高精度可训练的 PDE 算子：AI 参数数量存在唯一性的数学证明与实证

High-Precision Trainable PDE Operators: Mathematical Proof and Empirical Validation of Uniqueness in AI Parameter Count

真理是精确的。误差不能变成智慧，只能造成幻觉。

湘潭大学 信息与计算科学学院

杜秋实

中文预印本 V1，待进一步完善格式，实验分析

摘要 (Abstract)

1. 本文在数学上证明了，为消除分布外泛化(OOD)幻觉，AI 参数量的选取必须严格等于在训练集下 Galerkin 投影的非线性基底数，即严格遵循等式 $N_{AI} = \dim(V_h) = N_{basis}$（其中 V_h 是 Galerkin 投影的有限维子空间，N_{basis} 是该空间内的非线性基底数）。为了方便行文论述，我们将此等式命名为**自由度同构等式**。

2. 本文在数学上证明了，大于零的非平凡零空间维度，即 $\dim(Null\ Space) > 0$，是 AI 在分布外泛化 OOD 中产生幻觉的**充要**条件。

3. 本文在数学上证明了 AI 幻觉不是“优化问题（*Optimization Bug*）”，而是“拓扑结构缺陷（*Topological structural defects*）”，**无法**通过工程手段彻底消除幻觉(本文 *1.3.6*)。

4. 传统机器学习难以训练积分，本质上是难以使用简单参数量和训练集，对非线性函数进行有效训练。因此本文的全部实证都基于经典的**非线性**函数：**高斯钟形曲线、泰勒格林-涡以及 Q4 双线性形函数**。

在以上实验中，基于自由度同构等式的 *AI(参数量O(1))*实现了 $O(10^{-32})$ 的 *OOD* 泛化均方误差(*MSE Loss*)，触及 *FP64* 双精度浮点格式的精度极限(*MSE Loss*:$(10^{-16})^2$)，仅需**单个**训练集样本与最简单 *Adam* 迭代即可达成。相反，具有 $O(10^5)$ 参数数量的传统 *MLP* 对照组，在对比实验中泛化失败： $O(10^{+1})$到 $O(10^{-3})$。

5. 为了方便行文论述，本文将这种全新架构的高精度可训练的 *PDE* 算子命名为 ***Pure Science AI***

关键词：计算数学，*FEM*，*Galerkin* 投影，等参变换，秩-零化度定理，*AI4Science*

第一章 自由度匹配参数等式 $\dim(N) = \dim(N_{basis})$，即 *AI* 参数量存在唯一性的数学证明

本章证明思路如下：

1.1 Galerkin 投影——>在**局部**可通过**有限个非线性基底的线性组合**去拟合学习任务&可使用秩-零化度定理

1.2 等参变换与拉回——>*1.1* 的**局部性质变成全局性质**，即 *AI* 可通过有限个非线性基底的线性组合去拟合学习任务

1.3 由 *1.1* 得出的秩-零化度定理使用依据，由 *1.3.1*，*1.3.2*，*1.3.3*，*1.3.4* 得出证明**参数的存在唯一性**

- **1.1** 证明：状态非线性与参数非线性的数学解耦：通过 Galerkin 投影，实现有限参数量计算和使用**秩-零化度定理(Rank-Nullity Theorem)**在下文参与证明的数学依据：
    - Galerkin 投影：自然界的真实物理解$u(x,y)$存在于无限维 Sobolev 空间$\mathcal{V}$中，具有**可数无穷个**正交基函数$\{\Phi_i\}_{i=1}^{\infty}$。然而，在实际数值计算与算子学习中，我们通过 Galerkin 投影截断该无限维空间，将其降维至一个高度精细的**有限维**特征子空间$\mathcal{V}_h \subset \mathcal{V}$。

      假设物理系统是一个非线性偏微分方程：

      $$\mathcal{L}(u) = f \quad in\ \Omega$$

      其中 $\mathcal{L}$ 是包含高度非线性项（如对流项 $u \cdot \nabla u$）的微分算子。

      Step 1: 弱形式与测试函数 (The Weak Form)

      我们不在每一个点上强求方程成立，而是引入一个测试函数 $v \in V$，在全局积分下求内积：

      $$\int_{\Omega} \mathcal{L}(u) v \, dx = \int_{\Omega} f \, v \, dx$$

      Step 2: 伽辽金投影 (Galerkin Projection)

      我们将无限维空间 V 投影到有限维子空间 V_h。设 V_h 的非线性基底为 $\{\Phi_1, \dots, \Phi_N\}$。

      此时，我们将试探解 u_h 和测试函数 v_h 都展开为这组基底的线性组合：

      $$u_h(x) = \sum_{j=1}^{N} a_j \, \Phi_j(x), \quad v_h(x) = \Phi_i(x)$$

      代入弱形式，得到：

      $$\int_{\Omega} \mathcal{L}\left(\sum_{j=1}^{N} a_j \, \Phi_j(x)\right) \Phi_i(x) \, dx = \int_{\Omega} f \, \Phi_i(x) \, dx \quad for\ i = 1, \dots, N$$

      其离散近似解u_h可表示为一组有限数量的非线性物理基函数$\Phi_j(X)$的线性加权：

$$u_h(x,y) = \sum_{j=1}^{N} a_j \, \Phi_j(x,y)$$

状态非线性完全由基函数$\Phi_j(x,y)$（如高阶正交多项式或三角函数如$\sin(x)\cos(y)$）提供，这些基函数赋予 *PDE* 流形以“弯曲能力”。

参数非线性被完全解耦：网络仅需通过纯线性参数a_j组合这些物理基函数即可，此为伽辽金投影下的算子学习目标。

这种数学上的解耦确保了算子相对于参数空间 W 保持严格的线性映射关系，从而提供了使用**秩-零化度定理(Rank-Nullity Theorem)**进行证明的数学依据。

- **1.2** 证明：将局部性质推广到全局，*Pure Science AI* 所学习的特征映射具有全局不变性：
-
    - 我们引入**计算数学**中的等参变换和拉回操作进行证明：
    - **证明过程 (Proof):**

      ***Step 1*:建立不变的标准参考单元：**

      设定真实的全局计算域为 Ω，其由任意形状的畸变局部单元 Ω_e 构成，物理坐标记为 $\mathbf{x} = (x,y)$。

      引入一个不变的标准参考单元（*Reference Element*） $\widehat{\Omega}$（例如 $[-1,1] \times [-1,1]$），参考坐标记为 $\boldsymbol{\xi} = (\xi,\eta)$。

      存在一个微分同胚的几何映射 $F_e: \widehat{\Omega} \to \Omega_e$，使得：

      $\mathbf{x} = F_e(\boldsymbol{\xi}) = \sum_{k=1}^{N_{geo}} \mathbf{x}_k^{(e)} \, \widehat{\Psi}_k(\boldsymbol{\xi})$

      其中 $\mathbf{x}_k^{(e)}$ 是真实物理网格的节点坐标，$\widehat{\Psi}_k$ 是参考空间 $\widehat{\Omega}$上的几何形状函数。映射的雅可比矩阵（*Jacobian*）定义为：

$$J_e = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}$$

***Step 2*: 拉回**

根据等参变换原理，物理空间上的基底函数 $\Phi_j^{(e)}(\mathbf{x})$ 与参考空间上的标准基底 $\widehat{\Phi}_j(\boldsymbol{\xi})$ 是**同一个函数在不同坐标下的表示**：

$$\Phi_j^{(e)}(\mathbf{x}) \equiv \widehat{\Phi}_j(\boldsymbol{\xi}(\mathbf{x}))$$

因此，任意物理单元上的离散解 $u_h^{(e)}$，均可表达为同一组参考空间上标准基底$\widehat{\Phi}_j(\boldsymbol{\xi})$的线性加权：

$$u_h^{(e)}(\mathbf{x}) = \sum_{j=1}^{N} a_j^{(e)}\, \Phi_j^{(e)}(\mathbf{x}) = \sum_{j=1}^{N} a_j^{(e)}\, \widehat{\Phi}_j(\boldsymbol{\xi})$$

由此可得，无论真实网格 Ω_e 多么扭曲，在参考单元$\widehat{\Omega}$ 上，基底函数 $\widehat{\Phi}_j(\boldsymbol{\xi})$ 的解析形式、正交关系以及拓扑结构，是全域一致且与真实网格 Ω_e 几何无关的。

***Step 3*: 微分算子的几何解耦**

在 *PDE*（如对流扩散或泰勒涡）中，我们要计算基底的空间梯度（即微分算子）。根据链式法则引入雅可比矩阵（*Jacobian*），物理梯度的矩阵变换为：

$$\nabla_{\mathbf{x}} \Phi_j^{(e)}(\mathbf{x}) = J_e^{-T} \nabla_{\boldsymbol{\xi}} \widehat{\Phi}_j(\boldsymbol{\xi})$$

将其代入局部刚度矩阵的积分弱形式中：

$$A_{ij}^{(e)} = \int_{\Omega_e} \nabla_{\mathbf{x}} \Phi_i \cdot \nabla_{\mathbf{x}} \Phi_j \, d\mathbf{x} = \int_{\widehat{\Omega}} \left(J_e^{-T} \nabla_{\boldsymbol{\xi}} \widehat{\Phi}_i\right) \cdot \left(J_e^{-T} \nabla_{\boldsymbol{\xi}} \widehat{\Phi}_j\right) \det(J_e)\, d\boldsymbol{\xi}$$

***Step 4*: 结论：几何与拓扑的分离**

1. **几何畸变项（*PDE* 的表象）：** J_e^{-T} 和 $\det(J_e)$ 包含了所有的网格扭曲、坐标平移和非均匀性。这些是任意坐标空间提供的确定性张量，**不需要 *AI* 去学习**
2. **拓扑不变项（*AI* 的唯一学习对象）：** $\nabla_{\boldsymbol{\xi}} \widehat{\Phi}_i$ 和 $\nabla_{\boldsymbol{\xi}} \widehat{\Phi}_j$。这是在参考空间中，一组**固定**的**线性相关**的**非线性基底**所具有的。

3. **参数空间：** 因此，系数向量 $\mathbf{a} = [a_1, a_2, ..., a_N]^T$ 的非线性映射法则（即 *Pure Science AI* 学习的目标 $\mathcal{T}_{AI}(\mathbf{a})$），其数学形式**不再显式依赖于物理坐标 *x***，而是完全定义在**与几何解耦的参考空间系数域**上。

- **结论 (*Q.E.D.*):局部即是全局，单点即是世界**
- **1. 传统大模型在 x 空间(真实的全局计算域)拟合，因此无法拟合 J_e 产生的无限数量的变化。*Pure Science AI* 在 ξ 空间（参考单元内）直接对线性系数 a_j 建立算子的线性映射，从而在数学底层规避全局坐标畸变与介质非均匀性。局部即全局，此即全局不变性之铁证。**
- **2.这意味着，AI 仅需要一组有限个非线性基底的系数作为可训练参量(假设为 4 个)，组装成 $U(x,y) = a_1\Phi_1(x,y) + a_2\Phi_2(x,y) + a_3\Phi_3(x,y) + a_4\Phi_4(x,y)$，仅对非线性基底的系数做线性映射，即可完成对全局非线性函数的拟合（第二章实验的根基之一）**
- **3.这意味着，AI 仅需要一个参数作为训练集，完成对局部的训练，即可完成对全局非线性函数的拟合（第二章实验的根基之一）**

- 1.3 证明：使用秩-零化度定理证明 $\dim(N) = \dim(N_{\text{basis}})$
  - **1.3.1：证明** $\dim(N) = \dim(Image) + \dim(Null\ Space)$
    - 由 1.2 的结论 2：AI 仅需对非线性基底的系数做线性映射。对于任何一个线性映射 $T: \mathbb{R}^n \to \mathbb{R}^m$，有秩-零化度定理（*Rank-Nullity Theorem*）：定义域V的维数等于核空间$Ker(T)$的维数与值域$Ran(T)$的维数之和，

      即$dim(V) = \dim(Ker(T)) + dim(Ran(T))$

      AI 训练可视为训练**从输入数据到输出数据的映射** $f: X \to Y$。在物理场景中，给定输入观测数据 X_{data}，前向传播实际上构成了一个由**参数空间映射到物理观测空间**$Image$的映射$\mathcal{T}$：

$\mathcal{T}_{X_{data}}: \mathbb{R}^{\dim(N)} \to \mathbb{R}^{\dim(Image)}$

其中 $\mathcal{T}_{X_{data}}(W) = \Phi W$ ， $Y = \Phi(X_{data})W$，
W是AI参数构成的空间， Φ 是训练集下 Galerkin 投影的非线性基底$。

因为输入观测 X_{data} 与基底 Φ 在优化瞬间是固定的矩阵，那么该映射 $\mathcal{T}_{X_{data}}(W) = \Phi W$ 关于我们要优化的**参数空间** W 是严格的**线性映射（Linear Mapping）**

因此,我们就可以使用秩-零化度定理：

$$\dim(N) = \dim(Image) + \dim(Null\ Space)$$

物理意义： AI 的总参数量 $\dim(N)$，等于它能拟合出的有效物理特征维度 $\dim(Image)$，加上零空间维度 $\dim(Null\ Space)$。

- **1.3.2:我们给出幻觉的产生的数学证明、公式 $\dim(N) = Rank(Image)$的数学证明：**
  - 由线性代数，方程组 $Ax = b$ 的通解形式为：

    即（**最终参数 = 唯一正确匹配物理的特解 + 零空间里的任意向量**）

    对于具有庞大参数空间 $\mathbb{R}^n$ 的网络，设其最终训练学习到的参数向量为 $W \in \mathbb{R}^{\dim(N)}$。根据线性代数，该向量可被正交分解为真实物理特解 W_{true} 与零空间分量 W_{null}：

    $$W = W_{true} + W_{null}$$

    设训练集流场观测矩阵为 A_{train}，由于零空间的定义和隐式正则化，这无穷多个形态对训练集的损失函数（Loss）贡献恒为零：

    $$A_{train}W_{null} = \mathbf{0} \Longrightarrow Loss_{train}(W) = \| A_{train}W_{true} - Y_{true}\|^2$$

这意味着在训练阶段，W_{null} 是**不可观测且不受任何约束的（Unobservable and Unconstrained）**。

当模型进入分布外泛化 OOD 测试集时，随着观测测度（Measure）发生偏移，测试矩阵 A_{test} 所张成的行空间（Row Space）不可避免地与训练矩阵 A_{train} 的行空间发生**几何偏转**。这意味着隐藏在训练集正交补空间 $Ker(A_{train})$ 中的冗余分量 W_{null}，与 A_{test} 的行向量组不再保持正交。因此，W_{null}不再被 A_{test} 的核空间所包容：

即$W_{null} \notin Ker(A_{test}) \Longrightarrow A_{test}W_{null} \neq \mathbf{0}$

网络在测试集上的输出为：

$$Y_{pred} = A_{test}W = A_{test}W_{true} + A_{test}W_{null}$$

其中，误差项 $E_{hal} = A_{test}W_{null}$ 即为幻觉。因为 W_{null} 在训练时未受任何有界性约束，$\| E_{hal} \|$ 发散。

因此，非平凡零空间的存在，从数学上充要地决定了 OOD 测试集上幻觉的必然爆发。

为了从根本上消灭 E_{hal}，我们**必须且只能**施加强制性的数学约束，避免幻觉产生的可能性：

即 $\dim(Null\ Space) \equiv 0$

代入 $\dim(N) = Rank(Image) + \dim(Null\ Space)$

得到 $\dim(N) = Rank(Image)$

- **1.3.3:证明：$Rank(Image) \geq \dim(N_{basis})$:**
  - **反证法（假设发生欠拟合）：**

    假设 $Rank(Image) < \dim(N_{basis})$，即 $r < m$。

根据线性代数的基础子空间理论，映射矩阵 A 的列空间 $C(A)$ 仅仅是 m 维物理宇宙 $\mathbb{R}^m$ 中的一个 r 维子空间（一个低维的超平面）。

由于 $r < m$，必然存在属于大自然真实物理流形、但正交于 AI 列空间的维度。当大自然给出一个包含了这些维度的复杂物理状态 b 时，目标向量 b 不在矩阵 A 的列空间内，即：

$b \notin C(A)$

此时，方程 $Ax = b$ **无解（Inconsistent N）**。

因此，$Rank(Image) \geq \dim(N_{basis})$

- **1.3.4:证明：** $Rank(Image) \leq \dim(N_{basis})$**:**
  - AI 的最终输出，由 1.3.1 得出，是被表达为非线性基底的线性组合的：

    $U_h(\mathbf{x}) = \sum_{j=1}^{N_{basis}} a_j \, \Phi_j(\mathbf{x})$

    1. **母空间（The Codomain）：** 根据 Galerkin 投影原理，所有的基底函数 $\{\Phi_1, \Phi_2, ..., \Phi_{N_{basis}}\}$ 张成了一个有限维的试探子空间（Trial Space），我们记为 V_h。

       根据线性代数基底的定义，这个空间的维度是确定的：$\dim(V_h) \equiv \dim(N_{basis})$。

    2. **像空间（The Image）：** AI 无论怎么变幻参数生成的预测场，都绝对且只能是这 N_{basis} 个基底的某种组合。这意味着，AI 映射的**像空间（Image）**，是试探空间 V_h 的一个**子集（Subset）或子空间（Subspace）**。

       用数学符号表示极其冰冷：$Image(AI) \subseteq V_h$。

3. **子空间维度定理：** 任何有限维向量空间的子空间，其维度绝对不可能超过母空间。

   因此，我们得出了这个绝对不等式：

   $Rank(Image) = \dim(Image) \leq \dim(V_h) \equiv \dim(N_{basis})$

- **1.3.5：证明** $\dim(N) = \dim(N_{basis})$
  - 由 1.3.1，1.3.2，1.3.3，1.3.4 得，$\dim(N) = \dim(N_{basis})$

  命题得证，$\dim(N) = \dim(N_{basis})$，即 AI 参数量具有存在唯一性

- **1.3.6：推论：AI 幻觉是拓扑结构缺陷，不是优化问题**：
  - 任意 AI 训练方法，其每次参数矩阵N权重更新，依赖于训练数据的观测矩阵X_{data}的行空间(Row Space)。而零空间(Null Space)与行空间是正交补的。
  - 非平凡零空间的存在是 AI 幻觉的充要条件（1.3.2）
  - 假设零空间中存在任意一个非零向量 $V_{null} \in N(A_{train})$。

    即 $A_{train}V_{null} = \mathbf{0}$。

    我们在参数空间中，沿着这个零空间方向移动一段距离 ϵ。：

    $L(W + \epsilon V_{null}) = f(A_{train}(W + \epsilon V_{null}))$

    $= f(A_{train}W + \epsilon A_{train}V_{null})$

    $= f(A_{train}W + \mathbf{0})$

    $= L(W)$

因此，AI 幻觉无法通过训练手段规避

第二章 经典非线性物理流形的零幻觉算子逼近与对比实验

- **2.1** 实验设计：
    - **函数选取：**传统机器学习难以训练积分，本质上是难以使用简单参数量和训练集，对非线性函数进行有效训练。因此本文的全部实证都基于经典的**非线性**函数：***Q4* 双线性形函数、*2D* 泰勒-格林涡 *TGV* 和高斯钟形曲线**作为实验函数。

      其中：

        - ***Q4* 双线性形函数：** 这是有限元方法（*FEM*）的经典测试样例，以便验证 *Pure Science AI* 架构能完美复现前文基于有限元理论推导出的判断（*1.2* 节结论 *2*，*3*），即实现零幻觉的 *OOD* 泛化测试。
        - ***2D* 泰勒-格林涡 *TGV*：** 这是 *Navier-Stokes* 方程在宏观尺度下的经典非线性全局解，在此测试样例下，*Pure Science AI* 架构同样能实现零幻觉的 *OOD* 泛化测试。
        - **高斯钟形曲线：** 高斯曲线是热传导方程（扩散 *PDE*）的基础解，也是自然界最普遍的格林函数（*Green's Function*）形态。*Pure Science AI* 架构同样能精准捕捉其能量衰减的物理本质，即实现零幻觉的 *OOD* 泛化测试。

    - **训练集和测试集**：由第一章 *1.2* 的结论 *2* 和 *3*，我们选取实验函数解析解的单一点作为训练集，显然其他任意一点作为测试集，都属于 *OOD* 泛化测试

- **主体结构**：*Pure Science AI* 在程序中称为 *White Box AI*。作为对照组，$O(10^3)$ 至$O(10^4)$参数量的传统 *MLP* 神经网络在程序中被成为 *Black Box AI*。训练接口和输入输出上，黑盒白盒 *ai* 保持完全一致。区别在于，（例如在 *2D* 泰勒-格林涡 *TGV* 实验中，自由度同构等式要求 *AI* 的可训练参数数是 *4*），黑盒 *ai* 的 *4* 参数是通过 ***4* 头独立的 *mlp* 神经网络分别直接对应 *4* 个非线性基底的系数**，而白盒 *ai* 的 ***4* 参数分别直接对应 *4* 个非线性基底的系数**，即组成了严格对照实验。

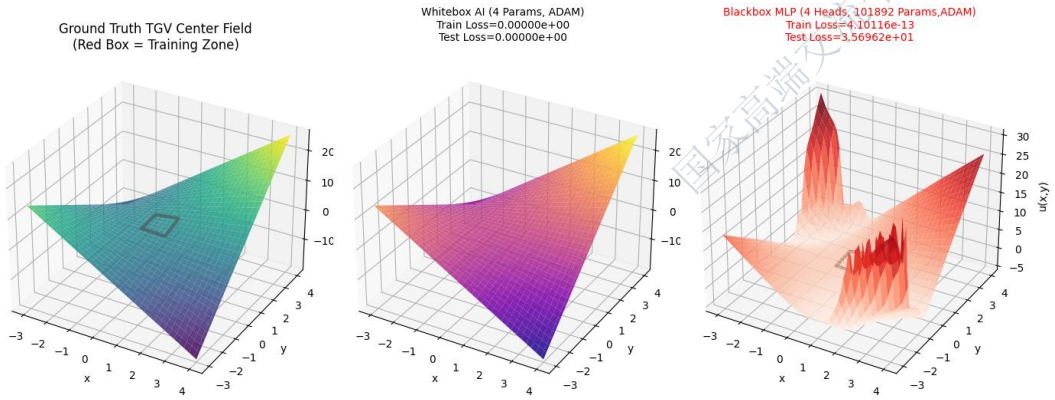
- **2.2** 极高训练精度与 *OOD* 泛化：*Train Loss=OOD Loss* 的完全一致性验证

  Q4 双线性形函数

  - 选取 *Q4* 双线性形函数的四个非线性基底：

    $\Phi1=1/4*(1-\xi)*(1-\eta)$, $\Phi2=1/4*(1+\xi)*(1-\eta)$, $\Phi3=1/4*(1+\xi)*(1+\eta)$, $\Phi4=1/4*(1-\xi)*(1+\eta)$
    即解析解系数为 `TRUE_PARAMS = torch.tensor([2.5, -1.2, 3.0, 0.8])`

    使用 *adam* 迭代 *10000* 次，结果如下：



  -

一图胜千言，*Pure Science AI* 表现出了完美的泛化能力，而同思路的黑盒 *MLP* 仅仅是参数量的不同，就造成了泛化灾难：

以下是本程序的输出数据：

```
train set training+hallucination test for areas outside the training re
gion : both Adam,10000 Epochs...
Epoch 500 | Blackbox Loss: 2.49813772e-23 | Whitebox Loss: 3.05895039e-
01
Epoch 1000 | Blackbox Loss: 1.72905711e-31 | Whitebox Loss: 4.63515359e
-04
Epoch 1500 | Blackbox Loss: 3.19535231e-28 | Whitebox Loss: 1.99171350e
-08
Epoch 2000 | Blackbox Loss: 4.72782302e-23 | Whitebox Loss: 9.13440280e
-15
Epoch 2500 | Blackbox Loss: 1.42531721e-14 | Whitebox Loss: 7.01882625e
-24
Epoch 3000 | Blackbox Loss: 1.95261943e-03 | Whitebox Loss: 2.78884928e
-29
Epoch 3500 | Blackbox Loss: 1.17769582e-26 | Whitebox Loss: 2.63569933e
-29
Epoch 4000 | Blackbox Loss: 4.31454368e-17 | Whitebox Loss: 1.80660788e
-29
Epoch 4500 | Blackbox Loss: 1.06583496e-16 | Whitebox Loss: 9.31978899e
-30
Epoch 5000 | Blackbox Loss: 4.91099367e-15 | Whitebox Loss: 6.53309676e
-30
Epoch 5500 | Blackbox Loss: 1.04405685e-15 | Whitebox Loss: 3.10613981e
-30
Epoch 6000 | Blackbox Loss: 4.32357819e-13 | Whitebox Loss: 2.01186922e
-30
Epoch 6500 | Blackbox Loss: 1.12955211e-15 | Whitebox Loss: 1.54074396e
-30
Epoch 7000 | Blackbox Loss: 2.84676999e-15 | Whitebox Loss: 7.77904504e
-31
Epoch 7500 | Blackbox Loss: 1.57665345e-17 | Whitebox Loss: 2.24606230e
-31
Epoch 8000 | Blackbox Loss: 2.62147276e-17 | Whitebox Loss: 2.24606230e
-31
Epoch 8500 | Blackbox Loss: 6.43311960e-30 | Whitebox Loss: 2.24606230e
-31
Epoch 9000 | Blackbox Loss: 1.86303335e-29 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 1.71795432e-17 | Whitebox Loss: 0.00000000e
+00
Epoch 10000 | Blackbox Loss: 4.65801922e-14 | Whitebox Loss: 0.00000000
e+00


 OOD training set:Out-of-region test set
-> blackbox 101892paramsOOD Loss: 3.56961502e+01
-> whitebox 4params OOD Loss: 0.00000000e+00
```

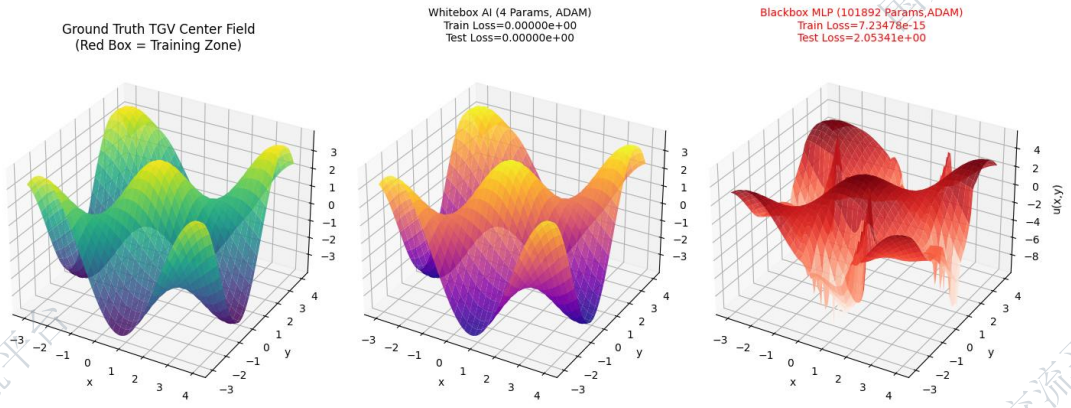
```
-> the parameters learned by the White Box AI: [   2.50000,   -1.20000,
   3.00000,    0.80000]
-> Taylor Green vortex analysis parameter:  [   2.50000,   -1.20000,
 3.00000,    0.80000]
```

值得注意的是，黑盒 *mlp* 虽然也取得了 *e-14* 的训练成绩，但是是典型的 *OOD* 严重幻觉

2D 泰勒-格林涡：

选取泰勒格林涡的四个非线性基底：

```
四个解析基底 M1=sin(x)*cos(y), M2=cos(x)*sin(y), M3=cos(x)*cos(y), M4=si
n(x)*sin(y)
解析解系数 TRUE_PARAMS = torch.tensor([2.5, -1.2, 3.0, 0.8])
```



如同第一个实验一样的结果：

```
train set training+hallucination test for areas outside the training re
gion : both Adam,10000 Epochs...
Epoch 500 | Blackbox Loss: 8.87819036e-24 | Whitebox Loss: 1.13881697e-
01
Epoch 1000 | Blackbox Loss: 1.20109551e-30 | Whitebox Loss: 1.94627264e
-04
Epoch 1500 | Blackbox Loss: 1.19013911e-30 | Whitebox Loss: 1.04116655e
-08
Epoch 2000 | Blackbox Loss: 9.87788069e-31 | Whitebox Loss: 6.69809033e
-15
Epoch 2500 | Blackbox Loss: 1.61122566e-18 | Whitebox Loss: 8.84806375e
-24
Epoch 3000 | Blackbox Loss: 3.12610124e-23 | Whitebox Loss: 1.48157939e
```

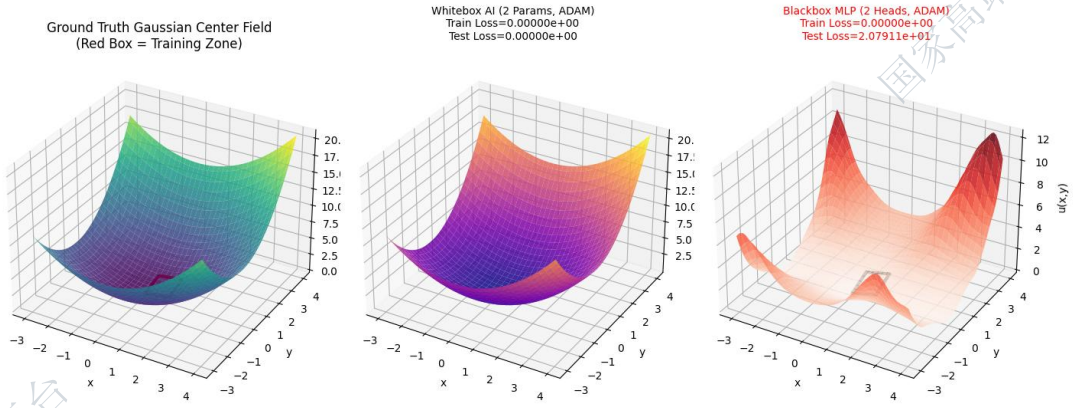
```
-29
Epoch 3500 | Blackbox Loss: 4.82665189e-16 | Whitebox Loss: 1.28874672e
-29
Epoch 4000 | Blackbox Loss: 5.61856309e-12 | Whitebox Loss: 7.93277705e
-30
Epoch 4500 | Blackbox Loss: 2.58827041e-09 | Whitebox Loss: 4.98687460e
-30
Epoch 5000 | Blackbox Loss: 8.03242668e-07 | Whitebox Loss: 2.53777649e
-30
Epoch 5500 | Blackbox Loss: 3.06394637e-06 | Whitebox Loss: 1.39283254e
-30
Epoch 6000 | Blackbox Loss: 2.45072315e-12 | Whitebox Loss: 1.17370451e
-30
Epoch 6500 | Blackbox Loss: 7.91082683e-17 | Whitebox Loss: 4.65647062e
-31
Epoch 7000 | Blackbox Loss: 5.20107335e-10 | Whitebox Loss: 2.84866438e
-31
Epoch 7500 | Blackbox Loss: 2.81007061e-22 | Whitebox Loss: 2.69801386e
-31
Epoch 8000 | Blackbox Loss: 2.34466991e-30 | Whitebox Loss: 1.15042215e
-31
Epoch 8500 | Blackbox Loss: 1.79169144e-08 | Whitebox Loss: 6.57384088e
-32
Epoch 9000 | Blackbox Loss: 4.57072058e-10 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 3.61324067e-13 | Whitebox Loss: 0.00000000e
+00
Epoch 10000 | Blackbox Loss: 5.41881218e-15 | Whitebox Loss: 0.00000000
e+00


 OOD training set:Out-of-region test set
-> blackbox 101892params OOD Loss: 2.05341280e+00
-> whitebox 4params OOD Loss: 0.00000000e+00


-> the parameters learned by the White Box AI: [   2.50000,   -1.20000,
    3.00000,    0.80000]
-> Taylor Green vortex analysis parameter:  [   2.50000,   -1.20000,
 3.00000,    0.80000]
```

高斯钟形曲线：

选取 2 个非线性基底：

```
# 两个非线性基底 M1 = X_grid**2，M2 = Y_grid**2
值得注意的是，解析式 y=exp（a1*M1+a2*M2）
a1，a2 的解析解系数为 TRUE_PARAMS = torch.tensor([0.5, 0.8])
```

这次实验结果更惊人，黑盒抢先训练到 *0.0000000e+00* 的机器极限经典，但仍然是泛化灾难，白盒依然证明无幻觉：



程序数据：

```
train set training+hallucination test for areas outside the training re
gion : both Adam,10000 Epochs...
Epoch 500 | Blackbox Loss: 1.61281190e-23 | Whitebox Loss: 7.16456170e-
02
Epoch 1000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.93928391e
-04
Epoch 1500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.01290169e
-06
Epoch 2000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.15350625e
-09
Epoch 2500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 4.86012800e
-13
Epoch 3000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25247896e
-18
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 500 | Blackbox Loss: 1.61281190e-23 | Whitebox Loss: 7.16456170e-
02
Epoch 1000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.93928391e
-04
Epoch 1500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.01290169e
-06
Epoch 2000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.15350625e
-09
Epoch 2500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 4.86012800e
-13
Epoch 3000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25247896e
-18
```

```
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 1000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.93928391e
-04
Epoch 1500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.01290169e
-06
Epoch 2000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.15350625e
-09
Epoch 2500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 4.86012800e
-13
Epoch 3000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25247896e
-18
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 1500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.01290169e
-06
Epoch 2000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.15350625e
-09
Epoch 2500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 4.86012800e
-13
Epoch 3000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25247896e
-18
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 2500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 4.86012800e
-13
Epoch 3000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25247896e
-18
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 4500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 5.99589070e
-30
Epoch 5000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 3.57178688e
-30
Epoch 5500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.09267268e
-30
Epoch 3500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.67525430e
-26
Epoch 4000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 6.76386596e
-30
Epoch 4500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 5.99589070e
-30
```

```
Epoch 5000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 3.57178688e
-30
Epoch 5500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.09267268e
-30
Epoch 4500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 5.99589070e
-30
Epoch 5000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 3.57178688e
-30
Epoch 5500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.09267268e
-30
Epoch 6000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25245364e
-30
Epoch 5500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.09267268e
-30
Epoch 6000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25245364e
-30
Epoch 6000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.25245364e
-30
Epoch 6500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.32024573e
-31
Epoch 7000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.47203808e
-31
Epoch 7500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.70508998e
-31
Epoch 8000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.67085122e
-31
Epoch 6500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.32024573e
-31
Epoch 7000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 2.47203808e
-31
Epoch 7500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.70508998e
-31
Epoch 8000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.67085122e
-31
Epoch 7500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.70508998e
-31
Epoch 8000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.67085122e
-31
Epoch 8500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.19013846e
-32
Epoch 8000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 1.67085122e
-31
Epoch 8500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.19013846e
-32
Epoch 9000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 8500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 7.19013846e
-32
```

```
Epoch 9000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 9000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 9500 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000e
+00
Epoch 10000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000
e+00
Epoch 10000 | Blackbox Loss: 0.00000000e+00 | Whitebox Loss: 0.00000000
e+00


 OOD training set:Out-of-region test set
-> blackbox 2heads OOD Loss: 2.07910732e+01
-> whitebox 2params OOD Loss: 0.00000000e+00


-> the parameters learned by the White Box AI: [0.5, 0.8]
```

综上，程序实证了前文证明

声明：

1.论文第一版中文预印本工作全部由本人独立完成，无任何人指导，本人只是湘潭大学信息与计算科学的大四本科生，更深层次的数据分析、理论进一步推导，纠错，仅以我个人能力已经无法继续进行，欢迎读者与我邮箱联系，进行学术交流。

2.本文程序大量使用 *vibe coding*，已经进行人工校对，受限于个人学术水平，如有问题请见谅。

附件：2D 泰勒格林涡流程序，读者可直接复现（其他两个简单修改一下解析解和白、黑盒的参数即可）

```
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np


# 锁定高精度最大精度e-16,后续MSE Loss 做开根运算会把下限打穿,变成e-30 的工程学奇迹
torch.set_default_dtype(torch.float64)
torch.set_printoptions(precision=8, sci_mode=False)


# 宇宙真实参数（4 参数）
TRUE_PARAMS = torch.tensor([2.5, -1.2, 3.0, 0.8])
#即泰勒格林涡流参数:u(x,y)=w1*sin(x)*cos(y)+w2*cos(x)*sin(y)+w3*cos(x)*cos(y)+w4*sin(x)*sin(y)
# 4 个解析基底：M1=sin(x)*cos(y), M2=cos(x)*sin(y), M3=cos(x)*cos(y), M4=sin(x)*sin(y)
dx = torch.tensor([-1., 0., 1., -1., 0., 1., -1., 0., 1.])
dy = torch.tensor([ 1., 1., 1.,  0., 0., 0., -1., -1., -1.])


def build_features_from_centers(x0, y0):
    """
    x0, y0: 形状 (N, 1)
    返回 features: (N, 4, 9)
    """
    X_grid = x0 + dx.unsqueeze(0)
    Y_grid = y0 + dy.unsqueeze(0)


    # 四个解析基底 M1=sin(x)*cos(y), M2=cos(x)*sin(y), M3=cos(x)*cos(y), M4=sin(x)*sin(y)
    M1 = torch.sin(X_grid) * torch.cos(Y_grid)
    M2 = torch.cos(X_grid) * torch.sin(Y_grid)
    M3 = torch.cos(X_grid) * torch.cos(Y_grid)
    M4 = torch.sin(X_grid) * torch.sin(Y_grid)


    features = torch.stack([M1, M2, M3, M4], dim=1)
    return features


def generate_data(num_samples, x_range, y_range):
    X0 = torch.rand(num_samples, 1) * (x_range[1] - x_range[0]) + x_range[0]
```

```
    Y0 = torch.rand(num_samples, 1) * (y_range[1] - y_range[0]) + y_ran
ge[0]


    X_grid = X0 + dx.unsqueeze(0)
    Y_grid = Y0 + dy.unsqueeze(0)


    M1 = torch.sin(X_grid) * torch.cos(Y_grid)
    M2 = torch.cos(X_grid) * torch.sin(Y_grid)
    M3 = torch.cos(X_grid) * torch.cos(Y_grid)
    M4 = torch.sin(X_grid) * torch.sin(Y_grid)


    features = torch.stack([M1, M2, M3, M4], dim=1)
    targets = TRUE_PARAMS[0]*M1 + TRUE_PARAMS[1]*M2 + TRUE_PARAMS[2]*M3
 + TRUE_PARAMS[3]*M4


    return features, targets, X0, Y0


def true_center_field(x, y):
    """中心点(dx=0,dy=0)的真值场,用于可视化。"""
    return (
        TRUE_PARAMS[0] * torch.sin(x) * torch.cos(y)
        + TRUE_PARAMS[1] * torch.cos(x) * torch.sin(y)
        + TRUE_PARAMS[2] * torch.cos(x) * torch.cos(y)
        + TRUE_PARAMS[3] * torch.sin(x) * torch.sin(y)
    )


# 训练集：狭小观测区域 [0,1]x[0,1],1000 个样本
x_train_features, y_train_targets, _, _ = generate_data(1, [0.0, 1.0],
[0.0, 1.0])
# 测试集：区域外 OOD（较近范围）[-3,4]x[-3,4],500 个样本
x_test_features, y_test_targets, _, _ = generate_data(500, [-3.0, 4.0],
 [-3.0, 4.0])


# ========================================
# 1. blackbox AI（25668 params MLP）
# ========================================
class BlackboxAI(nn.Module):
    def __init__(self):
        super().__init__()
        # 每个头：4 -> 128 -> 128 -> 64 -> 1
        # 单头参数: 640 + 16512 + 8256 + 65 = 25473
        # 四头总参数: 25473 * 4 = 101892
        def make_head():
            return nn.Sequential(
```

```
                nn.Linear(4, 128),
                nn.Tanh(),
                nn.Linear(128, 128),
                nn.SiLU(),
                nn.Linear(128, 64),
                nn.Tanh(),
                nn.Linear(64, 1),
            )

        self.heads = nn.ModuleList([make_head() for _ in range(4)])

    def forward(self, features):
        # features: (N, 4, 9)
        # 取中心点(索引4)的4 个基底作为4 维输入
        x4 = features[:, :, 4]  # (N, 4)

        # 四个头各输出一个系数，与四个基底一一对应
        w1 = self.heads[0](x4)
        w2 = self.heads[1](x4)
        w3 = self.heads[2](x4)
        w4 = self.heads[3](x4)
        w = torch.cat([w1, w2, w3, w4], dim=1)  # (N, 4)

        # 用输出的4 参数重建9 点场,保持与targets 形状一致 (N, 9)
        y9 = (features * w.unsqueeze(-1)).sum(dim=1)
        return y9

# ========================================
# 2. whitebox AI 4 params  (adam 版本没有物理公式约束,纯硬跑梯度下降)
# ========================================
class Whitebox4ParamAI(nn.Module):
    def __init__(self):
        super().__init__()
        self.w = nn.Parameter(torch.randn(4))#4 params



    def forward(self, features):
        return (features * self.w.view(1, 4, 1)).sum(dim=1)

'''训练接口上,黑盒白盒ai 保持完全一致,输入都是features,形状为(N,4,9),输出预测为(N,9)的9 点场,训练的也是四个正交基底向量的系数,都对同一个y_train_targets/y_test_targets 做MSELoss,区别在于黑盒ai 的4 参数是通过4 头单独的mlp 网络拟合出来，以用于完全模仿白盒的4 参数架构,本样例测试为黑盒ai 参数为25668,而白
```

```
盒ai 的4 参数则是直接作为可训练参数存在的'''

# ...existing code...

# ========================================
# 3. equal training：Both Adam
# ========================================
model_blackbox = BlackboxAI()
model_whitebox = Whitebox4ParamAI()
criterion = nn.MSELoss()

opt_blackbox = optim.Adam(model_blackbox.parameters(), lr=0.01)
opt_whitebox = optim.Adam(model_whitebox.parameters(), lr=0.01)

print("train set training+hallucination test for areas outside the trai
ning region : both Adam,10000 Epochs...")
for epoch in range(1, 10001):
    # blackbox
    opt_blackbox.zero_grad()
    loss_b = criterion(model_blackbox(x_train_features), y_train_target
s)
    loss_b.backward()
    opt_blackbox.step()

    # whitebox 4param
    opt_whitebox.zero_grad()
    loss_w = criterion(model_whitebox(x_train_features), y_train_target
s)
    loss_w.backward()
    opt_whitebox.step()

    if epoch % 500 == 0:
        print(f"Epoch {epoch} | Blackbox Loss: {loss_b.item():.8e} | Wh
itebox Loss: {loss_w.item():.8e}")

print("\n OOD training set:Out-of-region test set ")
with torch.no_grad():
    test_loss_b = criterion(model_blackbox(x_test_features), y_test_tar
gets)
    test_loss_w = criterion(model_whitebox(x_test_features), y_test_tar
gets)

    print(f"-> blackbox 101892params OOD Loss: {test_loss_b.item():.8e}
")
```

```
    print(f"-> whitebox 4params OOD Loss: {test_loss_w.item():.8e}")

    learned_str = "[" + ", ".join([f"{val:10.5f}" for val in model_whit
ebox.w.tolist()]) + "]"
    true_str = "[" + ", ".join([f"{val:10.5f}" for val in TRUE_PARAMS.t
olist()]) + "]"
    print("\n-> the parameters learned by the White Box AI:", learned_s
tr)
    print("-> Taylor Green vortex analysis parameter: ", true_str)



# ...existing code...
# ========================================
# 4. visualization (结构对齐三维平面脚本)
# ========================================
with torch.no_grad():
    train_loss_b = criterion(model_blackbox(x_train_features), y_train_
targets).item()
    train_loss_w = criterion(model_whitebox(x_train_features), y_train_
targets).item()
    test_loss_b = criterion(model_blackbox(x_test_features), y_test_tar
gets).item()
    test_loss_w = criterion(model_whitebox(x_test_features), y_test_tar
gets).item()


    x_line = np.linspace(-3, 4, 70)
    y_line = np.linspace(-3, 4, 70)
    X_mesh, Y_mesh = np.meshgrid(x_line, y_line)

    X_flat = torch.tensor(X_mesh.flatten(), dtype=torch.float64).unsque
eze(1)
    Y_flat = torch.tensor(Y_mesh.flatten(), dtype=torch.float64).unsque
eze(1)

    # 真值 (中心点场)
    Z_true = true_center_field(X_flat, Y_flat).view(-1, 1).numpy().resh
ape(X_mesh.shape)

    # 构建网格上的 3x3 输入特征,预测9 点,取中心点 index=4 可视化
    X_grid = X_flat + dx.unsqueeze(0)
    Y_grid = Y_flat + dy.unsqueeze(0)
    M1 = torch.sin(X_grid) * torch.cos(Y_grid)
    M2 = torch.cos(X_grid) * torch.sin(Y_grid)
    M3 = torch.cos(X_grid) * torch.cos(Y_grid)
    M4 = torch.sin(X_grid) * torch.sin(Y_grid)
```

```
    features_mesh = torch.stack([M1, M2, M3, M4], dim=1)

    Z_white = model_whitebox(features_mesh)[:, 4].cpu().numpy().reshape
(X_mesh.shape)
    Z_black = model_blackbox(features_mesh)[:, 4].cpu().numpy().reshape
(X_mesh.shape)

    # 训练区域框 [0,1]x[0,1]
    box_x = np.array([0, 1, 1, 0, 0], dtype=np.float64)
    box_y = np.array([0, 0, 1, 1, 0], dtype=np.float64)
    box_xt = torch.tensor(box_x, dtype=torch.float64).unsqueeze(1)
    box_yt = torch.tensor(box_y, dtype=torch.float64).unsqueeze(1)
    box_z = true_center_field(box_xt, box_yt).squeeze(1).cpu().numpy()

    fig = plt.figure(figsize=(18, 6))

    ax1 = fig.add_subplot(131, projection='3d')
    ax1.plot_surface(X_mesh, Y_mesh, Z_true, cmap='viridis', alpha=0.85)
    ax1.plot(box_x, box_y, box_z, color='red', linewidth=3)
    ax1.set_title("Ground Truth TGV Center Field\n(Red Box = Training Z
one)", fontsize=12)
    ax1.set_xlabel("x")
    ax1.set_ylabel("y")
    ax1.set_zlabel("u(x,y)")

    ax2 = fig.add_subplot(132, projection='3d')
    ax2.plot_surface(X_mesh, Y_mesh, Z_white, cmap='plasma', alpha=0.85)
    ax2.set_title(
        f"Whitebox AI (4 Params, ADAM)\nTrain Loss={train_loss_w:.5e}\n
Test Loss={test_loss_w:.5e}",
        fontsize=10,
        pad=14,
    )
    ax2.set_xlabel("x")
    ax2.set_ylabel("y")
    ax2.set_zlabel("u(x,y)")

    ax3 = fig.add_subplot(133, projection='3d')
    ax3.plot_surface(X_mesh, Y_mesh, Z_black, cmap='Reds', alpha=0.85)
    ax3.plot(box_x, box_y, box_z, color='black', linewidth=3)
    ax3.set_title(
        f"Blackbox MLP (101892 Params,ADAM)\nTrain Loss={train_loss_b:.
5e}\nTest Loss={test_loss_b:.5e}",
        color='red',
        fontsize=10,
        pad=14,
```

```
)
ax3.set_xlabel("x")
ax3.set_ylabel("y")
ax3.set_zlabel("u(x,y)")


plt.tight_layout()
plt.subplots_adjust(top=0.90)
plt.show()
```