Advances in Applied Mathematics 应用数学进展, 2025, 14(5), 348-356
Published Online May 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.145264

Hans汉斯

求解二维不可压Navier-Stokes方程的PINN算法

沈旭彬*，何流利

杭州师范大学数学学院，浙江 杭州

收稿日期：2025年4月23日；录用日期：2025年5月16日；发布日期：2025年5月28日

摘　要

本文采用物理信息神经网络(PINN)来求解不可压缩湍流Navier-Stokes方程。本研究引入了动态权重调整策略，使得各项误差在训练过程中得到适当的平衡，从而避免了某些误差项主导整个训练过程的问题。此外，为了加速训练收敛并提高精度，本研究还对网络结构进行了优化，结合物理约束优化过程，改变了优化方法，提高了模型的训练效率。

关键词

物理信息神经网络(PINN)，Navier-Stokes方程，不可压缩流体，深度学习

PINN Algorithm for Solving Two-Dimensional Incompressible Navier-Stokes Equations

Xubin Shen*, Liuli He

School of Mathematics, Hangzhou Normal University, Hangzhou Zhejiang

Received: Apr. 23rd, 2025; accepted: May 16th, 2025; published: May 28th, 2025

Abstract

In this paper, physical information neural networks (PINN) are used to solve the Navier-Stokes equations of incompressible turbulence. In this study, the dynamic weighting adjustment strategy is presented to make the errors properly balanced in the training process, so as to avoid the problem that

*通讯作者。

文章引用: 沈旭彬, 何流利. 求解二维不可压 Navier-Stokes 方程的 PINN 算法[J]. 应用数学进展, 2025, 14(5): 348-356.
DOI: 10.12677/aam.2025.145264

some error terms dominate the whole training process. In addition, in order to accelerate the training convergence and improve the accuracy, this study also optimized the network structure, combining with the physical constraint optimization process and changing the optimization method to improve the training efficiency of the model.

Keywords

Physics-Informed Neural Networks (PINN), Navier-Stokes Equations, Incompressible Fluid, Deep Learning

1. 引言

物理信息神经网络(PINN)是近年来由 Raissi 等人[1]提出的一种结合物理约束的深度学习方法。该方法的核心思想是将物理方程作为损失函数的一项加入到神经网络中，通过神经网络的训练过程求解偏微分方程(PDE)。与传统的基于数据的深度学习方法不同，PINN 不依赖大量的实验数据，而是直接利用物理定律(如 Navier-Stokes 方程)作为先验知识，在训练过程中结合物理信息。PINN 方法不仅可以有效地求解流体力学中的基础方程，还能够处理复杂的边界条件、非线性问题和多尺度现象，因此在流体力学中有广阔的应用前景。

PINNs 已经历了诸多扩展和改进，以增强其适用性并克服原始框架的局限性。包括融合其他网络架构如卷积神经网络(CNNs) [2]和循环神经网络(RNNs) [3]，进一步拓宽了 PINNs 的应用范围。神经网络具有强大的非线性映射能力和自适应特性，能够有效捕捉复杂系统中的潜在规律。在求解复杂流体问题，处理复杂的边界条件、噪声和不适定问题时，表现出较好的鲁棒性[4]。

因此，结合物理信息神经网络(PINN)的方法，为不可压缩流体的 Navier-Stokes 方程提供了一种新的解法。通过结合深度学习的强大推理能力和物理方程的约束，PINN 有望克服现有数值方法的瓶颈，提供更为高效、精确的解决方案。

当前，PINNs 中标准的优化算法是 Adam 和拟牛顿方法(如 BFGS)。BFGS 通过引入可训练变量空间的二阶信息进行预处理，显著提升了 PINNs 的性能。尽管这两种优化器因其理论和数值特性不仅在 PINNs 中，而且在其他优化问题中也广为应用，但近期在 PINNs 中的实践表明，对于许多问题，其他算法可能更具优势。

2. 物理信息神经网络模型

2.1. 模型框架

简单的 PINN 网络主要有三个部分，输入层为时空坐标 $(\boldsymbol{x},t)$，中间层为全连接层，输出层为速度和压力 $(\boldsymbol{u},p)$。模型的损失函数的一般形式为：

$$L_{PINN} = \lambda_1 L_{IC} + \lambda_2 L_{BC} + \lambda_3 L_f + \lambda_4 L_{data} \tag{1}$$

其中，L_{data} 是数据误差项，L_{IC} 和 L_{BC} 分别是初始条件和边界条件的误差项，L_f 是物理方程残差项，其中包含问题所满足的物理条件。每个残差项的系数(如 $\lambda_1,\lambda_2,\lambda_3,\lambda_4$)用于平衡损失函数中各项的影响，确保训练过程中的各项条件得到适当的满足。图 1 展示了一个二维物理信息神经网络的结构示意图。

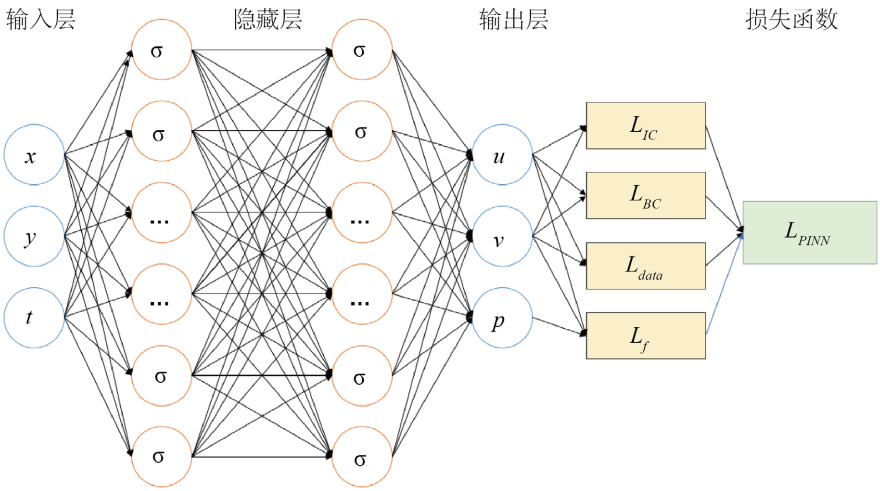

Figure 1. Structure of physics-information neural network
图 1. 物理信息神经网络结构

2.2. 初始条件与边界条件残差的处理

在 PINN 中，初始条件的残差处理体现在损失函数的设计中。设定初始条件时，网络需要确保在时刻 t_0 时，速度场和压力场满足初始条件。为了实现这一目标，在损失函数中加入初始条件的残差项，具体形式为：

$$L_{IC}=\frac{1}{N_i}\sum_{n=1}^{N_i}\left\|\boldsymbol{u}^n-\boldsymbol{u}_i^n\right\|^2,$$

其中，N_i 表示初始条件的训练点个数，$\boldsymbol{u}_i^n$ 是初始时刻 t_0 第 n 个数据点的给定速度，$\boldsymbol{u}^n$ 是 PINN 模型第 n 个数据点在初始时刻预测的速度，这里 $\|\bullet\|$ 表示 L_2 模。

边界条件的残差处理方式与初始条件类似，同样通过损失函数中的残差项来确保网络输出符合给定的边界条件。损失函数中与边界条件相关的残差项通常表示为：

$$L_{BC}=\frac{1}{N_b}\sum_{n=1}^{N_b}\left\|\boldsymbol{u}^n-\boldsymbol{u}_b^n\right\|^2,$$

其中，N_b 表示边界条件的训练点个数，$\boldsymbol{u}_b^n$ 是边界 $\partial\Omega$ 上第 n 个数据点的给定速度，$\boldsymbol{u}^n$ 是 PINN 模型第 n 个数据点在该边界点预测的速度。

2.2. 方程残差与物理约束结合

首先计算区域内部点的数据误差 L_{data}，计算过程如下：

$$L_{data}=\frac{1}{N_d}\sum_{n=1}^{N_i}\left\|\boldsymbol{u}^n-\boldsymbol{u}_d^n\right\|^2,$$

其中，N_d 表示数据点训练点个数，$\boldsymbol{u}_d^n$ 是第 n 个数据点的给定速度，$\boldsymbol{u}^n$ 是 PINN 模型在第 n 个数据点预测的速度。

在 PINN 中，为了确保网络的输出能够满足这些物理方程，将方程的残差引入到损失函数中。具体来说，方程残差的计算过程如下：

若问题满足 N 个物理方程 $f_n(\boldsymbol{x},t)(n=1,\cdots,N),x\in\Omega,t\in[0,T]$。可以根据神经网络的输出计算每个方程项的残差，计算过程如下：

$$L_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\sum_{n=1}^{N}\left|f_n\left(\boldsymbol{x}^i,t^i\right)\right|^2,$$

其中，$\left(\boldsymbol{x}^i,t^i\right)$表示第 i 个内部点的时空坐标。

通过计算这些残差，能够确定网络的输出与物理方程之间的偏差。

3. 改进后的物理信息神经网络模型

3.1. 动态权重调整策略

在 PINN 的训练过程中，由于初始条件、边界条件和方程残差项的数值范围和变化速率差异较大，若固定权重，某一项可能会过度主导训练，导致网络忽视其他约束条件，最终影响求解精度。因此，采用动态权重调整策略，实时调整损失函数中各项的权重，从而使得每个约束条件在网络训练的各个阶段发挥合适的作用。

残差项的大小反映了网络在某个方面的偏差程度，较大的残差表明该项在当前训练阶段对解的影响较大，需要更多的关注，较小的残差则表明该项对解的贡献较小，可以适当减小权重。一种常见的动态权重调整策略[5]是基于残差归一化的方式来更新每个项的权重。公式如下：

$$\lambda_i = \frac{L_i}{\sum_{i=1}^{K} L_i + \varepsilon} \tag{2}$$

其中，λ_i 是第 i 个损失项(如初始条件、边界条件或方程残差)的动态权重，L_i 是对应项的残差，ε 是一个小常数，避免分母为零，在本文中 ε 取 10^{-8}，避免对计算结果产生显著影响。本文中动态权重的调整频率为每训练 1000 轮次调整一次权重，既能避免每一轮调整权重给网络带来计算量和时间的增加，又可以确保了在训练过程中，较大的残差对应较大的权重，较小的残差对应较小的权重，从而实现平衡各个约束条件的作用。动态权重调整策略的算法如算法 1 所示。

算法 1 动态权重调整策略

输入：

每个数据点对应的空间坐标 $\boldsymbol{x}$ 和时间 t 。

输出：

每个时空点对应的速度 $\boldsymbol{u}$ 和压力 p 。

初始化: $\lambda_1=\lambda_2=\lambda_3=\lambda_4=1$

for $epochs = 1,\cdots,10000$ **do**

$$(\boldsymbol{u},p) = PINN(\boldsymbol{x},t), L_{IC} = \frac{1}{N_i}\sum_{n=1}^{N_i}\left\|\boldsymbol{u}^n - \boldsymbol{u}_i^n\right\|^2, L_{BC} = \frac{1}{N_b}\sum_{n=1}^{N_b}\left\|\boldsymbol{u}^n - \boldsymbol{u}_b^n\right\|^2,$$

$$L_{data} = \frac{1}{N_d}\sum_{n=1}^{N_i}\left\|\boldsymbol{u}^n - \boldsymbol{u}_d^n\right\|^2, L_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\sum_{n=1}^{N}\left|f_n\left(\boldsymbol{x}^i,t^i\right)\right|^2,$$

$$L_{PINN} = \lambda_1 L_{IC} + \lambda_2 L_{BC} + \lambda_3 L_f + \lambda_4 L_{data}$$

if $L_{PINN} > 10^{-5}$ **or** $epochs \bmod 100 = 0$ **then**

$$\lambda_i = \frac{L_i}{\sum_{i=1}^{4} L_i + \varepsilon}$$

end if

end for

return $(\boldsymbol{u},p)$

3.2. Adam 优化方法

神经网络中的优化问题主要寻找一组参数(权重和偏置) θ^*，使得损失函数 $L(\theta)$ 最小化：

$$\theta^* = \arg\min_{\theta} L(\theta)$$

Adam [6] (Adaptive Moment Estimation)优化算法是一种自适应的梯度优化方法。Adam 通过计算每个参数的梯度的一阶矩(均值)和二阶矩(方差)来动态调整学习率，从而在训练过程中实现快速收敛和较小的波动。Adam 算法在处理具有稀疏梯度的问题时，具有非常高的效率和稳定性。

Adam 算法的更新规则如下：

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

其中，$\hat{m}_t$ 和 $\hat{v}_t$ 分别表示梯度的偏差校正一阶矩和二阶矩，β_1 和 β_2 是衰减率，η 是学习率，ε 是一个小常数，用来避免分母为零。在 PINN 的训练过程中，Adam 算法通过自适应调整每个参数的学习率，使得网络在训练过程中既能保持较快的收敛速度，又能避免梯度消失或爆炸问题。

3.3. SSBroyden 优化方法

在优化问题中，需要求解的基本问题是 $\min_{\boldsymbol{x}\in R^n} f(\boldsymbol{x})$。对于这一问题，一个基本的挑战是确定从当前点过渡到改进点的最佳方向和步长。牛顿法是机器学习中用得比较多的一种优化算法。牛顿法的基本思想是利用迭代点 $\boldsymbol{x}_k$ 处的一阶导数(梯度)和二阶导数(Hessian 矩阵)对目标函数进行二次函数近似，然后把二次模型的极小点作为新的迭代点，并不断重复这一过程，直至求得满足精度的近似极小值。

拟牛顿法也是一类用于优化问题的迭代算法，旨在解决牛顿法计算复杂度高的问题，同时保留其快速收敛的特性。拟牛顿法通过构造 $\boldsymbol{x}_k$ 处 f 的 Hessian 矩阵的近似矩阵 $\boldsymbol{B}_k$，满足以下条件：

$$\boldsymbol{B}_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k,$$

其中，$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$。通过迭代公式更新 $\boldsymbol{B}_k$，避免直接计算 Hessian 矩阵，特别适用于高维优化问题。

Wolkowicz 等人[7] [8]提出的 BFGS 和其他 Broyden 家族更新的缩放方法家族进一步支持了缩放技术在某些优化问题中的有效性。直接和逆 Hessian 估计的自动缩放更新可以轻松扩展到一般 Broyden 家族，以下称为 SSBroyden 更新。这些更新由以下公式给出：

$$\boldsymbol{B}_{k+1} = \tau_k \left[\boldsymbol{B}_k - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{B}_k}{\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{B}_k \boldsymbol{s}_k} + \theta_k \left(\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{B}_k \boldsymbol{s}_k \right) \boldsymbol{w}_k \boldsymbol{w}_k^{\mathrm{T}} \right] + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{y}_k \boldsymbol{s}_k^{\mathrm{T}}},$$

$$\boldsymbol{H}_{k+1} = \frac{1}{\tau_k} \left[\boldsymbol{H}_k - \frac{\boldsymbol{H}_k \boldsymbol{y}_k \boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{H}_k}{\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{H}_k \boldsymbol{y}_k} + \Phi_k \left(\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{H}_k \boldsymbol{y}_k \right) \boldsymbol{w}_k \boldsymbol{w}_k^{\mathrm{T}} \right] + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{s}_k},$$

其中，θ_k 是一个标量参数，并且

$$\tau_k = \min\left\{1, \frac{1}{b_k}\right\},$$

$$\boldsymbol{w}_k = \frac{\boldsymbol{y}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k} - \frac{\boldsymbol{B}_k\boldsymbol{s}_k}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k\boldsymbol{s}_k},$$

$$\boldsymbol{v}_k = \frac{\boldsymbol{s}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k} - \frac{\boldsymbol{H}_k\boldsymbol{y}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{H}_k\boldsymbol{y}_k},$$

$$\Phi_k = \frac{1-\theta_k}{1+\left(h_k b_k - 1\right)\theta_k},$$

$$b_k = \frac{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k\boldsymbol{s}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k},$$

$$h_k = \frac{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{H}_k\boldsymbol{y}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k}.$$

在本文的数值实验中，我们将 SSBroyden 优化与自适应动态权重调整策略进行结合，观察权重策略和优化器同时改变之后损失函数的变化情况。

4. 数值实验

4.1. 数据集

本文研究的问题是二维圆柱绕流问题：该问题采用二维不可压缩的 Navier-Stokes 方程进行求解，方程形式为：

$$\nabla \boldsymbol{u} = 0, \tag{3}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\nabla\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \mu\nabla^2\boldsymbol{u}, \tag{4}$$

$$u\left(x,y,0\right) = u_0, v\left(x,y,0\right) = 0, \left(x,y\right) \in \Omega,$$

$$\frac{\partial u}{\partial \boldsymbol{n}} = 0, \frac{\partial v}{\partial \boldsymbol{n}} = 0, \text{在 } \Gamma_1 \text{ 上},$$

$$u = 0, v = 0, \text{在 } \Gamma_0 \text{ 上},$$

其中，$\boldsymbol{u} = \left[u,v\right]$ 是流速场，p 是压力场，μ 是运动粘度，实验中取 0.01。设 Ω_1 为方形计算区域，Ω_2 为 Ω_1 中圆的内部，记 $\Omega = \Omega_1 \setminus \Omega_2$，圆的边界记为 Γ_0，如图 2 所示。

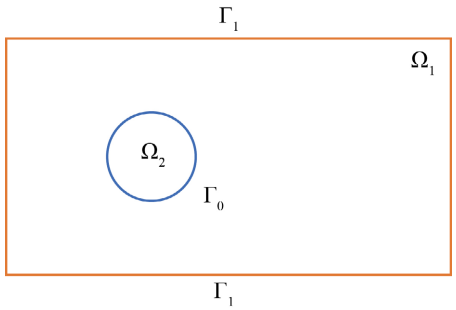

Figure 2. Computational area of two-dimensional cylinder wake
图 2. 二维圆柱绕流问题的计算区域

时空区域：计算域为二维矩形区域$\Omega_1=[-15,25]\times[-8,8]$，时间域为[0, 7]。边界条件为固定边界速度和无滑移条件。

由 Navier-Stokes 方程(3)、(4)可得，其质量守恒和动量守恒形式如下：

$$f_1(x,y,t):=\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0,(x,y)\in\Omega$$

$$f_2(x,y,t):=\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+\frac{\partial p}{\partial x}-0.01\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0,(x,y)\in\Omega$$

$$f_3(x,y,t):=\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+\frac{\partial p}{\partial x}-0.01\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}=0,(x,y)\in\Omega$$

则方程的残差L_f变为：

$$L_f=\frac{1}{N_f}\sum_{i=1}^{N_f}\left(\left|f_1\left(x^i,y^i,t^i\right)\right|^2+\left|f_2\left(x^i,y^i,t^i\right)\right|^2+\left|f_3\left(x^i,y^i,t^i\right)\right|^2\right),$$

其中，$\left(x^i,y^i,t^i\right)$是计算区域内部第 i 个内部点的时空坐标。

4.2. 实验环境与参数

本文实验环境为 Windows 10 家庭版(64 位)，处理器 11th Gen Intel(R) Core(TM) i5-1135G7 (2.42 GHz)，内存 16 GB，Python 版本为 3.10.8。数据集为 Raissi 等人[1]的二维层流圆柱绕流数值计算结果，来源于网站 https://github.com/maziarraissi/PINNs。实验研究选择了全连接神经网络(FCNN)作为 PINN 的基础架构。网络由输入层、8 个隐藏层(每层 16 个神经元)、输出层组成，激活函数使用 tanh。输出层包括两个速度分量和一个压力值。PINN 的损失函数使用公式(1)，其中包括数据误差、物理方程残差、初始条件和边界条件的误差。实验的数据点随机生成，其中内部点 6000 个，边界点 500 个，初始点 400 个，总采样点数量为 6900 个。为了确保每个部分对损失函数的贡献合理，本文使用公式(2)引入了动态权重调整策略，以平衡各部分误差在训练过程中的影响。

4.3. 实验评价指标

在进行 PINN 与传统数值方法(如有限元法和有限差分法)的比较时，本文选择了一个典型的二维流动问题作为实验对象，使用有限元法求解其数值解，并将结果与 PINN 求解结果进行比较。

4.4. 结果对比

本实验选择了一个二维圆柱绕流问题作为测试案例，计算域为一个矩形区域，流体的初始条件和边界条件已经定义。流动问题的求解依赖于 PINN 模型通过同时最小化数据误差、物理方程的残差和边界条件的残差来得到最优解。本文先根据 Raissi 等人[1]的文章对实验进行复现，原文采用固定权重 + Adam 优化的方法，本文采用动态权重 + Adam 优化的方法，对两者实验结果进行比较，观察动态权重调整策略对实验结果的影响，实验结果如表 1 所示。

从表 1 中可以看出，在训练 10,000 轮次之后，动态权重调整策略的总损失达到了 0.00083，比静态权重策略的总损失 0.0051 精度提高了一个数量级，并且训练时间只增加了 350 秒。表明在相同优化器的情况下，动态权重调整策略相比于固定权重策略，能够更有效地降低总损失，更快地提高精度。

表 1 可以证明动态权重调整策略相对于固定权重策略的优越性，本文接下来研究优化器的改变对于 PINN 实验结果的影响，本文在都采用动态权重调整策略的情况下，分别使用 Adam 优化方法和 SSBroyden 优化方法，观察不同优化方法对实验结果的影响，实验结果如表 2 和图 3 所示。

Table 1. Change of loss function and training time under different weighting strategies
表 1. 不同权重策略下的损失函数和训练时间的变化

训练轮次	固定权重策略 损失函数(L)	固定权重策略 训练时间(s)	动态权重策略 损失函数(L)	动态权重策略 训练时间(s)
1000	0.0753	153	0.0272	182
2000	0.0589	301	0.00835	375
3000	0.0431	448	0.00527	561
4000	0.0319	607	0.00352	750
5000	0.0223	755	0.00235	942
6000	0.0141	910	0.00153	1085
7000	0.0095	1061	0.00118	1276
8000	0.0072	1218	0.00101	1410
9000	0.0063	1364	0.00090	1633
10,000	0.0051	1522	0.00083	1872

Table 2. Comparison of losses of different optimizers under dynamic weighting strategy
表 2. 动态权重策略下不同优化器的损失比较

训练轮次	Adam 优化 损失函数(L)	Adam 优化 训练时间(s)	SSBroyden 优化 损失函数(L)	SSBroyden 优化 训练时间(s)
1000	0.0272	182	0.0115	215
2000	0.00835	375	0.00328	440
3000	0.00527	561	0.00102	659
4000	0.00352	750	0.00043	863
5000	0.00235	942	0.00031	1097
6000	0.00153	1085	0.00024	1336
7000	0.00118	1276	0.00018	1561
8000	0.00101	1410	0.00015	1772
9000	0.00090	1633	0.00012	1995
10,000	0.00083	1872	0.00009	2246

从表 2 和图 3 中可以看出，都采用动态权重调整策略的情况下，优化器从 Adam 改成 SSBroyden 之后，在训练 10,000 轮次之后的总损失从 0.00083 降低到了 0.00009，表明随着训练的进行，模型的误差逐渐减少。动态权重调整策略和 SSBroyden 优化方法相结合会在每个训练轮次中表现出更小的损失函数，表明该方法能够更有效地优化模型。本文结果表明，SSBroyden 方法显著提升了 PINNs 的收敛速度和精度，特别是在处理复杂和刚性 PDEs 时，收敛速度更快，且在复杂优化场景中表现出更强的鲁棒性。这些发现凸显 SSBroyden 方法在加速 PINNs 训练和增强数值稳定性方面的有效性。本研究表明，SSBroyden 方法是训练 PINNs 的高效且可靠的优化器。其处理复杂 PDEs 的能力和强大的收敛特性使其在基于深度学习的科学计算中具有广泛的应用前景。未来的研究可以专注于结合领域特定的适应性方法，以进一步提升 PINNs 在高维和多尺度问题中的准确性、效率和泛化能力。

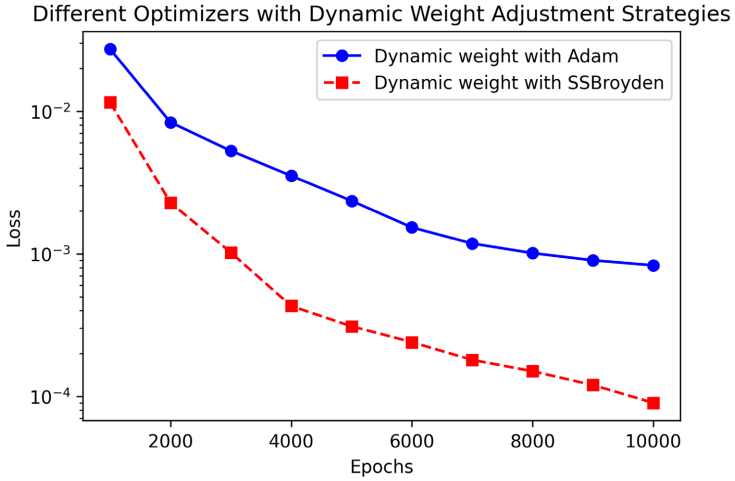

Figure 3. Loss reduction graph of different optimizers under dynamic weighting strategy
图 3. 动态权重策略下不同优化器的损失下降图

5. 总结与展望

本文在基于 PINN 的神经网络的结构下，使用动态权重调整策略，调整神经网络损失函数中的权重。同时，对神经网络当中的优化器进行改变，将 Adam 优化方法变成了 SSBroyden 优化方法。在优化方法改变之后，也成功地在训练轮次不变的前提条件下，将 PINN 的总损失降低了一个数量级，提高了模型训练效率。尽管 PINN 在流体力学问题中已经表现出较高的计算效率和精度，但在大规模复杂流体问题的求解中，仍然面临着计算量和精度的挑战。PINN 依赖于神经网络来逼近流体力学方程的解，这需要大量的训练数据和长时间的训练过程。特别是在三维高雷诺数湍流问题或多物理场耦合问题中，计算资源消耗仍然是一个瓶颈。因此，如何进一步提升 PINN 的计算效率和精度，成为未来研究的一个关键方向。

参考文献

[1] Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2017) Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations. arXiv: 1711.10561.

[2] Gao, H., Sun, L. and Wang, J. (2021) Phygeonet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parameterized Steady-State PDEs on Irregular Domain. *Journal of Computational Physics*, **428**, Article ID: 110079. https://doi.org/10.1016/j.jcp.2020.110079

[3] Ren, P., Rao, C., Liu, Y., Wang, J. and Sun, H. (2022) Phycrnet: Physics-Informed Convolutional-Recurrent Network for Solving Spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering*, **389**, Article ID: 114399. https://doi.org/10.1016/j.cma.2021.114399

[4] Tang, H., Rabault, J., Kuhnle, A., Wang, Y. and Wang, T. (2020) Robust Active Flow Control over a Range of Reynolds Numbers Using an Artificial Neural Network Trained through Deep Reinforcement Learning. *Physics of Fluids*, **32**, Article ID: 053605. https://doi.org/10.1063/5.0006492

[5] Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J. and Wu, J. (2014) MOEA/D with Adaptive Weight Adjustment. *Evolutionary Computation*, **22**, 231-264. https://doi.org/10.1162/evco_a_00109

[6] Kingma, D.P. and Ba, J. (2014) Adam: A Method for Stochastic Optimization. *Computer Science*, **10**, Article ID: 48550.

[7] Lukšan, L. (1994) Computational Experience with Known Variable Metric Updates. *Journal of Optimization Theory and Applications*, **83**, 27-47. https://doi.org/10.1007/bf02191760

[8] Wolkowicz, H. and Zhao, Q. (1995) An All-Inclusive Efficient Region of Updates for Least Change Secant Methods. *SIAM Journal on Optimization*, **5**, 172-191. https://doi.org/10.1137/0805009