arXiv:2303.02590v1 [math.NA] 5 Mar 2023

采用神经网络界面近似求解具有不同波数的时谐麦克斯韦方程的区域分解方法

T. Knoke [1],S. Kinnewig[1,3],S. Beuchler[1,3],A. Demircan[2,3],U. Morgner[2,3],和 T. Wick[1] [,3]

[1]汉诺威莱布尼茨大学，应用数学研究所，德国[2]汉诺威莱布尼茨大学，量子光学研究所，德国[3]汉诺威莱布尼茨大学，卓越集群 PhoenixD（光子学、光学与工程学——跨学科创新），德国

摘要

本文研究时谐麦克斯韦方程及其区域分解方法数值求解。作为一项创新，我们提出一种由前馈神经网络增强的子区域间界面条件近似方法。其优点在于，无需在每个更新步骤重新计算麦克斯韦系统，即可更新界面条件。本文的主要内容是详细介绍用于区域分解的神经网络的构建及其训练过程。为验证这一概念，我们通过若干低频数值实验研究了几个子区域，并将这一新方法与经典区域分解方法进行比较。此外，我们还阐述了使用不同波数进行训练和测试时面临的当前挑战，并介绍神经网络的行为特征，例如损失函数的收敛性以及不同的激活函数。

关键词： 时谐麦克斯韦方程、机器学习、前馈神经网络、区域分解法。

1 引言

用于描述电磁现象的麦克斯韦方程在当前的光学等研究领域中受到广泛关注。目前，麦克斯韦方程的一个应用实例可见于汉诺威莱布尼茨大学的卓越集群 PhoenixD（光子学 光学 工程学 跨学科创新）[1]，该集群正在开发用于光学模拟的现代方法。其中，一个重点是高效且准确地计算光学材料中的光分布，以设计微纳尺度的光学器件 [35, 26]。与固体力学或流体流动中的其他偏微分方程相比，麦克斯韦方程具有一些特殊性，例如旋度算子在二维问题中具有一维像，而在三维问题中具有三维像。此外，对离散化以及最终线性系统定性所提出的要求也具有特殊性。更具体地说，在数值数学中，麦克斯韦方程之所以受到关注，是因为其特定的数学结构[27, 9, 25, 33], 、对有限元的要求 [27, 25, 28, 31, 21, 7, 29], 、其数值求解[18, 16, 14, 10] ，以及后处理过程（如后验误差控制和自适应性） [34, 6]。由于其病态性，其数值求解颇具挑战性，例如， [4], 因此必须采用适当的技术。文献中最主要的方法基于区域分解（DD）技术 [36, 10]。Hiptmair 开发的几何多重网格求解器 [18] 只能应用于时域问题（即适定问题）。

[1]https://www.phoenixd.uni-hannover.de/en/

在本工作中，我们专注于使用区域分解方法进行数值求解。具体而言，我们的出发点是 [4], 中开发的方法，该方法基于 [11], 中的思想，并在现代开源有限元库 deal.II [1, 2]中实现。区域分解方法的关键在于界面算子的推导 [11]。本工作的主要目标是设计一个概念验证，借助前馈神经网络（神经网络）来近似界面算子 [5, 17, 20]。我们仔细推导控制算法，并聚焦于一个双区域问题，以研究新方法的机制和性能。在实现方面，前述 deal.II 库（使用 C++）与 PyTorch（使用 Python） [32]库相耦合；PyTorch 是用于神经网络计算的标准软件包之一。我们的主要目的是展示该方法的可行性，并说明其可以作为未来进一步扩展的出发点。我们注意到，当前工作是会议论文集论文 [22]的扩展，其中包含更多数学和算法细节，以及不同的数值测试，具体包括对不同波数的研究和两种神经网络激活函数的比较。

本文的结构如下：第 2 节介绍时谐麦克斯韦方程及本文采用的记号。接着，第 3 节介绍区域分解和神经网络近似。随后，第 4 节详细讨论训练过程。第 5 节通过一些数值测试展示概念验证。最后，第 6 节总结本文工作。

2 方程

为简化起见，我们仅考虑二维时谐麦克斯韦方程。下面将详细介绍这些方程。

2.1 基本算子

为全面描述该问题，我们介绍描述二维电磁问题所需的基本算子。因此，设标量函数 $\phi: \mathbb{R} \to \mathbb{R}$ 和$\vec{v} \in \mathbb{R}^2$ 为二维向量。那么， ϕ的梯度定义为 $\nabla\phi = \begin{pmatrix} \frac{\partial\phi}{\partial x_1}, & \frac{\partial\phi}{\partial x_2} \end{pmatrix}$, 而 v的散度定义为 $\mathrm{div}(v) := \nabla \cdot v := \sum_{i=1}^{2} \frac{\partial v_i}{\partial x_i}$. 接下来， $a \cdot b = (a_1, a_2)^T \cdot (b_1, b_2)^T = a_1 b_1 + a_2 b_2$表示标量积。此外，我们还可以写出二维旋度算子的表达式

$$\mathrm{curl}(\vec{v}) = \frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{\partial x_2}, \tag{1}$$

以及作用于标量函数的旋度算子

$$\underline{\mathrm{curl}}(\phi) = \begin{pmatrix} \frac{\partial\phi}{\partial x_2} \\ -\frac{\partial\phi}{\partial x_1} \end{pmatrix}. \tag{2}$$

2.2 时间谐波麦克斯韦方程

设 $\Omega \subset \mathbb{R}^2$ 是边界 Γ足够光滑的有界区域。该边界划分为$\Gamma = \Gamma^{\infty} \cup \Gamma^{\mathrm{inc}}$。主要的控制函数空间定义为

$$H(\mathrm{curl}, \Omega) := \{v \in \mathcal{L}^2(\Omega) \mid \mathrm{curl}(v) \in \mathcal{L}^2(\Omega)\},$$

其中 $\mathcal{L}^2(\Omega)$ 是勒贝格意义下众所周知的平方可积函数空间。为定义边界条件，我们引入迹

$$\gamma^t : H(\mathrm{curl}, \Omega) \to H_{\times}^{-1/2}(\mathrm{div}, \Gamma),$$
$$\gamma^T : H(\mathrm{curl}, \Omega) \to H_{\times}^{-1/2}(\mathrm{curl}, \Gamma),$$

其定义为

$$\gamma^t(\phi) = \begin{pmatrix} \phi\, n_2 \\ -\phi\, n_1 \end{pmatrix} \quad \text{and} \quad \gamma^T(v) = v - (n \cdot v) \cdot n,$$

其中， $n \in \mathbb{R}^2$ 是 Ω 的法向量， $H_{\times}^{-1/2}(\text{div } \Gamma) := \{v \in H^{-1/2}(\Gamma) \mid v \cdot n = 0, \ \text{div}_{\Gamma} v \in H^{-1/2}(\Gamma)\}$是定义良好的曲面散度场空间，而 $H(\text{curl } \Gamma) := \{v \in H^{-1/2}(\Gamma) \mid v \cdot n = 0, \ \text{curl}_{\Gamma}(v) \in H^{-1/2}(\Gamma)\}$ 是定义良好的曲面旋度空间，参见 [27, 第 3 章。4]。下文首先给出方程组的强形式。时谐麦克斯韦方程定义如下：求电场 $E: \Omega \to \mathbb{C}^2$，使得

$$\begin{cases} \underline{\text{curl}}\left(\mu^{-1} \text{curl}\left(\vec{E}\right)\right) - \varepsilon\omega^2 \vec{E} & = \vec{0} \quad \text{in } \Omega \\ \mu^{-1}\gamma^t\left(\text{curl}\left(\vec{E}\right)\right) - i\kappa\omega\gamma^T\left(\vec{E}\right) & = \vec{0} \quad \text{on } \Gamma^{\infty} \\ \gamma^T\left(\vec{E}\right) & = \vec{E}^{\text{inc}} \quad \text{on } \Gamma^{\text{inc}}, \end{cases} \tag{3}$$

其中， $\vec{E}^{\text{inc}}: \mathbb{R}^2 \to \mathbb{C}^2$ 是给定的入射电场， $\mu \in \mathbb{R}^+$ 是相对磁导率，$\kappa = \sqrt{\varepsilon}$、 $\varepsilon \in \mathbb{C}$ 是相对介电常数，$\omega = \frac{2\pi}{\lambda}$ 是波数， $\lambda \in \mathbb{R}^+$ 是波长， i表示虚数。方程组 (3) 及其弱形式称为时谐形式，因为其时间依赖性可表示为 $e^{i\omega\tau}$，其中 $\tau \geq 0$ 表示时间。

2.3 弱形式

本小节中，我们推导弱形式。这是有限元方法（FEM）离散化的起点。推导时，我们首先借助分部积分重写旋度乘积:

$$\int_{\Omega} \underline{\text{curl}}\,(\phi) \cdot \vec{u} \ \mathrm{d}x = \int_{\Omega} \phi \,\text{curl}\,(u) \ \mathrm{d}x + \int_{\partial\Omega} \gamma^t(\phi) \cdot u \ \mathrm{d}s, \tag{4}$$

例如，参见 [15, 27]。接下来，我们要从强形式 (3) 推导弱形式:

$$\begin{aligned} &\int_{\Omega} \underline{\text{curl}}\left(\mu^{-1} \text{curl}\left(\vec{E}\right)\right) \cdot \vec{\varphi} \ \mathrm{d}x - \varepsilon\omega^2 \int_{\Omega} \vec{E} \cdot \vec{\varphi} \ \mathrm{d}x = \vec{0}, \\ \overset{(4)}{\Rightarrow} &\int_{\Omega} \mu^{-1} \text{curl}\left(\vec{E}\right) \text{curl}\,(\vec{\varphi}) \ \mathrm{d}x - \varepsilon\omega^2 \int_{\Omega} \vec{E} \cdot \vec{\varphi} \ \mathrm{d}x + \int_{\partial\Omega} \mu^{-1}\gamma^t\left(\text{curl}\left(\vec{E}\right)\right) \cdot \vec{\varphi} \ \mathrm{d}s = \vec{0}. \end{aligned} \tag{5}$$

将方程 (3) 中边界 Γ^{∞} 和 Γ^{inc} 的定义应用于方程 (5)，即可得到时谐麦克斯韦方程的弱形式。求 $\vec{E} \in H(curl,\Omega)$，使得对于所有 $\vec{\varphi} \in H(curl,\Omega)$

$$\begin{aligned} \int_{\Omega} \left(\mu^{-1} \text{curl}\left(\vec{E}\right) \text{curl}\,(\vec{\varphi}) - \varepsilon\omega^2 \vec{E} \cdot \vec{\varphi}\right) \mathrm{d}x + i\kappa\omega \int_{\Gamma^{\infty}} \gamma^T\left(\vec{E}\right) \cdot \gamma^T(\vec{\varphi}) \ \mathrm{d}s \\ = \int_{\Gamma^{\text{inc}}} \gamma^T\left(\vec{E}^{\text{inc}}\right) \cdot \gamma^T(\vec{\varphi}) \ \mathrm{d}s. \end{aligned} \tag{6}$$

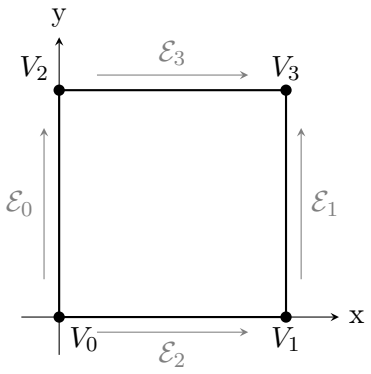
2.4 二维Nédélec元

为了借助 Galerkin有限元方法（有限元方法）进行实现，我们需要离散弱形式。根据 De-Rham上同调，我们必须从 Nédélec空间 V_h中选择基函数。因此，下面我们将基于 Zaglmayr [37, 第5章.2]介绍空间 V_h 的定义。

作为合适的多项式基，我们引入积分Legendre多项式。令 $x \in [-1, 1]$。以下递归公式定义了积分Legendre多项式:

$$\begin{aligned} L_1(x) &= x, \\ L_2(x) &= \tfrac{1}{2}\left(x^2 - 1\right), \\ (n+1)L_{n+1}(x) &= (2n-1)xL_n(x) - (n-2)L_{n-1}(x), \quad \text{for } n \geq 2. \end{aligned} \tag{7}$$

我们选择四边形参考单元作为 $Q = [0, 1] \times [0, 1]$。



$$\begin{aligned}
\lambda_0 &= (1-x)(1-y), & \sigma_0 &= (1-x)+(1-y),\\
\lambda_1 &= x(1-y), & \sigma_1 &= x+(1-y),\\
\lambda_2 &= (1-x)y, & \sigma_2 &= (1-x)+y,\\
\lambda_3 &= xy, & \sigma_3 &= x+y
\end{aligned}$$

图1：左：参考单元上的顶点和边的排序；右：参考单元的参数化。

接下来定义所有边的集合 $\mathrm{E} = \{\mathcal{E}_m\}_{0\le m<4}$，其局部边排序为 $\mathcal{E}_m = \{V_i, V_j\}$，其中 $(i,\ j) \in \{(0,2), (1,3), (0,1), (2,3)\}$，见图1。我们用局部顶点排序 $C = \{V_0, V_1, V_2, V_3\}$ 表示单元本身。多项式阶数为 $\vec{p} = \left(\{p_\mathcal{E}\}_{\mathcal{E}\in\mathfrak{C}},\ p_C\right)$。

$H(curl)$**协调基函数**	
基于顶点的形函数	
顶点上没有自由度。	
基于边的形函数 对于 $0 \le i < p_\mathcal{E}$, $\mathcal{E} \in \mathrm{E}$，其中 λ_α 和 σ_α, $\alpha \in \{0,1,2,3\}$ 定义于图1中	
最低阶	$\varphi_{\mathcal{E}_m}^{\mathcal{N}_0} = \frac{1}{2}\nabla(\sigma_{e_2} - \sigma_{e_1})(\lambda_{e_1} + \lambda_{e_2})$
高阶	$\varphi_i^{\mathcal{E}_m} = \nabla(L_{i+2}(\sigma_{e_2} - \sigma_{e_1})(\lambda_{e_1} + \lambda_{e_2}))$
基于单元的函数 $0 \le i,\ j < p_C$	
类型 1:	$\varphi_{(i,\ j)}^{C,1} = \nabla(L_{i+2}(\xi_F)L_{j+2}(\eta_F))$
类型 2:	$\varphi^{C,\ 2\ (i,\ j)} = \widetilde{\nabla}\left(L_{i+2}\left(\xi_F\right)\ L_{j+2}\left(\eta_F\right)\right)$ 其中 $\widetilde{\nabla}(a\ b) := (a'\ b - a\ b')$
类型 3:	$\varphi_{(0,j)}^{C,3} = L_{i+2}(2y-1)\vec{e}_x$ $\varphi_{(i,\ 0)}^{C,3} = L_{i+2}(2x-1)\vec{e}_y$

图2：参考单元上 H(curl) 基函数的定义，其重心坐标为 λ_α 和 σ_α，$\alpha \in \{0,1,2,3\}$。

借助这些基函数，我们定义二维 Nédélec 空间

$$V_h := V_h^{\mathcal{N}_0}(\mathcal{T}_h) \bigoplus_{\mathcal{E}\in\mathfrak{C}} V_h^{\mathcal{E}}(\mathcal{T}_h) \bigoplus_{C\in\mathcal{C}} V_h^{C}(\mathcal{T}_h), \tag{8}$$

其中，$V_h^{\mathcal{N}_0}$是最低阶 ***Nédélec*** 函数的空间，$V_h^{\mathcal{E}}$是边气泡函数的空间，而V_h^{C}是单元气泡函数的空间。图2展示了一个单元上所有具有重心坐标的基函数。图3展示了部分基函数的可视化结果。目前对 $V_h(\Omega)$ 的描述仍不完整；迄今为止，我们仅描述了 $V_h(Q)$，其中 Q 如前所述。接下来介绍皮奥拉变换，它用于将参考单元映射到任意给定的物理单元，参见 Monk [27] （引理3.57、推论3.58）。

2.5 离散弱形式

我们已经具备了写出时谐麦克斯韦方程离散弱形式所需的一切条件。通过对方程（6）应用伽辽金方法，我们得到离散弱形式

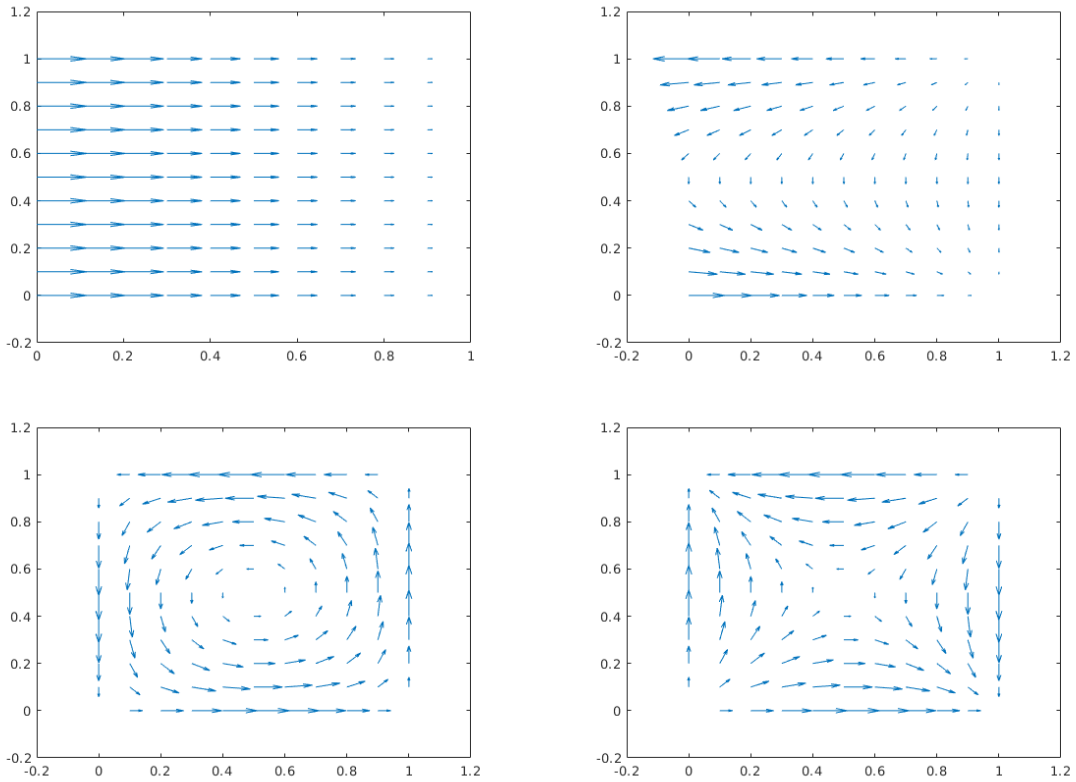


图3：$(0,\ 1)^2$上的基函数图：低阶边函数（左上），从 $p=2$ 到边 $\mathcal{E}_0$的高阶边基函数（右上），类型 1 和 2 的高阶单元基函数 $p=2$ （下方）。

求 $E_h \in V_h(\Omega)$，使得

$$\int_\Omega \left(\mu^{-1}\operatorname{curl}\left(\vec{E}_h\right)\operatorname{curl}\left(\vec{\varphi}_h\right)-\varepsilon\omega^2\vec{E}_h\cdot\vec{\varphi}_h\right)\ \mathrm{d}x+i\kappa\omega\int_{\Gamma^\infty}\gamma^T\left(\vec{E}_h\right)\cdot\gamma^T\left(\vec{\varphi}_h\right)\ \mathrm{d}s$$
$$=\int_{\Gamma^{\mathrm{inc}}}\gamma^T\left(\vec{E}^{\mathrm{inc}}\right)\cdot\gamma^T\left(\vec{\varphi}_h\right)\ \mathrm{d}s\quad\forall\varphi_h\in V_h(\Omega). \tag{9}$$

3 数值方法

本节首先介绍区域分解，然后介绍神经网络近似。后者还将概述如何用神经网络替代界面算子。

3.1 区域分解

由于麦克斯韦方程组 (3) 的解具有挑战性，正如引言中所述，我们采用一种非重叠区域分解方法 (DDM)[36] ，将区域划分为若干子区域，具体如下

$$\overline{\Omega}=\bigcup_{i=0}^{n_{\mathrm{dom}}}\overline{\Omega}_i\quad\text{with}$$
$$\Omega_i\cap\Omega_j=\varnothing\quad\forall i\neq j,$$

其中， $n_{\mathrm{dom}}+1$ 为子区域的数量。这样一来，每个子区域 Ω_i 都变得足够小，从而我们可以使用直接求解器处理它。电场的全局解 E 通过迭代法计算得到，其中我们在每个子区域上求解时谐麦克斯韦方程，并在不同子区域之间施加适当的界面条件。因此，我们得到一个解 E_i^k，适用于

每个子区域 Ω_i，其中 k 表示第 k 个迭代步。初始界面条件由下式给出

$$g_{ji}^{k=0} := -\mu^{-1}\gamma_i^t\left(\operatorname{curl}\left(E_i^{k=0}\right)\right) - i\kappa S\left(\gamma_i^T\left(E_i^{k=0}\right)\right) = 0, \tag{10}$$

其中，S 表示界面算子，i 是当前区域的索引，j 是相邻区域 [11]的索引。随后，在每个更新步骤中，通过求解以下方程组来计算电场E_i^{k+1}

$$\begin{cases} \underline{\operatorname{curl}}\left(\mu^{-1}\operatorname{curl}\left(E_i^{k+1}\right)\right) - \omega^2\varepsilon E_i^{k+1} & = 0 & \text{in } \Omega_i, \\ \mu^{-1}\gamma_i^t\left(\operatorname{curl}\left(E_i^{k+1}\right)\right) - i\omega\kappa\gamma_i^T\left(E_i^{k+1}\right) & = 0 & \text{on } \Gamma_i^{\infty}, \\ \gamma_i^T\left(E_i^{k+1}\right) & = \gamma_i^T\left(E_i^{\text{inc}}\right) & \text{on } \Gamma_i^{\text{inc}}, \\ \mu^{-1}S\left(\gamma_i^t\left(\operatorname{curl}\left(E_i^{k+1}\right)\right)\right) - i\omega\kappa\gamma_i^T\left(E_i^{k+1}\right) & = g_{ji}^k & \text{on } \Sigma_{ij}, \end{cases} \tag{11}$$

其中，$\Sigma_{ij} = \Sigma_{ji} : = \partial\Omega_i \cap \partial\Omega j$ 表示两个相邻单元的界面，界面条件通过以下方式更新

$$g_{ji}^{k+1} = -\mu^{-1}\gamma_i^t\left(\operatorname{curl}\left(E_i^{k+1}\right)\right) - i\kappa S\left(\gamma_i^T\left(E_i^{k+1}\right)\right) = -g_{ij}^k - 2i\kappa S\left(\gamma_i^T\left(E_i^{k+1}\right)\right). \tag{12}$$

如果成功，我们将得到 $\lim_{k\to\infty} E_i^k = E|_{\Omega_i}$，但这种收敛性在很大程度上取决于所选择的界面算子 S （见 [10, 11]）。该方法在 deal.II 中的实现见 [4]。

3.2 我们的新方法： S的神经网络近似

由于计算 S 的良好近似具有挑战性，我们研究了一种新方法，尝试借助神经网络（NN）对该算子进行近似。作为概念验证，我们选择了一个原型示例，并探究神经网络是否能够近似界面值。由于无法计算精确的界面算子 S，我们的目标是使用神经网络计算g_{ij}^{k+l}，$l > 0$，并将 g_{ij}^k 和 E_i^{k+1}作为输入。这种方法的另一项优势是，我们可以从经典区域分解方法中快速生成训练数据集，如第4.4节所述。为简单起见，在我们的经典区域分解方法中选择 $S = \mathbb{1}$。因此，该方法的优势在于，无需在每个更新步骤重新计算方程组（11），即可更新界面条件，这使我们有望降低计算成本。

4 神经网络训练

神经网络近似的第一步是训练过程，本节将对此进行介绍。除了数学实现之外，我们还需要选择软件库。我们使用 deal.II [2]，通过有限元方法对时谐麦克斯韦方程进行离散化。神经网络使用 PyTorch [32]进行训练。deal.II 代码与 PyTorch 代码所得结果之间的信息交换通过硬盘进行。

4.1 基本定义

首先，我们简要定义本文所采用的神经网络类型，并介绍基本参数。更多信息请参见 [5, 8, 24, 3, 30]。本小节所采用的符号和描述主要基于 [23]。

定义 4.1 （人工神经元）。一个（人工）神经元（也称为单元 *[12], [5][第 5.1]*节）u 是如下形式的元组（x,w σ）。各组成部分的含义如下：

- x =（x_0，.....，x_n）$\in \mathbb{R}^{n+1}$ 是输入向量。它包含神经元接收的信息。
- w =（w_0、...、w_n）$\in \mathbb{R}^{n+1}$ 是权重向量，它决定各个输入信息对神经元输出的影响。之后，w 表示所有神经元的权重向量。

- σ ： $\mathbb{R} \to \mathbb{R}$，其中 $\sigma = \sum_{i=0}^{n} x_i w_i \mapsto a$ 是激活函数。它根据输入和权重确定所谓的激活值 a，该激活值表示神经元的输出。

定义 4.2（神经网络）。（人工）神经（前馈）网络是一组神经元 U，并具有互不相交的分解 $U = U_0 \dot\cup \ldots \dot\cup U_l$。划分集合 U_k、$k = 0$、...、l 称为层。这里，U_0 是输入层，其中包含接收外部信息的神经元。此外，U_l 是输出层，其中的神经元负责返回输出。最后，U_1、...、U_{l-1} 是所谓的隐藏层。

从任意神经元 $u \in U_k$ 出发，都有一条连接通向每个神经元 $\hat{}u \in U_{k+1}$，其中 $k = 0$、...、$l-1$。这样的连接表示神经元 u 的输出 a 会传递给神经元 $\hat{}u$。这一特性正是其被称为前馈网络的原因。

每个 U_0、..、U_{l-1} 都包含一个形如 (0、0、1) 的偏置神经元。它没有输入和权重，输出值恒为 1，并且只将一个以权重形式表示的恒定偏置传递给后续层中的每个神经元。

注 4.3. 在以下示例中，U_k 层的所有神经元都使用相同的激活函数，该函数由 $\sigma^{(k)}$ 给出，适用于 $k = 0, \ldots, l$。这里，$D_k := |U_k| - 1$ 表示第 $k = 0, \ldots, l-1$ 层的神经元数量（不包括偏置神经元），而 $D_l := |U_l|$ 表示输出层的神经元数量。

4.2 区域分解

在构建神经网络之前，我们选择区域、分解方式以及用于求解方程组 (11) 以获得训练值的网格，因为它们会影响网络规模。在我们选定的示例中，区域定义如下：

$$\Omega = (0,1) \times (0,1),$$

它被划分为两个子区域

$$\Omega_0 = (0,1) \times (0,0.5) \quad \text{and} \quad \Omega_1 = (0,1) \times (0.5,1) \quad (\text{see Figure 4}),$$

应用有限元方法的网格由 32×32 个具有二次奈德莱克元的单元构成。因此，在这两个区域的界面上，各有 32 个单元，每个单元有 4 个自由度（dofs）

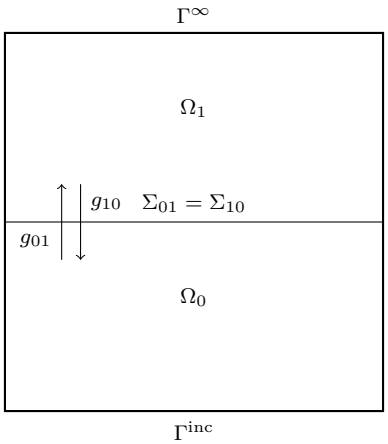


图 4：区域 Ω 及所选分解的可视化。

子区域。我们在每个自由度上评估界面条件和解，并将这些值用作神经网络的输入和目标。因此，输入包含 $4 \cdot \dim(g_{ij}) + 4 \cdot \dim(E_i) = 16$ 个值，输出由 $4 \cdot \dim(g_{ji}) = 8$ 个值组成，一次计算即可得到 32 对输入-目标对。

4.3 神经网络构建

鉴于上述考虑，我们需要一个包含 16 个神经元（无偏置）的输入层，以及一个包含 8 个神经元的输出层。此外，我们使用一个包含 500 个神经元（无偏置）的隐藏层。因此，对于控制网络，我们有

$$U = U_0 \dot{\cup} U_1 \dot{\cup} U_2$$

其中 $D_0 = 16$、$D_1 = 500$ 和 $D_2 = 8$。第 5 节给出的测试结果表明，对于我们的目的而言，这样的规模已经足够。每一层的激活函数选择如下：

$$\begin{aligned}
\sigma^{(0)} &= id \quad \text{(input layer)}, \\
\sigma^{(1)} &= \frac{1}{1+e^{-x}} \quad \text{(hidden layer)}, \\
\sigma^{(2)} &= id \quad \text{(output layer)},
\end{aligned}$$

其中，$\sigma^{(1)}$ 称为 Sigmoid函数。事实证明，它是最有效的函数，因为与我们测试过的其他函数（例如）相比，它能使误差更快地持续降低。

$$\begin{aligned}
\sigma^{(1)} &= \tanh(x), \\
\sigma^{(1)} &= \log\left(\frac{1}{1+e^{-x}}\right) \quad \text{(LogSigmoid)}, \\
\sigma^{(1)} &= \max(0,x) + \min(0, e^x - 1) \quad \text{(CELU)}, \\
\sigma^{(1)} &= a\left(\max(0,x) + \min\left(0, b\left(e^x - 1\right)\right)\right) \quad \text{(SELU)}, \\
&\quad \text{with } a \approx 1.0507, \text{ and } b \approx 1.6733.
\end{aligned}$$

一个例外是 ReLU函数，我们将在第 5.4 节中讨论。此外，我们为两种界面条件 g_{01} 和 g_{10} 分别采用形状相同的独立网络 U^{01} 和 U^{10}，因为事实证明，两者的近似方式不同，但都能快速、准确地完成近似。生成的程序代码如图 5 所示。

```
导入 torch.nn 作为nn
导入 torch.nn.functional 作为F
类 Maxwell(nn.Module):

    def __init__(自身):
        超级(Maxwell, 自身)。__初始化__()
        自身。线性1 = 神经网络。线性（16 , 500)
        自身.lin2 = nn.Linear(500 , 8)


    定义 前向传播(自身 , x):
        x  = torch.Sigmoid函数(self.lin1(x))
        #x  =F.relu 激活函数(self.lin1(x))
        x  = 自身。lin2(x)
        返回 x

网络 =Maxwell()
打印(网络)
```

图 5：网络构建实现的 PyTorch 代码。

4.4 训练

为了获得足够的训练数据，我们改变边界条件 E^{inc}，并创建训练值和测试值，以便在训练期间监控网络并避免过拟合。训练集和测试集由表1中列出的边界值生成。

E^{inc} 用于训练集		E^{inc} 用于测试集
$\begin{pmatrix} e^{\frac{-(x-0.7)^2}{0.008}} \\ 0 \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \sin(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	$\begin{pmatrix} e^{\frac{-(x-0.5)^2}{0.003}} \\ 0 \end{pmatrix}$
$\begin{pmatrix} e^{\frac{-(x-0.2)^2}{0.002}} \\ 1 \end{pmatrix}$	$\begin{pmatrix} \sin(\pi^2 x) + \sin(\pi^2 x)i \\ \sin(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \cos(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$
$\begin{pmatrix} e^{\frac{-(x-0.7)^2}{0.003}} \\ 1 \end{pmatrix}$	$\begin{pmatrix} \sin(\pi^2 x) + \sin(\pi^2 x)i \\ \sin(\pi^2 x) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	
$\begin{pmatrix} e^{\frac{-(x-0.8^{\ 2})}{0.003}} \\ \sin(\pi^2 x) \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \cos(\pi^2 x) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	
$\begin{pmatrix} e^{\frac{-(x-0.5^{\ 2})}{0.003}} \\ \cos(\pi^2 x) \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 x) + \sin(\pi^2 x)i \\ \cos(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	

表1：用于生成训练集和测试集的边界值

由于我们为训练集选择了10个不同的边界值，为测试集选择了2个，并且每个边界值都会生成一组包含32个训练值/测试值的数据（界面上的每个元素对应一个值），因此两个网络总共分别获得一组 $32 \cdot 10 = 320$ 个训练值和一组 $32 \cdot 2 = 64$ 个测试值。为简化第一组测试中的计算，我们选择较小的波数 $\omega = \frac{2\pi}{3}$，并通过迭代式DDM分4个更新步骤计算这些数据集。之后，我们使用结果 $\left(g_{ij}^1, E_i^2\right)$ 作为输入、g_{ji}^3 作为目标来训练神经网络，并采用均方误差作为损失函数，其定义为

$$\text{Loss}(\mathtt{w}) = \frac{1}{2}\sum_{i=1}^{N} \|t^{(i)} - y(\mathtt{x}^{(i)}, \mathtt{w})\|^2,$$

其中，N 表示输入-目标对的数量（在我们的案例中，训练集为 $N = 320$，测试集为 $N = 64$），$t^{(i)}$ 为目标向量，y 是网络生成的函数，因此 $y(\mathtt{x}^{(i)}, \mathtt{w})$ 表示神经网络的输出。具体实现请参阅第5.1节。

我们使用 Adam算法 [19], 作为优化器，它是一种基于以下迭代规则的线搜索方法

$$x^{\rho+1} = x^{\rho} + \alpha^{\rho} p^{\rho},$$

其中，p^{ρ} 称为搜索方向，α^{ρ} 是迭代步 ρ的步长（对于神经网络，也称为学习率）。Adam算法的搜索方向取决于四个参数 β_1、β_2、m_1和 m_2，其中 β_1 和 β_2 是区间 $[0, 1)$ 内的固定值，而 m_1 和 m_2 则在每个更新步骤中通过以下方式更新

$$\begin{aligned} m_1^0 = m_2^0 = 0, \quad & m_1^{\rho+1} = \beta_1 m_1^{\rho} + (1-\beta_1)\cdot \nabla \text{Loss}(\mathtt{w})^{\rho} \\ \text{and} \quad & m_2^{\rho+1} = \beta_2 m_2^{\rho} + (1-\beta_2)\cdot \|\nabla \text{Loss}(\mathtt{w})^{\rho}\|^2. \end{aligned}$$

因此，搜索方向为

$$p^{\rho-1} = -\widehat{m_1^{\rho}} \Big/ \sqrt{\widehat{m_2^{\rho}} + \varepsilon}$$

$\widehat{\ }$，其中 $m_1^{\rho} = m_1^{\rho}/\left(1 - (\beta_1)^{\rho}\right)$、$m_2^{\rho} = m_2^{\rho}/\left(1 - (\beta_2)^{\rho}\right)$ 以及 $0 < \varepsilon \ll 1$。

图 6 展示了该训练过程在 PyTorch 中的实现。

```
导入torch.optim作为optim
导入time
开始_时间 =time。time()
容差 = 3e-3
max_迭代 = 20000
迭代次数 = 0
loss_测试 = 容差 + 1
优化器 = optim.Adam(net.parameters(),lr=1e-5)
损失函数 = nn.均方误差损失()
当(迭代次数 < max_iter 且 损失_测试 > 容差):

    输出 = 网络(输入_训练)
    优化器.zero_grad()
    损失 = 损失函数(out, 目标_训练)
    损失_测试 = 损失函数(网络(inp_测试), 目标_测试)
    打印("损失: %.5f" %损失,    "测试-损失: %.5f"% 损失_测试))
    损失.反向传播()
    优化器.更新步骤()
    迭代次数 += 1
打印("最终损失: %.5f" % 损失_测试)
打印("迭代次数: %.0f"% 迭代次数)
时间_耗时 =时间.时间()-开始_时间
打印("运行时间: %.4f秒" % 时间_耗时)
```

图 6：网络训练实现的 PyTorch 代码。

网络 U^{01} 以学习率 10^{-5}进行训练。经过 29 843 个训练步数后，初始训练误差 3.12 和测试误差 5.87 分别降至 $1.7 \cdot 10^{-4}$ 和 $3 \cdot 10^{-3}$。在 U^{10} 时，初始训练误差 0.72 和测试误差 1.28 在以学习率 10^{-5} 经过 20 326 个更新步骤后分别降至 $3 \cdot 10^{-4}$ 和 $4 \cdot 10^{-3}$；随后以学习率 10^{-6} 继续训练 3706 个更新步骤，最终得到训练误差 $2.9 \cdot 10^{-4}$ 和测试误差 $3 \cdot 10^{-3}$。

5 数值测试

本节通过若干数值实验，展示我们的方法目前具备的能力。此外，我们还将指出并分析其不足与挑战。

5.1 新方法与经典 DDM 的比较

在第一个数值示例中，我们将已实现并训练好的神经网络应用于以下边界条件

$$E^{\mathrm{inc}}(x,y) = \begin{pmatrix} \cos\left(\pi^2\left(y-0.5\right)\right) + \sin\left(\pi^2 x\right) i \\ \cos\left(\pi^2 y\right) + 0.5\sin\left(\pi^2 x\right) i \end{pmatrix},$$

并通过使用表2中给出的参数求解（11）和（12），计算初始界面条件 g_{10}^1 和 g_{01}^1 以及解 E_1^2 和 E_0^2。随后，将这些值传递给网络 U^{01}和 U^{10}。网络返回的输出随后被视为新的界面条件，并用其再次求解方程组（11）。由此，我们得到最终解。此外，我们还使用 DDM 分4个更新步骤计算了同一示例。图7所示的结果显示出极好的一致性。

参数	定义	值
μ	相对磁导率	1.00
ε	相对介电常数	1.49^2
κ		$\sqrt{\varepsilon} = 1.49$
λ	波长	3.00
ω	波数	$\frac{2\pi}{\lambda} = \frac{2\pi}{3.00}$
	网格尺寸	$\frac{1}{32}$

表2：DDM 的参数

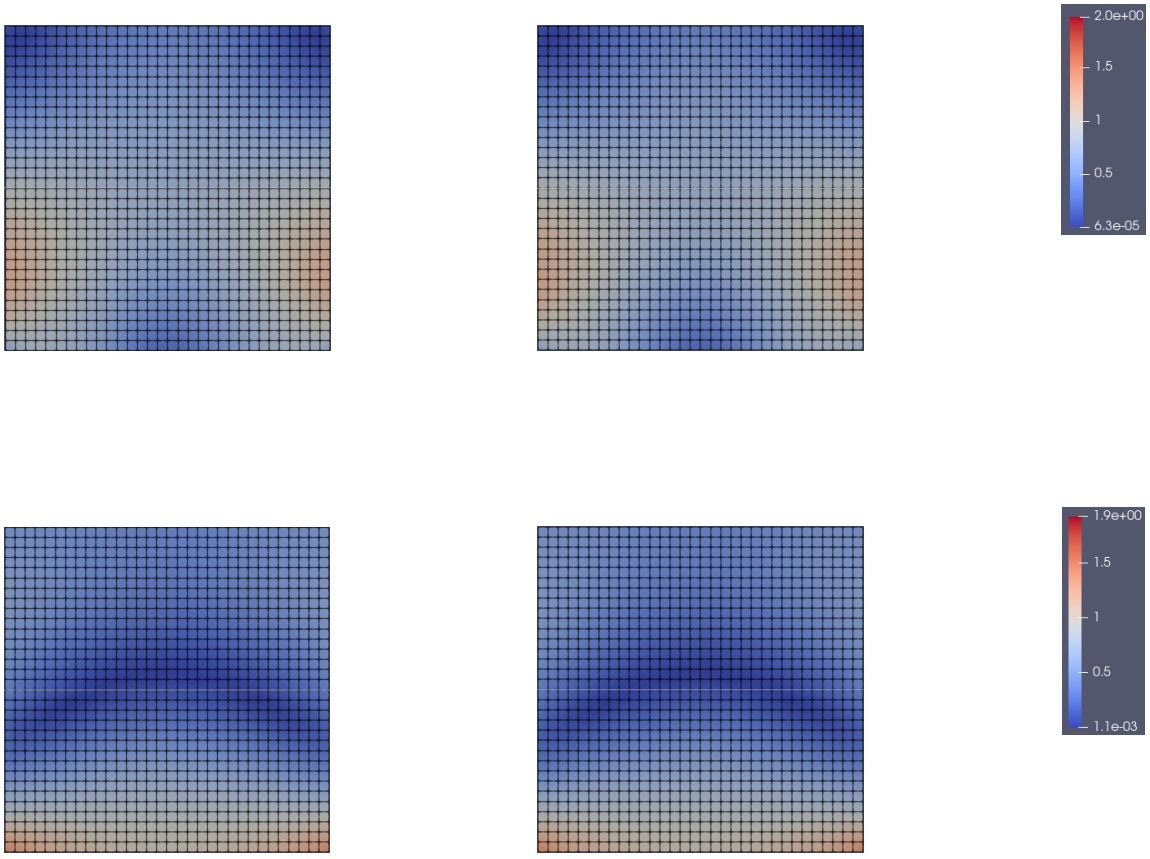


图7：第一个示例：神经网络解（左）和区域分解法解（右）的实部（上）和虚部（下）。

5.2 更高波数

作为第二个示例，我们增大波数，这会导致问题更加复杂。因此，我们使用 $\omega = \pi$ 重复相同的计算，并保持其他参数（尤其是神经网络的参数和超参数）不变。与前一个示例相比，图8所示的结果存在差异。虽然虚部仍得到了较好的近似，但神经网络解的实部与区域分解法解存在显著差异，并且在界面处表现出不连续性。

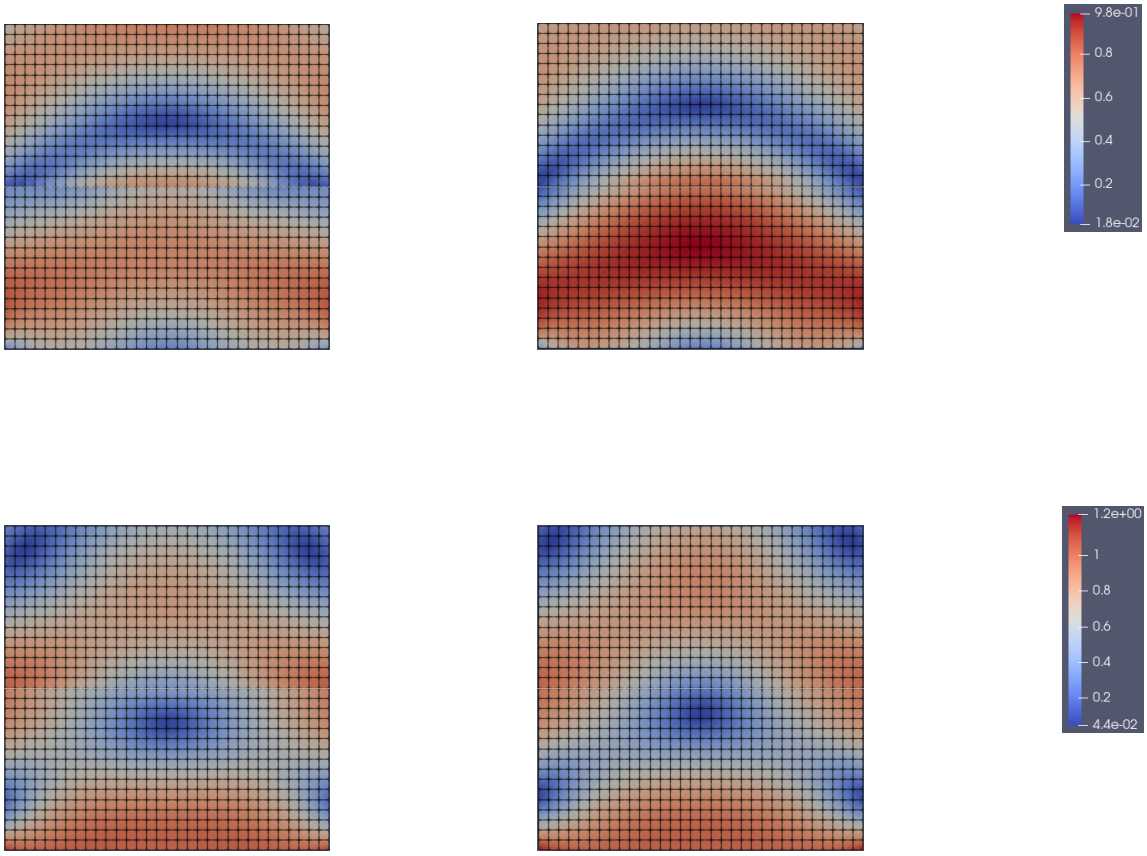


图8：第二个示例：神经网络解（左）和区域分解法解（右）的实部（上）与虚部（下）。

5.3 中间波数的精细计算分析

第5.2节中结果不匹配的一种可能原因是“波数过大问题”，这一问题在亥姆霍兹型问题中已得到充分研究 [13]。同样的问题也适用于麦克斯韦方程 [4]。为验证这一猜想，并鉴于第5.1节和第5.2节的结果截然不同，我们尝试使用另外两个波数进行计算，即 $\omega = \frac{2\pi}{2.9}$ 和 $\omega = \frac{2\pi}{3.1}$。图9和图10展示了结果。为使差异更加明显，我们省略了网格的表示。结果表明，无论波数略大于还是略小于训练时使用的波数，近似都会变得不准确。因此，近似效果不佳并非由波数过大造成。相反，可以认为神经网络针对训练时使用的特定波数进行了专门化，并在训练过程中“学习并适应”这一数值。

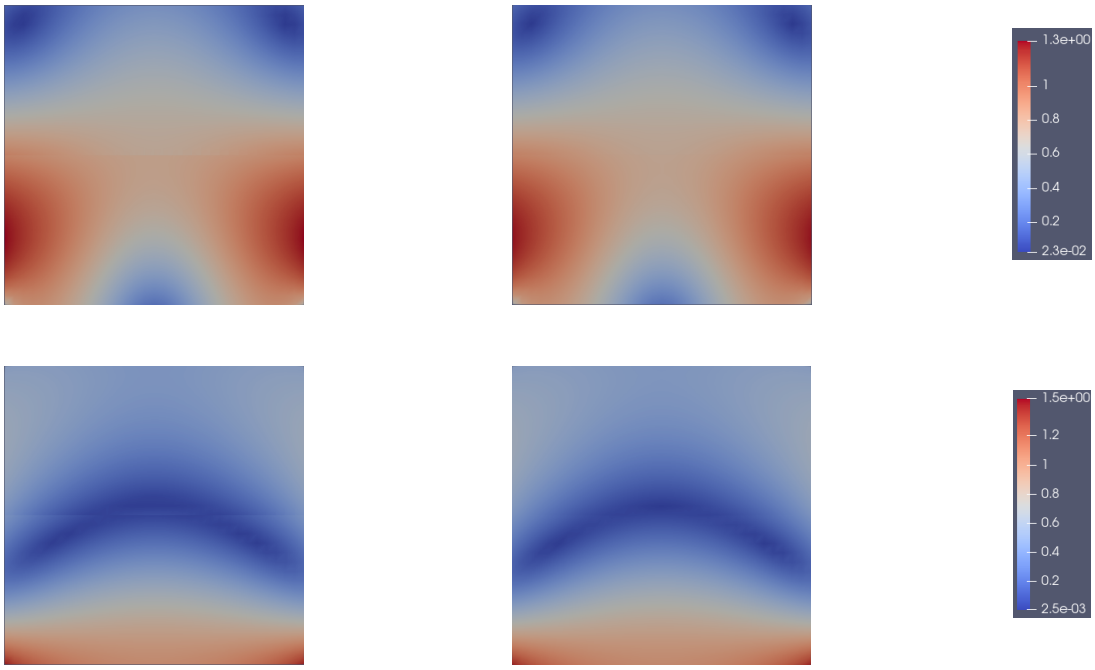


图9：第三个算例：神经网络解（左）和区域分解法解（右）的实部（上）与虚部（下），其中 $\omega = \frac{2\pi}{2.9}$ —

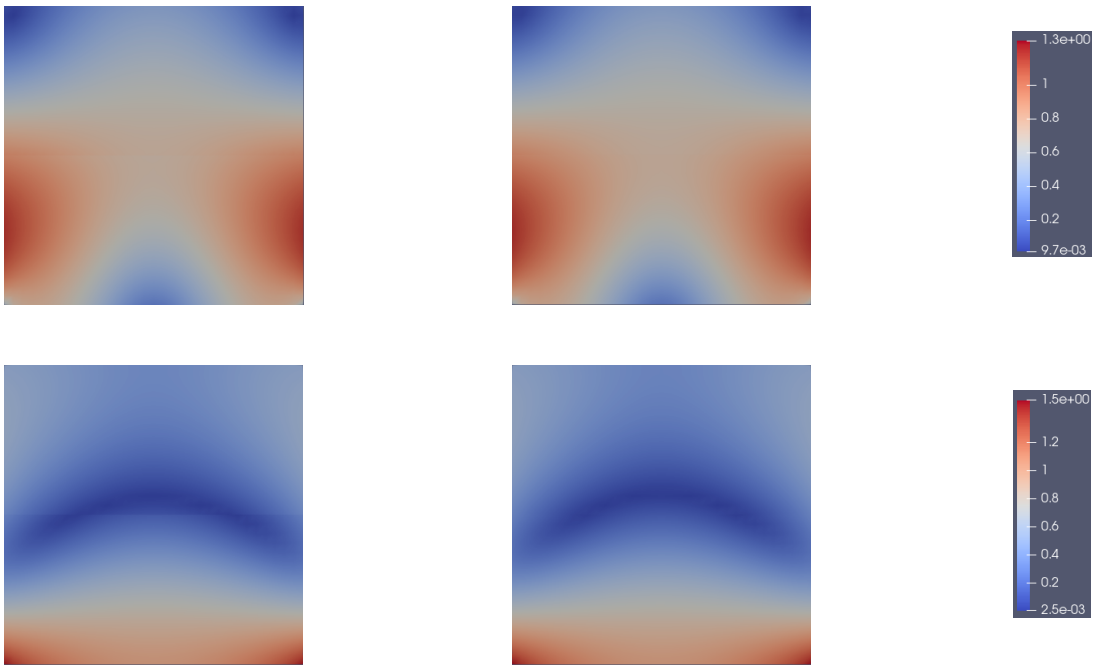


图10：第四个算例：神经网络解（左）和区域分解法解（右）的实部（上）与虚部（下），其中 $\omega = \frac{2\pi}{3.1}$。 —

5.4 不同神经网络激活函数的比较：Sigmoid函数与ReLU函数

如第4.3节所述，在使用 Sigmoid函数之前，我们测试了不同的激活函数来训练神经网络。其中之一是下式给出的 ReLU函数：

$$f(x) = \max(0, x),$$

该函数在 PyTorch 类 torch.nn.functional 中实现。与我们测试过的其他函数（包括 Sigmoid函数）相比，它能够更大幅度、更快速地降低误差。在大多数情况下，使用学习率 10^{-5} 时，网络 U^{01} 的测试误差约经过 16000 个更新步骤后即可降低；使用学习率 10^{-6} 时，约经过 6500 个更新步骤即可将测试误差降至 $8 \cdot 10^{-4}$。与采用 Sigmoid函数作为激活函数训练同一神经网络时最终得到的误差相比，这几乎只有四分之一（见第4.4节）。此外， U^{10} 的测试误差也能更快降低：使用学习率 10^{-5}时，约经过 3500 个更新步骤即可降至 $2 \cdot 10^{-3}$。不过，我们也观察到，在其他情况下，测试误差经过短暂的下降阶段后会有所上升。与此相反，训练误差仍持续减小，这表明我们的 ReLU网络更容易过拟合。当我们按照第5.1节所述的相同步骤，将训练成功的 ReLU网络应用于第一个示例时，这一猜测得到了进一步印证。图11所示的结果显示，界面处存在不连续性。这表明，即使在测试误差大幅降低的幸运情况下，我们遇到的仍是过拟合，最终得到的神经网络无法准确捕捉实际问题。鉴于 ReLU神经网络的训练不稳定，采用 Sigmoid函数作为激活函数是合理的。

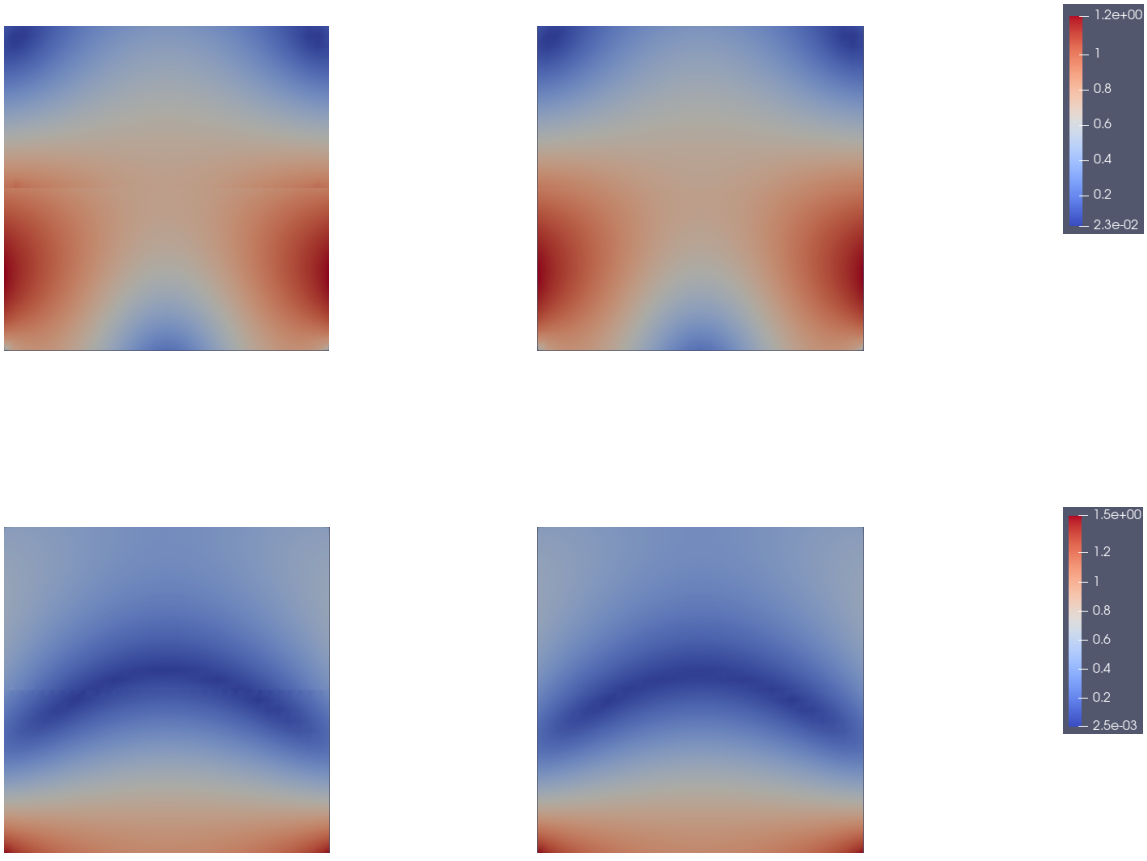


图11：使用 ReLU函数（左）和 Sigmoid函数（右）作为激活函数时，神经网络解的实部（上）和虚部（下）。

6 结论

本文对利用前馈神经网络近似区域分解中的界面算子进行了概念验证和可行性研究。这些概念应用于时谐麦克斯韦方程。我们从算法和实现两个角度，对数值框架进行了细致说明。在实现中，我们将用于求解麦克斯韦方程的 deal.II（C++）与用于神经网络求解的 PyTorch 相结合。随后，我们进行了多项数值测试，包括将这一新方法与经典区域分解方法进行比较。接着，我们更详细地研究了较高波数的情况。在此过程中，我们发现了一些困难，并进一步进行了研究，结果表明，神经网络的训练和测试对具体波数高度敏感。最后，我们比较了两种不同的神经网络激活函数。展望未来，我们计划增加子区域的数量，以进一步研究其他波数，并将该方法应用于三维麦克斯韦方程。

致谢

本研究由德国研究联合会（DFG）通过德国卓越战略下的卓越集群 PhoenixD（EXC 2122，项目编号 390833453）资助。

参考文献

[1] D. Arndt、W. Bangerth、D. Davydov、T. Heister、L. Heltai、M. Kronbichler、M. Maier、J.-P. Pelteret、B. Turcksin 和 D. Wells。《deal.II 有限元库：设计、特性与见解》。*Computers & Mathematics with Applications*，2020年。[2] D. Arndt、W. Bangerth、M. Feder、M. Fehling、R. Gassmöller、T. Heister、L. Heltai、M. Kron- bichler、M. Maier、P. Munch、J.-P. Pelteret、S. Sticko、B. Turcksin 和 D. Wells。《deal.II 库，9.4 版》。*Journal of Numerical Mathematics*，30(3):231–246，2022年。[3] S. Ben-David 和 S. Shalev-Shwartz。《理解机器学习：从理论到算法》。剑桥大学出版社；剑桥，2014年。[4] S. Beuchler、S. Kinnewig 和 T. Wick。《时谐麦克斯韦方程的并行区域分解求解器》，载于《计算科学与工程讲义》第145卷，第615–622页。施普林格，2023年。[5] C. M. Bishop。《模式识别与机器学习》。施普林格，2006年。[6] M. Bürg。麦克斯韦方程 hp 有限元方法的基于残差的后验误差估计器。*Applied Numerical Mathematics*，62:922–940，2012年8月。[7] C. Carstensen、L. Demkowicz 和 J. Gopalakrishnan。DPG 方法的断裂空间与微分形式及其应用（包括麦克斯韦方程）。*Computers & Mathematics with Applica-tions*，72(3):494–522，2016年。[8] S. Copony。《神经网络中的动态行为》。汉堡大学，2007年。[9] L. Demkowicz。《hp 自适应有限元计算。第1卷：一维和二维椭圆问题与麦克斯韦问题》。Chapman and Hall/CRC，2006年。[10] V. Dolean、M. Gander 和 L. Gerardo-Giorda。麦克斯韦方程的优化 Schwarz 方法。31(3):2193–2213。出版方：工业与应用数学学会。[11] M. El Bouajaji、B. Thierry、X. Antoine 和 C. Geuzaine。时谐麦克斯韦方程的准最优区域分解算法。*Journal of Computational Physics*，294:28–57，2015。

[12] S. Ellacott。神经网络数值分析的若干方面。*Acta Numerica*，3:145–202，1994年。[13] O. G. Ernst 和 M. J. Gander。《为何经典迭代方法难以求解亥姆霍兹问题》，第325–363页。《计算科学与工程讲义》。施普林格。[14] M. Faustmann、J. M. Melenk 和 M. Parvizi。时谐麦克斯韦方程有限元矩阵逆的 $\mathcal{H}$-矩阵可逼近性。计算数学进展，48(5)，2022年。[15] V. Girault 和 P.-A. Raviart。《纳维–斯托克斯方程的有限元方法》，《施普林格计算数学丛书》第5卷。施普林格出版社，柏林，1986年。理论与算法。[16] S. Henneking 和 L. Demkowicz。长波导仿真的污染误差与 DPG 自适应性数值研究。*Computers & Mathematics with Applications*，95:85–100，2021年。[17] C. F. Higham 和 D. J. Higham。深度学习：应用数学家的入门介绍。《*SIAM*评论》，61(4):860–891，2019年。[18] R. Hiptmair。麦克斯韦方程的多重网格方法。《*SIAM*数值分析杂志》，36(1):204–225，1998年。[19] D. P. Kingma 和 J. Ba。Adam：一种随机优化方法。载于 Y. Bengio 和 Y. LeCun 编，《第三届国际学习表征会议，ICLR 2015，圣迭戈（美国加利福尼亚州），2015年5月7日至9日，会议论文集》。[20] S. Kinnewig、L. Kolditz、J. Roth 和 T. Wick。《算法系统与神经网络的数值方法》。汉诺威：莱布尼茨汉诺威大学机构知识库，讲义。莱布尼茨汉诺威大学应用数学研究所，2022年。[21] S. Kinnewig、T. Wick 和 S. Beuchler。解决 hp 六面体 Nédélec 元悬挂节点的符号冲突问题，2023年。准备中。[22] T. Knoke、S. Kinnewig、T. Wick 和 S. Beuchler。应用于麦克斯韦方程的区域分解方法中的神经网络界面条件近似，2022年。审稿中。[23] T. Knoke 和 T. Wick。通过人工神经网络求解微分方程：模型问题中的发现与失败。《例子与反例》，1:100035，2021年。[24]D. Kriesel。《神经网络简明入门》。http://www.dkriesel.com/en/science/neural_networks，2005年。[25] U. Langer、D. Pauly 和 S. Repin（编）。《麦克斯韦方程：分析与数值计算》。De Gruyter，2019年。[26] O. Melchert、S. Kinnewig、F. Dencker、D. Perevoznik、S. Willms、I. Babushkin、M. Wurz、M. Kues、S. Beuchler、T. Wick、U. Morgner 和 A. Demircan。非线性金刚石光子学中的孤子压缩与超连续谱。2022年，arXiv 预印本 arXiv:2211.00492。[27] P. Monk。《麦克斯韦方程的有限元方法》。牛津科学出版物，2003年。[28] J. C. Nédélec。$\mathbb{R}3$中的混合有限元。《数值数学》，35(3):315–341，1980年9月。[29] S. Nicaise 和 J. Tomezyk。《9. 多面体区域中带阻抗边界条件的时谐麦克斯韦方程》，第285–340页。De Gruyter，柏林、波士顿，2019年。

[30] A. Nielson. 神经网络与深度学习. Determination Press, 2015.

[31] J. C. Nédélec. ℝ3中的一类新型混合有限元。《数值数学》, 50(1):57–81, 1986年1月。[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch：一种命令式风格、高性能的深度学习库。载于神经信息处理系统进展 *32*，第8024–8035页。Curran Associates, Inc., 2019年。[33] A. A. Rodriguez, E. Bertolazzi, and A. Valli. *1.* 旋度–散度系统：理论与有限元近似，第1–44页。De Gruyter, 柏林、波士顿, 2019年。[34] J. Schöberl. 麦克斯韦方程的后验误差估计。77(262):633–649。出版商：美国数学会。[35] L. Shi, I. Babushkin, A. Husakou, O. Melchert, B. Frank, J. Yi, G. Wetzel, A. Demircan, C. Lienau, H. Giessen, M. Ivanov, U. Morgner, and M. Kovacev. 非绝热隧穿区间中由飞秒场驱动的片上单向电子电流。《激光与光子学评论》, (15), 2021。[36] A. Toselli and O. Widlund. 区域分解方法——算法与理论。施普林格计算数学丛书第34卷。施普林格, 柏林、海德堡，2005年。[37] S. Zaglmayr. 用于电磁场计算的高阶有限元方法。林茨约翰·开普勒大学，2006年。