Advances in Applied Mathematics 应用数学进展, 2025, 14(10), 189-198
Published Online October 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.1410431

Hans汉斯

基于LM算法的多尺度神经网络求解椭圆界面问题

粟钊阳[1,2]

[1]长沙理工大学数学与统计学院，湖南 长沙
[2]长沙理工大学工程数学建模与分析湖南省重点实验室，湖南 长沙

收稿日期：2025年9月11日；录用日期：2025年10月4日；发布日期：2025年10月13日

摘　要

近几年以来，利用深度学习方法解决偏微分方程问题引起了广泛关注和研究。本文介绍了一种基于改进的Levenberg-Marquardt (LM)优化方法的多尺度神经网络，该方法在求解椭圆界面问题的精度方面显示出不俗的潜力。文章通过单个神经网络框架解决界面问题，并选用不同的激活函数进行对比，通过几个具有规则和不规则界面的数值实验，以验证神经网络效果及相关理论。

关键词

多尺度神经网络，LM算法，椭圆界面问题

Solving Elliptic Interface Problems Using a Multi-Scale Neural Network Based on the LM Algorithm

Zhaoyang Su[1,2]

[1]School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha Hunan
[2]Hunan Provincial Key Laboratory of Mathematical Modeling and Analysis in Engineering, Changsha University of Science and Technology, Changsha Hunan

Received: September 11, 2025; accepted: October 4, 2025; published: October 13, 2025

Abstract

In recent years, the application of deep learning methods to solve partial differential equations has garnered significant attention and research. This paper introduces a multi-scale neural network

文章引用: 粟钊阳. 基于 LM 算法的多尺度神经网络求解椭圆界面问题[J]. 应用数学进展, 2025, 14(10): 189-198.
DOI: 10.12677/aam.2025.1410431

based on an improved Levenberg-Marquardt (LM) optimization method, which demonstrates remarkable potential in achieving high accuracy for solving elliptic interface problems. The paper addresses boundary problems within a single neural network framework, comparing different activation functions. Numerical experiments involving both regular and irregular boundaries validate the neural network's performance and related theoretical foundations.

Keywords

Multi-Scale Neural Network, LM Algorithm, Elliptic Interface Problem

1. 引言

近年来，随着计算机技术的飞速进步和科学计算能力的显著提升，深度学习技术得以不断发展。基于神经网络(Neural Networks)的深度学习(Deep Learning)方法在多学科领域特别是在计算机视觉[1] [2] (Computer Vision)和自然语言处理[3] [4] (Natural Language Processing)等领域取得了令人瞩目的成就。这些成就如雨后春笋般层出不穷，吸引了众多学者深入探索。神经网络因其卓越的拟合能力而广受认可，被视为一种功能强大的通用函数逼近工具。最近，这一强大的工具也被引入到偏微分方程的求解中，成为了一个新兴的研究方向。应用深度学习求解偏微分方程的基本思想是通过训练神经网络而非求解代数方程，近似求解偏微分方程的解函数或解算子。目前，该领域已涌现出众多研究进展，并取得了令人瞩目的成果。然而，在利用神经网络求解带有界面条件的偏微分方程时，仍存在精度不足的问题。这类方程因界面上的不连续性，使得普通的神经网络方法难以准确捕捉到界面上的跳跃性特征，这成为求解过程中的一大难点。因此，探索使用更高精度的神经网络方法来求解带有界面条件的偏微分方程，成为了一个值得深入研究的方向。

2018 年，Jiequn H 等[5]将深度学习方法用于求解高维偏微分方程，19 年 Cai W 等人[6]运用多尺度深度神经网络求解高维偏微分方程，文[7]说明深度时间神经网络(DTNN)是求解高维偏微分方程的有效深度学习方法。而[8]-[11]分别介绍了求解低维非线性偏微分方程的深度学习初始化迭代方法(Int-Deep)、深度最小二乘法这一基于无监督学习的求解椭圆偏微分方程的数值方法、用于完全非线性偏微分方程的深分支求解器及使用深度神经网络求解椭圆偏微分方程的非梯度方法。2024 年，Paola F 等人[12]介绍用于加速 3D 椭圆偏微分方程的有限元求解器中的代数多重网格方法，[13] [14]中提出了单位网络自适应分区(APUNet)这一用于解决偏微分方程的本地化深度学习方法及基于改进欧拉法的非线性偏微分方程神经网络求解器。彭杰，张玉武[15]对基于自适应神经网络的偏微分方程进行研究。

与全连接神经网络和 ResNet 的结果相比，多尺度融合网络[16] [17]作为一种新的深度学习结构来解决椭圆界面问题能够更好地捕捉到界面上的跳跃性特征，从而提高了准确性。此外，它的数值解可以保持解的连续性，同时保持磁通量跳跃通过不同的界面，从而保持微分方程的物理特性。基于此背景，我们利用多尺度神经网络来求解椭圆界面方程。在本文中，我们使用单个网络来解决椭圆界面问题，从而规避了使用两个或多个网络的复杂性。相较于[16] [17]中所使用的 Adam 优化算法而言，我们借鉴[18]中的 LM 优化算法，将多尺度融合网络与 LM 算法结合，并考虑该情况下不同激活函数对求解椭圆界面问题精度的影响效果，并通过数值实验验证我们所采用方法的有效性与价值性。

本文的内容安排如下。在第二节部分介绍我们的多尺度神经网络及 LM 优化算法。在第三节基于我们所使用的神经网络对规则和不规则界面的二维问题进行了数值实验，以验证我们的理论结果，随后进一步将其用于一个三维问题，展示了该方法的有效性。最后，第四节给出我们的相关结论及展望。

2. 深度学习方法

在这一节，我们以椭圆偏微分方程为例，介绍使用多尺度神经网络求解界面问题的具体流程。考虑以下具有不连续系数的椭圆偏微分方程：

$$\begin{cases} -\nabla\cdot\left(a(x)\nabla u(x)\right)=f(x), & x\in\Omega \\ u\left(s^{+}\right)=u\left(s^{-}\right)+p(s), & s\in\Gamma \\ \alpha\left(s^{+}\right)\dfrac{\partial u\left(s^{+}\right)}{\partial \mathbf{n}(s)}=\alpha\left(s^{-}\right)\dfrac{\partial u\left(s^{-}\right)}{\partial \mathbf{n}(s)}+q(s), & s\in\Gamma \\ u(s)=g(s), & s\in\partial\Omega \end{cases} \tag{1}$$

其中 $\Omega=\Omega_p\cup\Omega_s\cup\Gamma$，界面 Γ 为 Ω_p 和 Ω_s 的交界处，是 $\mathbb{R}^d$ ($d\geq 2$ 为给定的正整数)上的有界凸域，$g(s)$ 和 $p(s)$ 及 $q(s)$ 分别是边界 $\partial\Omega$ 和界面 Γ 上的给定函数，设系数 $\alpha(x)$ 为跨越界面 Γ 的分段常数函数，

$$\alpha(x)=\begin{cases}\alpha_1, & x\in\Omega_p \\ \alpha_2, & x\in\Omega_s\end{cases} \tag{2}$$

在给定的边界条件和界面条件下近似偏微分方程(1)的解。对于具有界面条件的椭圆问题，直接应用前面描述的深度学习方法无法得到准确的数值解。为了克服这个精度问题，学者们提出了几种技术，例如域分解法、扩展和投影法。在这里，我们没有使用域分解方法，而是采用了扩展和投影方法，使用基于这种扩展和投影方法的一个网络可以比使用多个网络获得更好的结果，同时在必要时它可以保持底层量的连续性。给定一个神经网络，为了保证准确性，根据扩展投影法，我们首先用指示函数 $z(x)$ 将之扩展到更高的空间维度。也就是说，我们尝试在 $\tilde{\Omega}\equiv\Omega\times\mathbb{R}\in\mathbb{R}^{d+1}$ 上定义 $\tilde{u}(x,z)$ 满足 $u(x)=\tilde{u}(x,z(x))$。其中，指示函数 $z(x)$ 用于区分不同的子域 Ω_p 和 Ω_s。对于界面上的情况，我们可以将指标函数 $z(x)$ 视为以下分段常数

$$z(x)=\begin{cases}-1, & x\in\Omega_p \\ 1, & x\in\Omega_s\end{cases} \tag{3}$$

通过使用以下最小二乘损失函数来描述椭圆界面问题的解 $u(x)$：

$$\min_{v\in H_g^2(\Omega_p\cup\Omega_s)} L(v)=\min_{v\in H_g^2(\Omega_p\cup\Omega_s)}\sum_{i=1}^{5}\beta_i L_i(v) \tag{4}$$

其中 $\beta_i, i=1,2,3,4,5$ 是给定的正权重常数，$L_i(v), i=1,2,3,4,5$ 是对应的泛函，分别定义如下：

$$\begin{aligned} L_1(v)&=\int_{\Omega_p}\left|-\nabla\cdot\left(\alpha_1\nabla v_1(x)\right)-f(x)\right|^2 \\ L_2(v)&=\int_{\Omega_s}\left|-\nabla\cdot\left(\alpha_2\nabla v_2(x)\right)-f(x)\right|^2 \\ L_3(v)&=\int_{\Gamma}\left|v_1(s)+p(s)-v_2(s)\right|^2 \\ L_4(v)&=\int_{\Gamma}\left|\alpha_2\frac{\partial v_2(s)}{\partial \mathbf{n}(s)}-\alpha_1\frac{\partial v_1(s)}{\partial \mathbf{n}(s)}-q(s)\right|^2 \end{aligned} \tag{5}$$

$$L_5(v)=\int_{\partial\Omega}\left|v_2(s)-g(s)\right|^2$$

2.1. 多尺度神经网络

深度学习方法训练过程的本质是寻找最优参数使损失函数最小。在深度学习的过程中，神经网络需要对损失函数中的 5 个积分进行求值，通过蒙特卡洛算法对相应域中的随机点求和，可以有效且高效地逼近损失函数。在每个区域中引入均匀分布的随机点 $\{x_i\}_{i=1}^{M_p}\in\Omega_p,\{x_i\}_{i=1}^{M_s}\in\Omega_s,\{x_i\}_{i=1}^{M_\Gamma}\in\Gamma,\{x_i\}_{i=1}^{M_b}\in\partial\Omega$，相应的优化任务变为：

$$\min_{\Theta=(\Theta_p,\Theta_S)}\sum_{i=1}^{5}\beta_i l_i \tag{6}$$

函数 $l_i,(i=1,2,3,4,5)$ 是 L_i 的离散形式，形式如下：

$$\begin{aligned}
l_1(\tilde{v})&=\frac{|\Omega_p|}{M_p}\sum_{i=1}^{M_p}\left|-\nabla\cdot\left(\alpha_1\nabla\tilde{v}_1(x_i)\right)-f(x_i)\right|^2\\
l_2(\tilde{v})&=\frac{|\Omega_s|}{M_s}\sum_{i=1}^{M_s}\left|-\nabla\cdot\left(\alpha_2\nabla\tilde{v}_2(x_i)\right)-f(x_i)\right|^2\\
l_3(\tilde{v})&=\frac{|\Gamma|}{M_\Gamma}\sum_{i=1}^{M_\Gamma}\left|\tilde{v}_1(s_i)+p(s_i)-\tilde{v}_2(s_i)\right|^2\\
l_4(\tilde{v})&=\frac{|\Gamma|}{M_\Gamma}\sum_{i=1}^{M_\Gamma}\left|\alpha_2\frac{\partial\tilde{v}_2(s_i)}{\partial\mathbf{n}(s)}-\alpha_1\frac{\partial\tilde{v}_1(s_i)}{\partial\mathbf{n}(s)}-q(s_i)\right|^2\\
l_5(\tilde{v})&=\frac{|\partial\Omega|}{M_b}\sum_{i=1}^{M_b}\left|\tilde{v}_2(s_i)-g(s_i)\right|^2
\end{aligned} \tag{7}$$

其中 $\tilde{v}_i(x),(i=1,2)$ 是函数 $v_i(x)$ 的近似值，$\tilde{v}(x)$ 是函数 $v(x)$ 的近似值，$|\cdot|$ 代表的是测度。

由于逼近能力不同，神经网络中不同的拓扑或架构会影响所提出的深度学习方法的性能。在这里，我们没有使用常用的 ResNet 结构或前馈神经网络，而是使用了[16] [17]中的多尺度融合神经网络(MSFN)，它已被数值证明能够更好地捕捉“急转弯”，从而提高准确性。作为前馈神经网络的推广，MSFN 结构基于小波理论中的多尺度基函数思想提出，并结合改进的信息融合模块，大大增强了其逼近能力。神经网络结构见图 1：

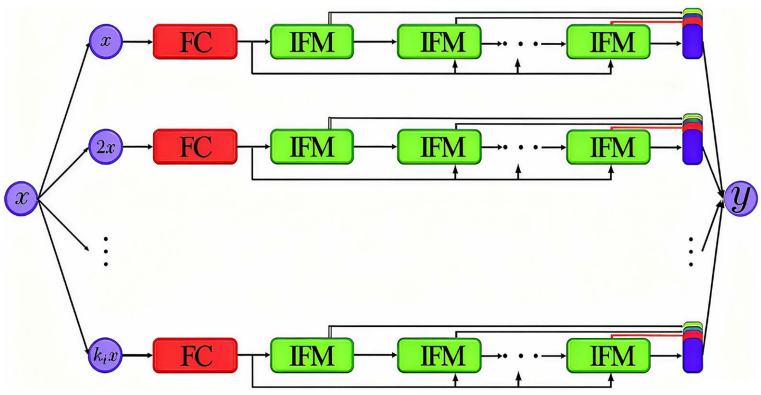

Figure 1. Multi-scale neural network architecture diagram
图 1. 多尺度神经网络结构图

MSFN 在输入层接收到信息后，将其划分成多个并行子网络，并在每个子网络中还进一步加入了改进后的信息融合模块，以增强其逼近能力。最后，MSFN 结构的输出是子网络的输出向量之和，加权平均作为可训练参数。基于多尺度神经网络求解椭圆界面问题，选择模型对参数进行更新时，使用 LM 优化算法。选择不同的激活函数对多尺度神经网络进行对比实验，并进行相关应用。

2.2. LM 优化算法

在神经网络的参数训练过程中，必须借助优化方法解决目标函数的最小化问题。当前，Adam 优化器因良好的稳定性与收敛效率，成为深度学习领域的主流选择。然而，这一选择在精度提升方面存在明显局限，若将 LM [18]方法作为优化器引入深度学习框架，模型精度可实现显著突破，其相对误差能够达到 10^{-6} 量级甚至更高，且在求解精度层面，表现可能优于传统的有限元方法。LM 这类近似二阶方法在求解 PDE 问题时精度优势的根本在于：它通过利用损失函数的曲率信息，能够更准确地模拟损失曲面的局部几何形状，从而计算出比一阶梯度方向更优的更新方向和步长。这对于由平方和构成、且常呈病态条件的 PDE 损失函数尤为有效。

为进一步提升模型性能，本文选择 LM 方法作为核心优化器。考虑到内容完整性，下文将在具体实现部分简要阐述 LM 方法的基本原理与关键技术细节。针对深度学习场景中常见的函数最小化问题，我们考虑以下通用损失函数：

$$\mathrm{Loss}(P)=\frac{1}{M}\sum_{i=1}^{M}\left(\phi(x_i)-\phi_{\mathrm{aug}}(x_i;P)\right)^2 \tag{8}$$

其中 $P\in\mathbb{R}^{N_p}$ 表示指定神经网络中的超参数向量，$x_i\,(i=1,\cdots,M)$ 是采样点。我们利用 LM 算法得到其最小值，则有如下迭代方案：

$$P^{(k+1)}=P^{(k)}+\delta \tag{9}$$

其中 $P^{(k)}$ 表示 P 的当前近似值，δ 满足以下修正的正态方程问题

$$\left(J^TJ+\mu I\right)\delta=J^T\left(\Phi-\Phi_{\mathrm{aug}}\left(P^{(k)}\right)\right) \tag{10}$$

通过归一化处理，我们将损失函数重写如下：

$$\mathrm{Loss}(P)=\sum_{i=1}^{M}\left(\frac{1}{\sqrt{M}}\phi(x_i)-\frac{1}{\sqrt{M}}\phi_{\mathrm{aug}}(x_i;P)\right)^2 \tag{11}$$

接下来我们介绍 LM 算法的步骤，从而找到上述损失函数的最小点，具体如下：

LM 算法：

步骤一：初始化容差 ϵ 和一个较大的 μ （$\mu=\mu_{\max}=10^8$），以便快速下降；设定 $\mu_{div}=3$ 和 $\mu_{\mathrm{mul}}=2$；初始化超参数 P 、loss 等；

步骤二：使用 LM 方法更新超参数 P 的值，计算新的损失，雅可比矩阵 J ，并设 $\mu=\max\left\{\mu/\mu_{div},10^{-15}\right\}$；

步骤三：对于当前近似值 P^k，通过 LM 算法更新超参数 $P^{(k+1)}$，计算增量 δ 和新的损失；

1. 如果新的损失小于之前的值，则设置 $\mu=\max\left\{\mu/\mu_{div},10^{-15}\right\}$，然后返回到步骤三，得到新的近似值 $P^{(k+1)}$；

2. 否则计算 $\cos\theta=\dfrac{\langle\delta,\delta_{\mathrm{old}}\rangle}{\|\delta_{\mathrm{old}}\|\cdot\|\delta\|}$，即前一个和当前搜索方向之间夹角的余弦值：

(a). 如果 $(1-\cos\theta)$ 乘以当前损失大于先前的损失，则放弃当前搜索方向并设置

$\mu = \max\{\mu * \mu_{\text{mul}}, \mu_{\max}\}$。使用新的 μ 求解修正的方程获得新的搜索方向 δ 并返回到步骤三；

(b). 否则，接受当前搜索方向，然后更新相关信息，设置 $\mu = \max\{\mu / \mu_{\text{div}}, 10^{-15}\}$ 并返回步骤三；

(c). 当损失小于给定的容差 ϵ 时停止迭代。

通过上述的算法步骤从而找到损失函数最小值。

3. 数值实验

在这部分中，我们进行数值测试来验证理论结果。为此，考虑了三种数值实验，一种是规则界面的情况，一种是不规则界面的情况，以及一种三维椭球界面的问题。这里我们使用 Python 来实现我们的数值实验，并对每种界面使用了三种不同的激活函数进行对比实验。我们采用相对 L^2 误差来测量神经网络的精确解 u^* 和近似解 $\hat{u}$ 之间的差异，其定义如下：

$$\varepsilon_{RL^2} = \frac{\|u-\hat{u}\|_{L^2}}{\|u\|_{L^2}} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|u^*(x_i)-\hat{u}(x_i)\right|^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|u^*(x_i)\right|^2}} \tag{12}$$

3.1. 规则界面

首先我们考虑一个圆心为(1, 1)，半径为 2 的圆域，界面 Γ 为相同圆心且半径为 1 的圆，其坐标为 $(x_1, x_2) = (1+\cos\theta, 1+\sin\theta), \theta \in [0, 2\pi)$。其中系数 α 为

$$\alpha = \begin{cases} 1, & x \in \Omega_p \\ 2, & x \in \Omega_s \end{cases} \tag{13}$$

其精确解为

$$u(x_1, x_2) = \begin{cases} 2\tanh(x_1 + x_2), (x_1, x_2) \in \Omega_p \\ \tanh(x_1 + x_2), (x_1, x_2) \in \Omega_s \end{cases} \tag{14}$$

在 Ω 内选取 400 个采样点，界面 Γ 和边界 $\partial\Omega$ 选取 200 个采样点进行训练，测试数据由 Ω 内均匀分布的 10,000 个采样点生成。考虑 Adam 和 LM 优化算法下的多尺度神经网络及 Tanh、Sigmoid、SiLU 三种激活函数的误差见表 1：

Table 1. Circular interface error
表 1. 圆形界面误差

优化算法	激活函数	相对 L^2 误差
Adam	Tanh	2.1810e-04
LM	Tanh	1.3241e-11
LM	Sigmoid	4.3823e-12
LM	SiLU	3.9786e-12

可以看出来，相对于 Adam 算法而言，我们所采用的 LM 算法下的误差量级显著下降，不同的激活函数对误差也有着一定程度的影响。其中 LM 算法下，激活函数为 SiLU 时我们得到的精确解、近似解和误差分布见图 2：

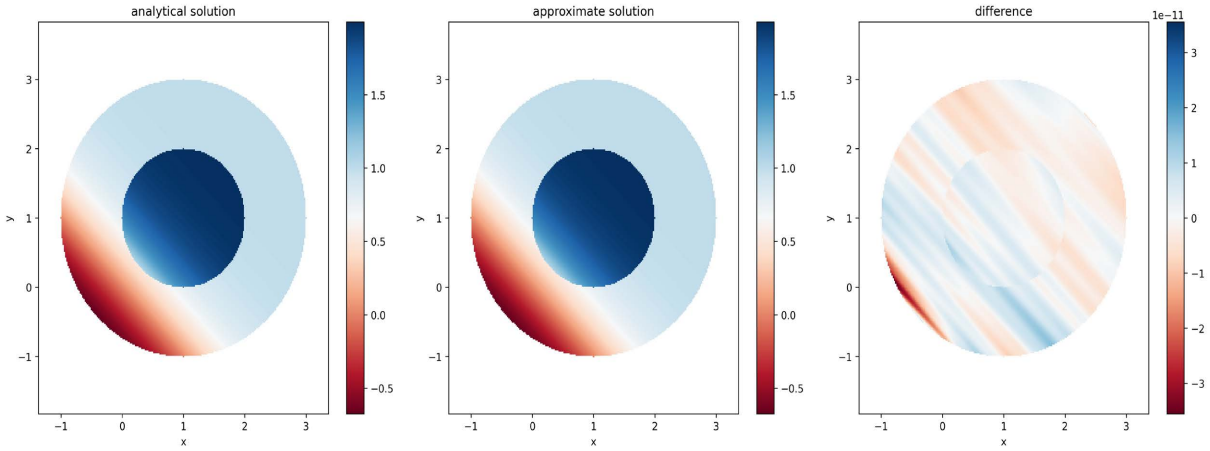

Figure 2. Circular interface diagram
图 2. 圆形界面图

3.2. 不规则界面

我们进一步考虑不规则花型界面问题，其精确解如下：

$$u(x)=\begin{cases} e^{r^2}, & x\in\Omega_p \\ 0.1r^4-0.01\log(2r), & x\in\Omega_s \end{cases} \tag{15}$$

r 表示点与原点之间的距离，计算域 Ω 为圆心为原点半径为 1 的圆，花形界面定义为：$r=0.5+\dfrac{\sin(5\theta)}{7}$，其中系数 α 为

$$\alpha=\begin{cases} 1, & x\in\Omega_p \\ 10, & x\in\Omega_s \end{cases} \tag{16}$$

在 Ω_p 选取 160 个采样点，Ω_s 选取 320 个采样点，界面 Γ 和边界 $\partial\Omega$ 选取 128 个采样点进行训练，在 $[-1,1]^2$ 的平方域上生成 201×201 的均匀网格用作测试。同样考虑两种优化算法及三种激活函数下的多尺度神经网络误差见表 2：

Table 2. Floral interface error
表 2. 花形界面误差

优化算法	激活函数	相对 L^2 误差
Adam	Tanh	1.4700e-03
LM	Tanh	4.1374e-05
LM	Sigmoid	7.4543e-06
LM	SiLU	9.9306e-08

可以发现，当界面不规则时，同为 Tanh 激活函数时我们所采用的 LM 优化算法依旧比 Adam 算法有不小的精度提升；不同激活函数对误差的影响较大，当激活函数为 SiLU 时的近似效果最好，此时我们得到的花形界面图形见图 3：

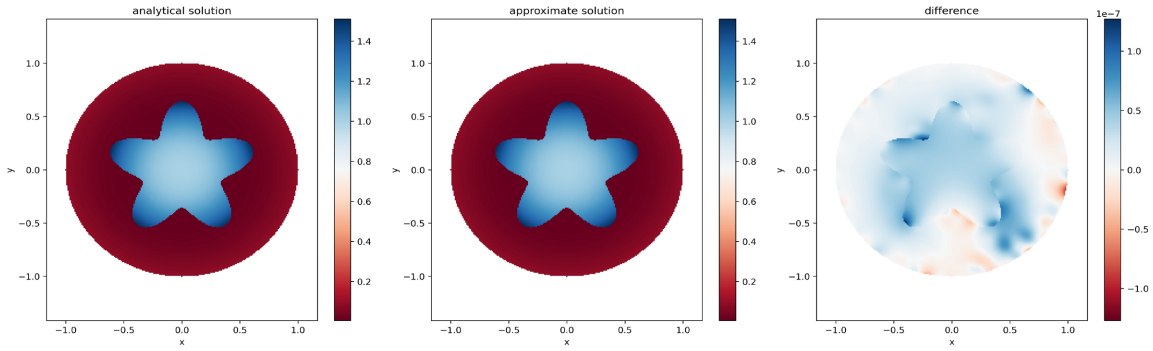

Figure 3. Flower pattern interface diagram
图 3. 花形界面图

3.3. 三维案例

最后，我们考虑一个三维的例子。在这个例子中，考虑一个系数为常数的 3D 椭球界面问题。计算域定义为$[-1,1]^3$的立方体。界面采用以原点为中心的椭球体表面的形式，其特征是长度为 0.7、0.5 和 0.3 的半主轴。这个椭球体由水平集函数$\varphi=(x/0.7)^2+(y/0.5)^2+(z/0.3)^2-1$表示。对于该问题，子域系数$\alpha$都取 1，精确解为

$$u(x,y,z)=\begin{cases}\exp\left(x^2+y^2+z^2\right) & (x,y,z)\in\Omega_p\\ 0.1\left(x^2+y^2+z^2\right)^2-0.01\log\left(2\sqrt{x^2+y^2+z^2}\right), & (x,y,z)\in\Omega_s\end{cases} \tag{17}$$

在Ω选取 1500 个采样点，界面Γ和边界$\partial\Omega$选取 240 个采样点进行训练，在$[-1,1]^3$的平方域上生成 128 × 128 × 128 的均匀网格用作测试。和前面的数值实验采用一样的对比实验，得出三维案例下的相关误差见表 3：

Table 3. Three-dimensional ellipsoid interface error
表 3. 三维椭球界面误差

优化算法	激活函数	相对 L^2 误差
Adam	Tanh	3.1283e-03
LM	Tanh	3.1265e-05
LM	Sigmoid	4.1338e-07
LM	SiLU	2.3768e-07

其中，该三维案例在$z=0$切平面处的精确解、近似解和误差分布见图 4：

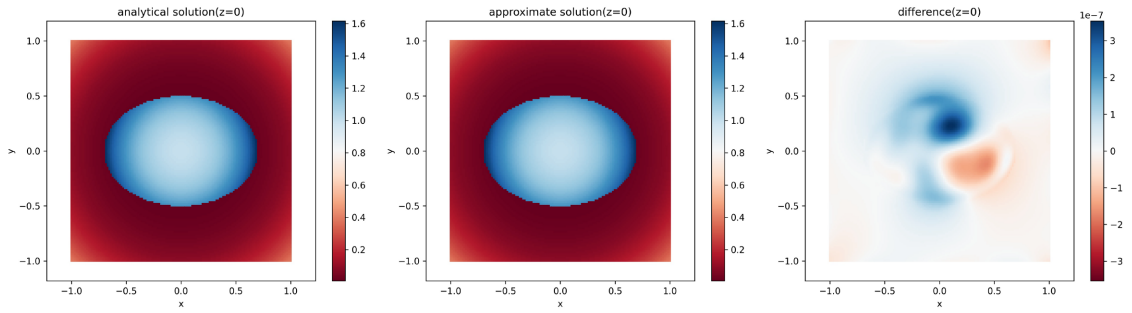

Figure 4. Cross-section of a three-dimensional ellipsoid interface
图 4. 三维椭球界面切面图

4. 结论与展望

本文基于 LM 算法的多尺度神经网络求解椭圆界面问题，并将其与常规的 Adam 优化算法展开对比分析。研究结果表明，在三个典型数值案例的求解过程中，所采用的 LM 算法展现出更优异的近似效果，相较于 Adam 算法，其求解精度的数量级实现了显著提升，这一结果充分验证了本研究的可行性与有效性。

然而，需要客观指出的是，LM 算法在处理高精度优化问题时，也存在明显的性能瓶颈。当将该算法拓展至大规模椭圆界面问题求解场景时，其计算效率不足的问题逐渐凸显，对硬件计算设备的算力、存储容量等性能指标提出了更高要求——这不仅增加了实际工程应用中的部署成本，也在一定程度上限制了该方法的适用范围。因此，如何优化 LM 算法的计算流程以提升效率、降低设备依赖，成为当前方案亟待突破的核心问题，有待进一步深入探索与完善。

参考文献

[1] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) Imagenet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, **60**, 84-90. https://doi.org/10.1145/3065386

[2] He, K.M., Zhang, X.Y., Ren, S.Q., *et al*. (2016) Deep Residual Learning for Image Recognition. 2016 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), Las Vegas, 27-30 June 2016, 770-778. https://doi.org/10.1109/cvpr.2016.90

[3] Vaswani, A., Shazeer, N., Parmar, N., *et al*. (2017) Attention Is All You Need. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6000-6010.

[4] Devlin, J., Chang, M.W., Lee, K., *et al*. (2018) BERT: Pre Training of Deep Bidirectional Transformers for Language Understanding.

[5] Han, J.Q., Jentzen, A. and Weinan, E. (2018) Solving High-Dimensional Partial Differential Equations Using Deep Learning. *Proceedings of the National Academy of Sciences*, **115**, 8505-8510. https://doi.org/10.1073/pnas.1718942115

[6] Cai, W. and Xu, Z.Q.J. (2019) Multi-Scale Deep Neural Networks for Solving High Dimensional PDEs.

[7] Aghapour, A., Arian, H. and Seco, L. (2024) Deep-Time Neural Networks: An Efficient Approach for Solving High-Dimensional PDEs. *Applied Mathematics and Computation*, **488**, Article 129117. https://doi.org/10.1016/j.amc.2024.129117

[8] Huang, J., Wang, H. and Yang, H. (2020) Int-Deep: A Deep Learning Initialized Iterative Method for Nonlinear Problems. *Journal of Computational Physics*, **419**, Article 109675. https://doi.org/10.1016/j.jcp.2020.109675

[9] Cai, Z., Chen, J., Liu, M. and Liu, X. (2020) Deep Least-Squares Methods: An Unsupervised Learning-Based Numerical Method for Solving Elliptic PDEs. *Journal of Computational Physics*, **420**, Article 109707. https://doi.org/10.1016/j.jcp.2020.109707

[10] Nguwi, J.Y., Penent, G. and Privault, N. (2022) A Deep Branching Solver for Fully Nonlinear Partial Differential Equations.

[11] Peng, Y., Hu, D. and Xu, Z.J. (2023) A Non-Gradient Method for Solving Elliptic Partial Differential Equations with Deep Neural Networks. *Journal of Computational Physics*, **472**, Article 111690. https://doi.org/10.1016/j.jcp.2022.111690

[12] Caldana, M., Antonietti, P.F. and Dede', L. (2024) A Deep Learning Algorithm to Accelerate Algebraic Multigrid Methods in Finite Element Solvers of 3D Elliptic PDEs. *Computers & Mathematics with Applications*, **167**, 217-231. https://doi.org/10.1016/j.camwa.2024.05.013

[13] Barbara, I., Masrour, T. and Hadda, M. (2024) Adaptive Partition of Unity Networks (APUNet): A Localized Deep Learning Method for Solving PDEs. *Evolving Systems*, **15**, 1137-1158. https://doi.org/10.1007/s12530-023-09544-7

[14] 黄冠男, 王靖岳, 王美清. 基于改进欧拉法的非线性偏微分方程神经网络求解器[J]. 福州大学学报(自然科学版), 2024(4).

[15] 彭杰, 张玉武. 基于自适应神经网络的 PDEs 求解研究[J]. 佳木斯大学学报(自然科学版), 2024, 42(3):174-177.

[16] Ying, J., Liu, J., Chen, J., Cao, S., Hou, M. and Chen, Y. (2023) Multi-Scale Fusion Network: A New Deep Learning Structure for Elliptic Interface Problems. *Applied Mathematical Modelling*, **114**, 252-269. https://doi.org/10.1016/j.apm.2022.10.006

[17] Ying, J., Li, J., Liu, Q. and Chen, Y. (2024) Improved Multi-Scale Fusion Network for Solving Non-Smooth Elliptic Interface Problems with Applications. *Applied Mathematical Modelling*, **132**, 274-297. https://doi.org/10.1016/j.apm.2024.04.039

[18] Ying, J., Xie, Y., Li, J., *et al*. (2024) Accurate Adaptive Deep Learning Method for Solving Elliptic Problems.