arXiv:2602.03404v1 [physics.comp-ph] 3 Feb 2026

神经霍奇校正求解器：混合迭代—神经框架

预印本

Arjun Puthli
应用力学系，IIT Delhi；数学系；计算机科学与信息系统系，比尔拉理工学院KK Birla果阿校区，桑科阿莱，403726，印度
f20212249@goa.bits-pilani.ac.in

Somdatta Goswami
土木与系统工程系
约翰斯·霍普金斯大学
马里兰州巴尔的摩，21218
somdatta@jhu.edu

Souvik Chakraborty应用力学系 亚迪人工智能学院(ScAI) 印度理工学院德里分校（IIT） 豪兹卡斯，110016，印度
souvik@am.iitd.ac.in

2026年2月4日

摘要

我们提出神经霍奇修正求解器（NHCS），这是一种用于偏微分方程的混合迭代—神经框架，将学习得到的修正算子嵌入离散外微分演算（DEC）公式中。该方法结合经典的雅可比–理查森迭代与数据驱动修正，在保留底层拓扑与度量结构的同时细化数值解。NHCS采用两阶段训练策略。第一阶段通过相对残差最小化，从数据中学习DEC算子。第二阶段将这些算子集成到迭代求解器中，并通过学习修正更新来提升收敛性；即使中间解不准确，这些更新仍能发挥作用。这种交错训练能够实现稳定、渐进的细化，同时保持DEC离散化的保结构特性。为增强多尺度自适应性，NHCS引入了一种基于卷积神经网络的修正项，该修正项能够借助融合全局上下文信息的局部更新捕捉细尺度解特征，从而提升相较于网格分量级神经方法的可扩展性。此外，所提出的框架通过避免基于牛顿–拉夫森法的训练及其相关的参数化算子雅可比矩阵计算，大幅降低了计算成本。由此得到的求解器在不损害数值稳定性的前提下，提高了效率、鲁棒性和精度。

关键词 离散外微积分 · 混合方法 ·科学机器学习

1 引言

偏微分方程（PDEs）构成了广泛物理与工程系统的数学基础，支配着诸如流体动力学等现象 [1], 传热 [2], 静磁学 [3, 4], 以及波传播[5]。因此，准确而高效地求解偏微分方程仍是科学计算的核心目标。经典数值方法，包括有限差分法 [6], 有限元法 [7], 有限体积法 [8], 以及谱方法 [9],提供了成熟且具有理论基础的框架，对一致性、稳定性以及

收敛。然而，使用这些方法获得高精度通常需要精细的空间离散化，并且对于时变问题还需要较小的时间步长，从而导致巨大的计算成本。这一挑战在多查询场景中尤为突出：偏微分方程必须针对变化的初始条件、边界条件或系统参数反复求解，使得传统求解器难以用于实时决策或大规模参数化研究。受这些局限的推动，基于深度学习的数据驱动方法近来受到关注，可作为分摊重复偏微分方程求解成本的一种途径。特别是，神经算子框架[10] 旨在学习无限维函数空间之间的映射，从而逼近解算子本身，而非单个解实例。早期例子如深度算子网络（DeepONet） [11],及其分支-主干架构，以及随后包括傅里叶 [12], 小波 [13],和拉普拉斯 [14] 神经算子在内的基于积分核的模型，已作为复杂物理系统（包括热传导和断裂力学）的代理模型展现出强劲的经验性能。尽管取得了成功，这些方法在很大程度上仍作为黑箱逼近器发挥作用，与经典数值离散化相比，在可解释性、结构保持和理论保证方面较为有限，从而凸显了对混合方法的需求，即将基于学习的模型的效率与传统数值求解器的鲁棒性和结构相结合。

混合方法试图将经典数值方法的稳定性和可解释性与深度学习的适应性和计算效率相结合，以弥合这一差距。总体而言，这些方法分为两类，即深度学习引导（DL-G）方法和深度学习辅助（DL-A）方法。在 DL-G 方法中，神经网络架构用于自适应地提升迭代求解器的收敛性。例如，柔性共轭梯度神经算子 [15] 采用谱神经算子 [16]预测共轭梯度更新方向，从而加速收敛。傅里叶神经求解器 [17]将定常迭代方法与基于快速傅里叶变换的频域校正模块相结合，其中频域滤波器由神经网络根据作为输入的偏微分方程参数进行预测。 [18] 提出通过求解局部加权最小二乘模型来学习代数多重网格方法的延拓算子，并将其应用于随机扩散方程以及具有小波数的一维亥姆霍兹方程。额外的神经网络项充当算子的非线性校正因子，进一步加速收敛。深度多重网格方法[19] 对几何多重网格方法的收敛过程进行建模，然后优化深度神经网络以模拟类似的收敛模式。另一方面，DL-A 方法以更直接的方式将神经近似器与经典求解器相结合。在算子学习方法发展之前，Int-Deep [20]使用经过训练的深度残差网络 [21] 提供初始猜测，并将其传递给传统有限元求解器。混合迭代神经可迁移求解器（HINTS） [22] 在经典更新（例如雅可比或 Gauss–Seidel）与预训练的 DeepONet 之间交替进行。HINTS 在具有规则和不规则几何的一维及二维泊松方程和亥姆霍兹方程数据集上进行了训练和评估。 [23] 通过用 MIONet[24], 替代 DeepONet 扩展了这一方法，从而能够处理多个输入函数。这些混合方法旨在实现严格的收敛保证与学习模型适应性之间的平衡。尽管已有研究表明它们能够泛化到不同的网格离散化，但其中的深度学习组件仍然充当黑箱近似器。

离散外微分演算（DEC） [25, 26] 为在离散网格上表示微分算子提供了一种无坐标且几何一致的框架，并已在计算机图形学应用中取得广泛成功[27, 28, 29]。DEC 的一个显著特点是，它能通过斯托克斯定理等基本定理的离散类比，自然地编码向量分析守恒原理。因此，DEC 尤其适用于设计偏微分方程求解器，因为它能在离散层面强制满足关键的物理守恒性质。由此，基于 DEC 的求解器能够保留底层区域的几何与拓扑结构，确保数值近似忠实于物理定律 [30]。文献中已将这些方法广泛应用于各种流体流动问题 [31, 32]。 [33] 提出了一种用于求解偏微分方程的数据驱动的离散外微分演算框架，其中度量信息由数据学习得到，并与纯拓扑的 DEC 算子相结合。尽管他们通过二维达西流和静磁学问题的数值实验，展现了数据引导的几何学习的潜力，但该方法主要作为概念验证，并表现出明显的可扩展性局限。

针对这一挑战，我们提出了**神经霍奇修正求解器（NHCS）**，这是一种深度学习辅助混合方法，将学习得到的修正算子融入 DEC 框架，并结合理查森迭代与雅可比迭代更新解的估计值。本文的主要贡献在于算法层面，具体如下：

- **两阶段训练：**我们采用交错的两阶段训练方法：首先利用给定数据，通过相对残差最小化学习 DEC 算子。接下来，将学习得到的算子用于混合雅可比–理查森框架，逐步改进数值求解器得到的解。此阶段的训练使模型能够迭代改进其预测结果，即使当前估计不准确，也能确保稳健收敛，同时保留底层的拓扑与度量结构。

- **增强的多尺度自适应性：** 与 [33] 中使用的网格分量神经网络相比，我们引入了基于卷积神经网络的校正项，以增强模型捕捉解空间中细尺度变化的能力。这种统一的卷积架构使校正项能够结合全局上下文学习局部调整，同时提升可扩展性和效率。
- **降低计算成本：** [33] 所采用的训练方法同时执行牛顿–拉夫森迭代并通过梯度下降更新模型参数，因此计算成本很高；每次牛顿–拉夫森更新都需要反复计算参数化微分算子的雅可比矩阵（或雅可比向量积）。我们采用的两阶段训练策略结合类似理查森的迭代更新，无需进行这些开销巨大的雅可比矩阵计算，在保持精度和稳定性的同时显著降低了总体计算成本。

本文其余部分安排如下：第2节介绍 NHCS 方法，并回顾必要的背景知识。第3节介绍评估流程和数值实验，以验证我们方法的有效性和鲁棒性。所研究的问题包括多尺度泊松方程和亥姆霍兹方程，以及具有几何不连续性的二维区域上的线弹性问题。最后，第4 节回顾我们方法的主要特征和关键发现，并在结尾讨论未来可能的研究方向。

2 神经霍奇校正求解器

2.1 经典离散外微分形式

考虑如下形式的偏微分方程:

$$\begin{aligned} \mathcal{L}(\boldsymbol{u}) &= f \quad \text{in } \Omega, \\ g(\boldsymbol{u}) &= \boldsymbol{u_b} \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

其中， f 是一个强迫函数项， u_b 是某个边界条件， $\boldsymbol{u}$ 是我们感兴趣的解，而 $\mathcal{L}$ 是一个线性椭圆微分算子。为了以保持结构的方式离散化 $\mathcal{L}$ ，我们采用离散外微分演算（DEC）。

令 Ω 表示一个代表计算区域的单纯复形。一个 k-链 C_k 是有向 k-单纯形的线性组合，而**边界算子** ∂_k : $C_{k+1} \to C_k$ 将每个（$k+1$）-单纯形映射为其 k维面的有向和。 k-链的对偶空间是 k-余链空间 C^k : $C_k \to \mathbb{R}$。**余边界算子** δ_k : $C^k \to C^{k+1}$ 编码局部量沿高维单纯形的累积方式。边界算子和余边界算子满足斯托克斯定理的离散类比,

$$\langle \delta_k \alpha, \beta \rangle = \langle \alpha, \partial_k \beta \rangle, \quad \alpha \in C^k,\ \beta \in C_{k+1}, \tag{2}$$

并满足 $\delta_{k+1} \circ \delta_k = 0$，对应于经典向量微积分恒等式（例如，对于 $k=0$，$\text{curl} \circ \text{grad} = 0$）。**离散霍奇星算子**算子 $\bigstar_k$: $C^k \to C^{n-k}$ 引入了度量结构:

$$(\bigstar_k)_{ii} = \frac{|D(\sigma_i^k)|}{|\sigma_i^k|}, \tag{3}$$

其中， $D(\sigma_i^k)$ 是与 σ_i^k 对应的对偶元素， $|\cdot|$ 表示其几何测度。它定义了一个离散内积

$$\langle x, y \rangle_{\bigstar_k} = x^\top \bigstar_k y, \tag{4}$$

并诱导出**余微分** $\delta_k^* = \bigstar_k^{-1} \delta_k^\top \bigstar_{k+1}$。将这些结合起来可得到**离散霍奇拉普拉斯算子**:

$$L_k = \delta_{k-1}\delta_{k-1}^* + \delta_k^* \delta_k = \delta_k \bigstar_{k-1}^{-1} \delta_{k-1}^\top \bigstar_k + \bigstar_k^{-1} \delta_k^\top \bigstar_{k+1} \delta_k. \tag{5}$$

虽然 DEC 保留了拓扑与度量结构，但经典离散化可能收敛缓慢，并且无法捕捉多尺度问题中的细尺度变化。

2.2 数据驱动离散外微分演算算子（DDEC）

为克服这些局限，我们引入 DEC算子的**数据驱动对应形式**。**数据驱动余边界**和**余微分**定义为

$$d_k := B_{k+1} \delta_k B_k^{-1}, \quad d_k^* := D_k^{-1} \delta_k^\top D_{k+1}, \tag{6}$$

其中， B_k 和 D_k 是元素为正的对角矩阵（通过对可训练参数平方来保证）。数据驱动的内积是

$$\langle x, y \rangle_{D_k B_k^{-1}} := \langle D_k B_k^{-1} x, y \rangle, \tag{7}$$

它保留了 a 它们之间的伴随关系，并将经典离散外微分演算中的 $\langle\cdot,\ \cdot\rangle_{\bigstar_k}$ 推广。在 $\mathbb{R}^2$ 中，我们有

$$GRAD = B_1\delta_0 B_0^{-1},\quad CURL = B_2\delta_1 B_1^{-1},\quad GRAD^* = D_0^{-1}\delta_0^\top D_1,\quad CURL^* = D_1^{-1}\delta_1^\top D_2, \tag{8}$$

和 $DIV = -GRAD^*$。这些算子满足

$$d_{k+1}\circ d_k = 0,\quad d_k^*\circ d_{k+1}^* = 0. \tag{9}$$

备注 2.1。 可学习矩阵 $\mathbf{B_k}$ 和 $\mathbf{D_k}$ 充当数据驱动的霍奇星算子，无需显式构造对偶网格即可捕捉原始网格中的度量信息。

数据驱动的 **霍奇拉普拉斯算子**是

$$\Delta_k = \begin{cases} d_0^* d_0, & k = 0, \\ d_{k-1}d_{k-1}^* + d_k^* d_k, & k \geq 1. \end{cases} \tag{10}$$

图1 总结了经典 DEC 算子与数据驱动 DEC 算子之间的关系。

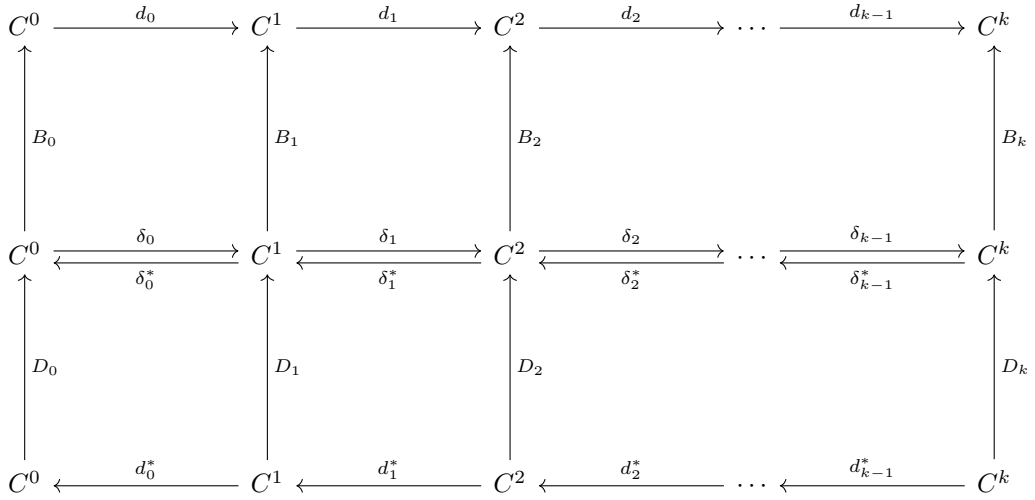


图 1：展示经典 DEC 算子与数据驱动 DEC 算子之间关系的交换图。

2.3 神经 Hodge 校正求解器与训练

为增强多尺度自适应性，我们在数据驱动离散外微分演算算子的算子表示中加入一个基于卷积神经网络的校正项：

$$\tilde{\Delta_k} = \Delta_k + \mathcal{CNN}(\Delta_k), \tag{11}$$

其中，卷积神经网络能够捕捉网格上的局部变化。深度网络中的谱偏差使其倾向于低频分量 [34, 35], , 而迭代求解器能够快速降低高频误差 [36]。卷积神经网络由四个二维卷积层（3×3卷积核，ReLU）组成，输出为单通道。

图2展示了训练与评估流程，算法1概述了训练过程，算法2则详细说明了迭代评估过程。我们将模型参数统称为 $\boldsymbol{\theta}$，它由B_k、D_k 以及合并在一起的卷积神经网络参数 $\boldsymbol{\eta}$ 组成。我们将偏微分方程相关参数与训练参数统记为 $\mathcal{P}$。此外，为避免符号歧义， $\boldsymbol{u}^i$ 表示解的第 i^{th} 次迭代近似解， $\boldsymbol{u}_i$ 表示来自数据集 $\mathcal{D}$的第 i^{th} 个真实解。在第一训练阶段，我们专注于学习算子的准确表示，通过联合更新度量表示参数$\mathbf{B_k}$ 和$\mathbf{D_k}$以及神经网络校正项来实现。这确保残差 $\mathcal{L}[\boldsymbol{u}] - f$ 足够小，从而为下一阶段提供稳定的算子表示。在第二阶段，我们将算子集成到迭代求解框架中。从解的初始猜测开始，先通过 $\mathcal{N}$ 次雅可比迭代进行细化，从而快速降低高频误差。随后进行 $\mathcal{M}$ 次神经霍奇校正迭代（即 $\mathcal{L}(\boldsymbol{u}^n) - f$），以捕捉经典迭代单独无法解析的多尺度特征和低频分量。通过将保结构的数据驱动离散外微分算子与卷积校正项相结合，求解器以极小的计算开销实现快速收敛。交错式两阶段训练确保了对数据驱动

算子的稳定且可解释学习，而评估过程避免了代价高昂的雅可比矩阵计算并降低了内存占用。请注意，$GD(L,\theta)$表示梯度下降步骤，其中我们更新参数 θ并最小化损失 L。我们所采用的损失函数是相对均方误差（Relative MSE），定义为：

$$\mathrm{L}(x,y)=\frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-y_i)^2}{\frac{1}{N}\sum_{i=1}^{N}y_i^2}. \tag{12}$$

以应对数据中因系数随机性导致的尺度变化。总体而言，该框架提供了一种统一、轻量且高度自适应的偏微分方程求解器，能够高效求解复杂的多尺度变化，同时不牺牲数学严谨性或计算效率。

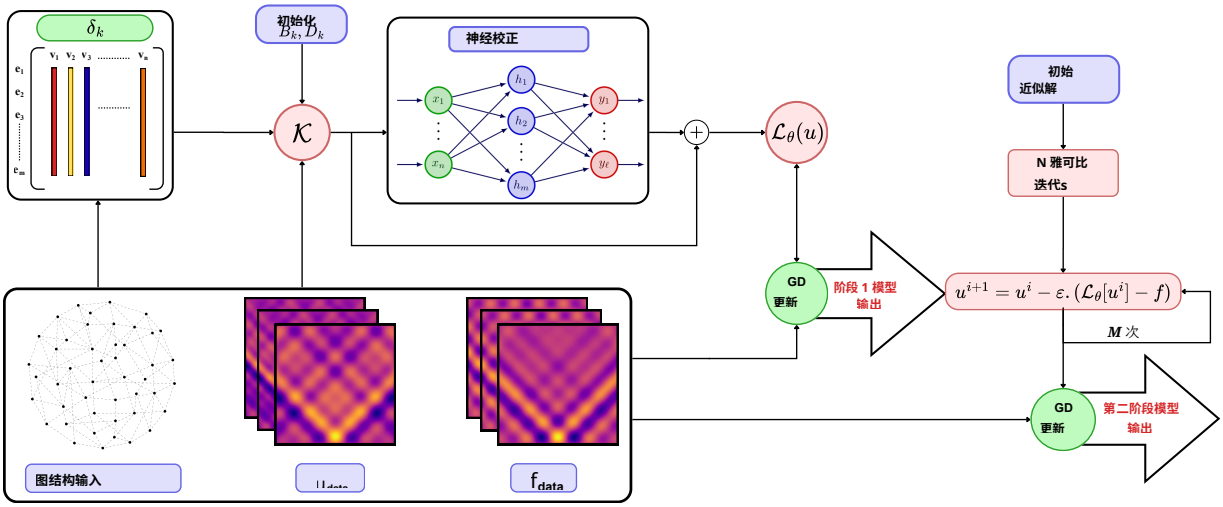


图2：阶段 1 和阶段 2 的 NHCS 训练流程。阶段 1 最小化 $\|f_{\text{est}}-f_{\text{data}}\|$ 以更新 DDEC 算子，而阶段 2 则优化混合雅可比 +神经霍奇校正求解器。

3 数值实验

为验证所提出的方法，我们针对三种典型的椭圆型偏微分方程开展数值实验：泊松方程、亥姆霍兹方程和线弹性方程。这些方程在计算科学与工程中无处不在，广泛应用于不可压缩流体动力学、静电学、声学、电磁学和结构设计等领域。例如，在不可压缩纳维–斯托克斯求解器中，压力泊松方程和亥姆霍兹型动量方程构成主要的计算瓶颈；而在波传播和散射问题中，亥姆霍兹方程决定频域行为。由于这些方程应用广泛且具有良好的解析可处理性，因此是检验我们方法有效性和鲁棒性的理想测试案例。我们通过设定带随机系数的解析傅里叶型解，并精确推导相应的强迫项，构建合成数据集。这既能确保真实值监督的精确性，又能引入受控变化以防止过拟合。最后，我们在一个具有物理意义的问题——二维线弹性方程上展示所提框架的有效性，从而凸显其通用性和适用性。

由于我们在弹性问题中并未施加任何外力，第一阶段训练采用均方根误差（均方根误差）。我们使用了 AdamW 优化器（权重衰减为 1×10^{-5}），它将权重衰减与梯度更新解耦 [37],，而非 Adam，因为它能为我们的模型提供更稳定的学习曲线。对于泊松方程和亥姆霍兹方程，我们还使用了平台期降低学习率（RoP）调度器。三个问题的训练配置汇总于表 1。所有训练与评估均在单块 NVIDIA GeForce RTX-3050 GPU 上进行，配备 $4GB$ 内存。

图 3 展示了在大小为 512 的泊松方程数据集上，中提出的偏微分方程优化算法每轮所需的时间。在此场景中使用的扰动网络包含一个宽度为 10 的隐藏层，连续层之间采用 ReLU 激活函数。这凸显了原始 DDEC 算法的巨大计算开销：NHCS 的整个训练过程所需时间显著更短（4480 秒），甚至少于原始 DDEC 方法单个训练轮次所需的时间。为进行公平比较，NHCS 训练也使用了大小相同的数据集。

算法1： NHCS 训练

数据： 偏微分方程和训练参数 $\mathcal{P}$，图结构 $\mathcal{G}$// 阶段 1：**初始化：** $\leftarrow$余边界和余微分算子 δ_k， δ_k^*，卷积神经网络 $\mathcal{CNN}$，模型参数 θ**对于** $e \leftarrow 1$**至**训练轮次**执行**$\mathcal{D} \leftarrow GenerateDataset(\mathcal{P})$**对于** u_i， $f_i \in \mathcal{D}$ **执行**$f_{est} = \mathcal{L}(u_i)$ 损失 $\leftarrow L(f_i, \quad f_{est})\theta \leftarrow GD$(损失 θ)

// 阶段 2：
初始化： $\leftarrow \varepsilon$，雅可比迭代**I**，神经霍奇迭代 **M**，雅可比模板 **A**
对 $e \leftarrow 1$**到**训练轮次**执行**
  $\mathcal{D} \leftarrow GenerateDataset\ (\mathcal{P})$
  对于 u_i， $f_i \in \mathcal{D}$ **执行**
    $u^0 \leftarrow 0$
    // 执行 I 次雅可比迭代
    对于 $n \leftarrow 1$**到*I* 执行**
      $u^n \leftarrow\ u^{n-1} + A^{-1}\ (f_i - Au^{n-1})$
    对于 $n \leftarrow 1$**到*M* 执行**
      $f^{est}\ = \mathcal{L}(u^{I+n-1})$
      $u^{I+n} \leftarrow u^{I+n-1}\ -\ \varepsilon.(f^{est} - f_i)$
    损失 $\leftarrow L(u^{I+N}, \quad u_i)$
    $\theta \leftarrow GD$(损失 θ)
返回 θ

算法2： NHCS 测试

输入： 偏微分方程参数 $\mathcal{P}$，强制项 f，雅可比模板 A，学习型 NHCS模型 $\mathcal{L}$，迭代配置（N， M），步长 ε**初始化：** $u^0 \leftarrow 0$// 雅可比预迭代**对于** $i \leftarrow 1$**到** N **执行**$u^i \leftarrow u^{i-1} + A^{-1}(f - Au^{i-1})$ // 学习型校正迭代**对于** $j \leftarrow 1$**到** M **执行**$u^{i+j} \leftarrow u^{i+j-1} - \varepsilon(\mathcal{L}(u^{i+j-1})\ - f)$**返回** u^{N+M}

我们将提出的混合模型与 HINTS算法 [38]进行比较。HINTS框架中使用的 DeepONet 由两个子网络组成：分支网络和主干网络。分支网络由一系列卷积层构成，通道尺寸为 $[2, 40, 60, 100, 180]$, ，卷积核尺寸为 3，步幅为 2。随后将生成的特征图展平，并传入三个带有 ReLU 激活函数的全连接层，得到大小为 80 的输出。主干网络由三个大小为 80 的全连接层组成，层间使用 Tanh激活函数。使用 Adam优化器进行 10,000 个训练轮次，初始学习率为 1×10^{-3}。采用阶梯式学习率调度器，每 100 个训练轮次将学习率降低至原来的 0.5 倍，最低学习率界限为 1×10^{-5}。弹性问题采用相同的架构，但分支网络的输入维度还包含杨氏模量、泊松比和顶层位移。DeepONet 训练了 20,000 个轮次，学习率为 5×10^{-3}。DeepONet 的这些架构细节及训练超参数汇总于附录 A 的表格中。HINTS 使用 [38], 中推荐的参数进行评估，并采用固定校正频率（首次迭代校正，此后每 200 次迭代校正一次），以避免偏袒 NHCS。

训练参数	泊松		亥姆霍兹		弹性力学	
	阶段 1	阶段 2	阶段 1	阶段 2	阶段 1	阶段 2
训练轮次	120	60	120	30	500	1000
优化器	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
学习率调度器	RoP	RoP	RoP	RoP	-	-
初始学习率	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-4}	1×10^{-4}
最小学习率	5×10^{-6}	1×10^{-5}	5×10^{-6}	1×10^{-5}	-	-
调度器衰减因子	0.8	0.8	0.8	0.8	-	-
调度器容差	0.1	0.02	0.1	0.02	-	-
理查森更新步 ε	-	2.5×10^{-5}	-	1.5×10^{-2}	-	1×10^{-1}

表1：泊松、亥姆霍兹和弹性力学问题在两个训练阶段中的训练配置。RoP 是学习率平台期衰减调度器。

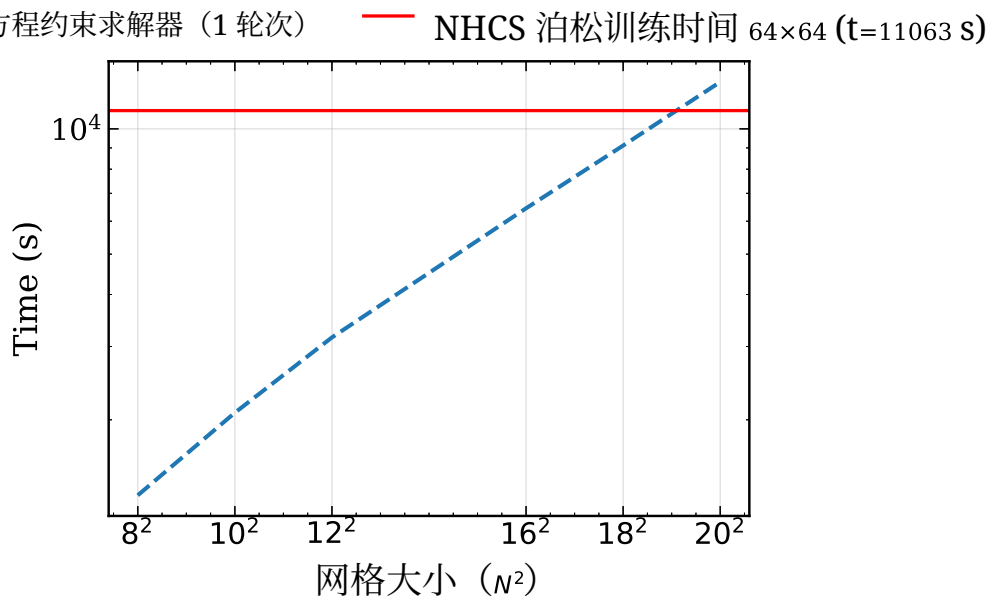


图3：泊松方程训练时间图。偏微分方程约束优化算法每轮次的训练时间（以秒为单位） [33] 随网格大小变化，由蓝色虚线表示。分量式神经网络包含一个隐藏线性层，其宽度为10，并在连续层之间使用 ReLU 激活函数。使用包含512 个样本且批量大小为1 的数据集。红线表示 $t = 11063s$，即阶段 1 +训练中阶段 2 所耗费的时间，针对 64×64 模型。

我们使用计算所得解的最终相对均方误差（Relative MSE）比较这两种方法，并考察求解器过程中的相对残差变化。Relative MSE 定义为

$$\text{Relative MSE}(\boldsymbol{u}_{\text{pred}}, \boldsymbol{u}_{\text{ref}}) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(u_{\text{pred},i} - u_{\text{ref},i}\right)^2}{\frac{1}{N}\sum_{i=1}^{N} u_{\text{ref},i}^2}, \tag{13}$$

其中，N 是我们进行评估的空间网格点数量。对于线性系统 $\mathbf{Au} = \mathbf{f}$，第 k 次迭代时的相对残差定义为

$$r_k = \frac{\|\mathbf{Au^{(k)}} - \mathbf{f}\|_\mathbf{2}}{\|\mathbf{f}\|_\mathbf{2}}. \tag{14}$$

3.1 泊松方程

二维泊松方程写为：

$$\begin{aligned} \Delta u(x,y) &= -f(x,y) \quad \forall\ (x,y) \in \Omega, \\ u(x,y) &= 0 \qquad\qquad \forall\ (x,y) \in \partial\Omega, \end{aligned} \tag{15}$$

其中，$u(x,\ y)$ 是定义在二维有界域上的标量函数，而 $f(x,\ y)$ 是该偏微分方程的强迫函数。我们使用以下解析表示合成生成数据集：

$$\begin{aligned} u(x,y) &= \frac{1}{N}\sum_{n=1}^{N} a_n \sin(n\pi x)\cos(n\pi y), \\ f(x,y) &= \frac{1}{N}\sum_{n=1}^{N} 2a_n (n\pi)^2 \sin(n\pi x)\cos(n\pi y), \end{aligned} \tag{16}$$

其中 $x \in [0,1]$，$y \in [-0.5, 0.5]$，以及 $N = 15$。这符合方程 15 中规定的形式。为生成训练数据的随机 $(u,\ f)$ 对，我们对每个 a_n 进行均匀采样，取值范围为 $(0,\ 1)$。就数据驱动 DEC 而言，上述内容的偏微分方程表示相当直接，实际上就是方程 11 中 $k = 0$ 的情形。离散 Hodge 拉普拉斯方程写为：

$$\mathcal{L}(u) \ = \ \Delta_0 u \ + \ \mathcal{CNN}(\Delta_0 u). \tag{17}$$

对应的网格间距为 h 的雅可比迭代模板如下：

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{18}$$

网格大小	泊松		亥姆霍兹	
	相对均方误差	时间（秒）	相对均方误差	时间（秒）
30×30	9.498×10^{-5}	597.65	1.059×10^{-6}	590.25
35×35	8.271×10^{-5}	872.72	2.303×10^{-6}	708.58
40×40	7.379×10^{-5}	1197.85	6.538×10^{-7}	1281.26
45×45	6.470×10^{-5}	1373.65	8.630×10^{-7}	2166.07
50×50	5.273×10^{-5}	1892.27	5.493×10^{-6}	1644.41
64×64	2.339×10^{-3}	4045.40	4.309×10^{-6}	4001.35

表2：在120个训练周期内，不同网格尺寸下泊松方程和亥姆霍兹方程的阶段 1 训练性能（相对均方误差和训练时间）。

网格大小	泊松		亥姆霍兹	
	相对均方误差	时间（秒）	相对均方误差	时间（秒）
30×30	8.341×10^{-3}	407.00	5.282×10^{-7}	253.82
35×35	9.639×10^{-3}	625.36	5.208×10^{-7}	311.43
40×40	1.161×10^{-2}	815.10	3.163×10^{-7}	438.67
45×45	1.295×10^{-2}	1279.23	2.230×10^{-7}	635.65
50×50	1.457×10^{-2}	2111.71	3.028×10^{-6}	1017.62
64×64	2.006×10^{-2}	7017.64	1.805×10^{-6}	3490.56

表3：泊松方程与亥姆霍兹方程在不同网格尺寸下、分别经过60个训练轮次和30个训练周期的阶段 2 训练性能（相对均方误差和训练时间）。

图 4(a,b) 和表 2 总结了不同网格尺寸下的第一阶段训练性能。正如预期，随着网格离散密度增加，训练所需时间也会增加。图 5(a, b) 和表 3 总结了第二阶段训练性能，并采用了 $\mathcal{N}$：$\mathcal{M} = 30 : 20$配置。图 6(a) 展示了计算

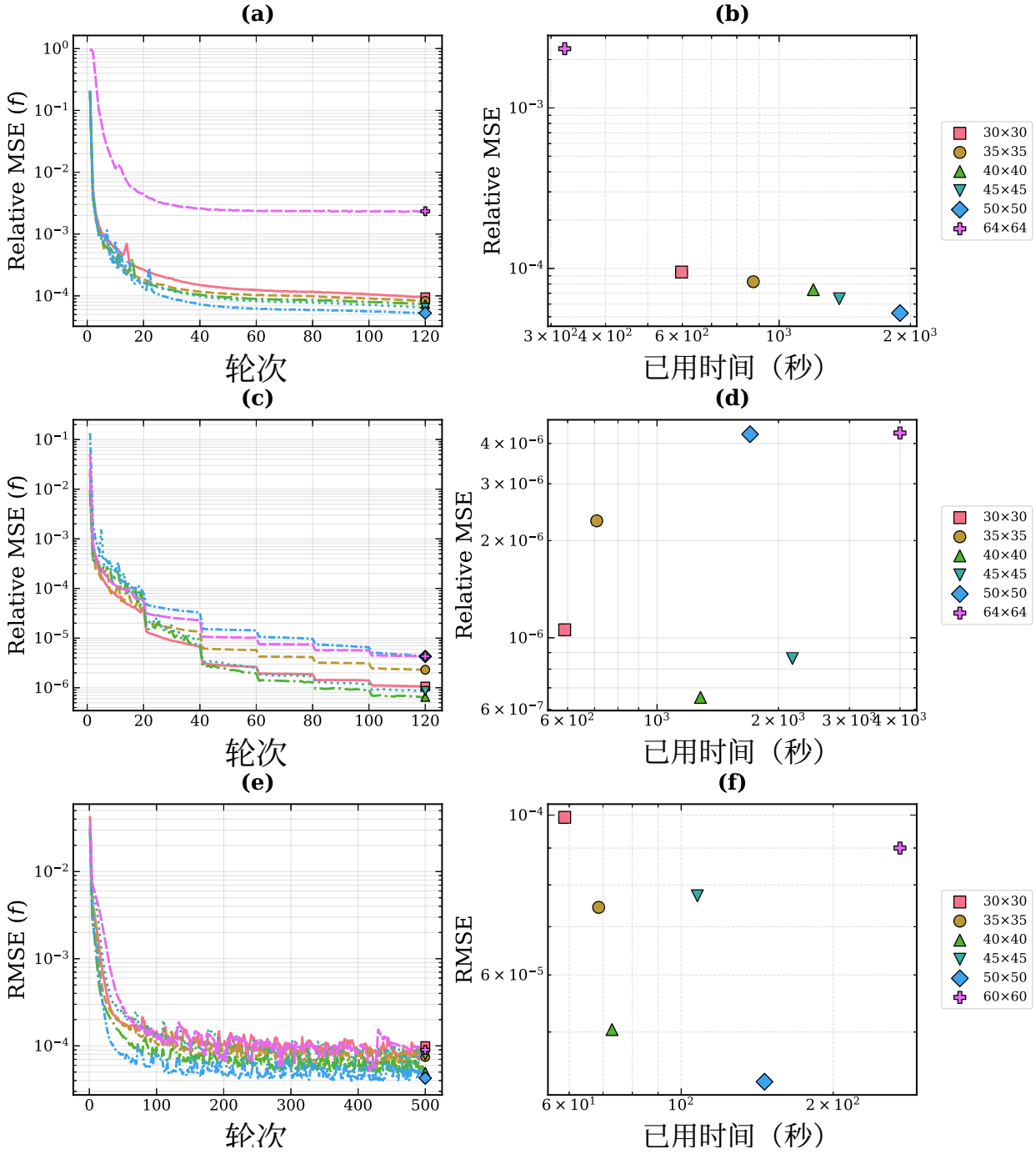


图 4：二维泊松方程 (a, b)、二维亥姆霍兹方程 (c, d) 和二维线弹性问题 (e, f) 的阶段 1 训练性能。(a, c) 强迫项的相对均方误差与训练轮次的关系。(b, d) 强迫项的相对均方误差与训练时间（秒）的关系。对于亥姆霍兹问题，我们针对波数 $\in \{4, 8, 16, 20, 24, 28\}$ 进行训练。(e) 强迫项的均方根误差与训练轮次的关系。(f) 强迫项的均方根误差与训练时间（秒）的关系。

评估解所需的时间随网格大小变化的情况。随着网格分辨率从 30×30 增加到 64×64，HINTS 的运行时间逐渐增加。相比之下，在相同网格尺寸范围内，NHCS 的运行时间几乎保持不变，表明其对网格细化的敏感性较低。图 7(a) 比较了 NHCS 和 HINTS 在泊松方程问题上的评估性能。正如最终的第二阶段训练图所示，随着网格细化，两种方法的误差都逐渐增加。具体而言，HINTS 表现出

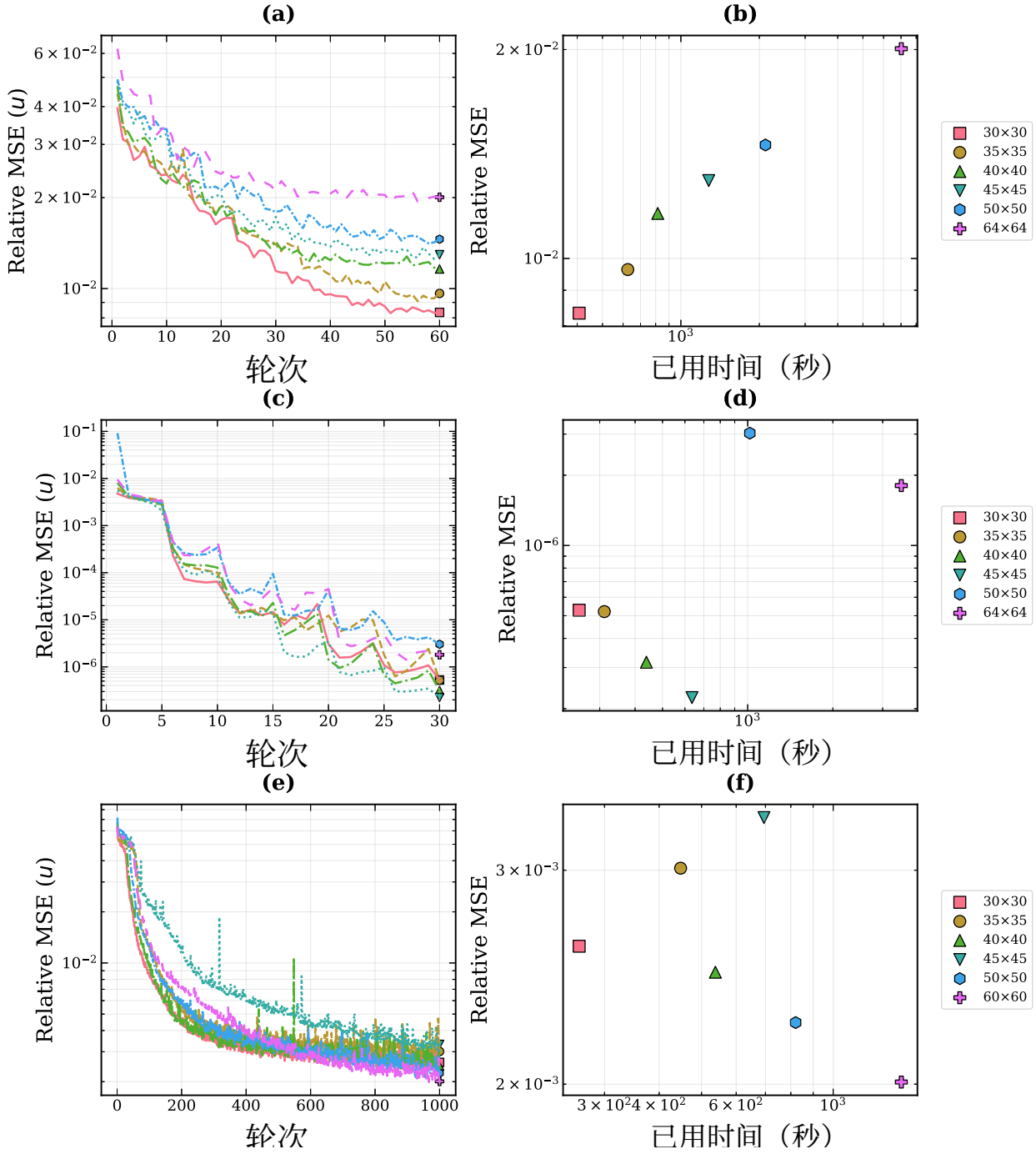


图 5：二维泊松方程（a、b）、二维亥姆霍兹方程（c、d）和二维线弹性问题（e、f）的第二阶段训练性能。（a、c、e）解的相对均方误差随训练轮次的变化。（b、d、f）解的相对均方误差随训练时间（秒）的变化。

相对误差从约 0.05 （在 30×30 网格上）增至超过 0.1 （在 64×64上），而 NHCS 的相对误差则从略低于 0.01 增至 0.02。图 8通过展示随机生成样本的预测解，直观呈现了两种方法的性能。两种方法都捕捉到了整体解结构；然而，与 HINTS 相比，NHCS 的场近似更准确，伪影更少，且与真值更为吻合。此外，HINTS 往往会产生过度平滑的预测，而 NHCS 虽然并非总能精确捕捉势场，却能成功再现局部梯度。图9 展示了其演变

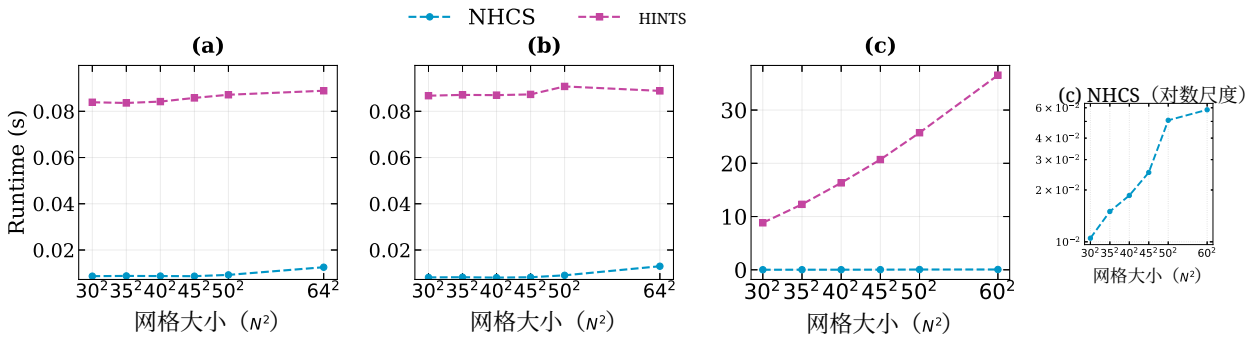


图6：NHCS 与 HINTS 在以下问题上的评估运行时间比较：(a) 泊松方程、(b) 亥姆霍兹方程和 (c) 弹性方程。最右侧的图展示了 (c) 中 NHCS 运行时间的对数尺度可视化。在这三个问题中，NHCS 的运行时间均快于 HINTS。在弹性问题 (c) 中，这种差异尤为明显：HINTS 所采用的雅可比求解器必须在计算中适应孔洞边界，而 NHCS 中经过学习的 DEC 算子则能考虑域结构。

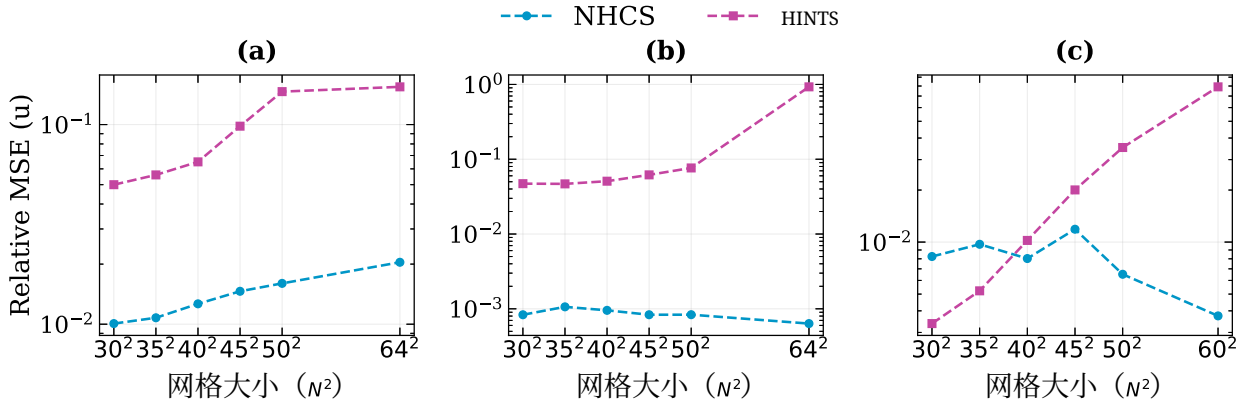


图7：NHCS 与 HINTS 在以下问题上的相对均方误差比较：(a) 泊松方程、(b) 亥姆霍兹方程和 (c) 弹性方程。在泊松方程 (a) 和亥姆霍兹方程 (b) 中，NHCS 在各种网格分辨率下均稳定优于 HINTS。由于该弹性方程 (c) 属于低频问题，HINTS 起初优于 NHCS；其求解器中的雅可比组件在较粗网格上能够更好地消除误差。

残差范数随迭代次数的变化情况。在 9(a) 中，蓝线表示 NHCS 的相对残差，黑色虚线表示基于DEC的理查森更新取代雅可比更新的迭代位置，粉色虚线表示雅可比迭代残差轨迹的延续。在此处可以观察到相对残差急剧下降，这可归因于符合物理规律的DEC算子对解近似更新的贡献。此后逐渐上升的趋势，则可归因于理查森迭代的更新特性，其结果是逐步出现轻微的过度校正。9(b) 中的残差峰值出现在 HINTS算法进行 DeepONet 校正期间。这是预期之中的，因为 DeepONet 经训练后能够将输入映射到解函数空间，而其优化目标并非残差最小化。由于雅可比迭代更新规则具有压缩性，雅可比更新的迭代残差范数持续下降。

3.2 亥姆霍兹方程

二维亥姆霍兹方程写作：

$$\begin{aligned}(\alpha - \Delta)u(x,y) &= f(x,y) \quad \forall\ (x,y) \in \Omega,\\ u(x,y) &= 0 \qquad\quad \forall\ (x,y) \in \partial\Omega,\end{aligned} \tag{19}$$

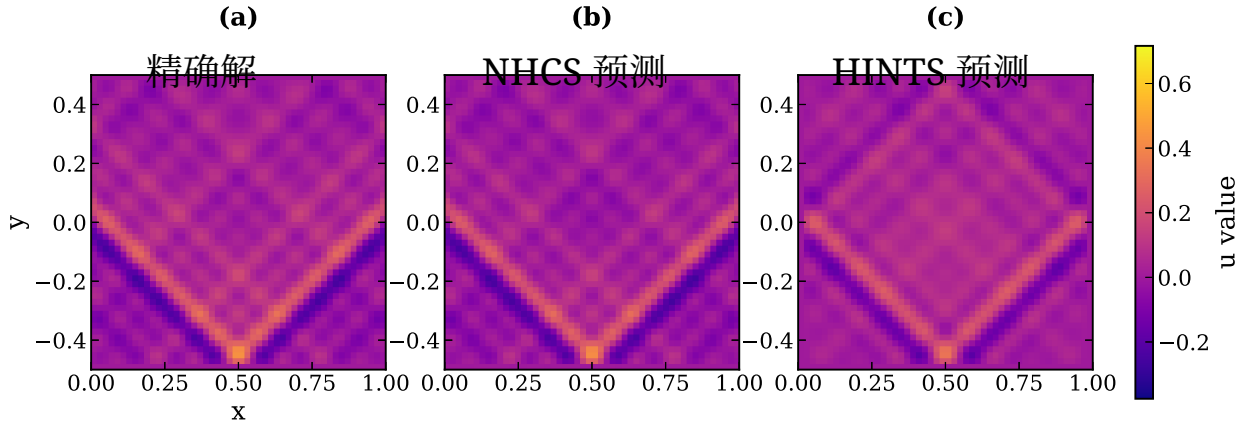


图 8：在 50×50 分辨率下，泊松方程解的估计结果与精确解 (a) 的对比：(b) NHCS，(c) HINTS。与 HINTS 的近似解 (c) 相比，NHCS 的近似解 (b) 更接近真值 (a)；HINTS 的预测结果往往过于平滑。NHCS 能够更精确地捕捉细微变化和局部梯度。

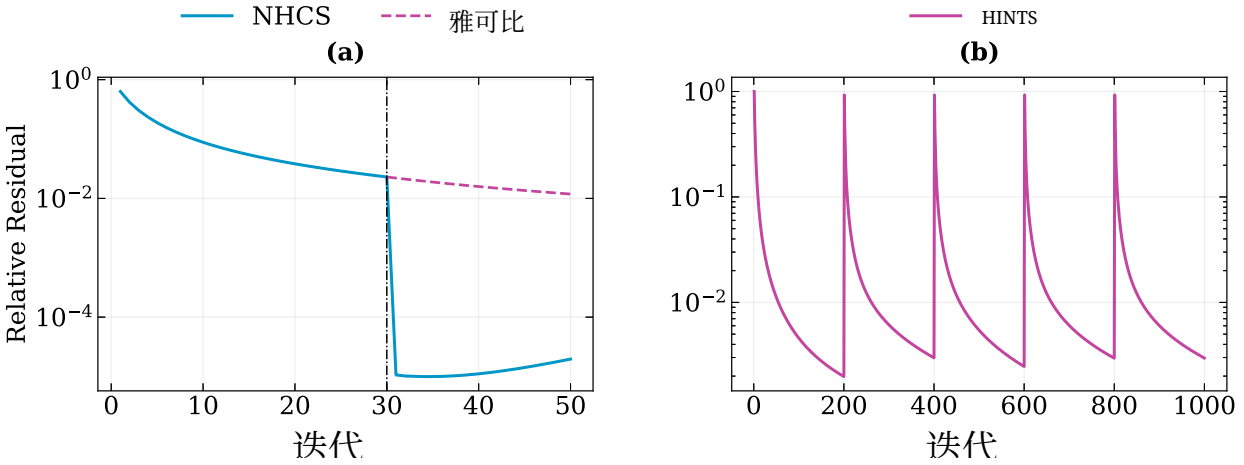


图 9：NHCS (a) 和 HINTS (b) 的泊松方程相对残差随迭代次数的变化。在 (a) 中，蓝线表示 NHCS 的残差范数，黑色虚线表示基于 DEC 的理查森更新接替雅可比更新的迭代位置，粉色虚线表示雅可比迭代残差轨迹的延续。(a) 中残差的骤降可归因于学习得到的 DEC 算子对近似校正的贡献，而 (b) 中的尖峰则出现在 HINTS 执行 DeepONet 校正期间。

其中 α 为波数。我们采用与泊松方程相同的方法生成带标签数据。合成的解-强迫函数对通过以下方式生成：

$$\begin{aligned} u(x,y) &= \frac{1}{N}\sum_{n=1}^{N} a_n sin(n\pi x).cos(n\pi y), \\ f(x,y) &= \frac{1}{N}\sum_{n=1}^{N} (2a_n(n\pi)^2 \ + \ \alpha) sin(n\pi x) \cdot cos(n\pi y). \end{aligned} \tag{20}$$

我们考虑与泊松问题相同的区域，$x \in [0,1]$，$y \in [-0.5, 0.5]$，以及 $\alpha \geq 2^2$。该算法的偏微分方程表示为：

$$\mathcal{L}(u) \ = \ \alpha u \ - \ (\Delta_0 u \ + \ \mathcal{CNN}(\Delta_0 u)). \tag{21}$$

对应的网格间距为 h 的雅可比迭代模板为

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4+\alpha h^2 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{22}$$

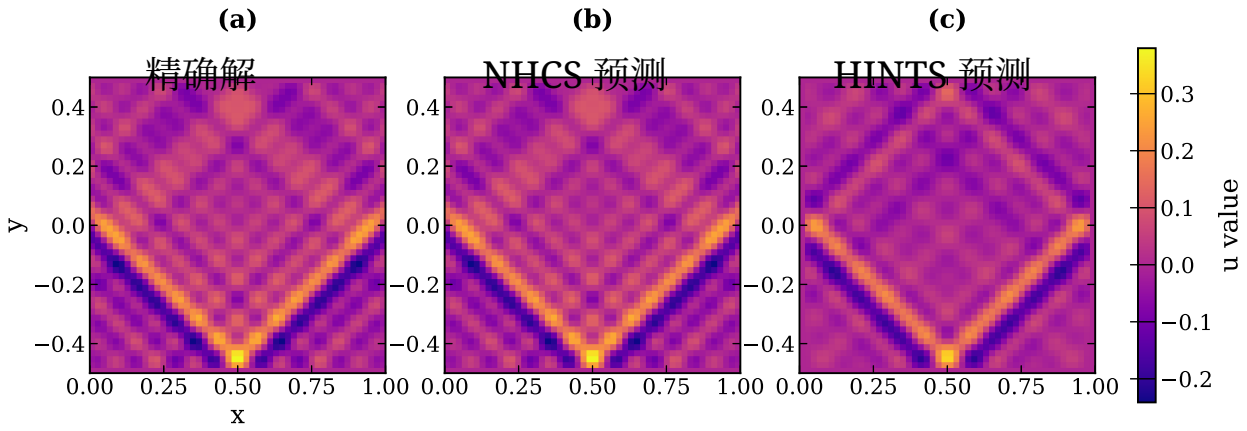


图 10：亥姆霍兹方程解的估计值对比：(b) NHCS 和 (c) HINTS 与精确解 (a) 的对比。同样，NHCS 对解场的近似比 HINTS 更准确；HINTS 往往只能捕捉高势能区域中解场的主要特征，同时会过度平滑局部高频变化。

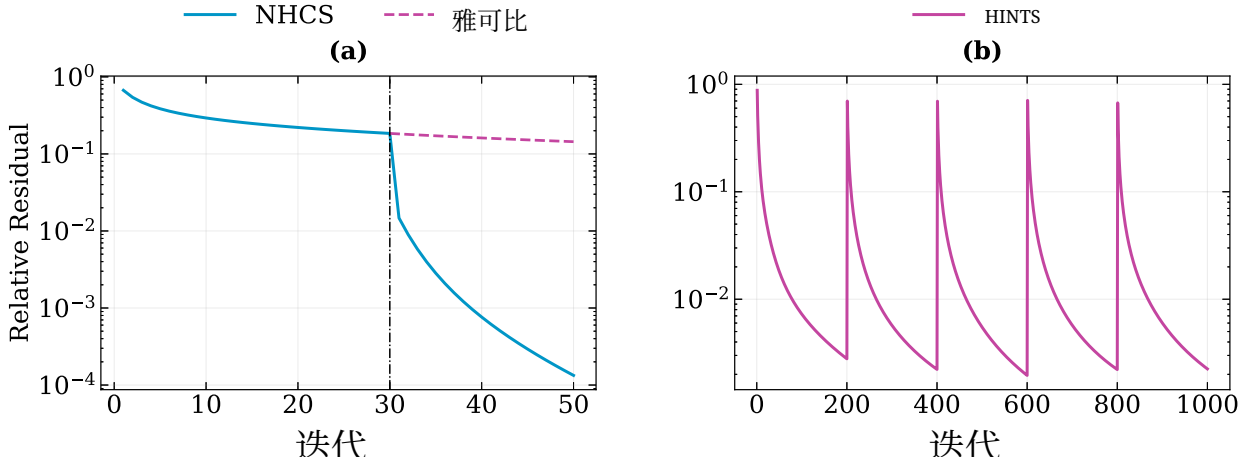


图 11：NHCS (a) 和 HINTS (b) 的亥姆霍兹方程相对残差随迭代次数的变化。在 (a) 中，蓝线表示 NHCS 的残差范数，黑色虚线表示基于DEC的理查森更新接替雅可比更新的迭代位置，粉色虚线表示雅可比迭代残差轨迹的延续。(a) 中残差的急剧下降可归因于符合物理规律的离散外微分算子在理查森更新期间对校正的贡献；(b) 中的尖峰则出现在 HINTS 的 DeepONet 校正期间。

图 4(c, d) 和表 2总结了不同网格尺寸下的第一阶段训练性能。与泊松方程类似，网格离散密度越高，训练所需的时间越长。图 5(c, d) 和表 3 总结了第二阶段训练性能，并采用了 $\mathcal{N}$： $\mathcal{M} = 30$： 20 配置。图 6(b) 展示了评估解所需的计算时间与网格大小之间的关系。同样，与泊松方程类似，随着网格分辨率从 30×30 增至 64×64，HINTS 的运行时间逐渐增加；而在相同网格尺寸范围内，NHCS 的运行时间几乎保持不变，表明其对网格细化的敏感性较弱。图 7(b) 比较了 NHCS 和 HINTS 在亥姆霍兹方程上的评估性能。除 35×35 情况外，误差通常随网格细化而降低，在 6×10^{-4} 处达到约 64×64的最低值。值得注意的是，即使是 35×35 模型，误差也低至约 1×10^{-3}。相比之下，HINTS 的误差稳步上升至 50×50，随后急剧增至近 0。9 （位于 64×64处）。

NHCS呈现的非单调行为可归因于波数项的存在：它会改变算子的谱结构，导致振荡模态的相对贡献随分辨率变化。因此，某些网格尺寸与学习到的算子表示更加匹配，从而在中等分辨率下产生较低的误差。图10展示并比较了HINTS和NHCS对随机生成样本的解进行近似的能力。与泊松方程的结果类似，NHCS能够准确地近似场表示。相比之下，HINTS仅能捕捉某些主要特征，尤其是高势能的对角区域，同时对其他区域过度平滑，因此其近似明显较不精细且不一致。图11的(a)和(b)分别展示了NHCS和HINTS的相对残差随迭代次数的变化。与泊松方程类似，在NHCS切换为理查森更新后，可以观察到残差急剧上升。然而，与泊松方程不同的是，此后残差逐渐下降。同样与泊松方程的情况类似，HINTS在执行DeepONet校正时会出现残差尖峰。

3.3 线弹性方程

我们考虑一块尺寸为 1 m $\times$ 1m 的方形板，其中心有一个边长为 0.2 m $\times$ 0.2 m 的方形切口。方形板的底边固定（位移为零），顶边施加均匀的竖直位移 Δ_y，而左、右边界无牵引力。体力 f 设为零。二维线弹性问题由偏微分方程（PDE）控制，写作：

$$\begin{cases} \nabla \cdot \sigma(x,y) + f(x,y) = 0 \\ \sigma(x,y) = C : \varepsilon(x,y) \qquad \text{in } \Omega, \\ \varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^\top\right) \end{cases} \tag{23}$$

使得 $u = 0$ 在 $\partial\Omega_b$上，$u_y = \Delta_y$ 在 $\partial\Omega_t$上，以及 $\sigma \cdot n = 0$ 在 $\partial\Omega_l \cup \partial\Omega_r \cup \partial\Omega_h$上，其中 $u(x,y) = [u_x, u_y]^\top$ 是位移场，$\boldsymbol{\varepsilon}$ 是应变张量，Δ_y 是所施加的位移，$\boldsymbol{\sigma}$ 是柯西应力张量。$\partial\Omega_t$、$\partial\Omega_b$、$\partial\Omega_l$、$\partial\Omega_r$和 $\partial\Omega_h$ 分别表示方形板的顶边、底边、左边、右边和孔边界。四阶弹性张量 $\mathcal{C}$ 由材料属性——杨氏模量 E 和泊松比 ν——定义如下：

$$\mathcal{C} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}. \tag{24}$$

为生成数据，我们在界限 [7MPa,10MPa], 内对杨氏模量进行均匀采样，泊松比从 $[0.15, 0.35]$采样，Δ_y 从 $[0.01, 0.25]$采样，并使用Matlab的偏微分方程求解器生成目标解。应力张量的1-链式DDEC表示写为：

$$\varepsilon_k(u) = \frac{1}{2}\begin{bmatrix} 2\,(\text{GRAD}\,u_x)_k & (\text{GRAD}\,u_y)_k \\ (\text{GRAD}\,u_y)_k & 2\,(\text{GRAD}\,u_y)_k \end{bmatrix}, \tag{25}$$

并且该算法的偏微分方程表示为

$$\begin{aligned} \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)}, \\ \mu &= \frac{E}{2(1+\nu)}, \\ \mathcal{K}u &= \text{GRAD}^*\mathcal{C} : \varepsilon(u) \\ &= \text{GRAD}^*(\lambda \text{tr}(\varepsilon(u))\mathbf{I} \;+\; 2\mu\varepsilon(u), \\ \mathcal{L}(u) &= \mathcal{K}u \;+\; \mathcal{CNN}(\mathcal{K}u). \end{aligned} \tag{26}$$

图4(e, f) 和表4 总结了不同网格尺寸下的第一阶段训练性能。与泊松方程和亥姆霍兹方程类似，训练所需时间随网格离散密度的增加而增加。图5(e, f) 和表5 总结了采用经修改的$1:7$ 配置时的第二阶段训练性能。我们不再执行单次雅可比迭代，而是先考虑一个没有孔洞的正方形区域，并通过在 y 方向施加单轴拉伸，以及在 x 方向施加由泊松比决定的横向收缩来近似求解。如果 Δ_y 是顶层的 y 轴位移，且 ν 是泊松比，则初始近似解计算如下

$$\begin{aligned} u_x(x,y) &= -\,\nu\,\frac{\Delta}{L}\,x, \\ u_y(x,y) &= \;\;\frac{\Delta}{L}\,y, \end{aligned} \tag{27}$$

网格大小	RMSE	时间（秒）
30×30	9.927×10^{-5}	58.77
35×35	7.446×10^{-5}	68.62
40×40	5.039×10^{-5}	72.93
45×45	7.726×10^{-5}	107.65
50×50	4.266×10^{-5}	146.21
60×60	9.000×10^{-5}	271.21

表 4：阶段 1 训练性能（均方根误差和训练时间），针对弹性问题（平面应力，中心孔），在不同网格尺寸上经过 500 个训练轮次。

网格大小	相对均方误差	时间（秒）
30×30	2.597×10^{-3}	262.52
35×35	3.012×10^{-3}	447.77
40×40	2.473×10^{-3}	537.76
45×45	3.315×10^{-3}	694.60
50×50	2.248×10^{-3}	820.62
60×60	2.008×10^{-3}	1431.45

表 5：阶段 2 训练性能（相对均方误差和训练时间）：不同网格尺寸下经过 1000 轮训练的结果。

其中 u_x和u_y 分别是 x 和 y 方向的位移。随后将此初始近似解输入理查森更新循环。图5(f) 显示，对于弹性问题，随着模型大小增加，最终训练误差总体呈下降趋势。尽管区域中的几何不连续性可能导致局部误差累积，但更精细的网格能使模型更准确地捕捉这些变化，并消除由此产生的近似误差。此外，采用半解析初始近似解可使所有训练样本的收敛行为更加一致。

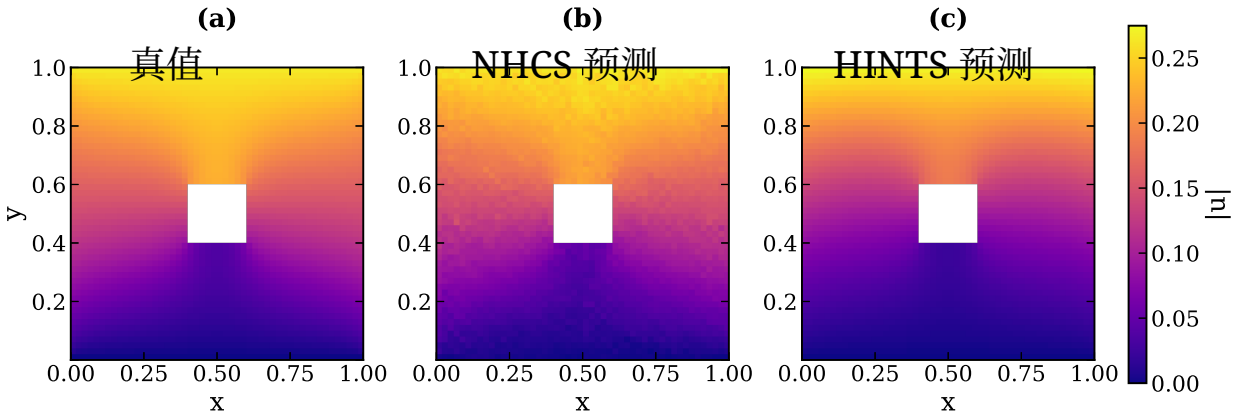


图 12：弹性问题解的估计结果对比：(b) NHCS 和 (c) HINTS 与精确解 (a) 的对比。NHCS 近似解中可清楚看到较粗糙的局部误差。尽管如此，即使存在几何不连续性，它仍能准确近似解场。HINTS 近似解明显更加平滑，但靠近孔洞边界时误差会增大。再进行几次雅可比迭代即可进一步优化 NHCS 预测，平滑较粗糙的误差。

图6(c) 展示了评估解所需的计算时间随网格大小的变化。此处性能差异尤为明显：HINTS 的运行时间从8秒左右（分辨率为 30×30 ）增加到30 秒以上（分辨率为 60×60）。虽然 NHCS 的运行时间略有增长，但在所有网格大小下均低于0.06 秒，因此相比 HINTS 提速了数个数量级。为更清晰地展示 NHCS 的运行时间，还提供了放大的对数坐标图。图7(c) 比较了 NHCS 和 HINTS 在二维弹性方程问题上的评估性能。正如第二阶段训练图所示，NHCS 的误差总体呈下降趋势。相比之下，HINTS 的相对误差却呈上升趋势

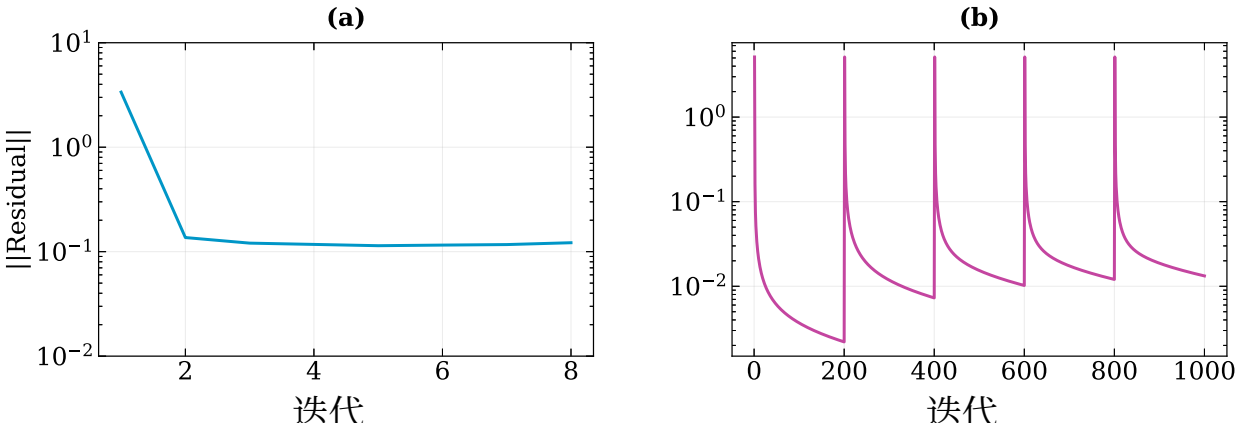


图 13：NHCS（a）和 HINTS（b）的弹性问题迭代残差范数与迭代次数的关系。与亥姆霍兹方程和泊松方程相比，（a）中的迭代残差演化显著更加稳定，这是由于解具有低频特性，从而带来了更稳定的更新。（b）中的尖峰同样是由解空间中的 DeepONet 校正造成的。

网格细化时的均方误差，这与我们此前在泊松和亥姆霍兹问题中的观察一致。除 45×45 这一情况外，NHCS 的误差维持在 10^{-3}量级，在 60×60 网格上观测到的最小误差低于 4×10^{-3}。对于 30×30 和 35×35 网格，HINTS 的精度确实高于 NHCS；然而，随着网格分辨率提高，其误差急剧上升，在0.08 左右（60×60 这一情况）。控制方程构成一个耦合偏微分方程组，而网格细化会改变这种耦合在离散化过程中的表示方式，以及 DEC 算子在理查森更新中的校正方式，因而导致 NHCS 呈现非单调行为。图12 通过展示随机生成样本的预测解，说明了两种方法的性能。NHCS 能够捕捉几何不规则性，但局部仍存在一些误差。它对解的近似较为准确，不过某些区域仍不够平滑。HINTS 得到的位移场较为平滑，但精度明显较低，且误差在孔洞边界附近逐渐累积。在最后一次迭代中，DeepONet 组件似乎还高估了顶部边界位移的近似值。这表明 NHCS 能够灵活适用于存在几何不规则性的问题。由于强迫项为零，我们绘制的是残差范数轨迹，而非相对残差。图13(a) 和 (b) 分别展示了 NHCS 和 HINTS 的迭代残差范数随迭代次数的变化轨迹。与亥姆霍兹方程和泊松方程相比，(a) 中的迭代残差演化显著更加稳定，这是因为解的频率较低，从而使更新更加稳定。(b) 中的尖峰同样源于解空间中的 DeepONet校正，而雅可比迭代则因其更新的压缩性而呈现单调下降趋势。本实验表明，NHCS 无需改变架构即可自然扩展到标量椭圆问题之外，处理具有复杂几何结构的向量值耦合偏微分方程，因为算子能够考虑区域的结构。

4 结论与讨论

我们介绍了神经霍奇修正求解器（NHCS），这是一种用于求解偏微分方程的神经网络与迭代混合框架，利用离散外微分演算（DEC）算子和数据驱动的度量学习。偏微分方程算子由 DEC 上边界算子和余微分算子构造，而卷积校正项则捕捉局部和全局变化，从而实现多尺度自适应性。NHCS 以系统化的方式将经典迭代求解器与学习得到的校正相结合，利用雅可比迭代高效降低高频误差，并通过神经霍奇校正迭代解决低频和多尺度特征。交错式两阶段训练确保 DEC 算子的学习稳定且可解释，而评估过程则避免了昂贵的雅可比矩阵计算，并降低了内存开销。通过对泊松、亥姆霍兹和线弹性问题进行数值实验，我们证明 NHCS 能够保留拓扑与度量结构，准确捕捉解中的高频和低频分量，并在存在几何不连续性的情况下仍保持解的尖锐特征。与 HINTS 相比，NHCS 收敛更快、解的保真度更高、多尺度分辨能力更强；残差轨迹表明，它能以更少的迭代实现更有效的迭代更新。

该框架轻量且适应性强，为整合经典方法与学习组件提供了统一途径，因此广泛适用于具有复杂解结构的偏微分方程。通过结合保结构算子与单个卷积校正项，NHCS 在计算效率与

数值精度之间实现了平衡，避免了以往混合方法中出现的过度平滑和高昂计算成本。所提出的方法表明，将几何与拓扑结构以原则性的方式融入学习型求解器，能够提升可解释性、收敛性和鲁棒性，为混合神经数值偏微分方程求解器指明了一个充满希望的方向。

尽管 NHCS 在规则的二维区域和几何复杂度适中的问题上表现出色，但仍有若干扩展方向有待未来探索。目前，扩展到更大的网格或非结构网格受到构建和应用余边界算子的计算成本限制，该成本随元素数量按 $\mathcal{O}(n^4)$ 增长。区域分解与并行化，或将不规则网格映射到更粗表示的编码策略，或可应对这些挑战。此外，通过稳定的混合时间积分器将 NHCS 扩展到时变或非线性偏微分方程，是一条很有前景的途径；时空网格上的离散外微积分近期进展也表明这一方向切实可行。最后，在混合框架中利用学习得到的自适应预条件器，或可进一步加速收敛，这可借鉴基于离散外微积分的方法在流形学习和数据分析中的成功经验。

致谢

S. Chakraborty 感谢港口与航运部通过编号为 ST14011/74/MT (356529) 的信函提供的资助，以及 IIT Delhi 提供的种子基金。

利益冲突

作者声明不存在任何利益冲突。

参考文献

[1] G. K. Batchelor，《流体动力学导论》，剑桥数学文库，剑桥大学出版社，2000年。[2] F. P. Incropera，《传热与传质基础》，John Wiley & Sons, Inc.，美国新泽西州霍博肯，2006年。[3] D. J. Griffiths，《电动力学导论》，培生，马萨诸塞州波士顿，2013年。[4] P. Monk，《麦克斯韦方程的有限元方法》，牛津大学出版社，2003年。[5] W. Strauss，《偏微分方程：导论》，威利，2007年。[6] J. C. Strikwerda，《有限差分格式与偏微分方程》，工业与应用数学学会，2004年。[7] T. J. Hughes，《有限元方法：线性静态与动态有限元分析》，Courier Corporation，2012年。[8] R. Eymard、T. Gallouët、R. Herbin，有限体积法，载于：$\mathbb{R}^n$ 中的方程求解（第3部分），科学计算技术（第3部分），《数值分析手册》第7卷，爱思唯尔，2000年，第713–1018页。[9] C. Canuto、M. Hussaini、A. Quarteroni、T. Zang，《谱方法：向复杂几何的演进及其在流体动力学中的应用》，科学计算，施普林格·柏林·海德堡，2007年。[10] N. Kovachki、Z. Li、B. Liu、K. Azizzadenesheli、K. Bhattacharya、A. Stuart、A. Anandkumar，神经算子：学习函数空间之间的映射及其在偏微分方程中的应用，《机器学习研究杂志》24（89）（2023）1–97。[11] L. Lu、P. Jin、G. Pang、Z. Zhang、G. E. Karniadakis，基于算子通用逼近定理通过DeepONet学习非线性算子，《自然·机器智能》3（3）（2021）218–229。[12] Z. Li、N. B. Kovachki、K. Azizzadenesheli、B. Liu、K. Bhattacharya、A. Stuart、A. Anandkumar，用于参数化偏微分方程的傅里叶神经算子，2020年。[13] T. Tripura、S. Chakraborty，用于求解计算力学问题中参数化偏微分方程的小波神经算子，《计算力学与工程方法》404（2023）115783。[14] Q. Cao、S. Goswami、G. E. Karniadakis，用于求解微分方程的拉普拉斯神经算子，《自然·机器智能》6（6）（2024）631–640。

[15] A. Rudikov、V. Fanaskov、E. Muravleva、Y. M. Laevsky、I. Oseledets，神经算子与共轭梯度的结合：用于高效偏微分方程求解的fcg-no方法，载于：第41届国际机器学习会议论文集，ICML’24，JMLR.org，2024年。[16] V. S. Fanaskov、I. V. Oseledets，谱神经算子，《数学报告》108（2）（2023）S226–S232。[17] C. Cui、K. Jiang、Y. Liu、S. Shu，用于大型稀疏线性代数系统的傅里叶神经求解器，《数学》10（21）（2022）4014。[18] F. Wang、X. Gu、J. Sun、Z. Xu，用于代数多重网格法的基于学习的局部加权最小二乘，《计算物理学杂志》493（2023）112437。[19] A. Katrutsa、T. Daulbaev、I. Oseledets，多重网格参数的黑盒学习，《计算与应用数学杂志》368（2020）112524。[20] J. Huang、H. Wang、H. Yang，Int-Deep：一种由深度学习初始化的非线性问题迭代方法，《计算物理学杂志》419（2020）109675。[21] K. He、X. Zhang、S. Ren、J. Sun，用于图像识别的深度残差学习，CoRR abs/1512.03385（2015）。[22] A. Kopaničáková、G. E. Karniadakis，基于DeepONet的预条件策略：求解参数化线性方程组，《SIAM科学计算杂志》47（1）（2025）C151–C181。[23] J. Hu、P. Jin，一种基于MIONet的偏微分方程混合迭代方法：理论与数值算例，《计算数学》（2025年4月）。[24] P. Jin、S. Meng、L. Lu，MIONet：通过张量积学习多输入算子，《SIAM科学计算杂志》44（6）（2022）A3490–A3514。[25] A. N. Hirani，离散外微积分，博士论文，美国，aAI3086864（2003）。[26] M. Desbrun、A. N. Hirani、M. Leok、J. E. Marsden，离散外微积分，arXiv:math/0508341（2005年8月）。[27] K. Crane、F. de Goes、M. Desbrun、P. Schröder，基于离散外微积分的数字几何处理，载于：ACM SIGGRAPH 2013课程，SIGGRAPH ’13，计算机协会，美国纽约州纽约，2013年。[28] F. de Goes、M. Desbrun、M. Meyer、T. DeRose，用于几何处理的细分外微积分，《ACM图形学汇刊》35（4）（2016年7月）。[29] M. S. Mohamed、A. N. Hirani、R. Samtaney，离散外微积分在任意曲面网格上的数值收敛，《工程科学与力学计算方法国际期刊》19（3）（2018）194–206。[30] D. N. Arnold，《有限元外微积分》，工业与应用数学学会，宾夕法尼亚州费城，2018年。[31] M. Wang、P. Jagad、A. N. Hirani、R. Samtaney，采用守恒相场法的两相不可压缩纳维–斯托克斯方程的离散外微积分离散化，《计算物理学杂志》488（2023）112245。[32] M. S. Mohamed、A. N. Hirani、R. Samtaney，曲面单纯形网格上不可压缩纳维–斯托克斯方程的离散外微积分离散化，《计算物理学杂志》312（2016）175–191。[33] N. Trask、A. Huang、X. Hu，在科学机器学习中强制精确物理：图上的数据驱动外微积分，《计算物理学杂志》456（2022）110969。[34] N. Rahaman、A. Baratin、D. Arpit、F. Dräxler、M. Lin、F. A. Hamprecht、Y. Bengio、A. C. Courville，论神经网络的谱偏差，载于：国际机器学习会议，2018年。[35] Z.-Q. J. Xu、L. Zhang、W. Cai，论理解并克服用于求解偏微分方程的深度神经网络学习方法的谱偏差，《计算物理学杂志》530（2025）113905。[36] I. Harimi、M. Saghafian，多重网格法加速迭代方法收敛能力的评估，《国际学术研究通报》2012（1）172687。[37] I. Loshchilov、F. Hutter，解耦权重衰减正则化，载于：国际学习表征会议，2019年。[38] E. Zhang、A. Kahana、A. Kopanicakova、E. Turkel、R. Ranade、J. Pathak、G. Karniadakis，在偏微分方程数值求解器中融合神经算子与松弛方法，《自然·机器智能》6（2024）1303–1313。

A 架构规格

分支网络规格				
问题	**输入通道**	**卷积通道**	**卷积核／步幅**	**线性层**
泊松	1	$[1, 40, 60, 100]$	$3/2$	$[80, 80, 80]$
亥姆霍兹	2	$[2, 40, 60, 100]$	$3/2$	$[80, 80, 80]$
弹性力学	5	$[5, 40, 60, 100]$	$3/2$	$[80, 80, 80]$

表 6：HINTS框架中使用的 DeepONet 的分支网络架构规格。卷积层的输出通过由三个全连接层组成的一组层映射为大小为 80 的最终输出，各层之间使用 ReLU 激活函数。

主干网络规格和 DeepONet训练超参数				
问题	**输入维度**	**线性层**	**激活函数**	**训练轮数 / LR**
泊松	2	$[80, 80, 80]$	Tanh	10，000 / 1×10^{-3}
亥姆霍兹	2	$[80, 80, 80]$	Tanh	10，000 / 1×10^{-3}
弹性力学	2	$[80, 80, 80]$	Tanh	20，000 / 5×10^{-3}

Table 7：HINTS 中使用的主干网络架构规格与 DeepONet 训练参数 .