arXiv:2304.03689v1 [physics.comp-ph] 7 Apr 2023

EPINN-NSE：用于求解纳维-斯托克斯方程的增强型物理信息神经网络*

阿尤布·法尔卡内 ‡, †，穆尼尔·戈格霍 †，穆斯塔法·乌达尼 †，和穆罕默德·布塔耶布 §, ‡

摘要。流体力学是工程和科学领域的基础学科。求解纳维-斯托克斯方程（NSE）对于理解流体的行为至关重要。然而，纳维-斯托克斯方程是一种复杂的偏微分方程，难以求解，而经典数值方法的计算成本可能很高。本文提出了一种使用物理信息神经网络（PINN）及多种改进其性能的新颖技术来求解纳维-斯托克斯方程的创新方法。第一个模型基于这样一种假设：通过采用流函数的导数来近似速度分量。该假设有助于简化系统，并保证速度满足无散方程。我们还开发了第二种更为灵活的模型，在不作任何假设的情况下近似求解方案。所提出的模型能够有效求解二维纳维-斯托克斯方程。此外，我们成功地将第二种模型应用于求解三维纳维-斯托克斯方程。结果表明，这些模型能够高效且准确地求解三维纳维-斯托克斯方程。这些方法具有多项优势，包括良好的可训练性、灵活性和效率。

关键词。纳维-斯托克斯方程、物理信息神经网络、深度学习、非线性偏微分方程、数值近似。

AMS主题分类。35Q35 ,65M99, 68T05

1.引言。流体流动在机械与化学工程、生物系统以及天体物理学中有着广泛的应用。从本质上说，所有运动的物体都与流体流动有关，例如车辆、飞机、卡车、火车、鸟类、海豚等。

纳维-斯托克斯方程是用于描述流体流动的偏微分方程。与许多偏微分方程一样，纳维-斯托克斯方程没有解析解，即使在三维情况下，它仍然是千年大奖难题之一。传统数值方法，例如有限差分法和有限元法（FEM），可以求得纳维-斯托克斯方程的近似解。然而，近年来，深度学习技术已广泛用于求解偏微分方程。其中一种方法是物理信息神经网络（PINN） [22, 19] ，它利用基于优化问题或残差损失函数的深度神经网络（DNN）来求解偏微分方程。文献中还提出了其他深度学习技术，例如深度 Galerkin 方法（DGM）[25] ，用于求解偏微分方程。DGM 对求解高维自由边界偏微分方程尤其有用，例如高维 Hamilton-Jacobi-Bellman 方程。然而，DGM 在其他类型的偏微分方程（椭圆型、双曲型和偏积分微分方程等）上的数值性能仍有待研究。为保证该神经网络收敛，人们对偏微分方程算子作出了不同假设。另一种利用包含深度神经网络的变分形式来近似求解偏微分方程的方法是深度 Ritz 方法（DRM） [27],，该方法将偏微分方程重构为具有变分表述的优化问题。DRM 具有天然的自适应性，对问题维数不太敏感，并且与随机梯度下降配合良好。尽管如此，仍有多种

*

†TICLab，拉巴特国际大学，拉巴特 11100，摩洛哥。 (ayoub.farkane@uir.ac.ma)，(mounir.ghogho@uir.ac.ma)， (mustapha.oudani@uir.ac.ma)。

‡CNRS-CRAN-7039，洛林大学，法国。（mohamed.boutayeb@univ-lorraine.fr）。§ INRIA-Nancy-LARSEN，法国。

此稿件仅供审阅。

DRM 存在一些缺点，例如凸性问题，以及局部极小值和鞍点问题。此外，DRM 对收敛速度没有一致的结论，且必要边界条件的处理比以往方法更加复杂。另外，网络结构和激活函数也存在一些值得关注的问题。

该研究 [5]通过对方程的扩散生成元施加条件，求解高维半线性偏微分方程。为此，该方法采用了 [16], 中建立的后向随机微分方程（BSDE）的解，以及 [2]中提出的时间离散化方案。该方法利用深度 BSDE 将偏微分方程与随机微分方程联系起来。然而，数值结果表明，该方法无法收敛到由全连接层组成的标准网络。其他研究则通过应用 Feynman–Kac 公式，基于偏微分方程与随机过程之间的联系展开 [10]。本文 [1]使用深度学习求解柯尔莫哥洛夫偏微分方程。Feynman-Kac 公式可扩展至完整的高阶线性抛物型方程类。

文献 [14] 提出了切比雪夫神经网络，用于求解二维（2D）线性偏微分方程。该方法采用单隐藏层前馈神经网络，并通过两个切比雪夫多项式的克罗内克积来近似解。然而，这种方法仍远不足以适用于非线性偏微分方程的求解。在一些相关论文 [20, 21], 中，偏微分方程的非线性项在时间域内进行了局部线性化。不过，这种方法仅适用于离散时间域，并且可能降低高度非线性系统的预测精度。另一种称为 MAgNet 的方法由文献 [3], 提出，该方法使用图神经网络，根据空间位置查询来预测空间连续的偏微分方程解。然而，MAgNet 使用一阶显式时间步进方案，可能会导致不稳定。

众所周知，求解纳维-斯托克斯方程等非线性动力学偏微分方程是一项艰巨的任务，尤其是在高维情形下。论文 [22] 提出了使用物理信息神经网络求解二维纳维–斯托克斯方程的方法。在本研究中，我们提出一种改进的、基于模型的神经网络来求解纳维-斯托克斯方程，并将其与物理信息神经网络 [22] 方法进行比较。我们还开展了多项实验，考察数据量和超参数变化的影响。最后，我们将这些增强模型用于求解三维（3D）纳维-斯托克斯方程，并采用一个测试解。

2. 控制方程。

2.1. 不可压缩流动的二维纳维–斯托克斯方程。纳维-斯托克斯方程可用于模拟多种流体流动 [6]，例如库埃特–泊肃叶流、贝尔特拉米流、管流和圆柱绕流，以及振荡平板流。考虑不可压缩流体二维纳维–斯托克斯方程的一般形式：

$$\frac{\partial u}{\partial t}+\beta\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t}+\beta\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right), \tag{2.2}$$

其中， u 是速度的 x 分量， v 是速度的 y 分量， t 是时间， p是流体压力， ν 是流体黏度[1] ， β 是一个固定常数。

[1]黏度 ν 表示为： $\nu=\frac{\rho u L}{R_e}$ ，其中 ρ 为流体密度， u 为流体速度，R_e 为雷诺数， L 为流体的长度或直径。

此稿件仅供审阅。

二维纳维–斯托克斯方程的解必须满足不可压缩性方程，该方程要求流体速度是一个无散函数：

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{2.3}$$

前一个方程描述了流体的质量守恒。

在研究流体动力学时，必须定义初始条件和边界条件，才能完整地确定一个边值问题。所采用的边界条件类型取决于所建模的具体流体现象。在某些情况下，可以采用狄利克雷边界条件，即指定解在边界处的值。在其他情况下，可以采用诺伊曼边界条件，即指定解在边界处的导数。也可以采用混合边界条件，将狄利克雷边界条件和诺伊曼边界条件结合起来。应根据所考虑的具体问题来选择适当的边界条件。

2.2. 三维纳维–斯托克斯方程。问题（2.1）-（2.3）可以在数学上扩展到三维，形式如下：

$$\frac{\partial u}{\partial t}+\beta\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \tag{2.4}$$

$$\frac{\partial v}{\partial t}+\beta\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\frac{\partial^2 v}{\partial z^2} \tag{2.5}$$

$$\frac{\partial w}{\partial t}+\beta\left(u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}\right)+\frac{\partial^2 w}{\partial z^2} \tag{2.6}$$

其中速度分量为 u， v和 w。其他参数的定义与前文相同。方程（2.6）给出了 w 速度分量的动量守恒方程。如前所述，可以在无散集合中求解：

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{2.7}$$

求解此类偏微分方程可能颇具挑战，但近期深度学习技术的进展已展现出解决这类问题的潜力。其中一种方法是物理信息神经网络方法，下一节将对此进行详细介绍。

3. 物理信息神经网络模型。物理信息神经网络是一种旨在求解偏微分方程的深度学习算法。它将神经网络的灵活性和表达能力与在模型中融入物理定律和约束条件的能力结合起来。

3.1. 方法论。物理信息神经网络方法由两个主要部分组成。第一部分涉及使用多层感知机（MLP）在预定义点生成一个候选解，该候选解称为“u”。第二部分包括计算与该候选解相关的偏微分方程项，并构建一个使用梯度下降算法进行优化的损失函数。下面我们将通过以下示例，更详细地考察这些方法的工作原理。考虑以下

本稿件仅供审阅。

一维偏微分方程：

$$\frac{\partial u}{\partial t}+\frac{\partial^2 u}{\partial x^2}=0,\quad \forall x\in\Omega,\,\forall t\in[0,T]. \tag{3.1}$$

$$u=0\ \text{ in }\Gamma_\Omega,\,\forall t\in[0,T] \tag{3.2}$$

$$u(t=0)=u_0,\quad \forall x\in\Omega \tag{3.3}$$

为了应用物理信息神经网络求解上述方程，神经网络将 x 和 t 作为输入，并生成候选 u 作为输出。通过最小化以下损失函数可以找到该解：

$$\mathrm{MSE}_u=\mathrm{MSE}_0+\mathrm{MSE}_b+\mathrm{MSE}_f, \tag{3.4}$$

其中：

$$\mathrm{MSE}_0=\frac{1}{N_0}\sum_{i=1}^{N_0}\left|u\left(0,x_0^i\right)-u_0^i\right|^2 \tag{3.5}$$

$$\mathrm{MSE}_b=\frac{1}{N_b}\sum_{i=1}^{N_b}\left(\left|u^i\left(t_b^i,.\right)-u_b^i\right|^2\right. \tag{3.6}$$

$$\mathrm{MSE}_f=\frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(t_f^i,x_f^i\right)\right|^2 \tag{3.7}$$

表示问题初始条件的数据点集合为 $\left\{x_0^i,u_0^i\right\}_{i=1}^{N_0}$，其中 N_0 是数据点的数量；边界上的配点由集合 $\left\{t_b^i\right\}_{i=1}^{N_b}$ 表示，其中 N_b 是点的数量；集合 $\{t_f^i,x_f^i\}_{i=1}^{N_f}$ 表示 f（t,x）上的配点。

均方误差 MSE_u 是计算预测值与实际值之间平方差平均值的一种度量。在当前语境中，MSE_0 表示专门与问题的初始条件相关联的 MSE 指标。类似地，MSE_b 是与边界条件相关联的MSE 指标，而 MSE_f 是一个确保解在定义域内满足方程的损失函数。函数 f 具体为：

$$f(t,x)=\frac{\partial u}{\partial t}+\frac{\partial^2 u}{\partial x^2} \tag{3.8}$$

这三种损失函数的组合对于获得当前问题的精确解至关重要。该方法可以扩展到不同类型的偏微分方程，但对于某些更复杂的问题，可能需要修改损失函数和神经网络架构。

3.2. 用物理信息神经网络求解二维纳维–斯托克斯方程。 一项值得关注的研究 [19] 采用物理信息神经网络方法求解二维纳维–斯托克斯方程。作者通过引入一个假设来简化不可压缩方程（2.3），即使用流函数 φ 来近似 u 和v 分量：

$$u=\frac{\partial\varphi}{\partial y},\quad v=-\frac{\partial\varphi}{\partial x} \tag{3.9}$$

流函数 φ 作为神经网络的输出进行计算，例如压力p。考虑满足以下条件的两个函数 f_1 和 f_2：

本稿件仅供审稿使用。

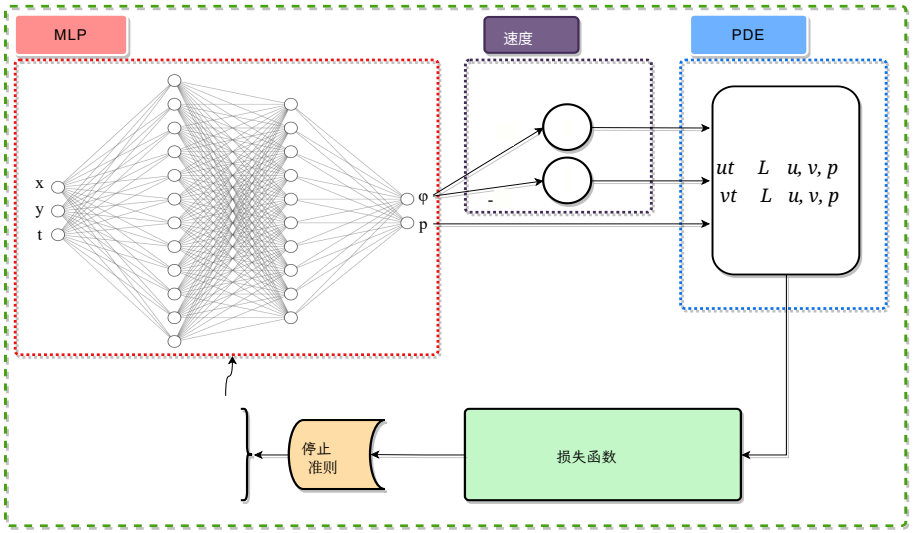

FIG. 1. *Method with the assumption for finding solutions to the 2D NSE using the PINN. The domain is represented by variables x and y, while t represents time. The stream function isrepresented by φ, and its partial derivatives with respect to x and y are represented by φ_x and φ_y, respectively. "Velocity" involves extracting a candidate solution velocity using the assumption. "MLP" is a multi-layer perceptron. "L_1, L_2" are the derivative operators associated with the equation.*

$$f_1(t,x,y) = \frac{\partial u}{\partial t} + \beta\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.10}$$

$$f_2(t,x,y) = \frac{\partial v}{\partial t} + \beta\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3.11}$$

使用先前定义的函数（3.10）和（3.11），使用以下损失函数训练模型:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\left|u\left(t^i,x^i,y^i\right) - u^i\right|^2 + \left|v\left(t^i,x^i,y^i\right) - v^i\right|^2\right) \tag{3.12}$$

$$+ \frac{1}{N}\sum_{i=1}^{N}\left(\left|f_1\left(t^i,x^i,y^i\right)\right|^2 + \left|f_2\left(t^i,x^i,y^i\right)\right|^2\right), \tag{3.13}$$

其中，$\left\{t^i,x^i,y^i,u^i,v^i\right\}$ 表示通过偏微分方程模拟或现实世界测量获得的准确数据。

该模型基于涉及不可压缩流体绕圆柱流动的实验数据进行训练。数据使用名为 Nektar++的数值求解器生成，该求解器将求解域离散为三角形单元，并将解近似为十阶分层半正交雅可比多项式展开的线性组合。本节所述方法用于求解本文中的二维纳维–斯托克斯方程 [22]。尽管物理信息神经网络方法在处理这一复杂问题方面已被证明具有优势，但当前研究正致力于优化模型并提升其有效性。下一节将全面概述增强模型，旨在改进用于求解纳维-斯托克斯方程的方法。

本稿件仅供审阅。

4.EPINN-NSE模型。物理信息神经网络方法基于深度学习技术，在求解各种偏微分方程方面展现出了良好的应用前景。然而，要将其应用于更复杂的问题，仍需开展进一步研究。文献中有大量论文致力于改进物理信息神经网络及其各种应用。物理信息神经网络面临的重大挑战之一是训练时长问题。对于简单情况，训练时长并不是关键问题；但对于复杂方程，这可能成为一个重大问题，训练可能需要数天而不是数小时。在此背景下， [24]有人提出了一种称为离散训练物理信息神经网络（DTPINNs）的技术，通过对偏微分方程进行离散化，并利用有限差分方法以高阶数值离散替代精确的空间导数，从而加速该方法。然而，在应用于涉及连续时间的问题之前，这种方法仍有很长的路要走。

物理信息神经网络方法缺乏确保模型收敛的理论基础，因此可能难以找到合适的解。例如，文献 [13] 指出，在某些偏微分方程问题中，物理信息神经网络无法找到准确解，例如反应扩散和对流问题。该论文讨论了这些网络可能失效的各种情形，例如网络无法准确捕捉所模拟系统的底层物理规律，或网络未得到恰当训练。此外，[13] 中的框架也为这些问题提供了潜在的解决方案，例如向网络中融入更多物理信息，或采用不同的训练方法。求解

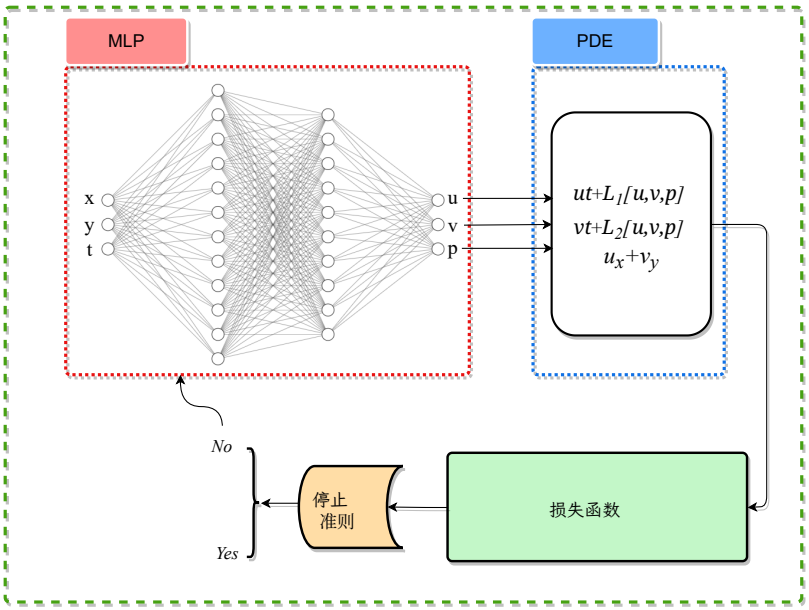

FIG. 2。 *Model of PINN without using any assumption for solving the 2D NSE.*

二维纳维–斯托克斯方程给物理信息神经网络带来了额外的挑战。例如， [22] 中介绍的模型需要较长的训练时长（超过 8 小时）。假设（3.9）基于流函数 φ 与速度分量之间的导数关系，分别为 u和 v。无需训练便学习压力这一因素，会增加模型生成不准确解的概率。与神经网络学习相关的另一个问题是，物理信息神经网络方法采用全

本稿件仅供审阅。

批量学习，这可能会增加陷入鞍点或局部极小值的概率。使用全批量学习来训练模型，在处理大型数据集时可能会面临挑战，从而对结果产生负面影响。

本文对先前提出的用于求解偏微分方程的物理信息神经网络模型进行了改进。具体而言，我们通过加入压力训练并采用先进的深度学习技术，增强了第 3.2 节所述的模型。这包括在方程（3.12）和（3.13）中引入压力损失函数：

$$\frac{1}{N}\sum_{i=1}^{N}\left(\left|p\left(t^i,x^i,y^i\right)-p^i\right|^2\right), \tag{4.1}$$

其中，$p\left(t^i,\ x^i,\ y^i\right)$ 表示压力预测值，p^i 表示从数据中获得的压力。加入压力训练旨在加快学习速度、提高学习精度，并改善压力近似结果。

除了这些改进之外，我们还开发了一种新模型，无需采用假设（3.9），即可直接近似速度分量和压力。取消这一基于流函数导数的假设，有助于缩短训练时间并提升整体效果。新模型使用以空间变量和时间变量为输入的神经网络，输出速度和压力的解。通过自动微分计算控制方程，从而确定网络将要最小化的损失函数各项。

Furthermore，我们的模型在神经网络架构方面进行了技术改进，并对超参数进行了优化。多项研究 [11, 9, 17] 强调了使用小批量梯度下降的优势。为了充分利用这些优势，我们在方法中采用了小批量学习。这使我们能够处理大型数据集并利用并行处理，从而实现更快的收敛并降低内存需求。我们采用了Adam优化器 [12], ，其对模型参数初始化不太敏感。与其他优化算法相比，Adam不易陷入局部极小值或鞍点。Adam根据每个参数过去的梯度调整其学习率，从而实现更快的收敛和更优的优化。为了有效训练模型并找出效果最佳的模型，我们使用验证损失函数在每个训练轮次对模型进行评估，并将结果与之前的损失值进行比较。随后，我们保存表现最佳的模型。此外，我们还引入了早停准则或“耐心值”。当模型经过预先设定次数的迭代后仍未表现出显著改进时，该技术会终止模型训练过程。

4.1. 求解二维纳维–斯托克斯方程。对于此案例，我们旨在使用这两个模型来求解问题。基于假设 (3.9) 的第一个模型如图1 所示，并加入了前文介绍的技术以及与压力相关的损失函数 (4.1)。在第二种方法中（图 2），神经网络生成一个候选解 u, v, 和 p。随后使用损失函数对该解进行评估。移除假设 (3.9) 需要添加与不可压缩性方程 (2.7) 相关的损失函数。训练神经网络以求得解，并通过最小化以下损失来学习函数 $f_1(t,x,y)$ 和 $f_2(t,x,y)$

function:

$$\text{MSE} = \text{MSE}_e + \text{MSE}_f + \text{MSE}_d \tag{4.2}$$

$$\text{MSE}_e = \frac{1}{N}\sum_{i=1}^{N}\left(\left|u\left(t^i,x^i,y^i\right)-u^i\right|^2+\left|v\left(t^i,x^i,y^i\right)-v^i\right|^2\right) + \frac{1}{N}\sum_{i=1}^{N}\left(\left|p\left(t^i,x^i,y^i\right)-p^i\right|^2\right) \tag{4.3}$$

$$\text{MSE}_f = \frac{1}{N}\sum_{i=1}^{N}\left(\left|f_1\left(t^i,x^i,y^i\right)\right|^2+\left|f_2\left(t^i,x^i,y^i\right)\right|^2\right) \tag{4.4}$$

$$\text{MSE}_d = \frac{1}{N}\sum_{i=1}^{N}\left(\left|u_x\left(t^i,x^i,y^i\right)+v_y\left(t^i,x^i,y^i\right)\right|^2\right), \tag{4.5}$$

其中， N是配点的数量，记为 $\left\{t^i,\ x^i,\ y^i\right\}$； $u(t^i,x^i,y^i)$、$v(t^i,x^i,y^i)$和 $p(t^i,x^i,y^i)$是正在评估的候选解。 u^i、 v^i和 p^i是通过偏微分方程求解器或真实世界观测获得的真实解。

方程（4.3）中指定的损失函数用于衡量精确解与近似解之间的差异。此外，方程（4.4）中定义的损失函数确保解符合该方程，而方程（4.5）中概述的损失函数表示无散方程。

4.2. 求解三维纳维–斯托克斯方程。三维纳维–斯托克斯方程是最难求解的偏微分方程之一。只有少数研究使用深度学习技术处理该方程，尤其是在三维问题中。在此，我们引用了一些使用有限元法（FEM）和与深度神经网络（DNN）相结合的有限体积法（FVM）等数值方法开展的研究 [15, 23] 。其他研究 [7, 26, 18] 则利用物理信息神经网络处理流体力学问题中纳维-斯托克斯方程的特定情形。在本节中，我们对 I_2 模型进行了调整，以近似求解三维问题。

考虑三维纳维–斯托克斯方程：

$$\frac{\partial u}{\partial t}+\beta\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)+\frac{\partial p}{\partial x}-\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)=h(t,x,y,z) \tag{4.6}$$

$$\frac{\partial v}{\partial t}+\beta\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)+\frac{\partial p}{\partial y}-\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)=g(t,x,y,z) \tag{4.7}$$

$$\frac{\partial w}{\partial t}+\beta\left(u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)+\frac{\partial p}{\partial z}-\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)=k(t,x,y,z), \tag{4.8}$$

其中 $(t,x,y,z)\in[0,T]\times\Omega$，满足 $\Omega\subset\mathbb{R}^3$（我们假设 $\Omega=[0,1]^3$、 $T=20$）。函数 h、 g和 k 表示施加于所研究系统的力分量，而其他变量的定义见第 2 节。所求解满足：

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=I(t,x,y,z) \tag{4.9}$$

该问题考虑齐次狄利克雷边界条件：

$$u(t,x,y,z)=v(t,x,y,z)=w(t,x,y,z)=0,\quad \forall(x,y,z)\in\Gamma_\Omega,\forall t\in[0,T], \tag{4.10}$$

本稿件仅供审阅之用。

其中， Γ_Ω 是区域 Ω 的边界。

对于三维纳维–斯托克斯方程，如图 3所示，神经网络的输入由空间变量 x、 y、 z 以及时间变量 t组成。神经网络的输出包括三个速度分量，分别表示为 u、 v和 w，以及压力 p。此外，损失函数的构建采用了与二维情况类似的方法。在这种情况下，损失函数表示为：

$$\text{MSE} = \text{MSE}_e + \text{MSE}_f + \text{MSE}_d \tag{4.11}$$

$$\begin{aligned}\text{MSE}_e = \frac{1}{N}\sum_{i=1}^{N}\Big(&\left|u\left(t^i,x^i,y^i,z^i\right)-u^i\right|^2+\left|v\left(t^i,x^i,y^i,z^i\right)-v^i\right|^2\\&+\left|w\left(t^i,x^i,y^i,z^i\right)-w^i\right|^2\Big)+\frac{1}{N}\sum_{i=1}^{N}\left(\left|p\left(t^i,x^i,y^i,z^i\right)-p^i\right|^2\right)\end{aligned} \tag{4.12}$$

$$\text{MSE}_f = \frac{1}{N}\sum_{i=1}^{N}\left(\left|f_1\left(t^i,x^i,y^i,z^i\right)\right|^2+\left|f_2\left(t^i,x^i,y^i,z^i\right)\right|^2+\left|f_3\left(t^i,x^i,y^i,z^i\right)\right|^2\right) \tag{4.13}$$

$$\begin{aligned}\text{MSE}_d = \frac{1}{N}\sum_{i=1}^{N}\Big(&\left|u_x(t^i,x^i,y^i,z^i)+v_y(t^i,x^i,y^i,z^i)+w_z(t^i,x^i,y^i,z^i)\right.\\&\left.-I(t^i,x^i,y^i,z^i)\right|^2\Big)\end{aligned} \tag{4.14}$$

神经网络生成的输出被标识为 $u\left(t^i,x^i,y^i,z^i\right)$, $v\left(t^i,x^i,y^i,z^i\right)$,$w\left(t^i,x^i,y^i,z^i\right)$, 以及 $p\left(t^i,x^i,y^i,z^i\right)$。另一方面， u^i、 v^i、 w^i和 p^i 表示从数据中获得的真实解。此外， f_1、 f_2 和 f_3 是定义如下的函数：

$$\begin{aligned}f_1(t,x,y,z) = \frac{\partial u}{\partial t}+\beta\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)+\frac{\partial p}{\partial x}-\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)\\-h(t,x,y,z)\end{aligned} \tag{4.15}$$

$$\begin{aligned}f_2(t,x,y,z) = \frac{\partial v}{\partial t}+\beta\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)+\frac{\partial p}{\partial y}-\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)\\-g(t,x,y,z)\end{aligned} \tag{4.16}$$

$$\begin{aligned}f_3(t,x,y,z) = \frac{\partial w}{\partial t}+\beta\left(u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)+\frac{\partial p}{\partial z}-\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)\\-k(t,x,y,z)\end{aligned} \tag{4.17}$$

用于训练和评估模型性能的数据集由

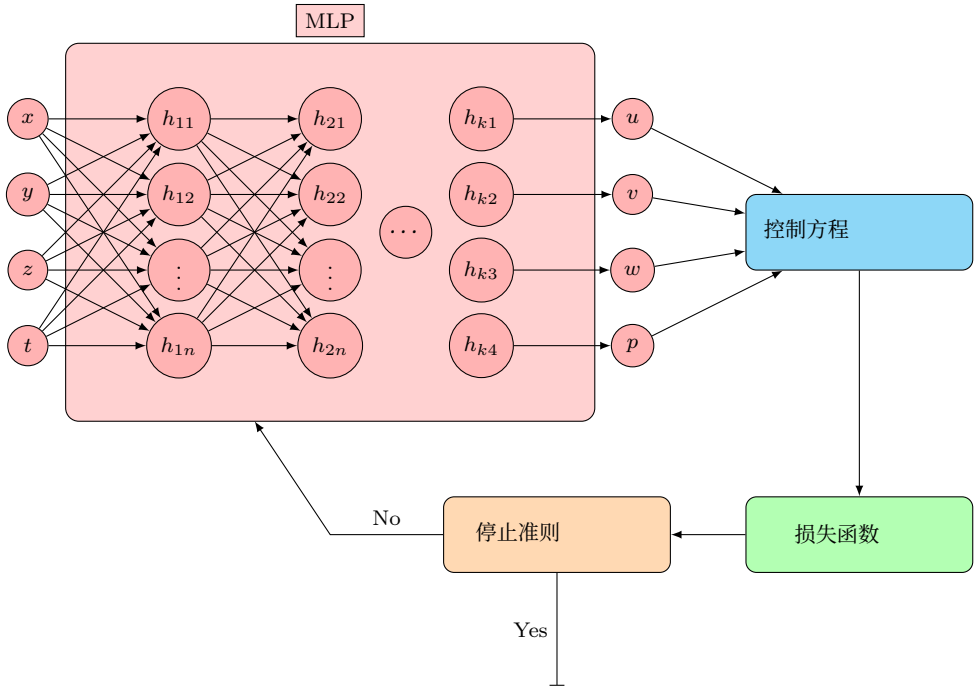

FIG. 3 *Approach to solve 3D NSE using the I_2 model. n represents the number of neurons per layer and k represents the number of hidden layers.*

一个测试解生成，该解由下式给出：

$$u := (t, x, y, z) \mapsto \mathrm{e}^{-t} \sin(\pi x) \sin(\pi y) \sin(\pi z) \tag{4.18}$$

$$v := (t, x, y, z) \mapsto \mathrm{e}^{-t} \left(x^2 - x\right) \left(y^2 - y\right) \left(z^2 - z\right) \tag{4.19}$$

$$w := (t, x, y, z) \mapsto \mathrm{e}^{-t} \sin(\pi x) \sin(\pi y) \left(z^2 - z\right) \tag{4.20}$$

$$p := (t, x, y, z) \mapsto \mathrm{e}^{-t} xyz \tag{4.21}$$

5. 实验结果与分析。本节评估改进模型在二维和三维空间中求解纳维-斯托克斯方程的性能。

5.1. 二维问题。

5.1.1.数据集。 在本研究中，用于训练和验证模型的数据来自对圆柱体周围不可压缩流体流动进行的计算流体动力学模拟。该模拟使用 Nektar++[4] 偏微分方程求解器完成，具体描述见论文 [22]。模拟在矩形区域 $\Omega = [-15, 25] \times [-8, 8]$中进行，左边界施加恒定速度，右边界假设压力为零，上下边界施加周期性条件。该模拟基于雷诺数，该数计算为自由来流速度、圆柱直径和运动黏度之比（$R = \frac{U_\infty L}{\nu}$，假设为 $U_\infty = 1$，$L = 1$，$\nu = 0.01$）。数据集中有 5000 个点，每个点代表 200 个时刻，并包含速度分量 u 和 v以及压力 p的值。数据以 $\{t^i, x^i, y^i, u^i, v^i, p^i\}$ 值的集合形式呈现。

本文稿仅供审阅。

5.1.2. 实现细节。 我们进行了多项实验[2] 以评估模型性能。这些实验通过考察损失函数、验证损失函数、预测解的相对误差，以及学习和预测解所需的训练时长，分析影响模型能力的各种因素。这些实验的目标是从全局数据集中提取数据子集。

	I_0	I_2
软件库	TF1.x	PyTorch
数据集	1000000	8334
训练数据	5000	5000
验证数据	-	1667
迭代次数-训练轮数	200000	**2000**
批量学习	全批次	**小批次**
批次大小	5000	64
训练损失	2.825×10^{-1}	$\mathbf{1.2}\times10^{-2}$
验证损失	-	**6.03e-5**
训练时长	>8h	**1小时40分钟**
GPU	Yes	Yes

TABLE 1
The initial training comparison of I_0 and I_2.

我们采用以下符号表示：

- I_0：表示在 [22] 中建立的物理信息神经网络初始实现。
- I_1：表示基于假设（3.9）开发的模型对应的实现。
- I_2：表示未使用任何假设的模型实现。

首先，我们比较了所提出模型之一的训练性能 ls 采用 [22] 中所述的方法。

表 1 展示了初始模型 I_0（在论文[22], 中介绍）与本文提出的模型之一 I_2之间的比较。模型 I_2 使用 PyTorch 训练，而另一个模型使用 TensorFlow $1.x$。两个模型均使用相同的 GPU 进行训练。在训练中， I_0 从全局数据集中随机选择了 5000 个点。为了与 I_2进行比较，我们从全局数据中选择 8334 个点，并取其中的 60%（$\simeq$ 5000）用于训练，取 20%（$\simeq$ 1667）用于验证。如表 1所示， I_2 仅使用 2000 个训练轮次就优于 I_0 ，且训练损失更低、训练时间更短； I_0 则需要 200,000 次迭代。

使用完整批次训练模型 I_0 可能会给开展额外实验带来挑战。虽然可以增加更多数据以延长训练时间，但这也会提高优化问题陷入鞍点或局部极小值的风险。为了解决这一问题，我们决定采用小批量学习、验证损失和提前停止准则来实现相同的模型，以缓解这些顾虑。此外，我们进一步改进了该模型，并将其称为模型 I_1，方法是将压力训练纳入其中。

表 2 展示了两个模型之间的通用参数 I_1 和 I_2。这些参数包括神经网络的超参数和优化参数。需要注意的是，两个模型均使用

[2]这些实现使用 Tesla T4 GPU 执行。

本稿件仅供审阅之用。

参数	详情
全局数据量	5000*200
隐藏层	8
每层神经元数	20
激活函数	Tanh
优化器	Adam
初始学习率	0.001
提前停止准则	100
训练数据量	60%
验证数据量	20%
测试数据量	20%
GPU	Yes

TABLE 2
Parameters shared in both models I_1 and I_2.

实验	E_1		E_2		E_3	
模型	I_1	I_2	I_1	I_2	I_1	I_2
Data size	9000		10000		20000	
提前停止标准	100					
批次 size	256					
训练 Loss	1.62e-3	1.65e-3	1.06e-3	1.53e-3	6.98e-4	1.38e-3
验证 loss	1.66e-3	1.86e-3	1.33e-3	1.78e-3	6.81e-4	1.18e-3
RE_u	2.19e-2	2.26e-2	2.03e-2	2.44e-2	1.44e-2	1.84e-2
RE_v	7.07e-2	6.14e-2	5.59e-2	6.45e-2	3.99e-2	5.12e-2
RE_p	7.30e-2	8.15e-2	7.30e-2	8.26e-2	4.81e-2	5.87e-2
训练持续时间	1小时15分钟	28分钟	1小时45分钟	32分钟	2小时16分钟	46分钟

TABLE 3
Numerical results of experiments E_1 E_2 and E_3 for solving the 2D NSE using models $I_1$and , I_2. Both models are executed with the same early stopping criteria and batch size.

PyTorch 软件库从头构建。

为了评估模型的精度，我们计算预测解与实际解之间的相对误差（RE）。相对误差（RE）使用以下公式计算：

$$RE_u = \frac{||predicted \;\; value(u) - true \;\; value(u)||}{||true \;\; value(u)||} \tag{5.1}$$

为使用更大的数据集开展更多实验，我们按照表 4 所示，增大了两种实现的批次大小，以避免训练时间过长。我们还计划在表 5 和 6 中开展进一步实验，以评估神经元数量和层数等主要超参数对模型的影响。训练和评估模型时，我们使用了 256 的批次大小，并采用

此稿件仅供审阅。

实验	E_4		E_5		E_6	
模型	I_1	I_2	I_1	I_2	I_1	I_2
Data size	30000		40000		60000	
提前停止标准	100					
批次 size	500					
训练 Loss	7.26e-4	7.11e-4	4.02e-4	7.33e-4	6.62e-4	7.00e-4
验证 loss	6.85e-4	6.82e-4	3.69e-4	6.41e-4	5.44e-4	6.09e-4
RE_u	1.37e-2	1.30e-2	9.20e-3	1.30e-2	1.19e-2	1.30e-2
RE_v	3.93e-2	3.82e-2	2.85e-2	3.61e-2	3.66e-2	3.48e-2
RE_p	4.53e-2	4.30e-2	4.29e-2	3.99e-2	4.23e-2	4.10e-2
训练持续时间	2小时30分钟	1小时12分钟	3小时30分钟	1小时40分钟	2小时55分钟	1小时30分钟

TABLE 4

Numerical outcomes obtained from experiments E_4 E_5 and E_6 after augmenting the data size , , and elevating the batch size.

实验	E_7		E_8		E_9	
模型	I_1	I_2	I_1	I_2	I_1	I_2
隐藏层	10					
每层神经元数层	10		20		30	
训练损失	3.05e-3	5.21e-3	9.60e-4	9.04e-4	2.96e-4	2.67e-4
验证损失	2.93e-3	5.00e-3	8.95e-4	9.00e-4	3.19e-4	2.62e-4
RE_u	2.82e-2	4.36e-2	1.65e-2	1.56e-2	8.91e-3	7.11e-3
RE_v	9.72e-2	1.14e-1	4.54e-2	4.37e-2	2.88e-2	2.34e-2
RE_p	1.19e-1	1.41e-1	5.60e-2	4.84e-2	3.08e-2	2.78e-2
训练持续时间	4小时40分钟	1小时58分钟	4小时40分钟	1小时58分钟	2小时28分钟	1小时20分钟

TABLE 5

Experiments with fixing the number of hidden layers and changing the number of neurons per layer.

100 个训练轮次的早停准则作为耐心值，并选取 20,000 个配点。根据表1、3、45和6中给出的数值结果，显而易见，所提出的模型 I_1 和 I_2 在求解二维纳维–斯托克斯方程方面优于其他模型。尽管通过整合深度学习技术， I_1 模型得到了显著改进，但训练过程仍然耗时较长。相比之下， I_2 模型展现出与 I_1 模型相媲美并取得良好结果的能力，同时其训练时间最多仅为 I_1的一半。此外，结果表明，调整架构和超参数会对最终结果产生明显影响。事实上，随着神经元数量和隐藏层数的增加，与模型 I_2 相关联的预测解收敛程度更高。

此稿件仅供审阅。

实验	E_{10}		E_{11}		E_{12}	
模型	I_1	I_2	I_1	I_2	I_1	I_2
隐藏层	12		15		18	
每层神经元数量 层	20					
训练损失	1.12e-3	1.49e-3	1.16e-3	8.86e-4	1.27e-3	9.40e-4
验证损失	1.07e-3	1.38e-3	1.10e-3	8.50e-4	1.11e-3	8.69e-4
RE_u	1.75e-2	2.05e-2	1.86e-2	1.54e-2	1.76e-2	1.68e-2
RE_v	5.11e-2	5.42e-2	5.21e-2	4.25e-2	5.59e-2	4.23e-2
RE_p	6.47e-2	5.85e-2	6.18-e2	5.08e-2	5.90e-2	4.96e-2
训练 时长	2小时20分钟	1h	3小时15分钟	1小时43分钟	4小时10分钟	2h

TABLE 6

Outcomes of fixing the number of neurons per layer and changing the number of layers.

在下一节中，我们尝试使用模型 I_2 求解三维区域内的非定常纳维–斯托克斯方程。我们选择这种方法，是因为它被认为能够有效应对当前问题的复杂性。

5.2. 三维问题。

5.2.1.数据集。 为了构建三维问题的模型，需要一个包含变量 $\{x, y, z, t, u, v, w, p\}$ 的数据集。在本研究中，使用一种名为 Gmsh 的流行且免费可用的网格生成器，在空间域内生成了总计 7318 个配点 [8]。时间区间被离散为 200 个时刻。与这些点相关联的解通过方程（4.18）-（4.21）计算得到。

随后，将所得数据集重新整理为 $\left\{x^i, y^i, z^i, t^i, u^i, v^i, w^i, p^i\right\}$，适用于所有 $1 \leq i \leq$（ 7318×200），从而能够对模型的结果进行更深入的分析和解读。

5.2.2. 数值结果。由于问题的复杂性，本实验使用了相当大的数据集[3]。图4展示了训练损失和验证损失随数据量变化的情况。根据已进行的实验，我们成功使用了整个数据集的 35% 来运行模型，该数据集包含超过一百万个样本。值得注意的是，使用 500,000 个样本，模型仅需 3 小时 30 分钟即可完成训练，展现出出色的效率。所有实验均采用表2 中列出的相同参数，唯一不同的是本例中的全局数据量为 7318×200。

表中展示的结果 7 采用训练轮次数作为停止标准，并选择 2000 训练轮次作为训练模型的最优值。我们认为这个训练轮次数足够大，能够为模型优化及探索其性能提供充足时间。通过考察图中所示的结果5，我们可以得出结论：随着数据量增加，相对误差呈下降趋势。这一观察结果符合预期，即数据量增加时，相对误差会降低，因为更大的数据集能够提供更多信息，从而实现更精确的计算。

[3]实现均使用 Tesla A40 GPU 运行。

本稿件仅供评审使用。

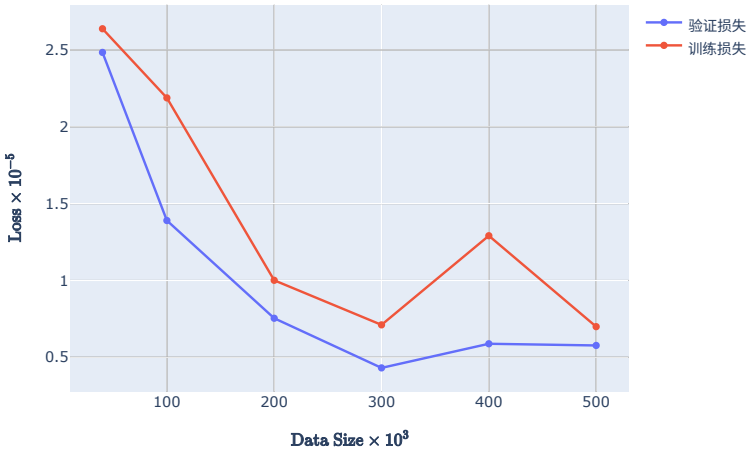

FIG. 4. *Effects of varying dataset size on training and validation loss.*

实验	E_1	E_2	E_3	E_4	E_5	E_6
数据量	40000	100000	200000	300000	400000	500000
损失函数	2.13e-5	7.23e-6	3.46e-6	2.56e-5	1.49e-5	5.37e-6
验证损失	1.99e-5	5.84e-6	3.23e-6	2.75e-6	1.98e-6	2.31e-6
RE_u	2.13e-2	1.20e-2	1.06e-2	1.13e-2	6.97e-3	5.69e-3
RE_v	3.89e-1	2.50e-1	4.56e-1	1.94e-1	9.65e-2	8.19e-2
RE_w	6.23e-2	2.25e-2	2.42e-2	2.48e-2	2.37e-2	1.35e-2
RE_p	1.58e-2	8.47e-3	4.24e-3	4.98e-3	3.99e-3	2.91e-3
训练时长	1小时5分钟	2小时36分钟	5小时27分钟	8h	11h	13小时41分钟

TABLE 7

The outcomes obtained through the implementation of the I_2 model for solving the 3D NSE, wherein the number of epochs was employed as a stopping criterion.

6. 结论与展望。纳维-斯托克斯方程作为偏微分方程，具有动态且非线性的特性，求解难度极高。近期研究表明，物理信息神经网络提供了一种颇具前景的解决方案。我们的研究证明，基于物理信息神经网络改进的模型在求解二维和三维情况下的纳维-斯托克斯方程时效果显著。我们表明，与原始物理信息神经网络方法相比，这些模型具有显著优势，包括更高的计算效率、更稳定的收敛性以及更低的相对误差。本文采用的方法是在三维问题中利用测试解进行训练，并评估旨在求解复杂偏微分方程的模型的效能。这种方法可作为开展真实实验或使用偏微分方程求解器的可行替代方案，后两者的成本可能更高。这些发现表明，进一步改进该方法有望为其应用于三维实验的真实世界数据开辟新的机遇。总的来说，

本稿件仅供审阅。

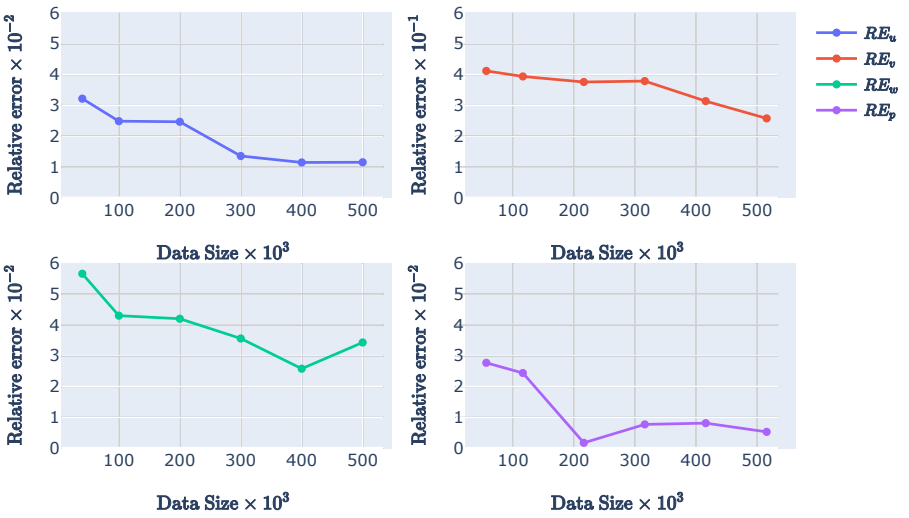

FIG. 5. *Relative errors of velocity and pressure components in various experiments of 3D NSE solved using the I_2 model.*

我们的工作凸显了物理信息神经网络作为解决复杂物理建模问题的强大工具的潜力，并为这一领域令人期待的未来发展铺平了道路。

除了本研究取得的可喜成果之外，未来研究还有若干令人振奋的方向。一种可能是探索物理信息神经网络在求解更复杂问题方面的潜力，例如涉及多相流或湍流的问题。另一个研究方向是将本文提出的方法扩展到其他类型的偏微分方程，包括具有非线性边界条件或时变系数的偏微分方程。此外，物理信息神经网络还可应用于物理和工程领域的其他方向，例如固体力学、电磁学和量子力学等。

另一个值得考虑的视角是将物理信息神经网络与其他机器学习技术（例如深度学习或强化学习）相结合的潜力，从而进一步增强其能力。将物理信息神经网络与其他方法结合，可能有助于更高效、更准确地解决复杂的物理问题，同时还能深入了解其背后的物理现象。

最后，必须认识到，在科学和工程的各个领域中求解偏微分方程可能产生的影响。准确地对复杂物理现象建立模型并进行预测，对于新材料的设计、制造工艺的优化以及新技术的开发具有重大意义。因此，持续开发和完善偏微分方程求解器，有望改变我们在各个领域开展建模与仿真的方式，从而带来令人振奋的新发现和进步。

参考文献

[1] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, *Solving the kolmogorov pde by means of deep learning* 88 2021 1 28 ,《科学计算杂志》, ()，页码 – 。[2] B. Bouchard and N. Touzi, *Discrete-time approximation and monte-carlo simulation of backward stochastic differential equations*，《随机过程及其应用》，111 (2004)，页码 175–206。[3] O. Boussif, D. Assouline, L. Benabbou, and Y. Bengio, *Magnet: Mesh agnostic neural pde solver* 2210 05495, 2022 ，arXiv预印本 arXiv: 。 ( )。[4] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia S. Yakovlev J.-E. Lombard D. Ekelschot et al. *Nektar++: , , , „ An open-source spectral/hp element framework*，《计算物理通讯》，192 (2015)，页码 205–219。[5] Q. Chan-Wai-Nam, J. Mikael, and X. Warin, *Machine learning for semi linear pdes*，《科学计算杂志》，79 (2019)，页码 1667–1712。[6] P. G. Drazin and N. Riley, *The Navier-Stokes equations: a classification of flows and exact solutions* 334, 2006 ，编号。剑桥大学出版社, 。[7] H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, *Physics-informed neural networks for solving reynolds-averaged navier–stokes equations* 34 2022 075117 ，《流体物理学》, (), 页码。[8] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities*，《国际工程数值方法杂志》, 79 (2009), 页码 1309–1331。[9] G. Hinton, N. Srivastava, and K. Swersky, *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent* 14 2012 2 , 引用于,(), 页码 。[10] M. Kac, *On distributions of certain wiener functionals*，《美国数学学会汇刊》, 65 (1949), 页码 1–13。[11] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, *Mini-batch gradient descent: Faster convergence under data sparsity*, 载于 2017 IEEE 第56届决策与控制年度会议（CDC）, IEEE, 2017, 页码2880–2887。[12] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*，arXiv 预印本 arXiv:1412.6980, (2014)。[13] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, *Characterizingpossible failure modes in physics-informed neural networks*，神经信息处理系统进展, 34(2021), 页码 26548–26560。[14] M. Liu, M. Hou, J. Wang, and Y. Cheng, *Solving two-dimensional linear partial differential equations based on chebyshev neural network with extreme learning machine algorithm*，《工程计算》, (2020)。[15] N. Margenberg, D. Hartmann, C. Lessig, and T. Richter, *A neural network multigrid solver for the navier-stokes equations*，《计算物理学杂志》, 460 (2022), 页码110983。[16] E. Pardoux and S. Peng, *Adapted solution of a backward stochastic differential equation*，《系统与控制通讯》, 14 (1990)，页码 55–61。[17] Y. Qian and D. Klabjan, *The impact of the mini-batch size on the variance of gradients in stochastic gradient descent* 2004 13146, 2020 ，arXiv预印本 arXiv: 。 ( )。[18] R. Qiu, R. Huang, Y. Xiao, J. Wang, Z. Zhang, J. Yue, Z. Zeng, and Y. Wang, *Physics-informed neural networks for phase-field method in two-phase flow*，《流体物理学》, 34 (2022), 页码052109。[19] M. Raissi, *Deep hidden physics models: Deep learning of nonlinear partial differential equations* 19 2018 932 955 ，《机器学习研究杂志》, (), 页码 – 。[20] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations* 357 2018 125 141 ，《计算物理学杂志》, ( ), 页码 – 。[21] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Numerical gaussian processes for time-dependent and nonlinear partial differential equations*，《SIAM科学计算杂志》, 40 (2018), 页码 A172–A198。[22] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations* 378 2019 686 707 ，《计算物理学杂志》, ( ), 页码 – 。[23] R. Ranade, C. Hill, and J. Pathak, *Discretizationnet: A machine-learning based solver for navier–stokes equations using finite volume discretization* ，《计算方法与应用力学及工程》，378 (2021)，页码113722。[24] R. Sharma and V. Shankar, *Accelerated training of physics informed neural networks (pinns) using meshless discretizations* 2205 09332, 2022 , arXiv 预印本 arXiv: 。 ( )。

此稿件仅供审稿使用。

[25] J. Sirignano and K. Spiliopoulos, *Dgm: A deep learning algorithm for solving partial differential equations* 375 2018 1339 1364 ,《计算物理学杂志》, （）, 第 – 页。[26] H. Wang, Y. Liu, and S. Wang, *Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network*，《流体物理学》，34（2022），第 017116 页。[27] B. Yu et al.,*The deep ritz method: a deep learning-based numerical algorithm for solving variational problems* 6 2018 1 12 ,《数学与统计通讯》, （）, 第 – 页。

此稿件仅供审阅。