

一种求解偏微分方程的动态平衡物理信息神经网络

A dynamic balanced physics-informed neural network for solving partial differential equations

期刊：	中国科学: 信息科学
稿件ID：	SSI-2023-0195.R2
稿件栏目：	论文
作者提交日期：	2024-03-18
参与作者列表：	邓书超, 宋孝天, 钟旻宵, 李庆, 孙亚楠, 吕建成
关键词：	
英文关键词：	
学科领域：	人工智能与机器学习
专题：	

For Review Only

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

一种求解偏微分方程的动态平衡物理信息神经网络

邓书超[1†], 宋孝天[1†], 钟旻霄[1,2], 李庆[2], 孙亚楠[1*], 吕建成[1]

1. 四川大学计算机学院, 成都 610065
2. 中国核动力研究设计院, 成都 610213
* 通信作者. E-mail: ysun@scu.edu.cn
† 同等贡献

四川大学-中国核动力研究设计院创新基金 (批准号: SCU&DRSI-LHCX-11、SCU&NPIC-LHCX-8) 资助项目

摘要 近年来, 物理信息神经网络 (Physics-Informed Neural Networks, PINN) 在求解非线性偏微分方程 (Partial Differential Equations, PDEs) 中得到了大量应用. PINN 将物理信息作为正则化约束加入神经网络损失函数, 可以减少传统神经网络方法对训练数据的大量依赖. 然而, PINN 无法根据数据变化动态调整损失函数中各个损失项的权重, 导致其在求解非线性 PDEs 时存在求解误差较大的问题. 为此, 本文提出了一种动态平衡物理信息神经网络 (Dynamic Balanced PINN, DBPINN). 首先, DBPINN 为 PINN 损失函数的各个损失项设计了一种动态权重系数, 并使用随机函数对该系数进行动态更新, 能够显著提升 PINN 的精度. 其次, DBPINN 为 PNNN 损失函数的各个损失项之间建立了一种平衡求和方法, 该方法考虑了所有损失项之间的竞争关系, 使得 PINN 各损失项朝着有利于收敛的方向进行优化. DBPINN 通过动态权重系数和平衡求和方法使得 PINN 可以更好地进行优化, 进而解决了 PINN 在实际应用中求解误差较大的问题. 本文选择了科学机器学习领域中四个经典的非线性 PDEs 对 DBPINN 进行了数值验证和分析. 实验结果表明, 相比于 PINN, DBPINN 在 Schrodinger 和 Allen-Cahn 方程上误差分别降低了 46% 和 64%. DBPINN 在求解 Navier-Stokes 方程时将系数 λ_1 和系数 λ_2 的误差分别降低了 1 至 2 个数量级和约 50%. DBPINN 在 KdV 方程中能够在多项系数中将误差降低 1 个数量级. 最后, 本文在多种形式的 Burgers 方程和 Allen-Cahn 方程上进行性能和参数消融验证, 结果表明 DBPINN 不仅能够提升模型性能、处理小数据量以及拟合不同时间状态下的方程的能力, 而且 DBPINN 相比于 PINN 具有更好的稳定性、准确率以及收敛性. DBPINN 可以取代 PINN 被应用于各种非线性 PDEs 的高精度求解, 源代码见于:https://github. com/Xiaojiuwo168/DBPINN/.

关键词 物理信息神经网络, 非线性偏微分方程, 动态权重系数, 平衡求和方法, 科学机器学习

引用格式: 邓书超, 宋孝天, 钟旻霄, 等. . 中国科学: 信息科学, 在审文章
Shuchao DENG, Xiaotian SONG, Minxiao ZHONG, et al. (in Chinese). Sci Sin Inform, for review

For Review Only

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

1 介绍

非线性偏微分方程 (Partial Differential Equations, PDEs)[1] 通常用于描述力学、控制过程等各种现象, 对非线性 PDEs 的精确求解在自然科学和工程实践中具有重要的理论和应用价值[2]. 基于网格的方法例如有限差分法 (Finite Difference Method, FDM)[3] 和有限元法 (Finite Element Method, FEM)[4] 是求解非线性 PDEs 的传统方法. 这些方法在求解过程中普遍面临着数据约束的问题, 即求解方程的精度受限于网格划分的精细程度. 如果网格划分足够精细, 则方程求解的精度高; 反之方程求解的精度会降低.

近年来, 具有通用近似能力的神经网络[5,6] 常用于代替传统方法求解非线性 PDEs[7,8]. 基于神经网络求解非线性 PDEs 是一种无网格的方法, 即在求解中没有网格划分的约束, 能够较高精度地处理数学建模复杂的非线性问题. 然而, 使用神经网络求解非线性 PDEs 是一种数据驱动的求解方法, 其求解过程中需要大量关于非线性 PDEs 的数据. 在许多实际应用场景中, 由于求解非线性 PDEs 所需的数据采集周期长或成本高, 基于神经网络求解非线性 PDEs 的方法经常遇到数据量不足等问题. 这些问题限制了使用神经网络求解非线性 PDEs 方法的使用场景, 并且阻碍了该方法在自然科学和工程实践中的广泛应用[9,10].

为了解决基于神经网络求解非线性 PDEs 存在的数据量不足等问题. Raissi 等人[9] 提出了物理信息神经网络 (Physics-Informed Neural Networks, PINN). PINN 能够将非线性 PDEs 现象背后的物理信息作为经验约束, 加入神经网络求解非线性 PDEs 的损失函数中, 从而使得 PINN 能够使用少量数据求解非线性 PDEs, 并且能够能达到很好的求解效果. 由于 PINN 在求解非线性 PDEs 时具有简单高效的特点, 它成为了科学机器学习领域中受到广泛关注的新兴技术[11]. PINN 能有效地解决自然科学和工程实践中各种现象背后与非线性 PDEs 有关的问题[12], 包括方程求解[9]、参数反演[13]、模型发现[14]、控制[15] 和优化[16] 等. 虽然 PINN 能够解决传统神经网络求解非线性 PDEs 所需数据量大的问题, 但是由于 PINN 训练数据量远小于非线性 PDEs 在整个时空域上所包含的数据量, PINN 的求解精度容易受到训练数据的影响. 例如, Bajaj 等人[17] 指出, PINN 容易受到边界数据的干扰, 训练过程中边界数据的拟合误差会传播到整个时空域, 进而增大总体求解误差. Krishnapriyan 等人[11] 指出, PINNs 难以优化新加入正则项后形成的损失函数, 导致 PINNs 不同损失项梯度下降速度不平衡.

大量研究表明, PINNs 在求解非线性 PDEs 时存在应用局限性和预测误差大的问题, 其中最主要的原因是 PINNs 的损失函数中包含多个损失项, 通过梯度下降方法难以同时优化这些损失项[18~20]. 针对如何更好地使用梯度下降优化多个损失项进而降低求解误差, 研究人员提出了多种解决方案. 例如, Wight 等人[18] 提出了一种侧重单一权重赋值的方法, 通过在损失函数中增加某一项的权重, 进而提高模型的学习能力并降低方程求解误差. 但该方法无法选择恰当的权重, 并且无法求解初始条件缺失的非线性 PDEs . 随后, Wang 等人[19] 提出了自适应权重赋值的方法, 通过自适应地将权重赋值给损失函数的不同损失项, 这种方法很好地弥补了单一权重赋值的不足, 但仍存在梯度消失和梯度爆炸的问题, 这些问题限制了它在实际中的应用. 此外, Wang 等人[20] 基于神经切线核 (Neural Tangent Kernel, NTK) 提出了适应 NTK 加权方法, 通过这种方法能够加快单一权重赋值的收敛速度, 进而解决了损失函数存在收敛速度的不平衡并降低求解的误差. 但由于 NTK 特征值需针对不同问题进行设计, 因此该方法在实际应用中存在局限性, 限制了其解决实际问题的能力. 总而言之, 为

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

了解决 PINN 使用梯度下降算法难以优化多个损失项的问题, 研究人员开发了包括侧重单一权重赋值和自适应权重赋值在内的多种方法. 虽然这些方法在一定程度上解决了损失函数难以优化的问题, 但仍存在较大的预测误差以及应用范围局限.

这种应用局限性和误差大的问题在很多经典的算法中也经常出现, 如粒子群算法[21]. Shi 等人[22] 对原始的粒子群算法进行修正并设计了一种新的参数即惯性权重. 惯性权重的加入解决了原始粒子群算法应用的局限性和预测误差大的问题, 并受到学者的青睐. 本文受此启发, 提出了动态平衡物理信息神经网络 (Dynamic Balanced PINN, DBPINN),DBPINN 通过动态权重系数和平衡求和方法使得各损失项朝着有利于收敛的方向进行优化, 进而降低求解非线性 PDEs 的误差. 本文的主要贡献概述如下:

- DBPINN 为损失函数的各个损失项设计了一种动态权重系数, 并使用随机函数对权重系数进行动态更新. 当面对不同的数据或各种边界干扰时, DBPINN 能够通过动态权重系数为求解非线性 PDEs 提供动态抗干扰能力, 能够显著提升 PINN 的精度.
- DBPINN 在动态权重系数的基础上为损失函数建立了一种平衡求和方法, 并考虑了各损失项之间的竞争关系. 当 PINN 中新加入各种损失项以形成最终的损失函数后, DBPINN 在求解非线性 PDEs 时能够通过平衡求和方法让各个损失项朝着有利于收敛的方向进行优化.
- 本文选取科学机器学习领域中五个经典的非线性 PDEs 对 DBPINN 进行数值验证. DBPINN 相较 PINN 在求解非线性 PDEs 时能够很好地同时优化多个损失项, 并且可以更好地收敛, 进而具有更广泛的应用和更低的求解误差. 本文通过大量消融实验证明 DBPINN 不仅能够提升模型性能、处理小数据量以及拟合不同时间状态下的方程, 而且受动态权重系数的取值上下界、系数和以及初始值的影响.

本文第 2 节对相关工作进行简要概述, 包括非线性偏微分方程及物理信息神经网络. 第 3 节详细介绍本文提出的非线性 PDEs 求解方法 DBPINN . 第 4 节通过对五个经典的非线性 PDEs 进行数值验证和消融实验分析验证 DBPINN 的有效性. 第 5 节对全文进行总结.

2 相关工作

2.1 非线性偏微分方程

一般形式的非线性 PDEs[9] 如公式 (1) 所示.

$$u_t + \mathcal{N}[u;\lambda] = 0, x \in \Omega, t \in [0;T]. \tag{1}$$

其中 u 代表方程的解, 并且 u 是由时间-空间信息 t 和 x 组成即 $u(t,x)$. $\mathcal{N}[u;\lambda]$ 是对 u 进行非线性运算, 参数 λ 代表方程中的参数. 根据 λ 是否固定, 可将涉及非线性 PDEs 的问题分为两类: 若 λ 固定, 则称该非线性 PDEs 的问题为数据驱动型问题. 否则, 则称该非线性 PDEs 的问题为数据驱动发现型问题[9]. 其中, 数据驱动型问题主要用于推断系统的隐藏状态, 而数据驱动发现型问题主要用于发现可观测数据的最佳参数 λ . 其主要区别是, 数据驱动发现型问题比数据驱动型问题有更多的学习参数 λ.

非线性 PDEs 在许多具有重大意义的自然科学和工程实践中扮演着重要作用. 许多现实生活中的复杂问题都可以转换成非线性 PDEs 的形式, 尤其是物理、力学等研究领域[23~26]. 因此, 对非线性

For Review Only

PDEs 的精确求解具有很重要的理论和应用价值. 为此, 研究人员开发了许多求解非线性 PDEs 的方法. 在研究早期阶段, 研究人员提出了解析法求解非线性 PDEs, 该方法能够拟合任意时刻非线性 PDEs 的解. 然而, 对于很多实际问题, 由于涉及到多个物理因素的相互作用, 非线性 PDEs 的复杂度很高, 通过解析法无法获得精确的解析解. 随后, 研究人员基于网格形式提出了求解非线性 PDEs 数值解的方法, 即在计算机上通过数值计算的方法对非线性 PDEs 进行近似求解. 例如, 经典的有限差分法和有限元法. 这类方法通过对问题的求解区域进行网格划分, 然后对待求解的问题进行等价形式离散并归结为线性方程组, 最终通过计算机求得非线性 PDEs 在离散网格上的近似解. 但是这类方法对非线性 PDEs 的精确求解受限于网格划分的精细程度. 为此, 研究人员希望找到无网格划分的非线性 PDEs 的求解方法. 近年来, 随着深度学习的发展, 研究人员开始使用神经网络求解非线性 PDEs. 例如, Long 等人[27] 基于数据驱动提出了求解非线性 PDEs 的 PDe-Net, 其主要思想是采用前馈深度神经网络来近似方程的解. 随后, Long 等人[28] 在 PDe-Net 的基础上结合 Symnet (Symbolic neural network) 开发出了 PDe-Net2.0. PDe-Net2.0 能够在使用较少先验知识的情况下实现更准确的预测. 此外, Wu 等人[29] 提出了一个利用观测数据和深度神经网络来求解方程的框架. 具体地, Wu 等人[29] 将 ResNet 作为基本网络以求解非线性 PDEs, 并通过大量实验验证了该方法的拟合效果. 采用神经网络求解非线性 PDEs 的方法在求解过程中需要大量的数据, 然而在实际场景中, 数据的获取周期长或成本高, 这限制了这类纯数据驱动下的非线性 PDEs 神经网络求解方法.

2.2 物理信息神经网络

PINN 是一种求解非线性 PDEs 的神经网络方法, 与传统纯数据驱动的神经网络求解方法不同, PINN 在常规神经网络模型的训练过程中引入蕴含物理规律的正则项, 能够提高在数据量较少情况下非线性 PDEs 的求解效率. 具体地, PINN 通过最小化神经网络的损失函数来近似非线性 PDEs 的解. 其中, 神经网络的损失函数由初始条件的损失项、执行循环边界条件的损失项以及对方程进行控制的损失项等构成. PINN 经过训练后, 可以获得非线性 PDEs 在空间-时间点上的数值解. 一般受约束的非线性 PDEs 可以写成目标式 (2) 的形式.

$$\mathcal{F}(u(z);\lambda)=0, \quad z\in\Omega, \tag{2}$$

$$s.t. \quad \mathcal{B}(u(z))=0, \quad z\in\Omega, \tag{2.a}$$

$$\mathcal{I}(u(z))=0, \quad z\in\Omega. \tag{2.b}$$

其中 $z=(x_1,x_2,\ldots,x_n;t)$ 包含了非线性 PDEs 在空间和时间的坐标, $u(z)$ 表示方程的解, λ 是控制参数, Ω 是方程所在的区域, 函数 $\mathcal{F}$ 描述控制方程, 函数 $\mathcal{B}$ 表示边界条件, 函数 $\mathcal{I}$ 表示初始条件. 函数 $\mathcal{F}$, $\mathcal{B}$ 和 $\mathcal{I}$ 可以通过关于 z 的自动微分来构建[30]. PINN 通常采用多层感知机[31] 作为神经网络模型, 通过训练该网络以近似公式 (2) 的真实解 $u(z)$. 公式 (2.a) 和 (2.b) 代表边界条件和初始条件对非线性 PDEs 的约束. PINN 中采用平均平方误差 (Mean Squared Error, MSE)[32] 作为损失函数的约束标准, 损失函数 $MSE(\theta)$ 如公式 (3) 所示.

$$MSE(\theta)=MSE_o+MSE_b+MSE_f. \tag{3}$$

其中 MSE_o, MSE_b 和 MSE_f 分别代表对初始条件的损失项、执行循环边界条件的损失项以及对方程进行控制的损失项. 这三个约束项在实践中使用广泛[9,18]. 此外, 为了满足不同问题中的物理信

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

息, PINN 的损失函数还可以加入其它相关的正则项和各种先验信息. 通过将非线性 PDEs 的各个约束条件表示为不同损失项, PINN 求解非线性 PDEs 可以建模为公式 (4) 所示的优化问题, 采用神经网络方法求解.

$$\theta^* = \arg\min_{\theta} MSE(\theta). \tag{4}$$

许多文献指出, PINN 各损失项梯度下降速度不平衡, 这使得 PINN 很难优化新加入正则化后形成的损失函数, 并且导致模型出现较大的预测误差[11, 33]. 针对该问题, 研究人员提出了多种解决方案, 旨在探索如何更好地使用梯度下降算法优化多个损失项, 以进一步降低预测误差[18~20]. 例如, Wight 等人[18] 认为神经网络应该侧重满足损失函数中关于初始条件的损失项. 基于此原则, Wight 等人设计出一种损失函数的一般形式:$\mathcal{L}(\theta) = C\mathcal{L}_o + \mathcal{L}_b + \mathcal{L}_f$. 其中 $\mathcal{L}(\theta)$ 代表总的损失函数, $\mathcal{L}_o$, $\mathcal{L}_b$ 和 $\mathcal{L}_f$ 分别代表对初始条件的损失项, 执行循环边界条件的损失项以及对方程进行控制的损失项, C 代表为重点满足 $\mathcal{L}_o$ 代表的权重, 通常取值大于 1. 这种人为对损失项进行单一权重赋值的方法也称为非适应性加权方法. 尽管该方法在使用 PINN 求解非线性 PDEs 的初期获得了良好的近似效果, 但是权重 C 对于不同的非线性 $PDEs$ 往往具有不同的取值, 因此该方法缺少通用性, 无法广泛应用于求解不同的非线性 $PDEs$. 此外, 权重 C 由于需要作用在初始条件上, 因此该方法无法对实际中缺少初始条件的非线性 $PDEs$ 进行求解. Wang 等人[19] 进一步提出一种损失项权重自适应方法, 采用学习率退火算法动态调整损失函数不同损失项的权重, 进而解决权重 C 选择困难和发挥失效的问题. 该方法利用损失函数的反向传播梯度信息, 在训练 PINN 期间自适应地调整多个损失项的权重. 这种自适应调节损失函数的方法在一定程度上降低了预测误差, 并且这种方法可以推广到其它具有多个目标函数相互作用的任务中. 然而, 该方法中权重本身并非通过反向传播进行调整, 而是通过更新学习率影响权重的更新, 因此仍存在梯度消失和梯度爆炸等训练困难的问题, 限制了该方法的进一步应用. 进一步地, Wang 等人[20] 提出了一种适应性 NTK 加权方法, 从 NTK 的角度分析和解释了一些 PINN 模型难以训练的问题. 此外, 通过分析 PINN 的极限 NTK, Wang 等人发现不同损失项的收敛速率存在明显差异, 这可能导致预测误差较大的问题. 针对该问题, Wang 等人提出了一种新的梯度下降算法以解决不同损失项收敛速率不平衡的问题, 并利用 NTK 的特征值来适应性改善总训练误差的收敛速率. 尽管这种结合 NTK 的新型梯度下降算法能够提高 PINN 损失项的收敛速率和预测准确率, 但是在实践中这种方法无法改变 NTK 的特征值分布, 因此不能直接用于解决实际问题. 此外, 结合 NTK 的新型梯度下降算法对权重进行分配可能形成不确定的核, 从而导致训练过程不稳定并增大预测误差.

3 本文提出的 DBPINN

本节将介绍 DBPINN 求解非线性 PDEs 的具体实现. 首先, 我们将介绍 DBPINN 的四个算法步骤. 随后, 我们将介绍 DBPINN 求解非线性 PDEs 的详细过程. 最后, 我们将介绍 DBPINN 的动态权重系数和平衡求和方法.

For Review Only

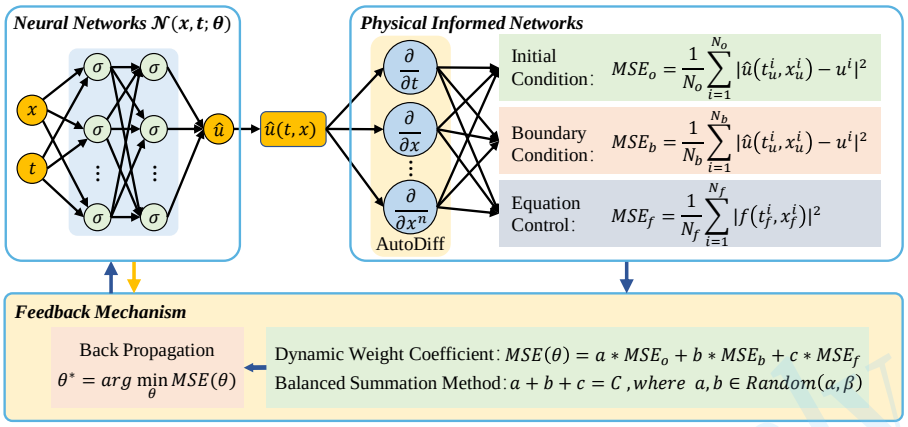


图 1 (网络版彩图) DBPINN 的工作原理示意图, 主要有三个主要部分: 神经网络、物理信息网络和反馈机制

Figure 1 (Color online)The overall framework of DBPINN, with three main components: Neural Networks, Physical Informed Networks and Feedback Mechanism

3.1 DBPINN 的算法步骤

DBPINN 求解非线性 PDEs 的过程可以概述为算法 1 中的四个步骤. 在步骤 1 中: 确定神经网络初始化所需参数, 构建带有参数的神经网络 $\mathcal{N}(x,t,\theta)$; 在步骤 2 中: 划分神经网络训练所需数据集, 明确三个训练数据即方程控制的数据 N_f、边界条件的数据 N_b 以及初始条件的数据 N_o; 在步骤 3 中: 根据划分出的三个训练数据集来构建对应损失项, 结合所提出的动态权重系数和平衡求和方法来构建最终的损失函数 $MSE(\theta)$; 在步骤 4 中: 通过最小化损失函数 $MSE(\theta)$ 找到最佳网络参数 θ, 实现 DBPINN 对非线性 PDEs 的求解.

算法 1 DBPINN 求解非线性 PDEs

1: 构建带有参数的神经网络 $\mathcal{N}(x,t,\theta)$;
2: 明确三个训练数据集: 方程控的数据 N_f、边界条件的数据 N_b 和初始条件的数据 N_o;
3: 通过 N_f, N_b 和 N_o 构建各损失项, 结合动态权重系数和平衡求和方法构建损失函数 $MSE(\theta)$;
4: 通过最小化 $MSE(\theta)$ 训练神经网络以找到最优网络参数 θ.

3.2 DBPINN 求解非线性 PDEs

我们假设 $f(t,x)$ 代表物理信息网络, 如公式 (5) 所示.

$$f := u_t + \mathcal{N}[u]. \tag{5}$$

DBPINN 可以通过构建带有参数的神经网络对 $u(t,x)$ 进行逼近. 其中, 物理信息网络 $f(t,x)$ 和神经网络求解 $u(t,x)$ 的网络参数相同. DBPINN 求解非线性 PDEs 的流程如图1所示, 求解过程主要分为三个部分: 神经网络、物理信息网络以及反馈机制.

神经网络部分的基本架构是多层感知机, 它通过对输入数据进行非线性拟合进而可以得到方程的解. DBPINN 通过构建带有参数的神经网络 $\mathcal{N}(x,t,\theta)$, 将 (t,x) 作为 $\mathcal{N}(x,t,\theta)$ 的输入, 经过神经网络求解得到方程的近似解 $\hat{u}(t,x)$. 该近似解与真实解 $u(t,x)$ 维度相同. 其中 $\theta = \{W^l, b^l\}_{(1\leqslant l\leqslant L)}$ 代表神经网络 $\mathcal{N}(x,t,\theta)$ 中的所有权重矩阵和偏置向量.

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

物理信息网络部分负责对神经网络拟合的解 $\hat{u}(t,x)$ 进行自动微分, 从而构建最终损失函数的各个损失项. 具体地, 在自动微分操作会将 $\hat{u}(t,x)$ 转化为一系列偏导数. 这些偏导数通过基础运算等组合方式形成了初始条件损失项 MSE_o、边界条件损失项 MSE_b 和方程控制损失项 MSE_f. 其中, DBPINN 能够很好地将物理信息融入到最终的损失函数中. 因此, DBPINN 在训练时只需要训练神经网络 $\mathcal{N}(x,t,\theta)$, 并且充足的物理信息使得 DBPINN 能够高效处理小数据量非线性 PDEs 求解问题.

反馈机制部分通过对神经网络 $\mathcal{N}(x,t,\theta)$ 进行训练以得到效果最优的参数 θ. 具体地, MSE_o, MSE_b 和 MSE_f 结合动态权重系数后求和得到了最终的损失函数 $MSE(\theta)$. 动态权重系数中动态调整的权重系数 a, b 和 c 满足平衡求和方法的约束, 这使得 DBPINN 在设计权重系数时兼具动态性和稳定性. 动态权重系数和平衡求和方法会在后面的两个小节详细说明.

为了进一步降低 DBPINN 求解非线性 PDEs 的误差, 我们针对 DBPINN 的反馈机制进行优化. 具体地, 我们提出了动态权重系数和平衡求和方法. 如图 1所示, 上述方法作用在 DBPINN 的反向传播阶段, 能够有效降低损失函数 $MSE(\theta)$ 的误差, 加速网络收敛并提升求解精度. 接下来, 我们将详细阐述这两种方法.

3.3 动态权重系数

首先是动态权重系数, 我们在物理信息网络中为各损失项设计了一种动态系数. 如图1所示, 通过随机初始化生成 $MSE(\theta)$ 的系数 a, b 和 c, 并在 DBPINN 训练过程中通过限定范围的随机函数对权重系数进行动态更新. 动态权重系数在 DBPINN 为求解非线性 PDEs 提供动态抗干扰能力, 进而解决现有方法存在应用范围局限的问题. 具体地, 在 DBPINN 的训练过程中, 动态权重系数能够动态地为损失函数中误差过大的损失项生成较小的系数, 以解决由于某损失项误差过大导致最终损失函数误差大的问题. 相应地, 动态权重系数也能够对损失函数中误差过小的损失项生成较大的系数, 以解决由于某损失项误差过小而忽略该采样点所蕴含的物理信息的问题. 总的来说, 动态权重系数使用随机函数进行动态更新能够使仅含单个损失项的网络更好地收敛. 当面对不同的数据或各种边界干扰时, DBPINN 能够通过动态权重系数为求解非线性 PDEs 提供动态抗干扰能力, 能够显著提升 PINN 的精度, 进而拓展了 DBPINN 在实际中的应用范围.

3.4 平衡求和方法

其次是平衡求和方法, 由于动态权重系数仅局限于损失函数中各个独立的损失项, 并未考虑到各个损失项之间的竞争关系. 通过动态权重系数方法为各个损失项设计权重系数后容易受到单一损失项的影响. 这表明在求解非线性 PDEs 时, 如果 MSE_0, MSE_b 和 MSE_f 这三项中某一项的权重系数过大或过小, 将导致 PDEs 的求解过程脱离实际, 使得求解精度降低. 针对该问题, 我们设计了平衡求和方法. 如图1所示, DBPINN 通过平衡求和方法将各个损失项在动态权重系数的基础上进行关联约束. 例如, 当 MSE_0 和 MSE_b 的值较大时, 动态权重系数可以对损失项分别生成两个较小的值, 通过平衡求和方法可以使得 MSE_f 损失项对应的系数值较大. 平衡求和方法考虑了各个损失项之间的竞争关系, 让各个损失项朝着有利于收敛的方向进行优化, 进一步降低了 DBPINN 在实际应用中的预测误差.

For Review Only

为了便于理解动态权重系数和平衡求和方法, 我们以求解一般形式非线性 PDEs 为例, 介绍这两种方法的具体流程. DBPINN 求解非线性 PDEs 的损失函数如公式 (6) 所示.

$$MSE(\theta) = a \times MSE_o + b \times MSE_b + c \times MSE_f, \tag{6}$$

$$s.t. \quad a \times MSE_o = a \times \frac{1}{N_o} \sum_{i=1}^{N_o} |u(t_u^i, x_u^i) - u^i|^2, \tag{6.a}$$

$$b \times MSE_b = b \times \frac{1}{N_b} \sum_{i=1}^{N_b} |u(t_u^i, x_u^i) - u^i|^2, \tag{6.b}$$

$$c \times MSE_f = c \times \frac{1}{N_f} \sum_{i=1}^{N_f} |u(t_f^i, x_f^i)|^2, \tag{6.c}$$

$$a, b \in Random(\alpha, \beta), \tag{6.d}$$

$$a + b + c = Num\{MSE_o, MSE_b, MSE_f\}. \tag{6.e}$$

该损失函数 $MSE(\theta)$ 由三部分求和组成, 如公式 (6.a)-(6.c) 所示, 分别代表初始条件的损失项, 执行循环边界条件的损失项以及对方程进行控制的损失项. 而 $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_o}$, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_b}$ 和 $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ 分别代表初始训练数据, 边界训练数据和方程的拟合点. 此外, 公式 (6.d) 代表权重大小的约束, 公式 (6.e) 代表权重求和的约束. 其中常数 $\alpha = 0.8$ 和 $\beta = 1.3$ 代表 a 和 b 的取值上下界. Num 代表损失项的数量, 通常取值等于 3. DBPINN 通过最小化公式 (6) 中的损失函数 $MSE(\theta)$, 求解 $u(t, x)$ 和物理信息神经网络 $f(t, x)$ 的共享参数, 最终达到求解非线性 PDEs 的目的. 在最小化损失函数 $MSE(\theta)$ 的过程中, 采用以下步骤对各损失项权重进行更新. 首先, 动态权重系数 a, b 和 c 是从区间 [0.8, 1.3] 之间随机生成的, 在每一次训练迭代中不断进行动态更新, 如公式 (6) 和 (6.d) 所示. 具体地, 训练中若 MSE_o 的值过大, 那么动态权重系数 a 在训练迭代中可能会生成较小的值如 $a = 0.81$ 等; 若 MSE_0 的值过小, 那么动态权重系数 a 在训练迭代中会生成较大的值如 $a = 1.28$ 等; 类似地, 这些系数能够解决由于某损失项误差过大导致总体误差过大的问题, 或解决由于某损失项误差过小导致忽略掉该采样点所蕴含的物理信息的问题, 进而能够提升 DBPINN 的抗干扰能力和拓展 DBPINN 的应用范围. 此外, 如公式 (6) 和 (6.e) 所示, 平衡求和方法将动态权重系数 a, b 和 c 进行关联约束, 该方法考虑了系数 a, b 和 c 之间的竞争关系, 例如, 训练中若 MSE_o 和 MSE_b 的值过大, 那么动态权重系数 a 和 b 在训练迭代中会生成较小的值如 $a = 0.91$ 和 $b = 0.83$ 等, 我们通过平衡求和方法 ($c = Num\{MSE_o, MSE_b, MSE_f\} - a - b$) 可以得出 $c = 1.26$. 即该方法可以在系数 a 和 b 值较小时, 使得生成系数 c 的值较大, 平衡求和方法能够让 DBPINN 忽略损失项误差相对较大的 MSE_o 和 MSE_b, 而更加关注损失项误差较小的 MSE_f, 从而进一步降低了 DBPINN 在实际应用中的预测误差. 最终, 经过多次的迭代重复得到最优的参数, 达到高效求解非线性 PDEs 的目的.

4 数值验证和实验分析

本节中, 我们使用五个经典的非线性 PDEs 对 DBPINN 进行数值验证和消融实验分析. 首先, 我们在连续时间模型和离散时间模型上对 DBPINN 进行了数值验证. 其次, 我们从多个维度对 DBPINN 进行了消融实验分析 (详细实验结果见补充材料). 我们使用 Tensorflow 框架编写实验代

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

码, 并在配有 NVIDIA 2080 Ti GPU 卡的 Ubuntu 服务器上进行实验.

4.1 连续时间模型

薛定谔方程 (Schrodinger equation). 在量子力学中, 每个微观系统都对应着一个薛定谔方程[34]. 通过求解微观系统对应的薛定谔方程, 可获得波函数的具体形式和相应的能量, 这有助于我们更好地了解微观系统的性质. 其中一维的非线性薛定谔方程常被用于研究量子力学系统, 例如非线性波在光束或波中的传播. 在此, 我们选用具有周期性边界条件且受到数据驱动的一维非线性薛定谔方程进行实验, 进而证明 DBPINN 能够处理这种带有周期性边界条件的非线性 PDEs, 并验证 DBPINN 在处理数据驱动问题上的有效性. 其中具有一维边界和初始条件的薛定谔方程如公式 (7) 所示.

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \tag{7}$$

$$s.t. \quad x \in [-5, 5], t \in \left[0, \frac{\pi}{2}\right], \tag{7.a}$$

$$h(0, x) = 2sech(x), \tag{7.b}$$

$$h(t, -5) = h(t, 5), \tag{7.c}$$

$$h_x(t, -5) = h_x(t, 5). \tag{7.d}$$

其中 i 是虚数的单位, $h(t, x)$ 为方程的解, h_t 代表 $h(t, x)$ 对 t 的求一次偏导数, h_{xx} 代表 $h(t, x)$ 对 x 求二次偏导数. 式子 $(7.a)$ 代表时间 t 和空间 x 的取值范围, 式子 $(7.b)$-$(7.d)$ 代表初始条件和周期性边界条件对非线性 PDEs 的约束. 为了方便我们后续的操作和融入物理信息, 我们定义了物理信息网络 $f(t, x)$, 如公式 (8) 所示.

$$f := ih_t + 0.5h_{xx} + |h|^2 h. \tag{8}$$

通过物理神经网络 $f(t, x)$ 完成对 $h(t, x)$ 的逼近. 如公式 (9) 所示, $h(t, x)$ 和 $f(t, x)$ 的共享参数可以通过最小化损失函数 $MSE(\theta)_{SDE}$ 来学习, $MSE(\theta)_{SDE}$ 中的 SDE 代表薛定谔方程. 具体地, DBPINN 的初始条件的损失项、边界条件的损失项以及对方程控制约束的损失项如公式 $(9.a)$-$(9.e)$ 所示.

$$MSE(\theta)_{SDE} = a \times MSE_o + b \times MSE_b + c \times MSE_f, \tag{9}$$

$$s.t. \quad a \times MSE_o = a \times \frac{1}{N_o} \sum_{i=1}^{N_o} |h(0, x_0^i) - h_0^i|^2, \tag{9.a}$$

$$b \times MSE_b = b \times \frac{1}{N_b} \sum_{i=1}^{N_b} \{|h^i(t_b^i, -5) - h^i(t_b^i, -5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2\}, \tag{9.b}$$

$$c \times MSE_f = c \times \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2, \tag{9.c}$$

$$a, b \in Random(\alpha, \beta), \tag{9.d}$$

$$a + b + c = Num\{MSE_o, MSE_b, MSE_f\} = 3. \tag{9.e}$$

For Review Only

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

表 1 DBPINN 在薛定谔方程上十次独立运行的结果以及均值和 PINN 比较

Table 1 Results of ten independent runs of DBPINN on Schrödinger's equation and comparison of average and PINN

Test	1	2	3	4	5	6	7	8	9	10	DBPINN	PINN
MSE	9.95e-04	9.64e-04	1.08e-03	1.14e-03	1.06e-03	1.08e-03	1.12e-03	1.04e-03	1.17e-03	9.46e-04	**1.06e-03**	1.97e-03

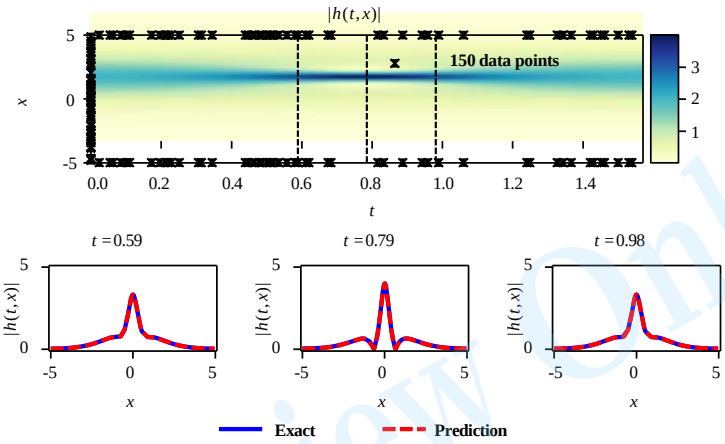


图 2 (网络版彩图) 顶部: 薛定谔方程的预测解 $h(t,x)$, 初始训练数据以及边界训练数据的位置. 底部: 上图中虚线所描述的三个时间点, 即 t 分别为 0.59, 0.79, 0.98 时预测解和真实值的比较

Figure 2 (Color online) Top: The predicted solution $h(t,x)$ of the Schrödinger equation, the initial training data and the location of the boundary training data. Bottom: Comparison between the predicted solution and the true value at the three time points depicted by the dashed lines in the above figure, i.e., t at 0.59, 0.79, and 0.98, respectively

其中 $\{x_0^i, h_0^i\}_{i=1}^{N_o}$, $\{t_b^i\}_{i=1}^{N_b}$ 和 $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ 分别代表初始数据、边界上的坐标点以及 $f(t,x)$ 上的坐标点. MSE_o, MSE_b 和 MSE_f 分别是初始数据的损失项, 强制执行循环边界条件的损失项和对方程进行控制的损失项.

我们使用 Raissi 等人[9] 通过传统光谱方法创建的高分辨率数据集以验证 DBPINN 的有效性. 在数据驱动的情况下, 我们在 $t = 0$ 时获取所有的训练数据, 其目的是推断出薛定谔方程在整个时空域上的解 $h(t,x)$. 训练数据被分为三部分: 从数据集中随机解析的初始条件 $N_o = 50$ 的初始数据点, 边界条件 $N_b = 50$ 的采样网格点以及约束方程 $N_f = 20,000$ 的采样网格点, 它们的位置分布如图2上方所示. DBPINN 对真实值的拟合情况以可视化的方式呈现在图2底部, 其 t 为 0.59、0.79 和 0.98 时蓝色线条代表真实值, 红色线条代表 DBPINN 的预测结果. 从图2中我们可以看出, 在不同的时间点 DBPINN 的预测曲线均与真实曲线高度重合, 这表明 DBPINN 预测值能够很好地拟合真实值. 此外, 我们在相同的条件下分别运行十次 DBPINN 和 PINN, 以验证 DBPINN 的稳定性和准确性. 十次独立运行的结果如表1所示. 其中 Tests 行中的十条数据代表 DBPINN 在薛定谔方程上十次独立运行的结果. 实验表明 DBPINN 在十次独立运行中得到的结果在 9.46e-04 和 1.17e-03 之间波动, 即波动范围很小, 这也说明 DBPINN 具有良好的稳定性. 此外, DBPINN 十次运行误差的平均值为 1.06e-03, 而 PINN 的结果为 1.97e-03. 结果表明 DBPINN 的平均误差比 PINN 降低了 46%, 即 DBPINN 能够大幅度降低误差.

纳维尔-斯托克斯方程 (Navier-Stokes equation). 纳维尔-斯托克斯方程是描述粘性不可压缩流体中动量守恒的运动方程. 纳维尔-斯托克斯方程反映了粘性流体 (也称为真实流体) 流动的基

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

For Review Only

本力学规律, 在流体力学中具有重要意义. 纳维尔-斯托克斯方程[35] 的应用范围广泛, 包括飞机、汽车和发电站等工程设计, 以及对血流和污染物分布的研究. 在此, 我们以二维 (2D) 的纳维尔-斯托克斯方程[35] 为例进行实验, 进而验证了 DBPINN 能够处理这种流体有关的非线性 PDEs, 并验证 DBPINN 在处理数据驱动发现问题上的有效性. 2D 的纳维尔-斯托克斯方程是一个具有边界和初始条件的非线性 PDEs, 它的具体形式如公式 (10)-(11) 所示.

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}). \tag{10}$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}). \tag{11}$$

其中 $u(t,x,y)$, $v(t,x,y)$ 和 $p(t,x,y)$ 分别表示速度场的 x 分量, y 分量和压力. λ_1, λ_2 是纳维尔-斯托克斯方程的未知系数. 纳维-斯托克斯方程的解可用无发散函数的方式求得, 如公式 (12) 所示.

$$u_x + u_y = 0. \tag{12}$$

为更容易表达各分量之间的联系, 我们假设一个解 $\psi(t,x,y)$ 由两个分量组成, 如公式 (13) 所示.

$$u = \psi_x \quad and \quad v = \psi_y. \tag{13}$$

有了这个假设, 方程的连续性即可得到保证. 为了定义出函数 $f(t,x,y)$ 和 $g(t,x,y)$, 我们需学习参数 λ 和压力 $p(t,x,y)$, 如公式 (14)-(15) 所示.

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy}). \tag{14}$$

$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy}). \tag{15}$$

我们使用具有两个输出的神经网络来近似 $[\psi(t,x,y), p(t,x,y)]$, 并构建物理信息网络 $[f(t,x,y), g(t,x,y)]$. 我们通过最小化 $MSE(\theta)_{NSE}$ 来训练网络, $MSE(\theta)_{NSE}$ 中的 NSE 代表纳维尔-斯托克斯方程. 此外, 我们在训练物理信息神经网络 $[f(t,x,y), g(t,x,y)]$ 时也可以学习参数 λ. 其中, 三个损失项和相关限制条件如方程 (16)-(16.e) 所示.

$$MSE(\theta)_{NSE} = a \times MSE_o + b \times MSE_b + c \times MSE_f, \tag{16}$$

$$s.t. \quad a \times MSE_o = a \times \frac{1}{N_o}\sum_{i=1}^{N_o}\{|u(t^i,x^i,y^i) - u^i|^2 + |v(t^i,x^i,y^i) - v^i|^2\} \tag{16.a}$$

$$b \times MSE_b = b \times \frac{1}{N_b}\sum_{i=1}^{N_b}\{|u(t^i,x^i,y^i) - u^i|^2 + |v(t^i,x^i,y^i) - v^i|^2\}, \tag{16.b}$$

$$c \times MSE_f = c \times \frac{1}{N_f}\sum_{i=1}^{N_f}\{|f(t^i,x^i,y^i)|^2 + |g(t^i,x^i,y^i)|^2\}, \tag{16.c}$$

$$a, b \in Random(\alpha, \beta), \tag{16.d}$$

$$a + b + c = Num\{MSE_o, MSE_b, MSE_f\} = 3. \tag{16.e}$$

其中 $\{u(t^i,x^i,y^i), v(t^i,x^i,y^i)\}_{i=1}^{N_o}$, $\{u(t^i,x^i,y^i), v(t^i,x^i,y^i)\}_{i=1}^{N_b}$ 和 $\{t^i,x^i,y^i\}_{i=1}^{N_f}$ 分别代表初始数据, 边界上的数据点和 $f(t,x)$ 上的数据点.

For Review Only

表 2 针对纳维尔-斯托克斯方程, DBPINN 在无噪声和 1% 噪声情况下, 通过学习 λ_1, λ_2 和 $p(t,x,y)$ 得到的拟合方程与 PINN 进行比较. 正确的方程中:$\lambda_1 = 1.00$, $\lambda_2 = 0.01$

Table 2 For the Navier-Stokes equations, the fitted equations obtained by learning λ_1, λ_2 and $p(t,x,y)$ for DBPINN in the no-noise and 1% noise cases are compared with the PINN. In the correct equation:$\lambda_1 = 1.00$, $\lambda_2 = 0.01$

Method	Noise	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy}), v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
PINN	No	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy}), v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
	1%	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy}), v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$
DBPINN	No	$u_t + 1.000(uu_x + vu_y) = -p_x + 0.01030(u_{xx} + u_{yy}), v_t + 1.000(uv_x + vv_y) = -p_y + 0.01030(v_{xx} + v_{yy})$
	1%	$u_t + 1.000(uu_x + vu_y) = -p_x + 0.01026(u_{xx} + u_{yy}), v_t + 1.000(uv_x + vv_y) = -p_y + 0.01026(v_{xx} + v_{yy})$

表 3 DBPINN 在纳维尔-斯托克斯方程上十次独立运行结果和 PINN 比较

Table 3 Comparison of DBPINN results and PINN for ten independent runs on the Navier-Stokes equations

Param.	0	1	2	3	4	5	6	7	8	9	DBPINN	PINN
λ_1	5.78e-05	1.74e-05	1.47e-05	1.43e-04	3.34e-05	5.78e-05	1.74e-05	1.47e-05	1.43e-04	3.34e-05	**5.33e-05**	7.80e-04
λ_2	2.56e-02	2.98e-02	2.24e-02	2.88e-02	2.65e-02	2.56e-02	2.98e-02	2.24e-02	2.88e-02	2.65e-02	**2.66e-02**	4.67e-02
$\lambda_1(\%)$	2.08e-04	8.20e-06	8.30e-05	3.06e-05	5.33e-05	2.08e-04	8.20e-06	8.30e-05	3.06e-05	4.88e-04	**7.66e-05**	1.70e-03
$\lambda_2(\%)$	3.20e-02	2.34e-02	3.29e-02	2.16e-02	2.66e-02	3.20e-02	2.34e-02	3.29e-02	2.16e-02	4.08e-02	**2.73e-02**	5.70e-02

如表2所示, DBPINN 在无噪声和 1% 噪声的情况下能很好地拟合正确的方程. 例如, 当拟合系数为 λ_1 时 DBPINN 与正确的方程一致. 此外, 如表3所示, 我们在相同的条件下独立运行 DBPINN 与 PINN 各十次. 从表3可以看出 DBPINN 在无噪声的情况下对参数 λ_1 和 λ_2 的拟合误差为 5.33e-05 和 2.66e-02, 而 PINN 对参数 λ_1 和 λ_2 的拟合误差为 7.80e-04 和 4.67e-02, DBPINN 在参数 λ_1 和 λ_2 上的拟合误差相比 PINN 降低了约 93% 和 43%. DBPINN 在 1% 的噪声情况下对参数 λ_1 和 λ_2 的拟合误差为 7.66e-05 和 2.73e-02, 而 PINN 对参数 λ_1 和 λ_2 的拟合误差为 1.70e-03 和 5.70e-02, DBPINN 在参数 λ_1 和 λ_2 上的拟合误差相比 PINN 降低了约 95% 和 52%. 总的来说, DBPINN 在参数 λ_1 和 $\lambda_1(1\%)$ 上比 PINN 降低了 1 个和 2 个数量级的误差. DBPINN 在参数 λ_2 和 $\lambda_2(1\%)$ 上比 PINN 分别降低了 43% 和 52% 的误差.

4.2 离散时间模型

艾伦-卡恩方程 (Allen–Cahn equation). 艾伦-卡恩方程[36] 是数学和物理学领域中经典的反应-扩散方程. 艾伦-卡恩方程具体描述了多组分合金体系的分离过程, 如有序-无序转换. 在此, 我们通过艾伦-卡恩方程来进行实验, 进而证明 DBPINN 能够处理这种反应-扩散有关的非线性 PDEs, 并验证 DBPINN 在处理数据驱动问题上的有效性. 带有周期性边界条件的艾伦-卡恩方程如公式 (17) 所示.

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \tag{17}$$

$$s.t. \quad x \in [-1,1], t \in [0,1], \tag{17.a}$$

$$u(0,x) = x^2 \cos \pi x, \tag{17.b}$$

$$u(t,-1) = u(t,1), \tag{17.c}$$

$$u_x(t,-1) = u_x(t,1). \tag{17.d}$$

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

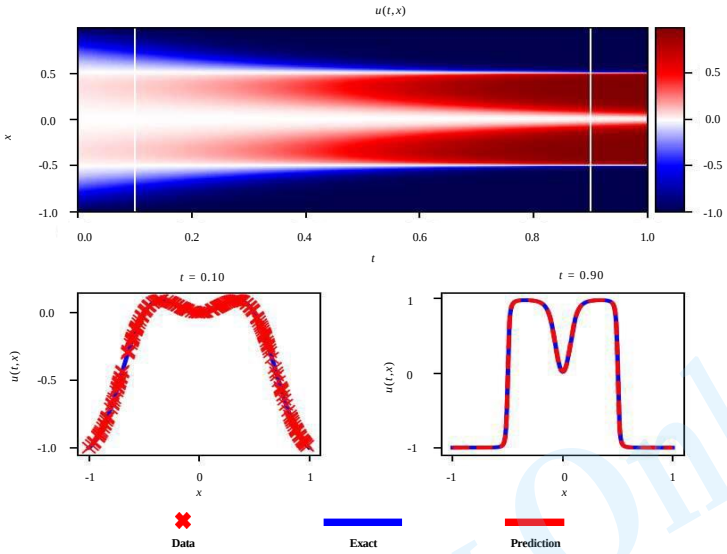


图 3 (网络版彩图) 顶部: 艾伦-卡恩方程的解 $u(t,x)$ 与初始训练时间点 $t=0.1$ 和预测时间点 $t=0.9$ 的位置. 底部: 左边是初始训练时的数据分布, 右边是预测值与真实值的拟合

Figure 3 (Color online) Top:The solution $u(t,x)$ of the Allen-Cahn equation with the initial training time point $t=0.1$ and the predicted time point $t=0.9$. Bottom:The data distribution at initial training on the left, and the fit of the predicted values to the true values on the right

表 4 DBPINN 和 PINN 对艾伦-卡恩方程的十次独立运行结果

Table 4 Results of ten independent runs of DBPINN and PINN for the Allen-Cahn equation

Model	0	1	2	3	4	5	6	7	8	9	Avg.
PINN	7.07e-03	1.90e-03	1.42e-02	5.52e-03	4.97e-03	8.54e-03	4.69e-03	6.41e-03	5.91e-02	7.57e-03	1.20e-02
DBPINN	2.29e-03	4.04e-03	4.51e-03	3.43e-03	6.97e-03	4.18e-03	5.70e-03	3.94e-03	2.85e-03	4.75e-03	**4.27e-03**

其中, 式子 (17.a) 代表时间 t 和空间 x 的取值范围, 式子 (17.b)-(17.d) 代表初始条件和周期性边界条件对非线性 PDEs 的约束. 针对艾伦-卡恩方程, 我们的目标是最小化 $MSE(\theta)_{ACE}$, $MSE(\theta)_{ACE}$ 中的 ACE 代表艾伦-卡恩方程. 该方程的三个损失项和相关限制条件如 (18)-(18.e) 所示.

$$MSE(\theta)_{ACE} = a \times MSE_o + b \times MSE_b + c \times MSE_f, \tag{18}$$

$$s.t. \quad a \times MSE_o = a \times \sum_{i=1}^{q} \{|u^{n+c_i}(-1) - u^{n+c_i}(1)|^2 + |u^{n+1}(-1) - u^{n+1}(1)|^2\}, \tag{18.a}$$

$$b \times MSE_b = b \times \sum_{i=1}^{q} \{|u_x^{n+c_i}(-1) - u_x^{n+c_i}(1)|^2 + |u_x^{n+1}(-1) - u_x^{n+1}(1)|^2)\}, \tag{18.b}$$

$$c \times MSE_f = c \times \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2, \tag{18.c}$$

$$a, b \in Random(\alpha, \beta), \tag{18.d}$$

$$a + b + c = Num\{MSE_o, MSE_b, MSE_f\} = 3. \tag{18.e}$$

预测艾伦-卡恩方程的解 $u(t,x)$ 是我们的目标. 例如, 可以在 $t=0.1$ 时训练 DBPINN, 然后使

For Review Only

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

表 5 针对 KdV 方程, DBPINN 在无噪声和 1% 噪声的情况下, 拟合 λ_1 和 λ_2 得到的方程与 PINN 进行比较

Table 5 For the KdV equation, DBPINN is compared with PINN by fitting λ_1 and λ_2 to the equations obtained in the no-noise and 1% noise cases

Method	Noise	$u_t + uu_x + 0.0025u_{xxx} = 0$
PINN	No Noise	$u_t + 1.000uu_x + 0.0025002u_{xxx} = 0$
	1% Noise	$u_t + 0.999uu_x + 0.0024996u_{xxx} = 0$
DBPINN	No Noise	$u_t + 1.000uu_x + 0.0025003u_{xxx} = 0$
	1% Noise	$u_t + 1.000uu_x + 0.0025023u_{xxx} = 0$

用大小为 $\delta t = 0.8$ 的单一时间步长对 $t = 0.9$ 时的情况进行预测. 在本实验中, DBPINN 中的物理神经网络含有 4 个隐藏层, 每层 200 个神经元.

DBPINN 对艾伦-卡恩方程的拟合效果如图3所示, 其中图3的上方代表解 $u(t, x)$ 的分布, 图3下方左侧中的红色的 "x" 代表数据点的分布图3下方中的蓝色线条代表真实的解, 图3下方右侧中红色线条代表 DBPINN 的预测结果, 我们可以看出图3下方右侧的红色线条和蓝色线条高度重合, 这说明 DBPINN 在单一时间步长中能够很好地拟合真实值. 我们在相同条件下将 DBPINN 和 PINN 独立运行了十次, 以验证其稳定性和准确性, DBPINN 和 PINN 十次独立运行的具体结果如表4所示. 从表4中可以看出, DBPINN 十次运行的平均值为 4.27e-03, 而 PINN 十次运行的平均值为 1.20e-02, DBPINN 相比 PINN 误差降低了 64%. 此外, 通过表4观察发现 DBPINN 在十次独立运行得到的结果在稳定性上优于 PINN. 总的来说, DBPINN 相比 PINN 可以提高稳定性并且降低误差.

Korteweg-de Vries 方程. Korteweg-de Vries(KdV) 方程[37] 在波的数学模型中起着重要作用, 它可以看作是增加了一个色散项的 Burgers 方程. KdV 方程与熟知的物理问题有着若干联系, KdV 方程可以描述许多物理环境中一维长波的演变. 在此, 我们使用 KdV 方程进行实验, 进而验证 DBPINN 能够处理这种涉及高阶导数的非线性 PDEs, 并验证 DBPINN 在处理数据驱动发现问题上的有效性. 常规的 KdV 方程如公式 (19) 所示.

$$u_t + \lambda_1 uu_x + \lambda_2 u_{xxx} = 0. \tag{19}$$

而 KdV 方程的非线性算子可由公式 (20) 给出.

$$\mathcal{N}\left[u^{n+c_j}\right] = \lambda_1 u^{n+c_j} u_x^{n+c_j} - \lambda_2 u_{xxx}^{n+c_j}. \tag{20}$$

神经网络的共享参数以及 KdV 方程的参数 $\lambda = (\lambda_1, \lambda_2)$ 可以通过最小化损失函数 $MSE(\theta)$ 来学习.

如表5所示, DBPINN 在 1% 的噪声情况下能够完全拟合参数 λ_1, 这说明 DBPINN 的拟合效果突破了 PINN 拟合的瓶颈. 无论训练数据是否被噪声破坏, DBPINN 仍能够正确识别未知参数. 在无噪声训练数据的情况下, DBPINN 对 λ_1 和 λ_2 的拟合误差分别为 0% 和 0.00003%. 在有 1% 的噪声训练数据的情况下, DBPINN 对 λ_1 和 λ_2 的拟合误差分别为 0% 和 0.00023%. 此外, 我们分别将 DBPINN 和 PINN 独立求解 KdV 方程十次, 结果如表6所示, 我们可以看出 DBPINN 在参数 λ_1, λ_2 和 $\lambda_1(1\%)$ 上拟合的误差均值都低于 PINN, 并且在 PINN 的基础上分别了降低了 97.6%, 77.5% 和 88.3%, 进而体现了所提的 DBPINN 能够大幅度带来误差的降低. 多次的独立运行表明 DBPINN 具有很好的稳定性.

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

表 6 DBPINN 和 PINN 对 KdV 方程进行十次独立运行的结果

Table 6 Results of ten independent runs of DBPINN and PINN for the KdV equation

Model	Param.	0	1	2	3	4	5	6	7	8	9	Avg.
PINN	λ_1	2.77e-05	1.54e-03	3.05e-03	4.72e-03	7.45e-03	5.73e-03	6.10e-03	9.37e-03	6.54e-03	6.55e-03	5.15e-03
	λ_2	6.53e-05	2.41e-04	7.51e-04	8.73e-04	1.06e-03	1.14e-05	3.82e-05	5.61e-04	1.11e-03	9.54e-04	5.06e-04
	$\lambda_1(1\%)$	1.48e-03	3.13e-03	4.15e-03	7.59e-03	6.05e-03	5.94e-03	9.14e-03	6.67e-03	7.79e-03	8.67e-03	5.99e-03
	$\lambda_2(1\%)$	4.53e-04	5.53e-04	2.19e-04	5.40e-04	7.73e-05	6.99e-04	4.75e-04	6.01e-04	1.52e-03	2.02e-03	**8.18e-04**
DBPINN	λ_1	1.55e-04	2.07e-04	1.47e-04	6.58e-05	1.92e-04	7.33e-05	1.01e-04	9.79e-05	6.72e-05	1.32e-04	**1.24e-04**
	λ_2	1.39e-04	1.52e-04	1.78e-04	5.87e-05	1.62e-04	6.14e-05	1.13e-04	1.09e-04	7.25e-05	9.63e-05	**1.14e-04**
	$\lambda_1(1\%)$	7.80e-04	3.70e-04	2.17e-04	6.73e-04	5.77e-04	1.19e-03	1.76e-03	5.53e-04	2.26e-04	6.66e-04	**7.01e-04**
	$\lambda_2(1\%)$	3.45e-04	1.68e-03	3.93e-03	5.56e-03	3.17e-03	3.34e-03	2.42e-03	4.25e-03	4.96e-03	5.96e-03	3.56e-03

5 总结

由于 PINN 不同损失项梯度下降速度的不平衡, PINN 难以优化新加入正则化后形成的损失函数, 这导致其存在应用的局限和求解误差大的问题. 尽管现有方法在一定程度上可以缓解损失函数难以优化的问题, 但在实践中仍存在应用局限性和求解误差大的问题. 为此, 我们提出了 DBPINN, 以拓展应用范围并降低求解非线性 PDEs 的误差. 具体地, DBPINN 通过提出的动态权重系数和平衡求和方法, 使得 DBPINN 相比于 PINN 能够更好地优化收敛, 进而扩展了 DBPINN 的应用范围和显著降低了 DBPINN 求解非线性 PDEs 的误差. 为了验证 DBPINN 的有效性, 我们在科学机器学习领域选择了五个经典的非线性 PDEs 进行数值验证和分析. 实验表明 DBPINN 比 PINN 具有更好的稳定性和更低的误差. 此外, 我们对 DBPINN 进行了大量消融实验分析. 一方面, 我们验证了模型性能提升与网络架构的无关性、DBPINN 具有处理小数据量的能力以及 DBPINN 能够拟合不同时间状态下的方程. 另一方面, 我们验证了 DBPINN 的性能与动态权重系数取值上下界关联、受所有动态权重系数的和影响、与动态权重系数的初始值关联以及 DBPINN 相比于 PINN 具有更好的稳定性、准确率和收敛性. 我们希望 DBPINN 可以在求解非线性 PDEs 时成为一个误差更低的选择. 本文选用的实验方程难以覆盖海量实际应用场景, 在未来的工作中我们会将 DBPINN 应用于更多的实际应用中以验证其性能表现.

参考文献

1 Logan J D. An Introduction to Nonlinear Partial Differential Equations[M]. John Wiley & Sons, 2008

2 Cha W S, Li D L, Shen L H, et al. Review of neural network-based methods for solving partial differential equations. Acta Mech Sinica-Prc, 2022, 54(3): 543-556 [查文舒, 李道伦, 沈路航, 等. 基于神经网络的偏微分方程求解方法研究综述 [J]. 力学学报, 2022, 54(3): 543-556]

3 Godunov S K, Bohachevsky I. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics[J]. Sb Math+, 1959, 47(3): 271-306

4 Reddy J N. An Introduction to the Finite Element Method[J]. New York, 1993, 27: 14

5 LeCun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436-444

6 Zhang B, Zhu J, Su H. Toward the third generation of artificial intelligence. Sci Sin Inform, 50(9):1281–1302, 2020 [张钹, 朱军, 苏航. 迈向第三代人工智能 [J]. 中国科学: 信息科学, 2020, 50(9): 1281-1302]

7 Ribeiro M D, Rehman A, Ahmed S, et al. DeepCFD: Efficient steady-state laminar flow approximation with deep convolutional neural networks[J]. arXiv preprint arXiv:2004.08826, 2020

8 Cai S, Wang Z, Lu L, et al. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation

For Review Only

by neural networks[J]. J Comput Phys, 2021, 436: 110296
9 Raissi M, Perdikaris P, Karniadakis G E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. J Comput Phys, 2019, 378: 686-707
10 Karniadakis G E, Kevrekidis I G, Lu L, et al. Physics-informed machine learning[J]. Nat Rev Phys, 2021, 3(6): 422-440
11 Krishnapriyan A, Gholami A, Zhe S, et al. Characterizing possible failure modes in physics-informed neural networks[J]. Advances in Neural Information Processing Systems, 2021, 34: 26548-26560
12 Folland G B. Introduction to Partial Differential Equations[M]. Princeton University Press, 2020
13 Zhang Y, Fu H, Qin Y, et al. Physics-informed deep neural network for inhomogeneous magnetized plasma parameter inversion[J]. IEEE Antenn Wirel Pr, 2022, 21(4): 828-832
14 Kissas G, Yang Y, Hwuang E, et al. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks[J]. Comput Method Appl M, 2020, 358: 112623
15 Arnold F, King R. State–space modeling for control based on physics-informed neural networks[J]. Eng Appl Artif Intel, 2021, 101: 104195
16 Mao Z, Jagtap A D, Karniadakis G E. Physics-informed neural networks for high-speed flows[J]. Comput Method Appl M, 2020, 360: 112789
17 Bajaj C, McLennan L, Andeen T, et al. Recipes for when physics fails: recovering robust learning of physics informed neural networks[J]. Mach Learn-Sci Techn, 2023, 4(1): 015013
18 Wight C L, Zhao J. Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks[J]. arXiv preprint arXiv:2007.04542, 2020
19 Wang S, Teng Y, Perdikaris P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks[J]. SIAM J Sci Comput, 2021, 43(5): A3055-A3081
20 Wang S, Yu X, Perdikaris P. When and why PINNs fail to train: A neural tangent kernel perspective[J]. J Comput Phys, 2022, 449: 110768
21 Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]//1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. IEEE, 1997, 5: 4104-4108
22 Shi Y, Eberhart R. A modified particle swarm optimizer[C]//1998 IEEE international conference on evolutionary computation proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360). IEEE, 1998: 69-73
23 Mishra S, Molinaro R. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs[J]. IMA J Numer Anal, 2022, 42(2): 981-1022
24 Kemeth F P, Bertalan T, Thiem T, et al. Learning emergent partial differential equations in a learned emergent space[J]. Nat Commun, 2022, 13(1): 3318
25 Gao H, Sun L, Wang J X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain[J]. J Comput Phys, 2021, 428: 110079
26 Wang S, Wang H, Perdikaris P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks[J]. Comput Method Appl M, 2021, 384: 113938
27 Long Z, Lu Y, Ma X, et al. Pde-net: Learning pdes from data[C]//International Conference on Machine Learning. PMLR, 2018: 3208-3216
28 Long Z, Lu Y, Dong B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network[J]. J Comput Phys, 2019, 399: 108925
29 Qin T, Wu K, Xiu D. Data driven governing equations approximation using deep neural networks[J]. J Comput Phys, 2019, 395: 620-635
30 Baydin A G, Pearlmutter B A, Radul A A, et al. Automatic differentiation in machine learning: a survey[J]. J Mach Learn Res, 2018, 18(153): 1-43
31 Gardner M W, Dorling S R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences[J]. Atmos Environ, 1998, 32(14-15): 2627-2636
32 Wang Z, Bovik A C. Mean squared error: Love it or leave it? A new look at signal fidelity measures[J]. IEEE Signal Proc Mag, 2009, 26(1): 98-117
33 Dwivedi V, Parashar N, Srinivasan B. Distributed physics informed neural network for data-efficient solution to partial differential equations[J]. arXiv preprint arXiv:1907.08967, 2019
34 Tsutsumi Y. Schrodinger equation[J]. Funkcialaj Ekvacioj, 1987, 30: 115-125
35 Temam R. Navier-Stokes equations: theory and numerical analysis[M]. American Mathematical Soc., 2001
36 Feng X, Prohl A. Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows[J]. Numer Math,

https://engine.scichina.com/doi/10.1360/SSI-2023-0195

2003, 94: 33-65
37 Kato T. On the Korteweg-de Vries equation[J]. Manuscripta Math, 1979, 28(1): 89-99

A dynamic balanced physics-informed neural network for solving partial differential equations

Shuchao DENG[1†], Xiaotian SONG[1†], Minxiao ZHONG[1, 2], Qing LI[2], Yanan SUN[1*] & Jiancheng LV[1]

1. *College of Computer Science, Sichuan University, Chengdu* 610065*, China*;
2. *Nuclear Power Institute of China, Chengdu* 610213*, China*
* Corresponding author. E-mail: ysun@scu.edu.cn
† Equal contribution

Abstract In recent years, physics-informed neural networks (PINN) have been used extensively in solving nonlinear partial differential equations (PDEs). PINN add physical information as a regularization constraint to the neural network loss function, which can reduce the extensive reliance on data by traditional neural network methods. However, this approach makes PINN unable to dynamically adjust the individual residual weights in the loss function according to data changes during training, which leads to the limitation that PINN have large solution errors in solving nonlinear PDEs. In this paper, a Dynamic Balanced PINN (DBPINN) is proposed. Firstly, DBPINN designs a dynamic weight coefficient for each loss term of the PINN loss function, and uses a stochastic function to dynamically update the coefficient, which makes the PINN with only a single loss term converge better. Secondly, DBPINN establishes a balanced summation method for the loss terms of the PNNN loss function, which takes into account the competition among all the loss terms and leads to better convergence of the PINN as a whole. DBPINN makes PINN better optimized by dynamic weighting coefficients and balanced summation method, which solves the problem of large solution error of PINN in practical applications. In this paper, four classical nonlinear PDEs in the field of scientific machine learning are selected for numerical validation and analysis of DBPINN. The experimental results show that DBPINN reduce the error by 46% and 64% than PINN on the Schrodinger and Allen-Cahn equations, respectively. The DBPINN reduces the error of coefficient λ_1 by 1 to 2 orders of magnitude and the error of coefficient λ_2 by about 50% in solving the Navier-Stokes equation, respectively. The DBPINN can reduce the error by 1 order of magnitude in multiple coefficients on the KdV equation. Finally, the performance and parameter ablation are verified on various forms of Burgers and Allen-Cahn equations, and the results show that DBPINN not only improves the model performance, handles small amount of data, and fits equations in different time states, but also has better stability, accuracy, and convergence than PINN. DBPINN can be used instead of PINN for high accuracy solution of various nonlinear PDEs, and the source code is available at: https://github. com/Xiaojiuwo168/DBPINN/.

Keywords physics-informed neural networks, nonlinear partial differential equations, dynamic weight coefficients, balanced summation method, scientific machine learning

For Review Only

https://engine.scichina.com/doi/10.1360/SSI-2023-0195