\mathcal{A}$	PDE 参数空间（即 $\eta \in \mathcal{A}$)
$G(\mathcal{A})$	所有 PDE 参数对应的真解组成的集合
u^{η_i}	PDE 参数 η_i 对应的方程解
Z	隐向量所在的函数空间（即 $z \in Z$)
$G_{\theta^*}(Z)$	预训练模型预测得到的方程解组成的集合
$u_{\theta^*}(\cdot, z_i)$	PDE 参数 η_i 对应的神经网络近似解

☑ 在一些复杂场景下，所有 PDE 参数对应的真解无法表示成某个流形（蓝色实线），而是表示成某个流形所在的邻域（浅蓝色的窄带）；

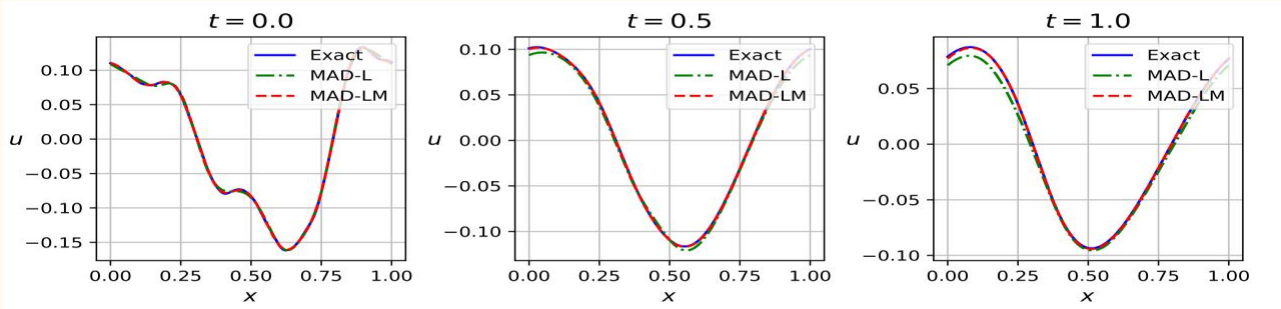
☑ 给定一个新的 PDE 参数 η_{new}，MAD-LM 方法不仅要在橙色实线上搜索，还要在橙色实线之外的地方搜索。而橙色实线上的搜索对应着微调隐向量 z，而在橙色实线之外的地方进行搜索对应着微调网络权重 θ；

☑ 由于模型权重 θ 在预训练阶段学到了好的初始化并且隐向量 z 的搜索空间较小，所以微调的收敛速度将依然非常的快。

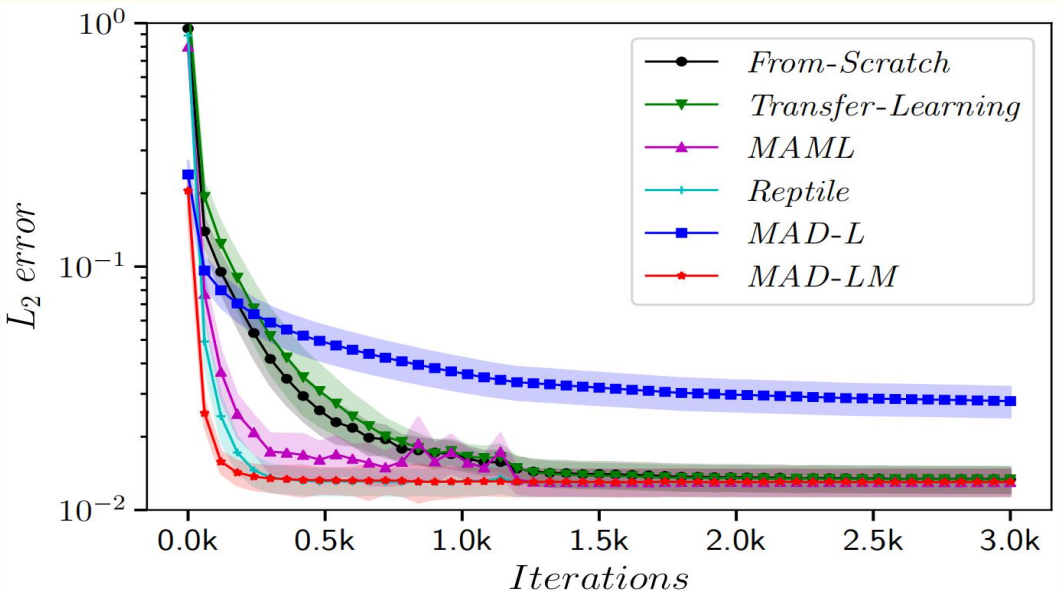
三、基于元学习思想高效求解参数化 PDE

- **数值实验 —— 初始条件可变的 Burgers 方程**

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



参考解 VS. 模型预测解



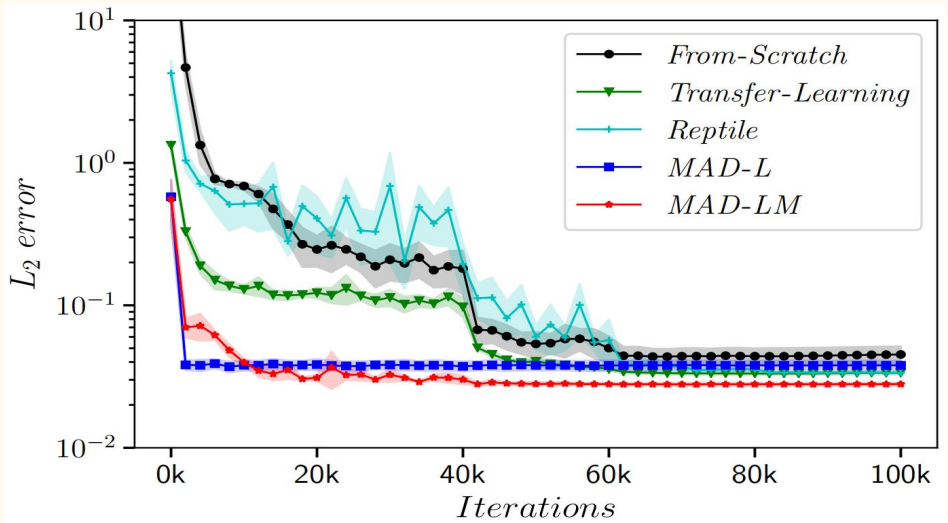
不同方法在微调阶段收敛速度的对比

方法	含义
From-Scratch	基于 PINNs 方法对每一个 PDE 参数从头开始训练模型
Transfer-Learning	随机的选择一个 PDE 参数用于预训练，在微调阶段会加载预训练好的模型权重
MAML	利用 MAML 算法进行预训练，在微调阶段会加载预训练好的模型权重
Reptile	利用 Reptile 算法进行预训练，在微调阶段会加载预训练好的模型权重
MAD-L	只微调隐向量 z
MAD-LM	同时微调模型权重 θ 和 隐向量 z

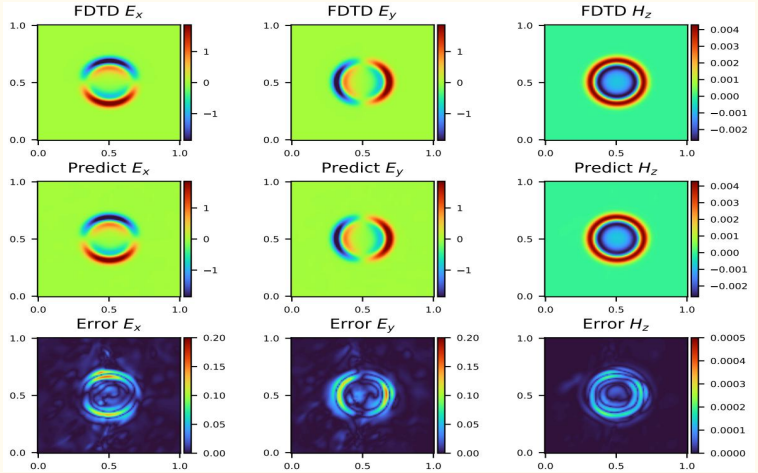
三、基于元学习思想高效求解参数化 PDE

- **数值实验 —— 方程系数可变的麦克斯韦方程组**

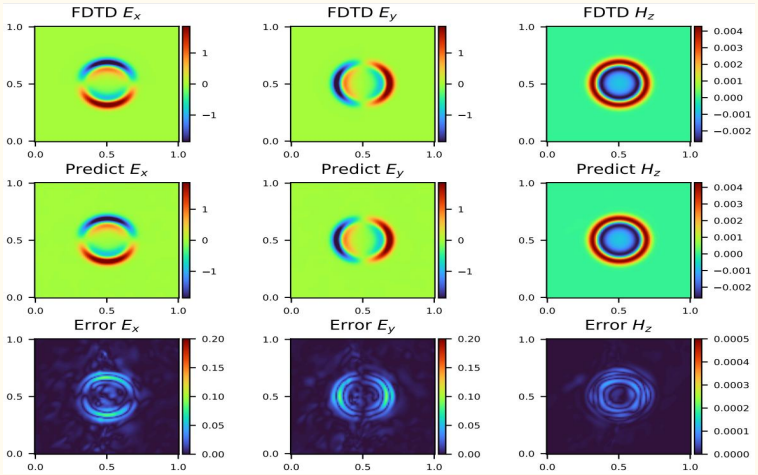
$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_0 \varepsilon_r}\frac{\partial H_z}{\partial y}$$
$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_0 \varepsilon_r}\frac{\partial H_z}{\partial x}$$
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu_0 \mu_r}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + J\right)$$
$$J = e^{-\left(\frac{t-d}{\tau}\right)^2}\delta(x-x_0)\delta(y-y_0)$$



不同方法在微调阶段收敛速度的对比



传统数值求解方法 FDTD VS. MAD-L

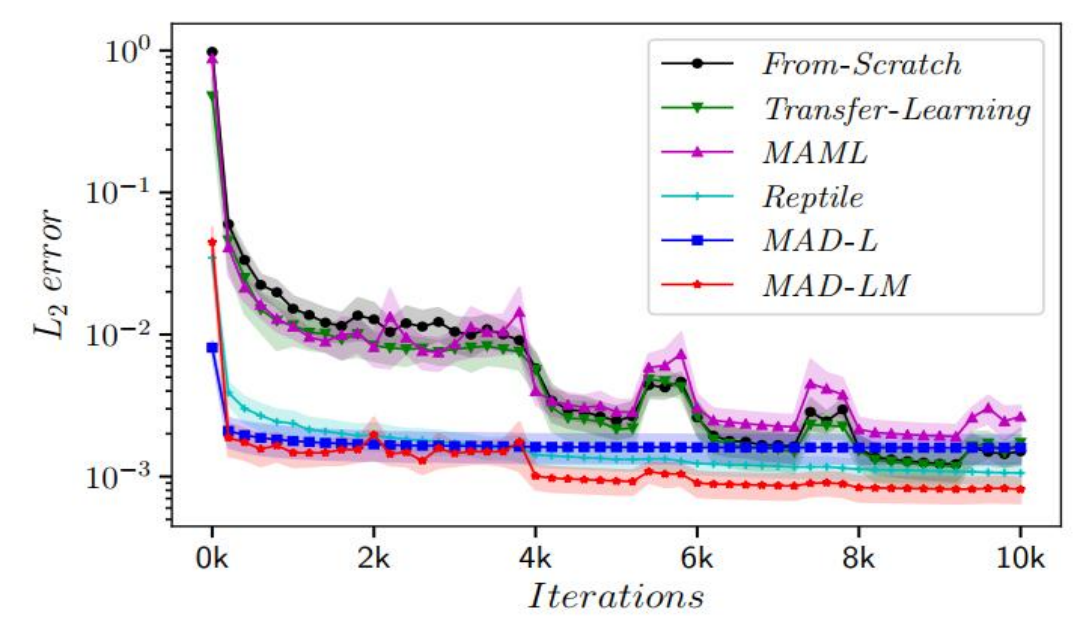


传统数值求解方法 FDTD VS. MAD-LM

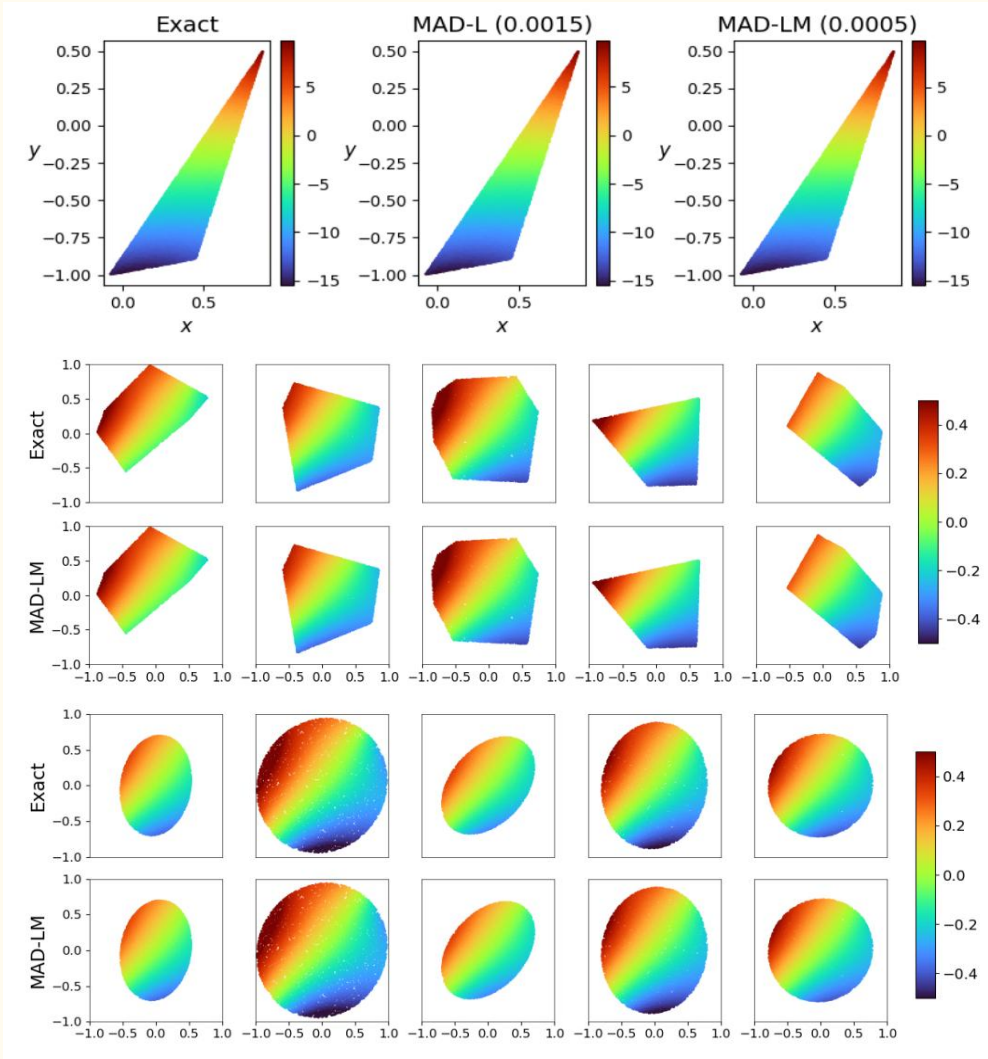
三、基于元学习思想高效求解参数化 PDE

- **数值实验 —— 求解域形状和边界条件可变的拉普拉斯方程**

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



不同方法在微调阶段收敛速度的对比



目录 Contents

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **控制方程部分已知的时空动力学系统**

☑ 时空偏微分方程的一般形式：

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}), \quad (\boldsymbol{x}, t) \in \Omega \times [0, T], \quad \text{(控制方程)}$$
$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{U}) = 0, \quad \boldsymbol{x} \in \Omega, \quad \text{(初始条件)}$$
$$\boldsymbol{\mathcal{B}}(\boldsymbol{x}, t, \boldsymbol{U}) = 0, \quad (\boldsymbol{x}, t) \in \partial\Omega \times [0, T]. \quad \text{(边界条件)}$$

☑ 假设 $\boldsymbol{\mathcal{L}}(\cdot)$ 是部分已知的，可以分解为已知部分 $\boldsymbol{\mathcal{L}}_{\text{known}}$ 和未知部分 $\boldsymbol{\mathcal{L}}_{\text{unknown}}$，即：

$$\boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}) = \boldsymbol{\mathcal{L}}_{\text{known}}(\boldsymbol{x}, \boldsymbol{U}) + \boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x}, \boldsymbol{U})$$

☑ 对于已知部分 $\boldsymbol{\mathcal{L}}_{\text{known}}$，我们知道其对应的确切形式：

$$\boldsymbol{\mathcal{L}}_{\text{known}}(\boldsymbol{x}, \boldsymbol{U}) = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{U}, \nabla \boldsymbol{U}, \nabla^2 \boldsymbol{U}, \cdots)$$

☑ 对于未知部分 $\boldsymbol{\mathcal{L}}_{\text{unknown}}$，我们没有任何的先验知识。

符号	含义
$\boldsymbol{x}$	空间坐标
t	时间坐标
$\boldsymbol{U}$	状态变量（方程解）
Ω	求解域
$\partial\Omega$	求解域 Ω 的边界
T	仿真时长
$\boldsymbol{\mathcal{L}}$	描述动力学规律的微分算子
$\boldsymbol{\mathcal{L}}_{\text{known}}$	$\boldsymbol{\mathcal{L}}$ 中的已知部分
$\boldsymbol{\mathcal{L}}_{\text{unknown}}$	$\boldsymbol{\mathcal{L}}$ 中的未知部分
$\boldsymbol{\mathcal{I}}$	初始条件对应的偏微分算子
$\boldsymbol{\mathcal{B}}$	边界条件对应的偏微分算子
$\boldsymbol{\Phi}$	$\boldsymbol{\mathcal{L}}_{\text{known}}$对应的确切函数形式

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **基于深度学习预测时空动力学系统演化过程的研究现状**

分类	方法	存在的问题
物理驱动方法	PINNs M. Raissi et al., JCP, 378:686-707, 2019.	• 在求解正问题时需要完整的 PDE 先验知识 • 优化困难 • 求解参数化 PDE 的效率低
	DGM Sirignano and Spiliopoulos, JCP, 375:1339-1364, 2018.	
	DRM W. E and B. Yu, CMS, 6(1), 1-12, 2018.	
	WAN Y. Zang et al., JCP, 411:109409, 2020.	
数据驱动方法	DeepONet L. Lu et al., NAT MACH INTELL, 3(3):218-229, 2021.	• 需要大量高质量的标签数据 • 长时间预测的性能较差 • 分布外任务的泛化能力较差 • 可解释性差
	FNO Z. Li et al., ICLR 2021.	
数据机理融合方法	CFD-GCN F. Belbute-Peres et al., ICML 2020.	• 针对特定问题而设计，通用性较差
	TF-Net R. Wang et al., KDD 2020.	
	JAX-CFD D. Kochkov et al., PNAS, 118(21):e2101784118, 2021.	
	PDE-Net Z. Long et al., ICML 2018.	• 处理复杂非线性项及未知项能力弱

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net**

时空偏微分方程的一般形式：

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}), \quad (\boldsymbol{x}, t) \in \Omega \times [0, T], \quad \text{（控制方程）}$$
$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{U}) = 0, \quad \boldsymbol{x} \in \Omega, \quad \text{（初始条件）}$$
$$\boldsymbol{\mathcal{B}}(\boldsymbol{x}, t, \boldsymbol{U}) = 0, \quad (\boldsymbol{x}, t) \in \partial\Omega \times [0, T]. \quad \text{（边界条件）}$$

☑ 采用前向欧拉方法对一阶时间偏微分项 $\frac{\partial \boldsymbol{U}}{\partial t}$ 进行离散化：

$$\boldsymbol{U}_{j+1} = \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t$$

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net**

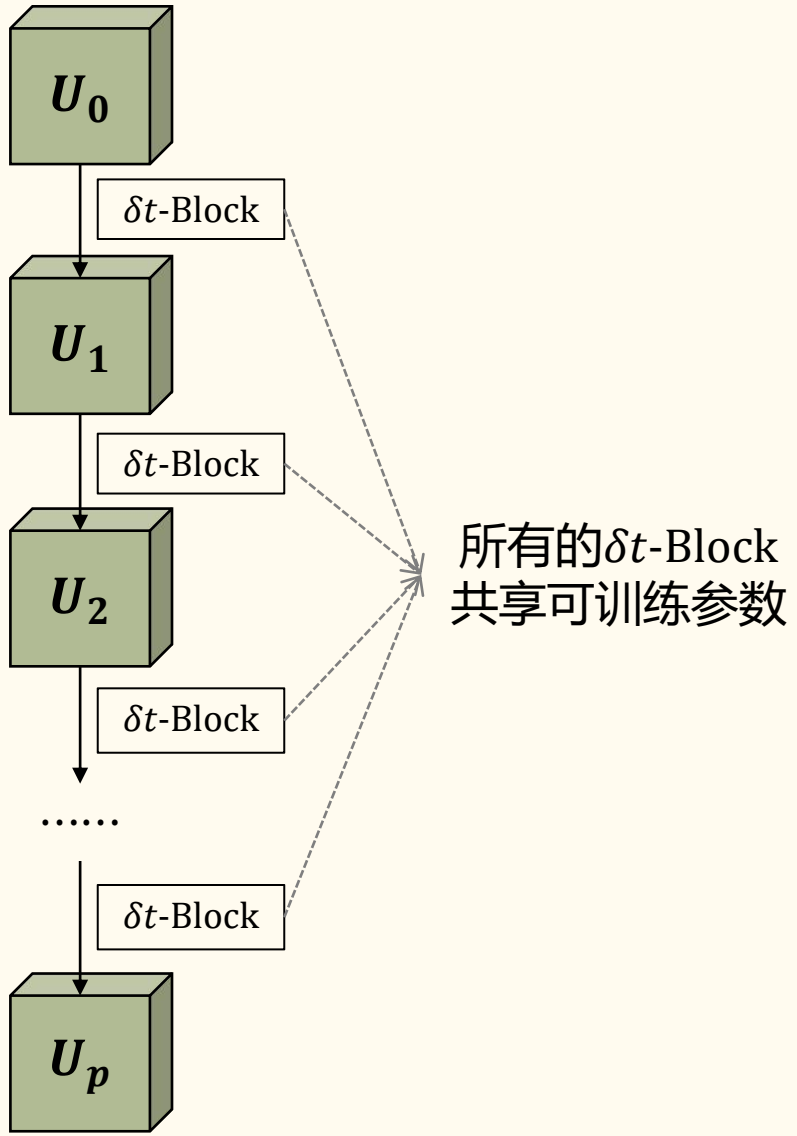
时空偏微分方程的一般形式：

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}), \quad (\boldsymbol{x}, t) \in \Omega \times [0, T], \quad \text{（控制方程）}$$
$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{U}) = 0, \quad \boldsymbol{x} \in \Omega, \quad \text{（初始条件）}$$
$$\boldsymbol{\mathcal{B}}(\boldsymbol{x}, t, \boldsymbol{U}) = 0, \quad (\boldsymbol{x}, t) \in \partial\Omega \times [0, T]. \quad \text{（边界条件）}$$

☑ 采用前向欧拉方法对一阶时间偏微分项 $\frac{\partial \boldsymbol{U}}{\partial t}$ 进行离散化：

$$\boldsymbol{U}_{j+1} = \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t$$

☑ 将 $\boldsymbol{U}_j$ 到 $\boldsymbol{U}_{j+1}$ 的演化过程等效成一个卷积神经网络 (δt-Block) 的前向过程；



四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net**

时空偏微分方程的一般形式：

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}), \quad (\boldsymbol{x}, t) \in \Omega \times [0, T], \quad \text{（控制方程）}$$
$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{U}) = 0, \quad \boldsymbol{x} \in \Omega, \quad \text{（初始条件）}$$
$$\boldsymbol{\mathcal{B}}(\boldsymbol{x}, t, \boldsymbol{U}) = 0, \quad (\boldsymbol{x}, t) \in \partial\Omega \times [0, T]. \quad \text{（边界条件）}$$

☑ 采用前向欧拉方法对一阶时间偏微分项 $\frac{\partial \boldsymbol{U}}{\partial t}$ 进行离散化：

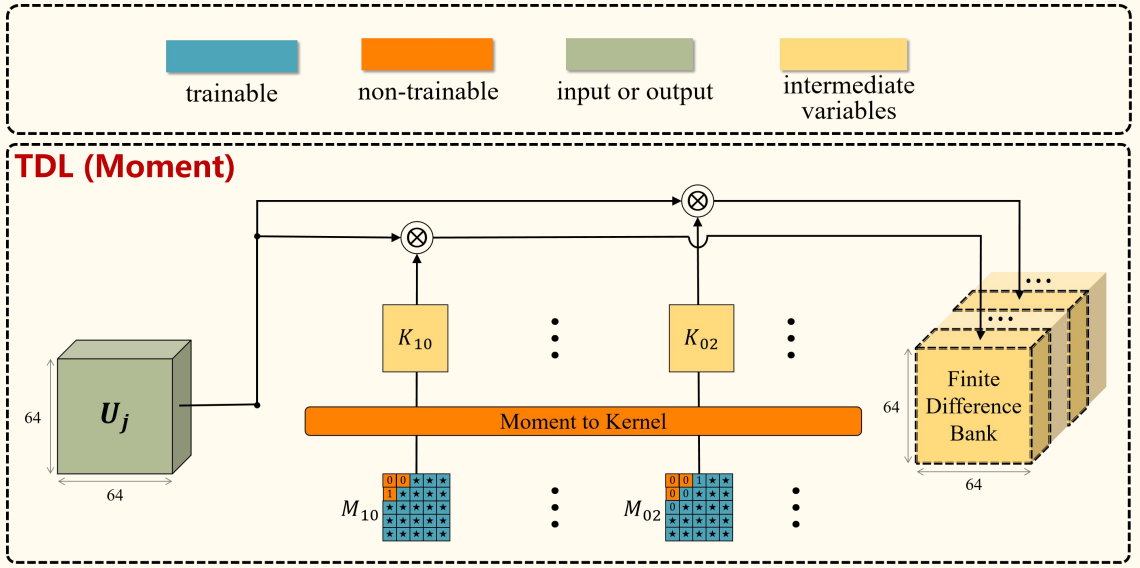
$$\boldsymbol{U}_{j+1} = \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t$$

☑ 将 $\boldsymbol{U}_j$ 到 $\boldsymbol{U}_{j+1}$ 的演化过程等效成一个卷积神经网络 (δt-Block) 的前向过程；

☑ $\boldsymbol{\mathcal{L}}(\cdot)$ 中的所有的偏导数项都用受 Moment 约束的卷积层来计算得到，称该卷积层为 TDL (Moment)，并将所有计算得到的偏导数项存储在有限差分库 (Finite Difference Bank) 中；

$$\boldsymbol{K}_{10} = \frac{1}{2\Delta x}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \Longrightarrow \boldsymbol{K}_{10} \circledast \boldsymbol{U} \approx \frac{\partial \boldsymbol{U}}{\partial x}$$

$$\boldsymbol{K}_{02} = \frac{1}{(\Delta y)^2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} \Longrightarrow \boldsymbol{K}_{02} \circledast \boldsymbol{U} \approx \frac{\partial^2 \boldsymbol{U}}{\partial y^2}$$



- $\boldsymbol{M}_{10}$和 $\boldsymbol{K}_{10}$ 表示偏导数 $\frac{\partial \boldsymbol{U}}{\partial x}$ 对应的 Moment 矩阵和卷积核；
- $\boldsymbol{M}_{02}$ 和 $\boldsymbol{K}_{02}$ 表示偏导数 $\frac{\partial^2 \boldsymbol{U}}{\partial y^2}$ 对应的 Moment 矩阵和卷积核；
- $\boldsymbol{M}_{ij}$ 中的“★”符号表示可训练的参数；
- “Moment to Kernel”运算可以将 Moment 矩阵变换为其对应的卷积核。

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net**

时空偏微分方程的一般形式：

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}), \quad (\boldsymbol{x}, t) \in \Omega \times [0, T], \quad \text{（控制方程）}$$
$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{U}) = 0, \quad \boldsymbol{x} \in \Omega, \quad \text{（初始条件）}$$
$$\boldsymbol{\mathcal{B}}(\boldsymbol{x}, t, \boldsymbol{U}) = 0, \quad (\boldsymbol{x}, t) \in \partial\Omega \times [0, T]. \quad \text{（边界条件）}$$

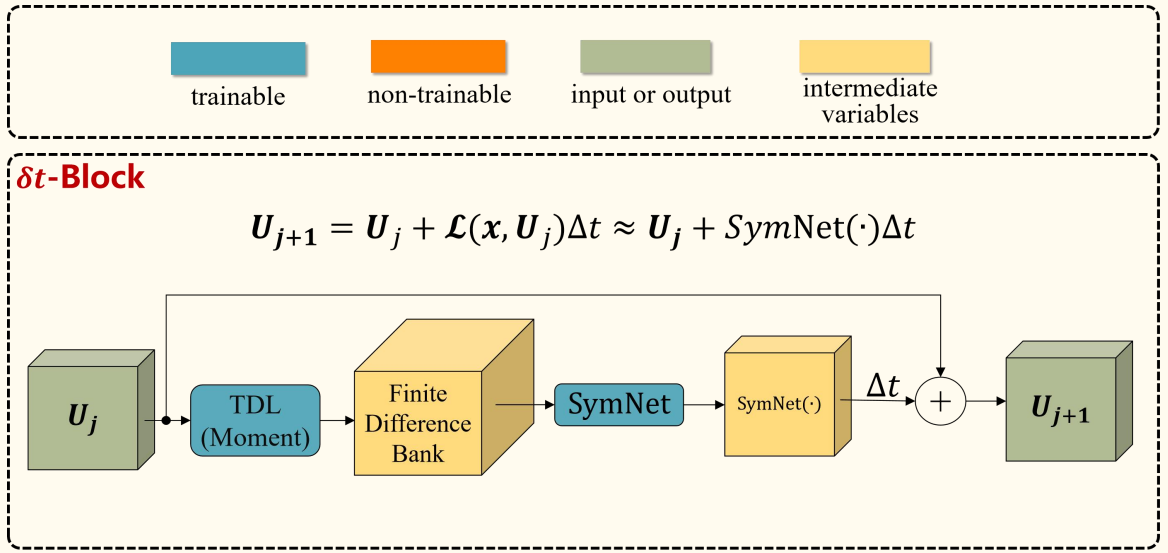
☑ 采用前向欧拉方法对一阶时间偏微分项 $\frac{\partial \boldsymbol{U}}{\partial t}$ 进行离散化：

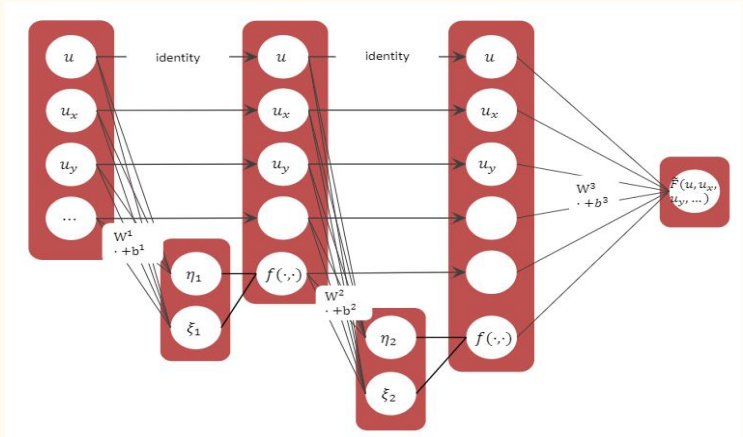
$$\boldsymbol{U}_{j+1} = \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t$$

☑ 将 $\boldsymbol{U}_j$ 到 $\boldsymbol{U}_{j+1}$ 的演化过程等效成一个卷积神经网络 (δt-Block) 的前向过程；

☑ $\boldsymbol{\mathcal{L}}(\cdot)$ 中的所有的偏导数项都用受 Moment 约束的卷积层来计算得到，称该卷积层为 TDL (Moment)，并将所有计算得到的偏导数项存储在有限差分库 (Finite Difference Bank) 中；

☑ 将有限差分库输入到 SymNet (Symbolic Neural Network) 以进行多项式计算，生成 $SymNet(\cdot)$ 以作为 $\boldsymbol{\mathcal{L}}(\cdot)$ 的近似。

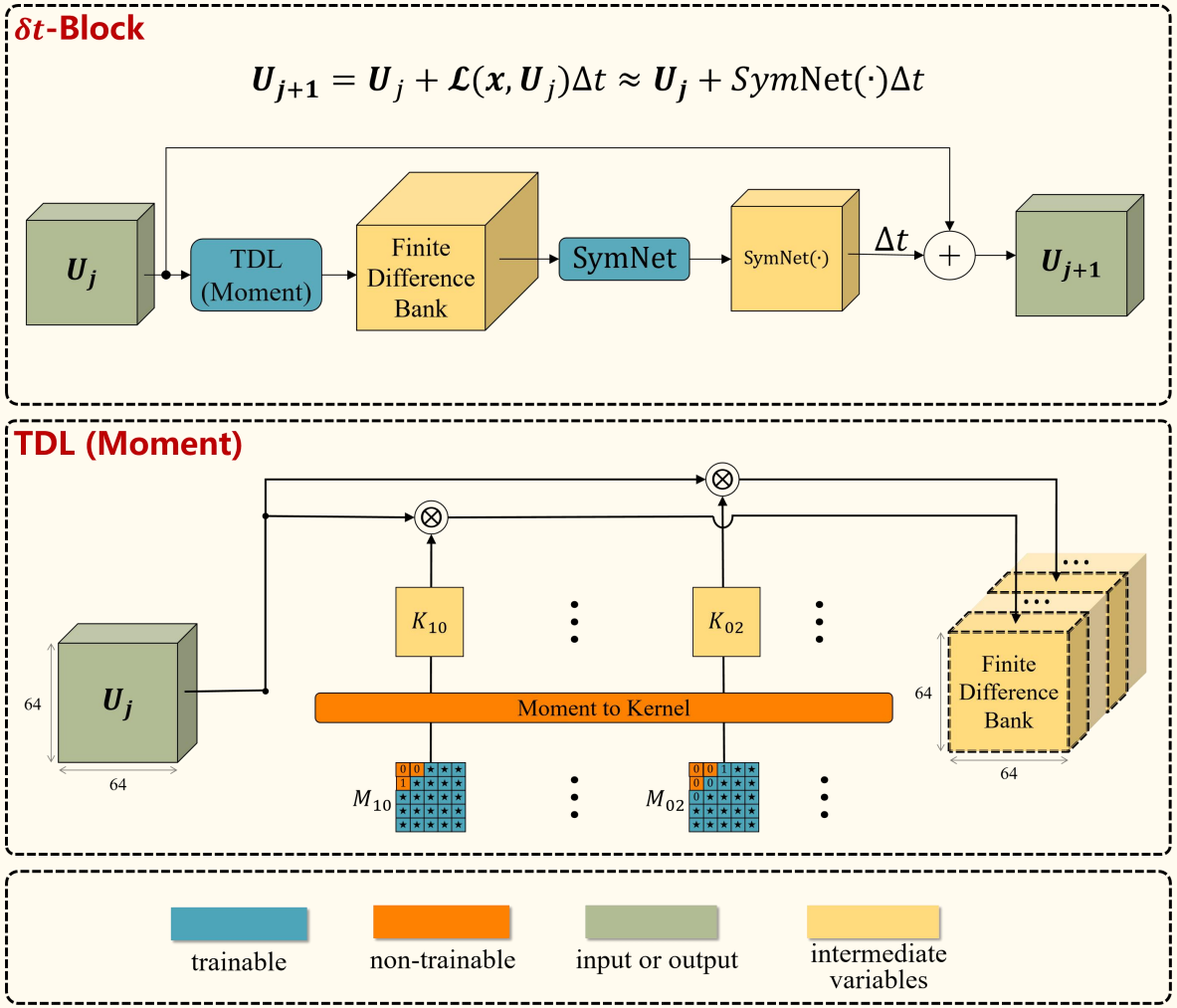




SymNet

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net**



PDE-Net 存在的问题：

☑ PDE-Net 并没有显式地对控制方程的已知部分进行多项式计算，而是用 SymNet 从标签数据中学习，当 PDE 中存在复杂非线性项时（如非线性项对应的方程系数随空间变化），SymNet 会失效；

☑ PDE-Net 没有设计专门的模块处理方程中的未知项，当 PDE 中存在复杂未知项时（如未知项是关于状态变量或多个偏导数项的复杂函数），PDE-Net 也无法处理。

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **PDE-Net++**

☑ 采用前向欧拉方法对一阶时间偏微分项 $\frac{\partial \boldsymbol{U}}{\partial t}$ 进行离散化:

$$\begin{aligned}\boldsymbol{U}_{j+1} &= \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t \\ &= \boldsymbol{U}_j + \boldsymbol{\mathcal{L}}_{\text{known}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t + \boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t \\ &= \boldsymbol{U}_j + \boldsymbol{\Phi}(x, \boldsymbol{U}_j, \nabla \boldsymbol{U}_j, \nabla^2 \boldsymbol{U}_j, \cdots)\Delta t + \boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x}, \boldsymbol{U}_j)\Delta t\end{aligned}$$

☑ $\boldsymbol{\Phi}(\cdot)$ 中的所有的空间偏导数项都用差分算子来生成:

$$\nabla \boldsymbol{U}_j \approx D(\boldsymbol{U}_j), \quad \nabla^2 \boldsymbol{U}_j \approx D^2(\boldsymbol{U}_j), \quad \cdots$$

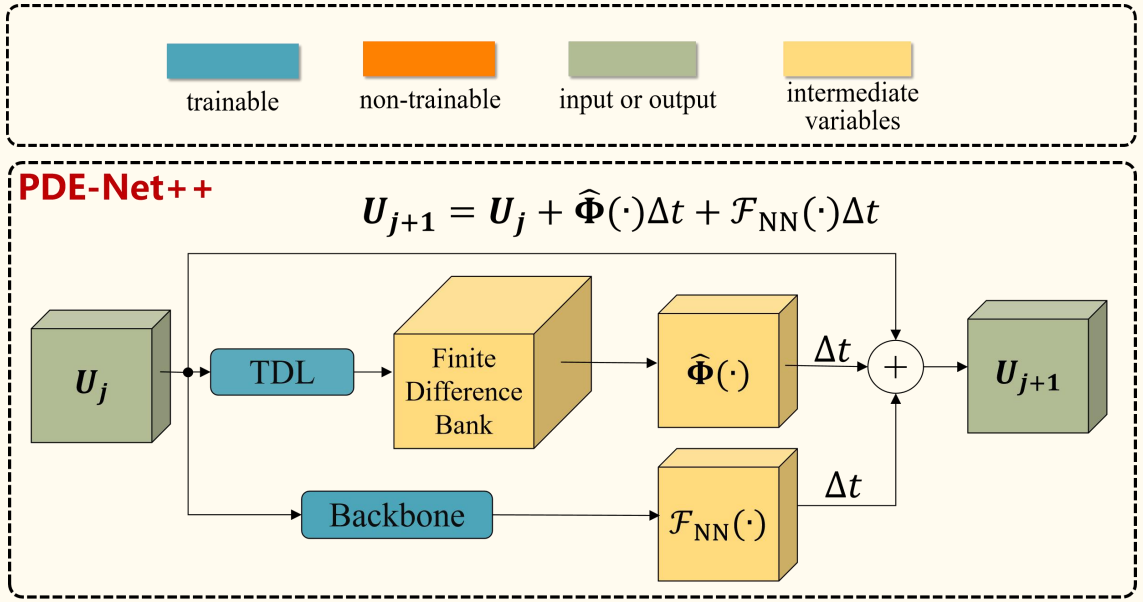
$$\Longrightarrow \quad \boldsymbol{\Phi}(\cdot) \approx \widehat{\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{U}_j, D(\boldsymbol{U}_j), D^2(\boldsymbol{U}_j), \cdots)$$

☑ 使用神经网络（称作“Backbone”）来近似方程中的未知项:

$$\boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x}, \boldsymbol{U}) = \mathcal{F}_{\text{NN}}(\boldsymbol{x}, \boldsymbol{U}_j)$$

☑ 综上，PDE-Net++ 的迭代公式:

$$\boldsymbol{U}_{j+1} = \boldsymbol{U}_j + \widehat{\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{U}_j, D(\boldsymbol{U}_j), D^2(\boldsymbol{U}_j), \cdots)\Delta t + \mathcal{F}_{\text{NN}}(x, \boldsymbol{U}_j)\Delta t$$



Backbone 的选择:

- ☑ U-Net
- ☑ ConvResNet
- ☑ FNO
- ☑ F-FNO
- ☑ Galerkin Transformer

差分算子的选择:

- ☑ FDM
- ☑ TDL (Moment)
- ☑ TFDL
- ☑ TDDL

两种新的可训练差分层

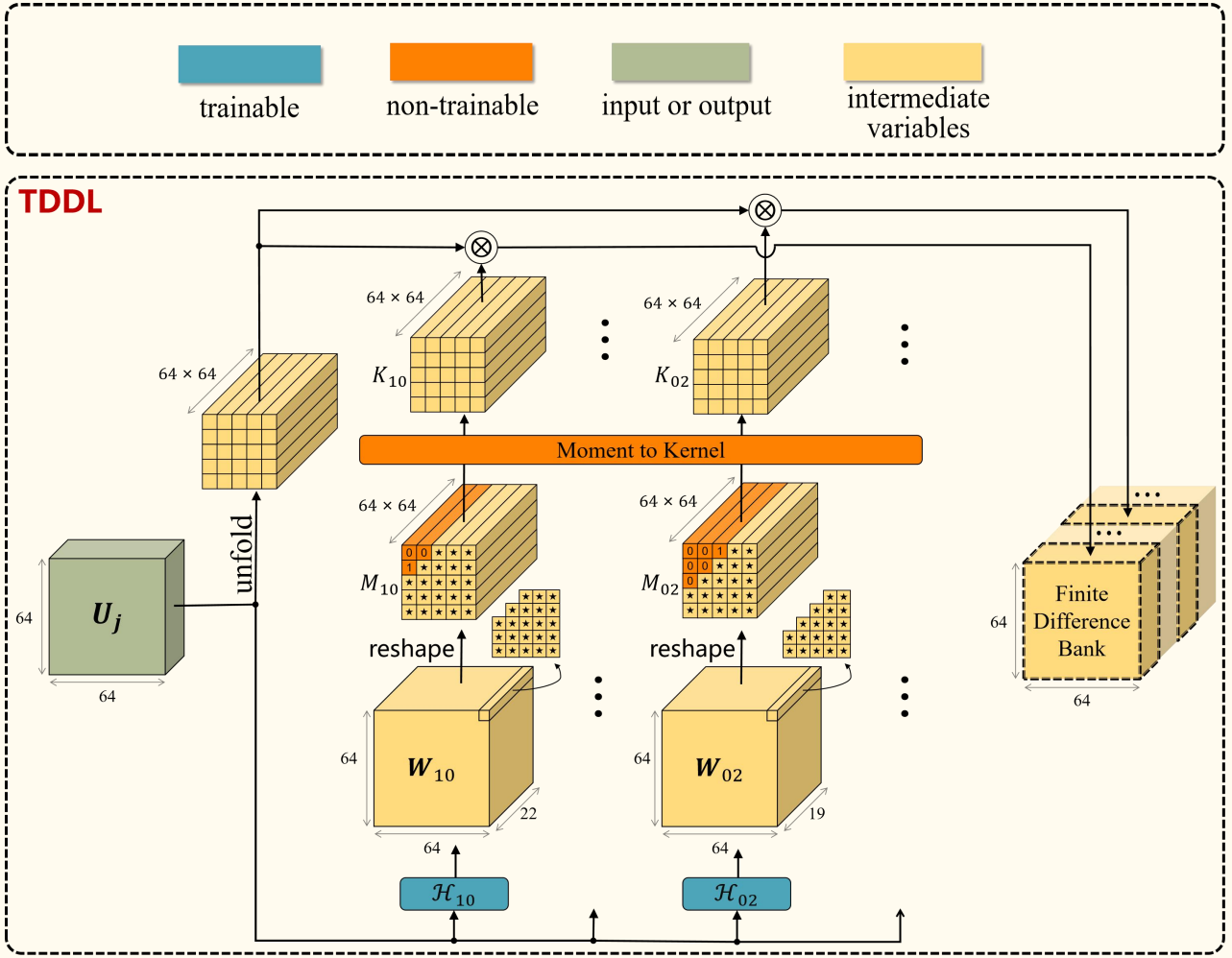
四、预测时空动力学系统演化过程的通用数据机理融合方法

- **TDDL (Trainable Dynamic Difference Layer)**

☑ 基于动态卷积思想：Moment矩阵 $\boldsymbol{M}_{pq}$ 是由超网络 $\mathcal{H}_{pq}$ 生成的；

☑ 超网络 $\mathcal{H}_{pq}$ 本质是一个简单的卷积神经网络（包含 3 个卷积层，激活函数为 ReLU）；

☑ 超网络 $\mathcal{H}_{pq}$ 会为每个离散网格点自适应地生成一个差分方案：为了捕捉物理场的局部特征，超网络会为每个网格点生成一个Moment矩阵 $\boldsymbol{M}_{pq}$，实际上也就是为每一个网格点生成一个卷积核 $\boldsymbol{K}_{pq}$；

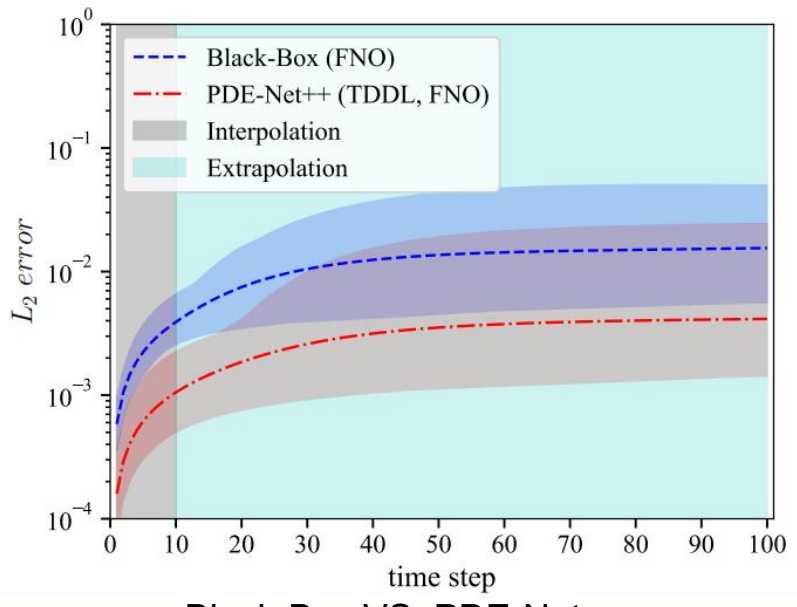


四、预测时空动力学系统演化过程的通用数据机理融合方法

- **数值实验 —— Burgers 方程**

$$\frac{\partial \boldsymbol{U}}{\partial t} = -\boldsymbol{U}\cdot\nabla\boldsymbol{U} + \nu\Delta\boldsymbol{U} + \boldsymbol{f}(x,y,\boldsymbol{U}),\quad \boldsymbol{U}=(u,v)^{\boldsymbol{T}}$$
$$\boldsymbol{U}|_{\boldsymbol{t=0}} = \boldsymbol{U}_0(x,y).$$

- $\nu = 0.05,\quad (t,x,y)\in[0,1]\times[0,2\pi]^2$
- $\boldsymbol{f}(x,y,\boldsymbol{U}) = (sin(v)cos(5x+5y), sin(u)cos(5x-5y))^T$
- $\boldsymbol{\mathcal{L}}_{\text{known}}(\boldsymbol{x},\boldsymbol{U}) = \boldsymbol{\Phi}(x,y,\boldsymbol{U},\nabla\boldsymbol{U},\nabla^2\boldsymbol{U},\cdots) = -\boldsymbol{U}\cdot\nabla\boldsymbol{U} + \nu\Delta\boldsymbol{U}$
- $\boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x},\boldsymbol{U}) = \boldsymbol{f}(x,y,\boldsymbol{U})$



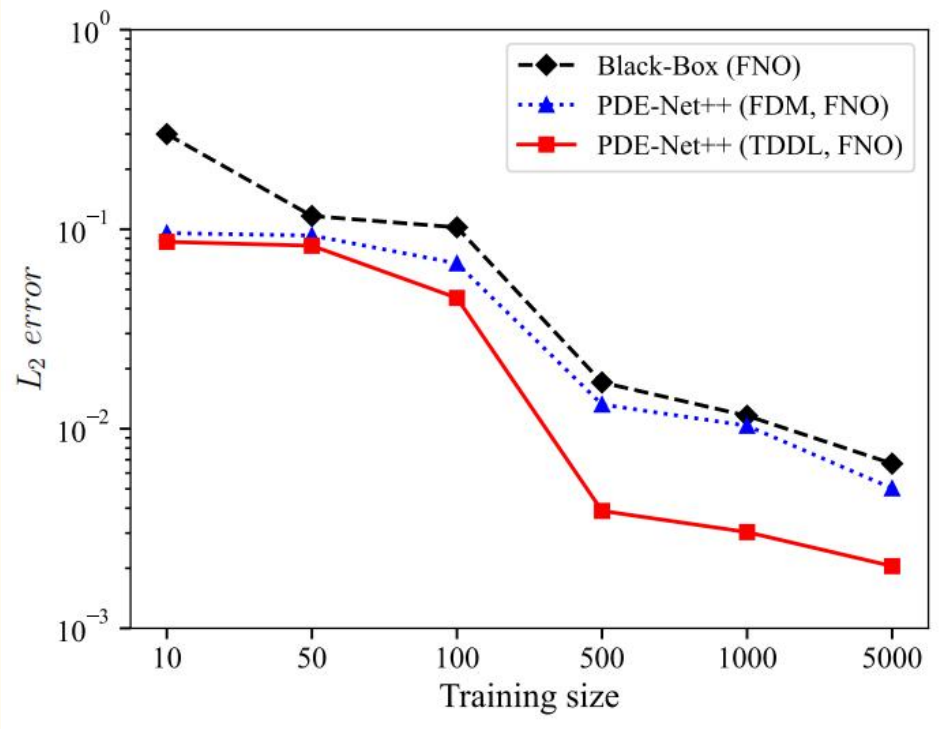
Black-Box VS. PDE-Net++

Backbone	Method	Derivatives	*SR*	L_2 *error*
U-Net	Black-Box	-	100%	9.718e-2
	PDE-Net++	FDM	96%	7.624e-3
		Moment	100%	4.210e-3
		TFDL	100%	4.970e-3
		TDDL	100%	3.114e-3
ConvResNet	Black-Box	-	100%	1.050e-1
	PDE-Net++	FDM	96%	1.204e-2
		Moment	97%	6.151e-3
		TFDL	100%	7.767e-3
		TDDL	100%	4.481e-3
FNO	Black-Box	-	100%	1.162e-2
	PDE-Net++	FDM	98%	1.036e-2
		Moment	99%	4.345e-3
		TFDL	100%	4.664e-3
		TDDL	100%	3.034e-3
F-FNO	Black-Box	-	100%	2.064e-2
	PDE-Net++	FDM	96%	9.545e-3
		Moment	100%	4.529e-3
		TFDL	100%	4.456e-3
		TDDL	100%	2.920e-3
Galerkin Transformer	Black-Box	-	100%	8.919e-2
	PDE-Net++	FDM	96%	9.316e-3
		Moment	100%	5.473e-3
		TFDL	100%	5.307e-3
		TDDL	100%	5.755e-3

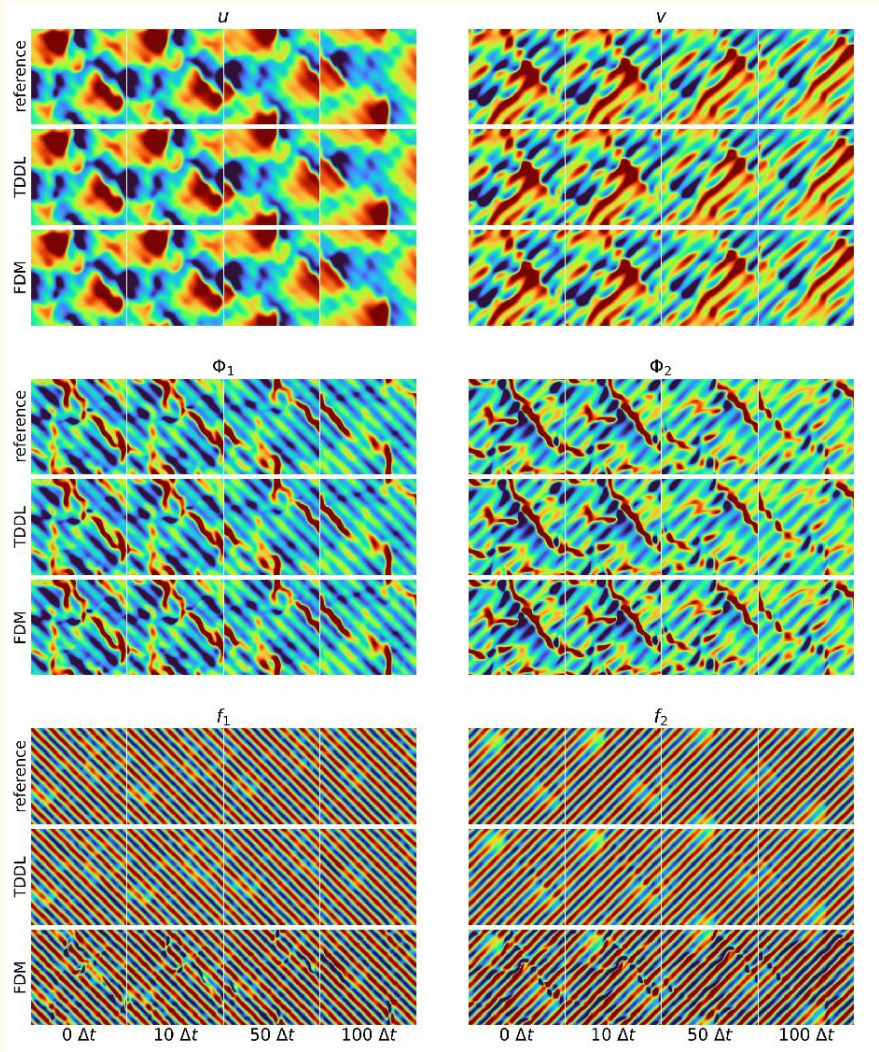
不同 Backbone 下 Black-Box 和 PDE-Net++ 的对比

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **数值实验 —— Burgers 方程**



不同训练样本数目下不同模型的比较



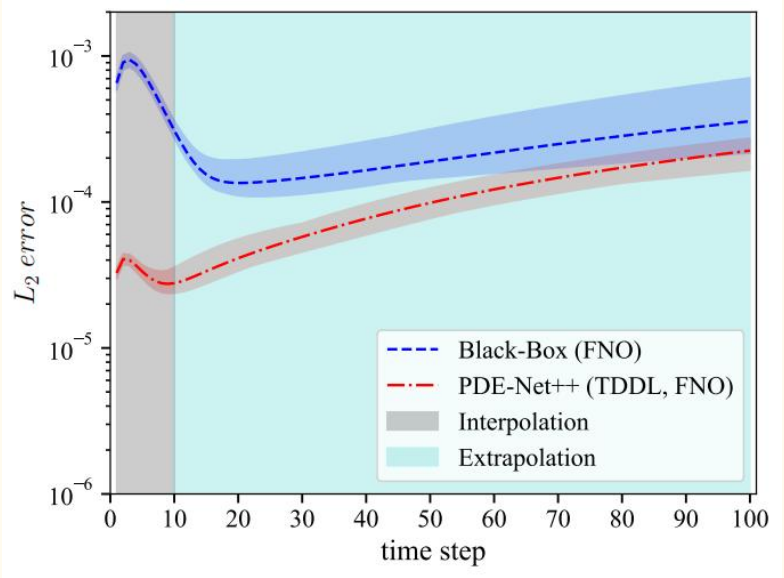
参考解 VS. TDDL VS. FDM

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **数值实验 —— FitzHugh–Nagumo 对流扩散方程**

$$\frac{\partial \boldsymbol{U}}{\partial t} = \gamma \Delta \boldsymbol{U} + \boldsymbol{R}(\boldsymbol{U}), \quad \boldsymbol{U} = (u, v)^{\boldsymbol{T}}$$
$$\boldsymbol{U}|_{\boldsymbol{t}=\boldsymbol{0}} = \boldsymbol{U}_0(x, y).$$

- $\gamma = 1.0, \quad (t, x, y) \in [0, T] \times [0, 6.4]^2$
- $\boldsymbol{R}(\boldsymbol{U}) = (u - u^3 - v + \alpha, \beta(u - v))^T, \ \alpha = 0.01, \beta = 0.25$
- $\boldsymbol{\mathcal{L}}_{\text{known}}(\boldsymbol{x}, \boldsymbol{U}) = \boldsymbol{\Phi}(x, y, \boldsymbol{U}, \nabla \boldsymbol{U}, \nabla^2 \boldsymbol{U}, \cdots) = \gamma \Delta \boldsymbol{U}$
- $\boldsymbol{\mathcal{L}}_{\text{unknown}}(\boldsymbol{x}, \boldsymbol{U}) = \boldsymbol{R}(\boldsymbol{U})$



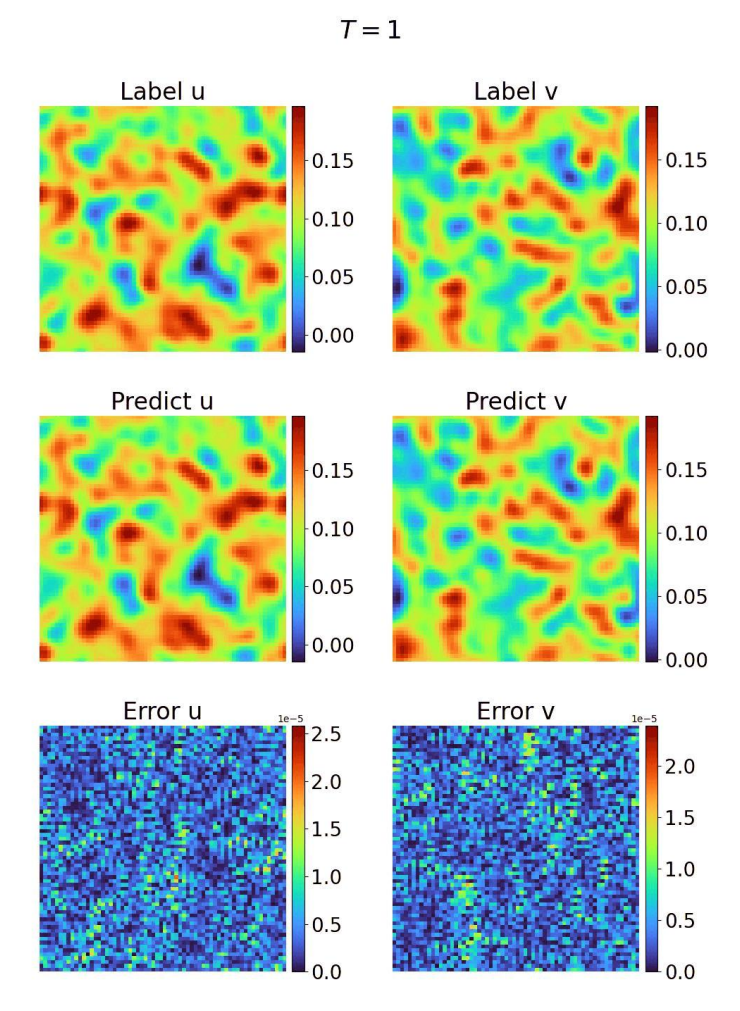
Black-Box VS. PDE-Net++

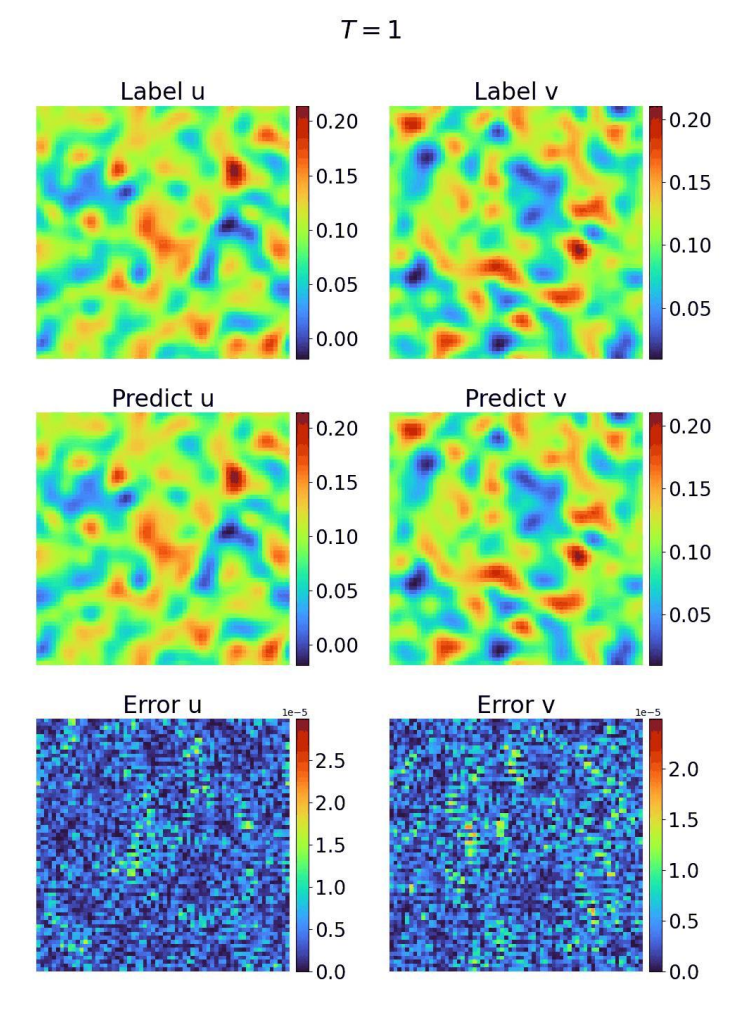
Backbone	Method	Derivatives	*SR*	L_2 *error*
U-Net	Black-Box	-	100%	2.103e-2
	PDE-Net++	FDM	100%	5.125e-4
		Moment	100%	3.373e-4
		TDDL	100%	2.080e-4
ConvResNet	Black-Box	-	100%	1.852e-2
	PDE-Net++	FDM	100%	1.893e-4
		Moment	100%	1.232e-4
		TDDL	100%	1.023e-4
FNO	Black-Box	-	100%	2.586e-4
	PDE-Net++	FDM	100%	8.200e-5
		Moment	100%	6.475e-5
		TDDL	100%	7.118e-5
F-FNO	Black-Box	-	100%	5.489e-3
	PDE-Net++	FDM	100%	6.305e-5
		Moment	100%	6.357e-5
		TDDL	100%	6.230e-5
Galerkin Transformer	Black-Box	-	100%	1.708e-2
	PDE-Net++	FDM	100%	1.133e-3
		Moment	100%	4.101e-4
		TDDL	100%	1.249e-3

不同 Backbone 下 Black-Box 和 PDE-Net++ 的对比

四、预测时空动力学系统演化过程的通用数据机理融合方法

- **数值实验 —— FitzHugh–Nagumo 对流扩散方程**





四、预测时空动力学系统演化过程的通用数据机理融合方法

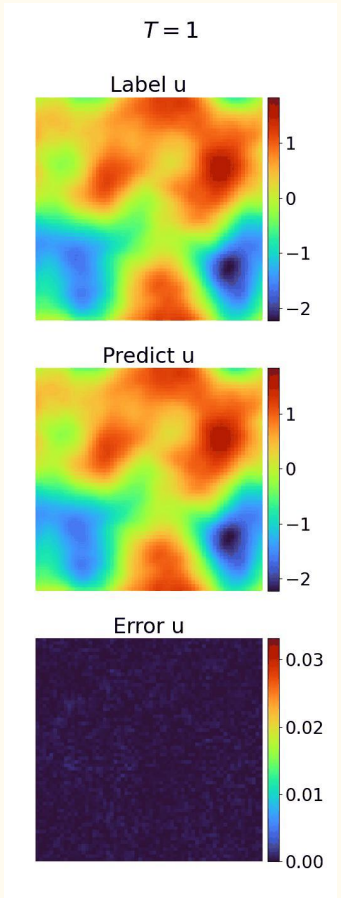
- **数值实验 —— 纳维-斯托克斯方程**

$$\frac{\partial \omega}{\partial t} = -\boldsymbol{U}\cdot\nabla\omega + \nu\Delta\omega + f(x,y),\quad (t,x,y)\in[T_1,T_2]\times[0,1]^2,$$
$$\nabla\cdot\boldsymbol{U}=0,\quad (t,x,y)\in[0,T]\times[0,1]^2,$$
$$\omega(0,x,y)=\omega_0(x,y),\quad (x,y)\in[0,1]^2.$$

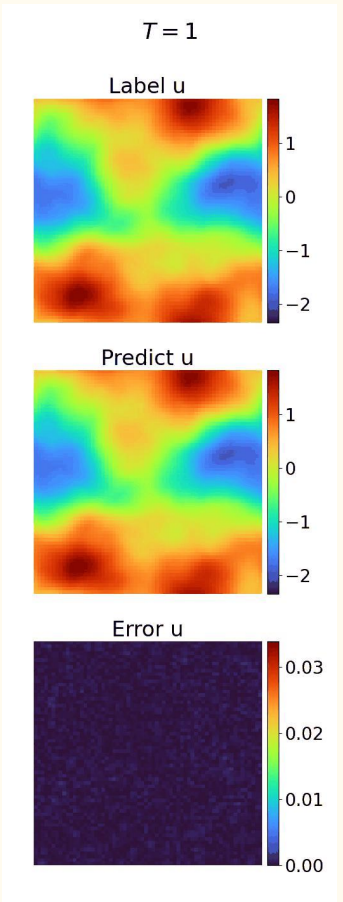
- $\boldsymbol{U}=(u,v)^T,\quad \omega=\nabla\times\boldsymbol{U}=\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}$
- $\nu=0.0001,\quad T_1=10,\quad T_2=30$
- $f(x,y)\sim\mathcal{N}(0{,}25(-\Delta+25)^{-3})$
- $\mathcal{L}_{\text{known}}(\boldsymbol{x},\omega)=\Phi(x,y,\omega,\nabla\omega,\nabla^2\omega,\cdots)=-\boldsymbol{U}\cdot\nabla\omega+\nu\Delta\omega$
- $\mathcal{L}_{\text{unknown}}(\boldsymbol{x},\omega)=f(x,y)$

Backbone	Method	Derivatives	*SR*	L_2 *error*
FNO	Black-Box	-	100%	1.345e-2
	PDE-Net++	FDM	63%	1.079e-2
		Moment	97%	1.001e-2
		TFDL	100%	9.478e-3
		TDDL	100%	7.194e-3

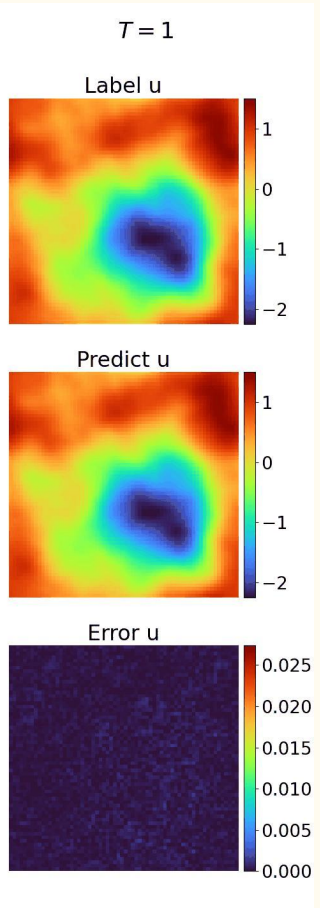
Black-Box VS. PDE-Net++



$\omega(t,x,y)$



$\omega(t,x,y)$



$\omega(t,x,y)$

目录 Contents

五、总结与展望

- **总结**

☑ 改进PINNs方法求解带点源 PDE

★ 利用对称单峰概率密度函数近似狄拉克函数，消除点源奇异性问题

★ 提出下界受限的不确定性加权方法，平衡多个损失函数项

★ 提出结合 Sine 函数的多尺度神经网络架构，增强神经网络的表达能力

☑ 基于元学习思想高效求解参数化 PDE

★ 提出预训练和微调相结合的参数化 PDE 求解方法 MAD，减小模型重训练的开销

★ 从流形学习的角度解释方法的工作原理

☑ 预测时空动力学系统演化过程的通用数据机理融合方法 PDE-Net++

★ 改进 PDE-Net，增加其处理复杂非线性项及未知项的能力

★ 结合有限差分方法和黑盒模型，将物理信息显式地嵌入神经网络架构

★ 提出两种新的可训练差分算子（TFDL 和 TDDL），提高长时序列预测的数值稳定性

五、总结与展望

- **进一步的工作**

☑ 对于新的方程实例，我们提出的 MAD 方法仍需进行微调。如何在完全避免重训或微调的同时保证较高的求解精度?

☑ 因为目前深度学习的可解释性较弱，所以它难以进行误差分析。因此，基于深度学习方法求解PDE如何进行误差分析?

☑ 针对多物理场、多尺度下的复杂 PDE，如何用深度学习方法进行高效求解?

欢迎感兴趣的小伙伴们加入

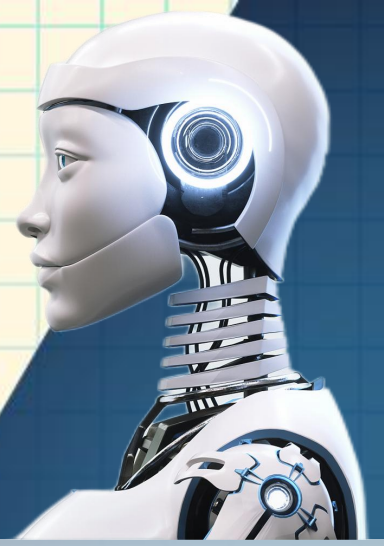
- **MindFlow SIG**
  - ☑ **愿景**：MindFlow SIG将致力于AI流体仿真端到端全覆盖，将以学界、业界的不同流体仿真需求牵引MindFlow功能的进一步开发和完善
  - ☑ **组织活动**：组织一场大型活动与数场小型活动，且固定在每个季度组织一场校园行活动；面向广大开发者举行MindFlow公开课程，与开发者共同学习交流
  - ☑ **暑期学校**：邀请组内的核心专家老师准备多个主题进行多天的授课。采用线上线下相结合授课形式，面向全国的高校师生与开发者传授AI+科学计算领域知识
  - ☑ **开源实习**：发布开源实习任务以及众智任务吸引高校老师和学生持续投入（https://gitee.com/mindspore/community/issues/I55B5A）
- **代码链接**
  - ☑ **MindScience:** https://gitee.com/mindspore/mindscience
  - ☑ **MindFlow:** https://gitee.com/mindspore/mindscience/tree/master/MindFlow
  - ☑ **改进PINNs方法求解带点源的泊松方程**：https://gitee.com/mindspore/mindscience/tree/master/MindFlow/applications/physics_driven/poisson_point_source
  - ☑ **改进PINNs方法求解带点源的麦克斯韦方程组**：https://gitee.com/mindspore/mindscience/tree/master/MindElec/examples/physics_driven/time_domain_maxwell
  - ☑ **MAD求解方程系数可变的麦克斯韦方程组**：https://gitee.com/mindspore/mindscience/tree/master/MindElec/examples/physics_driven/incremental_learning

欢迎感兴趣的小伙伴们加入

MindSpore Flow SIG
Gitee代码仓

MindSpore Flow SIG
开源实习任务列表

MindSpore Flow SIG
微信群



谢谢! Q&A

报告人: 黄翔

MindSpore