Solving Oscillator ODEs via Soft-constrained Physics-informed Neural Network with Small Data

卢凯良 [†]，苏雨蒙 [†]，毕卓 [⋆]

[⋆]xxx@163.com, 上海建桥学院信息技术学院

[1] [2]

Abstract

本文通过文献调研在解微分方程（ODE/PDE）上对比了 PINN 和常规神经网络方法、数值离散化方法。PINN 继承了神经网络的强表达能力，通过硬编码或软约束的方式将物理信息和知识融入神经网络拓扑结构或学习架构以及计算流程中，从而减少了数据冗余、改进了泛化性能，进而提高了计算效率、增强了可解释性和鲁棒性。另一方面，相较于数值离散化方法（例如龙格库塔法、有限元法），PINN 具有 1）mesh-free，2）可无缝合并实验数据且容忍噪声，3）有效和高效评估新数据点以及 4）可打破维度诅咒、适用于数值离散化方法不擅解决或求解代价昂贵的高维、非线性和非光滑以及逆问题，等优势。

我们归纳了软约束 PINN 方法求解微分方程（ODE/PDE）的一般架构和计算流程，并通过求解多种典型谐振 ODEs 从实验上验证了它的工作机制及其精度和效率。基于 DeepXDE 实现 PINN 方法，不仅轻代码、训练效率也高，并且跨平台灵活度高。PINN 大大减少了对标注数据的需要：当问题的非线性较弱时，非常少量的标注训练数据加上少量的配置点数据就足以预测解；极简情况下，对于一阶和两阶 ODE 仅分别需要（含初始值的）一个和两个训练点。强非线性问题也仅需要适当增加训练点和配置点，相比常规 NN 仍有显著优势。通过配置点的辅助和利用物理信息，PINN 具备了预测训练集所涵盖时域外数据的外推能力，而且对噪声数据鲁棒性良好，泛化能力获得增强。当伴随着数据量的减小获得的收益大于损失函数项增加引起的拖累时，训练速度是加快的。软约束 PINN 方法通过在总损失函数中加正则化项即可方便地施加

[1] [†] 为共同一作。(The first two authors contributed equally to this work.)

[2] {备选标题: Solving {Non-linear} Oscillator ODE via Physics-informed Neural Network(PINN) {Constrained by Energy Conservation on DeepXDE} with {Noisy} Small Data}

物理定律（例如能量守恒）约束，从而改善遵守该物理定律的 ODE 的求解性能。

不仅如此，PINN 还可用于刚性 ODE、PDE、IDE、PDE、SDE 等各类微分方程，正在成为数字双胞胎（Digital Twins）时代的有利催化剂。

Keywords: PINN; Oscillator ODE; DeepXDE; Soft-constrained; Minimalistic Data; Noise; Energy Conservation Regularization

1 Introduction

1.1 Why PINN: Where Neural Networks can be Improved

凭借爆炸式增长的可用数据和计算资源，深度学习已在图像识别、下棋等多个任务中取得了超越人类的表现 [LBH15, SHS+18]。其成功首先仰赖于大数据，却也存在过拟合和泛化性能差的问题 [BG21] 以及可理解性问题 [CYY+22]。另一方面，在现实的复杂科学和工程实践中（例如物理、生化、医学等），数据通常通过昂贵的实验或大规模仿真计算产生，数据往往是稀疏的且带有噪声；有时获取数据的成本高得令人望而却步，甚至根本无法获得大量实验数据 [KKL+21, YLRK20, CLS21]。在这种小数据环境下，大多数神经网络方法（包括卷积/递归神经网络）都缺乏鲁棒性，无法提供收敛保证 [RPK19]，因此我们不可避免地要面对在部分信息下预测或决策的挑战。好的消息是，在这些系统建模案例中，存在着大量的先验知识（即前人智慧的结晶），而这些信息在传统的神经网络中尚未得到充分利用；深度神经网络由于表达能力强，从而忽略了利用先验信息和知识，却也导致了过于强调和依赖数据以及算力。

Raissi 等人 [RPK19] 提出了在神经网络的学习过程中利用物理信息或先验知识的想法，即物理信息神经网络（Physics-informed/inspired Neural Network, PINN），这是一类新的通用函数近似器，继承了神经网络的强表达能力，能够编码支配给定数据集的任何基本物理定律（这经常用微分方程来描述），即是利用我们对世界的观察、经验、物理或数学理解所产生的先验知识来提高学习算法性能（包括减少数据冗余、改进泛化性能、提高计算效率、增加可解释性和鲁棒性等）的过程。

PINN 在结构化、模块化的神经网络学习架构中，将物理信息和知识融入网络拓扑结构或计算流程作为模型先验，具体实现主要分为硬约束 (Hard Constraints) [LPY+21] 和软约束 (Soft Constraints) 两种。硬约束一般通过将物理知识或定律（如微分方程、对称性、守恒律等）[WJX+22] 以及边界、初始条件直接嵌入作为神经网络架构或求解过程的一部分来确保它们被严格遵守，这需要通过特定的网络设计或训练策略来近似实现。本文所讨论的软约束则是通过将物理残差项或正则化项加入损失函数来间接实现的，较易实现，这种方法可被视为多任务学习的一种特定用例。软约束允许在满足物理定律或条件时有一定的灵活性或容忍度，但需要通过仔细调整损失函数中的各项权重来平衡预测性能和物理准确性。

将这些先验作为物理残差项或正则化项，可使所求解问题的可接受解的解空间限制在可管理的范围内（例如，在不可压缩流体动力学问题中，可以摒弃任何违反质量守恒原

理的非现实流动解 [KKL+21])。作为回报，将这种结构化信息编码到学习算法中，可以放大算法所看到的数据的信息含量，使其能够快速引导自己找到正确的解决方案，并在只有少量（监督的或标注）训练示例的情况下也能很好地泛化——这是 PINN 的关键特性。更进一步地，这些先验知识或约束条件可以产生更多可解释的学习方法，这些方法在数据不完善（如缺失或噪声值、异常值等）的情况下仍能保持稳健，并能提供准确且物理上一致的预测，甚至能用于外推任务。

1.2 PINN as a Complement or Alternative to Numerical Discretization

常微分方程（ODE）与偏微分方程（PDE）是表达科学和工程领域自然规律的重要工具，现实世界的物理系统可以使用基于不同领域特定的假设和简化的微分方程来建模，这些模型可以用来近似这些系统的行为 [GKLS24]，前者精准刻画一维动态，后者处理多维复杂变化，已成为构建自然规律的数学基础，推动了对自然现象及其蕴含的科学问题的深入理解和预测。解常/偏微分方程的常见代表性数值离散化求解方法有龙格库塔 (Runge-Kutta) 方法和有限元方法 (FEM)。从可理解性的角度来看，由于底层近似的简单性，这些方法都得到了很好的理解：具有现成的误差估计以及收敛性和稳定性保证。

通过数值离散化求解微分方程（ODE/PDE）以模拟各类场问题方面取得了巨大进步，理论基础完备，具有高效率、高精度、稳定性好的优势，但仍有一些严重限制其应用的瓶颈问题——包括维度灾难问题，网格生成仍然很复杂，在合并实验数据方面存在困难、仍然无法将噪声数据无缝纳入现有算法，而且无法解决由参数化 PDE 控制的高维问题等等 [KKL+21]。比如，有限元方法的一个明显的缺点是，它通常依赖于空间离散（空间网格或大多项式基），会遭遇维度诅咒问题：在三维空间，已经较难使用，遑论更高维问题；而且，某些非线性和非光滑的偏微分方程很难离散，例如，由于行为需要解决在一个非常细的网格，一般的非光滑行为，或奇点。由于有限元是一种基于网格的求解器，要在某些不规则域上求解，就必须采用具有设计和求解挑战性的定制方法 [GKLS24]。此外，解决逆问题（例如，用于推断功能材料中的材料性质或发现反应输运中缺失的物理学）往往是非常昂贵的，而且需要复杂的公式、新的算法和复杂的计算机代码；并且，通过传统的方法解决现实中缺失、间隙或噪声边界条件的物理问题目前是不可能的 [KKL+21]。

深度学习方法（包括 PINN [RPK19]、深度利兹方法 []、深度伽辽金方法 [] 等）在解决高维问题方面[3]取得了巨大成功，最近已成为各种偏微分方程数值解的常用方法。它们有潜力克服上述数值离散化经典方法所面临的一些挑战： 通过利用自动微分 []，消除了离散化的需要，从而具有不依赖于网格的优势。 神经网络能够表示比有限元基更一般的功能，可打破维度诅咒 [PMR+16]，为求解高维 PDE 提供了一个很有前途的方向。 虽然与有限元等数值求解器相比训练神经网络（非凸优化问题）可能变得需要计算，但它在评估新数据点时（即泛化能力）是非常有效和高效的。 其集成数据和知识的能力和灵活性为处理知识不完善或有限数据的问题提供了一个可扩展的框架。然而训练深度神经网络需要大量数据，而相关科学问题，如前所述，并不总能获得这些数据。

[3] 例如具有高分辨率的图像分类 []，等

其中，PINN 作为一种潜在的颠覆性技术，将物理信息和知识融入网络拓扑结构或计算流程中，PINN 可以遵守任何对称性、不变性或守恒性原则，这些原则源于由一般时变和非线性偏微分方程建模的、支配观测数据的物理定律 [RPK19]。这种简单而强大的构造使我们能够解决计算科学中的各种问题，从而开发出新的数据高效和物理信息学习机、新的常/偏微分方程数值解算器，以及新的数据驱动的模型反演和系统识别方法。需要强调的是，PINN 并非用于取代 FEM：PINN 在解决本节所述的假定问题和逆问题等方面特别有效；而对于不需要任何数据同化的正向问题，现有的基于网格的数值求解器目前的性能优于 PINN [KKL+21, GKLS24]。表 1 中归纳了求解微分方程的三类方法（即数值离散化、常规神经网络和 PINN 方法）的优缺点。

1.3 Advantages and Application Scenarios of PINN

PINN 具有在小数据环境中实现强泛化的独特优势。通过强制或嵌入物理学，神经网络模型被有效地限制在一个较低维度的流形，因此只需少量数据就能进行训练；PINN 不仅能进行内插，还能进行外推。

必须强调的是，与解决微分方程的任何经典数值方法（基于网格的方法，例如 RK4、有限差分法 FDM、有限元法 FEM）不同，这种预测是在没有对时空域进行任何离散化的情况下获得的，可以打破维度诅咒 [PMR+16, LMMK21]，能够以机器精度精确评估配置点上的微分算子 [BB23]。PINN 不需要处理令人望而却步的小步数，因此可以轻松处理不规则和移动域问题 [GKLS24]，并能在更高维度上良好扩展。自动微分是 PINN 相对于其他经典数值方法求解 DE 的一个显著优势，与传统数值方法相比，自动微分利用链式规则通过网络反向传播并计算导数，它并不依赖于离散该域的网格，但对其而言抽样方法变得更加重要 [WZT+23]。

PINN 将数据与数学物理模型无缝集成，即使在部分理解、不确定的情况下也是如此。由于 PINN 表述本身的平滑性或规律性，即使问题的假设并不完美，也有可能找到有意义的解决方案 [YLRK20]。

PINN [RPK19] 可以直接处理非线性问题（神经网络善于处理非线性问题的固有特性），而无需承诺任何先验假设、线性化或局部时间步进。PINN 在处理病态问题和逆问题时有效且高效，例如，没有指定初始条件或边界条件的正演问题和反演问题，或 PDEs 中某些参数未知的问题，在这些情况下，经典数值方法可能会失效 [KKL+21]。

1.4 Related Work & Our Contributions

Related Work on PINN Solving ODEs. [BB23] 基于 PyTorch 对 PINN 求解典型线性/非线性谐振 ODE 开展了系统、全面、对比性的标准测试[4]。[Bat23b] 还针对在求解刚性 ODE 的 IVP 时遇到的 已知解数据（即训练数据）非常有限、 积分区间如果过大将导致 PINN 难以准确预测解的问题，通过嵌入更多物理信息、优化训练数据损失以确保其更全面地考虑所有初始条件以及渐进式学习策略即逐步增加积分区间并在每个区间内优化 PINN 的参数、采用移动网格来最小化微分方程的残差等改进

[4] 重点关注了在训练过程中使用尽可能少的数据量的可能性

表 1: 求解微分方程三类方法的对比.

求解方法	数值离散化方法	常规神经网络	PINN
特点: √ 优点 × 缺点	√ 理论基础完备、可理解，具有现成的误差估计以及收敛性和稳定性保证 √ 高效率 √ 高精度 × 遭受维度灾难 × 网格生成仍很复杂 × 无法无缝整合噪声数据 × 解决逆问题的成本过高	√ 无网格 √ 克服维度灾难，能应对高维和非线性问题 √ 对新数据的泛化能力 × 需要大数据和强算力 × 过拟合 × 黑箱问题	√ 常规神经网络的优点 + √ 小数据、不易过拟合，泛化能力强 √ 可处理不完善数据和不完整模型 √ 处理病态问题和逆问题时有效且高效 √ 可理解性高 √ 网络参数少、速度快 √ 灵活性好、跨计算平台

策略使得改进后的 PINN 在求解刚性 ODE 的 IVP 具有更高的精度和更稳定的训练过程。此外，这些改进策略也被证明在解决边界值问题（BVP）时同样有效，例如高雷诺数稳态对流-扩散方程的求解。[Bat23a] 进一步比较了软约束和硬约束在解决高阶 Lane-Emden-Fowler 类型方程[5]时的性能。基准测试结果表明：软约束灵活性较高，能够处理多种不同的物理场景和边界条件，但可能需要更多的训练数据和迭代次数来收敛到满意的解；硬约束能够更直接地满足微分方程的约束，因此可能在较少的迭代次数内达到较高的精度，但可能需要更复杂的试验解选择和调整过程。

Our Contributions.

- 通过相关文献调研在主要关注的解微分方程（包括 ODE 和 PDE）领域综述了 PINN 相对于常规神经网络方法、数值离散化方法的优势（见表 1）。

- 从数学原理上描述了软约束 PINN 方法求解微分方程（ODE/PDE）的一般架构和计算流程；并从实验上探索了其工作机制，通过求解 Primer、Van der Pol、Duffing 等典型（涵盖一阶、二阶，线性、非线性）谐振 ODEs 论证了 PINN 方法的有效性和高效率：1) PINN 嵌入物理信息和先验知识从机理上减少数据冗余，具有在小数据环境下实现强泛化的独特优势，大大减少了对标注数据的需求，甚至能在极简数据下预测解（例如线性一阶和二阶微分方程仅分别需要含初始值的一个和两个训练点），还提高了计算效率；2）对噪声数据具有良好的鲁棒性，并能提供准确且物理上一致的预测甚至能在训练集时域外外推，泛化能力得到了增强；并且 3) 可理解性更好。

- 以二阶非线性 Duffing oscillator 在极简数据（含初始值的 2 个训练点加少量配置点）下的收敛性为例，验证了软约束 PINN 方法通过在总损失函数中加正则化项即可方便地施加物理定律（例如能量守恒）约束，从而改进遵守该物理定律的 ODE 求解性能。

- 展示了基于 DeepXDE 实现 PINN 方法的轻代码量、训练速度快等优势。

[5] 如二阶 Lane-Emden 方程、三阶 Emden-Fowler 方程和四阶 Lane-Emden-Fowler 方程

2 Method & Result

2.1 Soft-constrained PINN & Implementation

2.1.1 Governing Equations

考虑一个定义在空间或时空域 $\Omega \subseteq \mathbb{R}^d$ 上的物理系统，其中未知数 $u(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^m$ 为系统状态变量，是空间或时空坐标 $\boldsymbol{x} \in \Omega$ 的函数，对于时间无关系统 $\boldsymbol{x} = (x_1, \ldots, x_d)$，对于时间依赖系统 $\boldsymbol{x} = (x_1, \ldots, x_{d-1}, t)$。支配系统的物理规律常以 ODE/PDE 表征，这些方程称为控制方程 [HLZ+23]：

$$\text{Differential equation: } \mathcal{F}(u;\theta)(\boldsymbol{x}) \equiv \mathcal{F}(u, \boldsymbol{x};\theta) = 0, x \in \Omega. \tag{1}$$

$$\text{Initial Conditions: } \mathcal{I}(u;\theta)(x, t_0) = 0, x \in \Omega_0. \tag{2}$$

$$\text{Boundary Conditions: } \mathcal{B}(u;\theta)(x, t) = 0, x \in \partial\Omega. \tag{3}$$

对于时间依赖系统（即动态系统），需要为状态变量（有时还包括其导数）在初始时刻 t_0 设置初始条件 $\mathcal{I}(u;\theta)(x, t_0) = 0, x \in \Omega_0$，如式 (2)，$\theta \in \Theta$ 对系统进行参数化或控制，其中 θ 可以是一个矢量，也可以是一个包含在控制方程中的函数。对于以偏微分方程为特征的系统，我们还需要对空间域 $\partial\Omega$ 边界上的状态变量进行约束，以使系统适定。对于边界点 $x \in \partial\Omega$，我们有边界条件 $\mathcal{B}(u;\theta)(x, t) = 0, x \in \partial\Omega$，如式 (3)。如果对初始条件和边界条件没有约束，则有：$\mathcal{I}(u;\theta) \triangleq 0$ 和 $\mathcal{B}(u;\theta) \triangleq 0$。

2.1.2 Framework of PINN via Soft Constrains

如图 1所示，假设有一个服从方程 (1) 的系统和一个数据集 $\{u(\boldsymbol{x}_i)\}_{i=1,\ldots N}$，然后就可以构建神经网络 $u_w(\boldsymbol{x})$ 并用以下损失函数对其进行训练，即

$$\begin{aligned}
\mathcal{L}_{total} &= \mathcal{L}_{data} + \underbrace{\mathcal{L}_{gov} + \mathcal{L}_{IC} + \mathcal{L}_{BC} + \cdots}_{\mathcal{L}_{phys}} \\
&= \frac{\lambda_d}{N}\sum_{i=1}^{N} \|u_w(\boldsymbol{x}_i) - u(\boldsymbol{x}_i)\|^2 + \frac{\lambda_g}{|\Omega|}\int_{\Omega} \|\mathcal{F}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x} + \\
&\frac{\lambda_i}{|\Omega_0|}\int_{\Omega_0} \|\mathcal{I}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x} + \frac{\lambda_b}{|\partial\Omega|}\int_{\partial\Omega} \|\mathcal{B}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x} + \cdots .
\end{aligned} \tag{4}$$

式中，$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N} \|u_w(\boldsymbol{x}_i) - u(\boldsymbol{x}_i)\|^2$ 是 PINN 在匹配监督数据集的常规数据损失；$\mathcal{L}_{gov} = \int_{\Omega} \|\mathcal{F}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x}$ 是使得 PINN 网络 u_w 满足微分方程约束的残差损失；$\mathcal{L}_{IC} = \int_{\Omega_0} \|\mathcal{I}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x}$ 和 $\mathcal{L}_{BC} = \int_{\partial\Omega} \|\mathcal{B}(u_w;\theta)(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x}$ 分别是使得 PINN 满足初始条件和边界条件的相应损失。PINN 的损失具有灵活性和可扩展性：式 (4) 中的 $\cdots$ 可以是额外的正则化项 $\mathcal{L}_{reg}$，例如满足能量守恒定律的正则化项；如果没有可用数

据或初始/边界限制，可以直接省略相应的损失项。这些损失的学习率或权重可以通过设置超参数 λ_d, λ_g, λ_i, λ_b, *etc.* 来调整。

为了计算式 (4)，我们需要评估几个积分项，其中涉及 $u_w(\boldsymbol{x})$ 的高阶导数计算（PINN 利用计算图的自动微分来计算这些导数项）。然后，（使用 Monte-Carlo 抽样法）利用一组配置点对积分进行近似。我们使用 $\mathcal{D}_d$, $\mathcal{D}_g$, $\mathcal{D}_i$, $\mathcal{D}_b$ 来表示配置点数据集，用 N_d, N_g, N_i, N_b 表示数据量。那么，损失函数可以近似为 [HLZ+23]

$$\begin{aligned}\mathcal{L}_{total} = \frac{\lambda_d}{N_d}\sum_{i=1}^{N}\|u_w(\boldsymbol{x}_i) - u(\boldsymbol{x}_i)\|^2 + \frac{\lambda_g}{N_g}\sum_{i=1}^{N_g}\|\mathcal{F}(u_w;\theta)(\boldsymbol{x}_i)\|^2 + \\ \frac{\lambda_i}{N_i}\sum_{i=1}^{N_i}\|\mathcal{I}(u_w;\theta)(\boldsymbol{x}_i)\|^2 + \frac{\lambda_b}{N_b}\sum_{i=1}^{N_b}\|\mathcal{B}(u_w;\theta)(\boldsymbol{x}_i)\|^2 + \cdots.\end{aligned} \tag{5}$$

式 (5) 可以使用 SGD 等一阶方法和 L-BFGS 等二阶优化器进行高效训练。

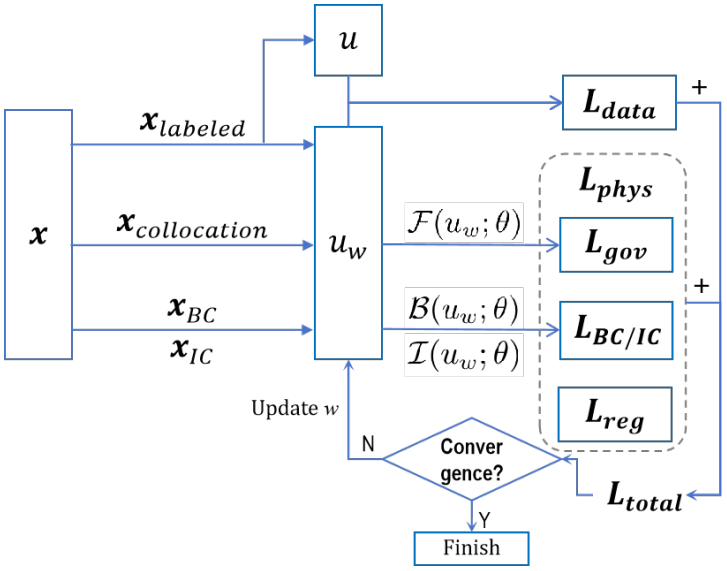


图 1: Schematic diagram of PINN via soft constrains.

2.1.3 DeepXDE Implementation

我们用 DeepXDE [LMMK21] 实现并与 [BB23] 用 Pytorch 的实现进行了结果对比（见2.2.2），用 DeepXDE 编写的代码更短、更接近于数学公式，而且运算效率更高。在 DeepXDE 中求解微分方程如同使用内置模块“搭积木”，包括指定计算域（几何和时间）、微分方程、边界/初始条件、约束、训练数据、神经网络结构和训练超参数。这些模块或组件都是松散耦合的，因此 DeepXDE 具有良好的结构和高度可配置性。

2.2 Results for PINN Solving Oscillator ODEs

为了便于直接对比，我们选用与 [BB23] 中相同的线性和非线性谐振 ODE（即原文中 Tutorial Example 和 Van der Pol osillators）求解。

2.2.1 PINN Solving Linear Oscillator ODE vs NN

线性谐振方程（Tutorial Example/Primer Oscillator）

$$\frac{du}{dt} + 0.1u - \sin(\pi t/2) = 0. \tag{6}$$

式中，$t \in [0, 30]$，初始条件 $u_0 = 1$。我们使用常规神经网络和 PINN 两种方法分别求解，采用 3 层 32 个神经元的全连接隐藏层为 Base 网络，Adam 优化器、学习率 $\eta = 3 \times 10^{-3}$。精确解以龙格库塔法（3000 步，至少约 300 步）所得结果为基准。

用常规神经网络求解时，如果训练点的数量不够或者分布不合理，例如： 图 2(a) 中在全时间域均匀采样 26 个训练点，收敛性不好；或者 图 3(a) 中在左半边时间域均匀采样 61 个训练点，尽管左半区间拟合良好，在无训练点的右半区间上可能会完全失败，即无外推能力。此情况下，需要在全时间域上均匀采样至少 101 个训练点[6]才能与精确解较好地吻合。

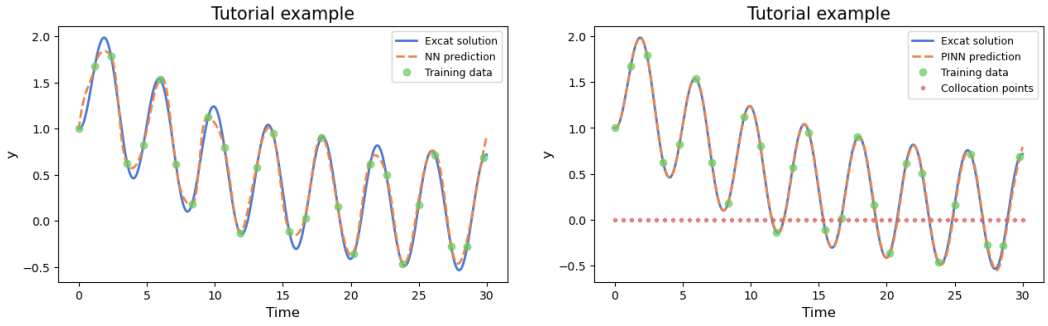
在本例中用 PINN 软约束方法求解与常规神经网络的唯一不同就是，总损失函数中增加了 $\mathcal{L}_{phys}$，因常规神经网络中初始条件已包含在训练集（即第一个训练点）中故只是增加了需满足式 (6) 的 $\mathcal{L}_{gov}$，这需要通过额外地增加配置点来实现（参见式 (5)），本例中在整个时间域均匀采样 50 个配置点。可以这么理解，训练点是有监督的，配置点是是无监督的。具体需要多少配置点取决于神经网络参数量即所求解问题的规模，配置点的分布（均匀/随机）对结果也有一定影响 [WZT+23]。学习速率和损失权值也会影响梯度下降算法的收敛性 [BB23]，这里通过超参数优化损失权值取 $\lambda_d = 1.0, \lambda_r = 6 \times 10^{-2}$。损失权值起到在数据和物理信息双驱动之间平衡的作用。

对于常规神经网络无法收敛的两种情况下，PINN 方法所得结果分别如图 2(b) 和 3(b) 所示，对比相应的 (a) 图可见明显的改善：1) 不仅所需的监督数据点从 101 个点减少为 26 个点（本例中极简情况实际只需要初始条件这 1 个点即可），而龙格库塔法因稳定性和精度要求小时间步长也至少需要约 300 个点，因此对采样点的需要是数量级地减少的——这是 PINN 方法的显著优势之一。2) 特别是通过配置点的辅助、利用 ODE 携带的物理信息，使 PINN 具备了预测训练集所涵盖时域外数据的外推能力，泛化能力得到了增强。此外，PINN 和常规神经网络方法一旦完成训练，对于新数据的预测是瞬时完成的，这是数值离散化（龙格库塔）方法所不具备的特性。

Minimalistic Training Data. 那么，PINN 方法所需的最小训练数据量是多少呢? 对于本例，如果只取初始值一个点，配置点保持不变，得到的解就很好，如图 4(a) 所示，损失函数下降过程见图 4(b) 。[BB23] 的研究也表明，当问题的非线性较弱时，非常少量的标注训练数据加上少量的配置点数据就足以预测解。极简情况下，正如经典的解析或数值积分方法求解一阶和二阶微分方程仅分别需要一个和两个初始条件，PINN 方法也仅分别需要一个训练点（初始值）和两个训练点（含初始值，见图 8 ）。

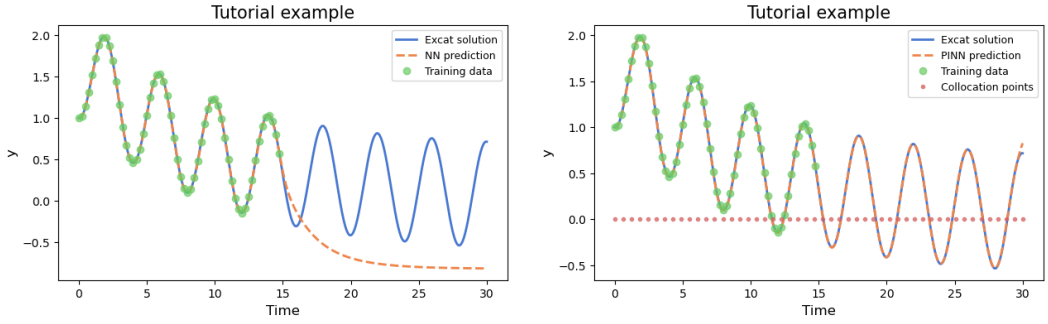
Accelerating Training Time. 相较于常规神经网络，PINN 方法因其只需要小数据量（极简的标注训练数据加上少量的配置点）故可明显缩短训练时长；另一方面，因其

[6] 原文为 101 个训练点，我们基于 DeepXDE 实现 NN 至少需要 50 个训练点?



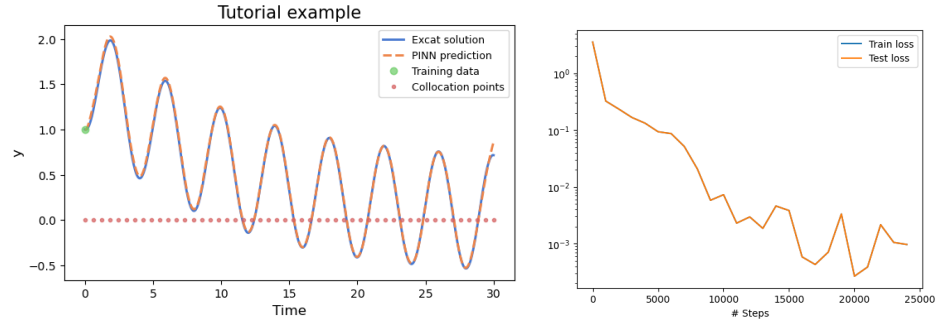
(a) (b)

图 2: Result Comparison of PINN vs NN Solving Linear Oscillator (Case)



(a) (b)

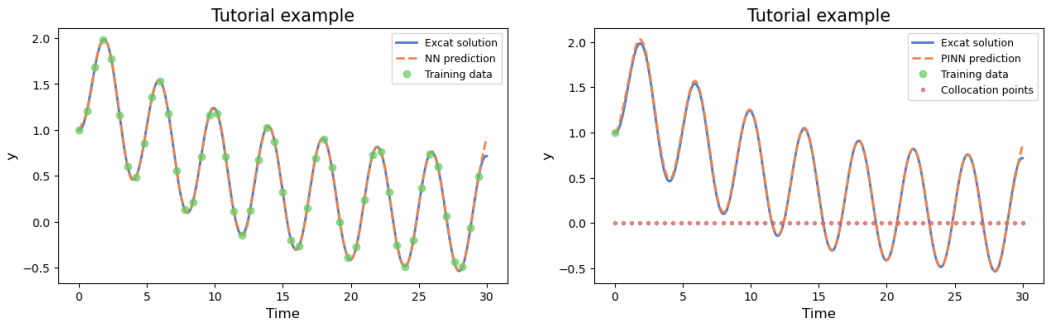
图 3: Result Comparison of PINN vs NN Solving Linear Oscillator (Case)



(a) (b)

图 4: Minimalistic Training Data Result of First-order Linear ODE.

损失函数加上了 $\mathcal{L}_{phys}$ 变得更为复杂，在迭代时需要更多的计算、甚至有时需要更多的迭代轮数，有可能反而拖累了训练效率。我们基于 DeepXDE 在【计算机配置简介】平台上对 Tutorial Example 进行了验证测试，结果如图 5 所示，对比可知：对于线性谐振 ODE，当减少数据量获得的收益大于损失函数引起的拖累时，软约束 PINN 方法是可以加快训练速度的。需要强调的是，因为 Primer Oscillator 是简单的线性 ODE，方便对比起见，我们使用了相同的 Base 网络；PINN 因其小数据量特性可以比 NN 使用参数量更少的网络，故能从减少网络参数量上更进一步地提高训练速度。



(a) (b)

图 5: Training Time of PINN vs NN Solving Primer Oscillator via DeepXDE: (a) 10.53s【训练时长显示在图片上】by NN with Min. 50 training points; (b) 8.82s by PINN with Min. 1 initial training points + 48 collocation points.

2.2.2 PINN Solving Non-linear Oscillator ODE

软约束 PINN 方法在求解非线性谐振 ODE 上同样具有良好的表现 [BB23]。我们基于 DeepXDE 成功求解了典型的、强非线性的 Van der Pol oscillator，并展示了 PINN 方法对噪声数据的容忍性以及 DeepXDE 在 PINN 实现上的低代码和求解速度优势。Van der Pol (VDP) oscillator ODE 如下

$$\frac{d^2u}{dt^2} + \omega_0^2 u - \epsilon\omega_0(1-u^2)\frac{du}{dt} = 0. \tag{7}$$

式中，ω_0 是归一化的角速度，取 $\omega_0 = 15$；ϵ 反映了 VDP 的非线性程度，ϵ 值越大非线性越强；$t \in [0, 1.5]$，初始条件 $u_0 = 1$。

Noisy Data Tolerated. 如图 6 所示，我们使用 PINN 方法求解含噪声小数据下的 VDP 非线性谐振子，依旧采用 3 层 32 个神经元的全连接隐藏层为 Base 网络，这体现了 PINN 继承了神经网络的强表达能力，相关参数设置（即 ϵ、训练点数、配置点数、正态噪声方差）见图 6 中说明，【其他超参数设置可参考开源代码】。精确解由 RK4 生成。图 6 结果表明，随着非线性程度的增加（通过增大 ϵ）需要更多的训练点和配置点点数布置在非线性特征明显的区域才能得到精确解【待图片定稿后补充简单数据分析】。并且随着非线性的增大，对噪声的容忍能力也逐渐下降：随着 ϵ 从 1 增大到 5，能容忍的 σ 却从 0.1 下降到 0.05；即使加大数据点数对提升噪声容忍度的作用也十分有限。

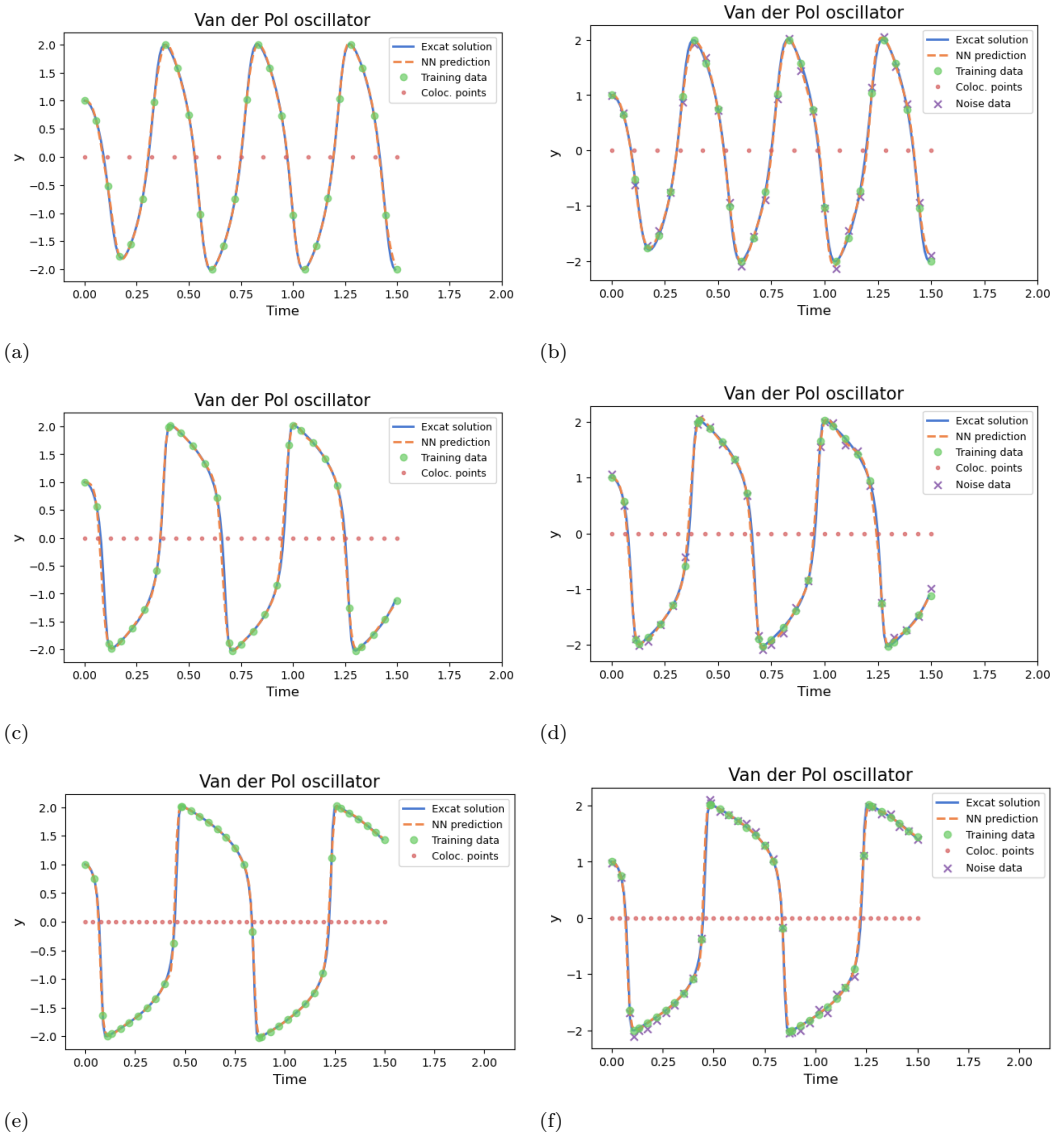


图 6: Results of PINN Solving VDP Oscillator via DeepXDE with Noisy Small Data. Top(a)(b): $\epsilon = 1$, No. of training data points $N_{data} = 28$, No. of collocation data points $N_c = 15$, Noise's normal variance $\sigma = 0.1$; Middle(c)(d): $\epsilon = 3$, $N_{data} = 32$, $N_c = 25$, $\sigma = 0.08$; Bottom(e)(f): $\epsilon = 5$, $N_{data} = 38$, $N_c = 40$, $\sigma = 0.05$. 【参数值配置尽量写进图片】

DeepXDE vs Pytorch of PINN Implementation Code. 对比图 7 可知，基于 DeepXDE 或 Pytorch 实现软约束 PINN 的总体流程（参见图 1 ）是一致的。DeepXDE 对 PINN 方法的定制化封装更好，因此所需的代码量更少，在求解 VDP 的相同设置下，DeepXDE 的代码行数（100 行）约是 Pytorch 实现的 $\frac{1}{3}$。此外，DeepXDE 的训练效率明显优于 Pytorch，对于图 5(b) 的例子，相同设置下，基于 Pytorch 训练耗时 17.30s，比 DeepXDE 的多一倍左右。这得益于 DeepXDE 的后端针对 PINN 方法所做的优化 [LMMK21]。

PINN Computational Efficiency @CPU vs @GPU. PINN 方法不依赖于大数据故可以在 CPU 或 GPU 等平台上高效运算。我们对比了基于 DeepXDE 分别在 CPU 和 GPU 上【CPU、GPU 参数简介】求解 VDP oscillator 的运算时间:【待补充】

2.2.3 Energy Conservation Regularization Improves Result of Solving Duffing Oscillator

PINN 方法的一大优势是可以充分利用物理信息或知识先验，例如能量守恒定律，软约束法通过在总损失函数中加正则化项 $\mathcal{L}_{reg}$ 来实现。我们用二阶非线性的 Duffing oscillator 来验证，其 ODE 如下

$$\frac{d^2u}{dt^2} + \alpha u + \beta u^3 = 0. \tag{8}$$

式中，$t \in [0, 1.5]$，取 $\alpha = 3.0, \beta = 1.0$?，初始条件为 $u_0 = 1.5, u_0' = 0$。同样，精确解以龙格库塔法所求得结果为准。式(8)的能量守恒项为 $E = \frac{1}{2}(\frac{du}{dt})^2 + \frac{1}{4}\beta^2 u^4$，代入正则化项 $\mathcal{L}_{reg}$。

考虑极简数据的情况，对于二阶 ODE 只用两个训练点（含初始值），均匀选取配置点数 $N_c = 37$?40? 【补充：超参数设置 (参见式 (4))。】

使用和不使用能量守恒正则化项的 PINN 求解 Duffing oscillator 结果分别如图 8(a) 和 (b) 所示，受非线性影响，极简数据情况下，没有用能量守恒正则化的 PINN 在整个 36000 轮次的训练中都无法完成收敛；而加上能量守恒正则化的 PINN 则取得了明显的改善，与精确解完美吻合。需要注意的是，能量守恒正则化对 Duffing 有用，因为 Duffing 遵守能量守恒；对 VDP 无用，因为 VDP 是能量耗散的。

3 Concluding Remarks

3.1 Summary

(1) 深度神经网络的成功仰赖于大数据和强算力，却也忽略了充分利用先验信息或知识，特别是在小数据环境下缺乏鲁棒性甚至无法收敛。PINN 正是深度神经网络在小数据场景下的有力补充，继承了神经网络的强表达能力，并通过硬编码或软约束的方式将物理信息和知识融入神经网络拓扑结构或学习架构以及计算流程中，从而减少了数据冗余、改进了泛化性能，进而提高了计算效率、增强了可解释性和鲁棒性。具体而言，PINN

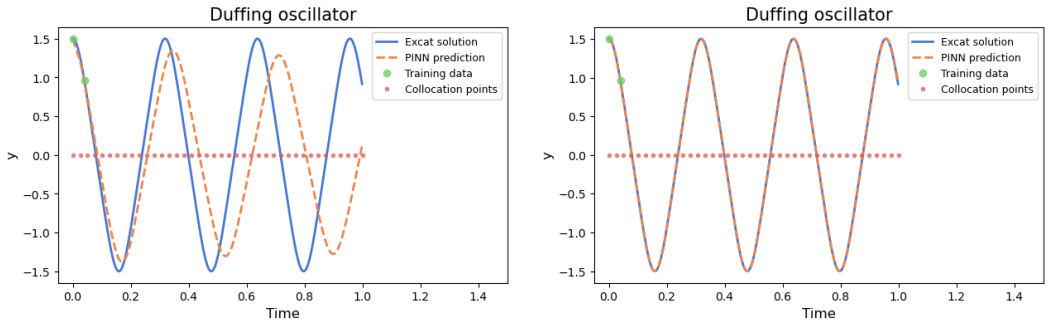
```python
import deepxde as dde
import numpy as np
from scipy.integrate import odeint

def VDP:
    """定义VDP的ODE"""

# 调用scipy中的odeint生成数值精确解
y = odeint(VDP)

# 从精确解中采样生成训练集TraingData(含初始点(t0,y0)和端点)
t_data=t[0:1.5:0.005]
y_data=y[0:1.5:0.005]

geom = dde.geometry.TimeDomain(0.0, 1.5)    # 定义区间

def boundary(t, on_boundary):
    return on_boundary and dde.utils.isclose(t[0], 0)

def error_derivative(inputs, outputs, X):
    return dde.grad.jacobian(outputs, inputs, i=0, j=None)

# 定义初始条件: y(0)=1 和 y'(0)=0
ic1 = dde.icbc.IC(geom, lambda x : 1, lambda _, on_initial: on_initial)
ic2 = dde.icbc.OperatorBC(geom, error_derivative, boundary)

# 定义训练集和边界条件(含边界条件)
t_y_data = dde.PointSetBC(t_data, y_data)

data = dde.data.TimePDE(geom, ode, [ic1, ic2, t_y_data], num_domain=40, num_boundary=1)     # num_domain为配置点数

net = dde.maps.FNN([1] + [32] * 3 + [1])   # 实例化神经网络
model = dde.Model(data, net)
model.train(epochs=24000)
```

(a)

```python
import torch
import numpy as np
from scipy.integrate import odeint

def VDP:
    """定义VDP的ODE"""

# 定义区间和边界条件
t = np.linspace(0, 1.5, 1000)
y0 = [1, 0]     # y0为初始条件

# 调用scipy中的odeint生成数值精确解
y = odeint(VDP, y0, t)

# 从精确解中采样生成训练集TraingData(含初始点(t0,y0)和端点)
t_data=t[0:1.5:0.005]
y_data=y[0:1.5:0.005]

# 选取配置点
x_collocation = torch.linspace(0.,1.5,48).view(-1,1).requires_grad_(True)

class FNN:
    """定义神经网络"""

model = FNN(1, 1, 32, 3)    # 实例化FNN

# PINN的训练流程:
for i in range(24000):

    # 数据损失
    yh = model(t_data)
    Loss_data = 1 * torch.mean((yh - y_data)**2)

    # 物理损失
    yhp = model(x_collocation)
    dx = torch.autograd.grad(yhp, x_collocation, torch.ones_like(yhp), create_graph=True)[0]
    dx2 = torch.autograd.grad(dx, x_collocation, torch.ones_like(dx), create_graph=True)[0]
    physics = (dx2 + 15 **2 * yhp - 5 * (1 - yhp**2) * dx)
    Loss_phys = (1e-4) * torch.mean(physics**2) * 1

    # 计算测试集的均方误差
    yhpp = model(t)
    mse = torch.mean((yhpp - y)**2)

    # 反向传播总损失
    Loss_total = Loss_data + Loss_phys
    Loss_total.backward()
```

(b)

图 7: (a) DeepXDE vs (b) Pytorch of PINN Implementation Code.



(a) (b)

图 8: PINN Solving Duffing Oscillator (a) w/o vs (b) with Energy Conservation Regularization.

在只有少量（监督的或标注）训练示例的情况下也能很好地泛化，在数据不完善（如噪声、缺失或异常值等）的情况下仍能保持稳健，并能提供准确且物理上一致的预测甚至能用于外推任务。

另一方面，相较于数值离散化方法（例如龙格库塔法、有限元法），PINN 具有 1）无需划分网格或离散时间步长，2）基于数据驱动的可无缝合并实验数据并且对噪声的容忍性好，3）评估新数据点时非常有效和高效（尽管训练神经网络可能变得需要计算）等优势，还可打破维度诅咒、适用于数值离散化方法不擅解决或求解代价昂贵的高维、非线性和非光滑以及逆问题等等。基于以上优势和特点，PINN 正成为数字双胞胎（Digital Twins）新兴时代的有利催化剂。

(2) 我们归纳了软约束 PINN 方法求解微分方程（ODE/PDE）的一般架构和计算流程，从损失函数的构成上可知 PINN 是受数据和物理信息双驱动的，比常规神经网络增加了微分方程（包括初始、边界条件）以及诸如能量守恒定律等先验信息和知识的约束或规范。并通过求解 Primer、Van der Pol、Duffing 等典型（涵盖了一阶、二阶，线性、非线性）谐振 ODEs 从实验上验证了软约束 PINN 方法的工作机制及其精度和效率。

PINN 大大减少了对标注数据的需要：当问题的非线性较弱时（e.g. Primer 和 Duffing），非常少量的标注训练数据加上少量的配置点数据就足以预测解；极简情况下，正如经典的解析或数值积分方法求解一阶和二阶微分方程仅分别需要一个和两个初始条件，PINN 方法也仅分别需要一个训练点（初始值）和两个训练点（含初始值）。强非线性问题（e.g. VDP）也仅需要适当增加训练点和配置点，相比常规 NN 仍有显著优势。

通过配置点的辅助、利用 ODE 携带的物理信息，使 PINN 具备了预测训练集所涵盖时域外数据的外推能力，而且对噪声数据具有良好的鲁棒性，泛化能力得到了增强。与此同时，当伴随着数据量的减小（PINN 的网络参数可以更少）获得的收益大于损失函数项增加引起的拖累时，训练速度是加快的。

软约束 PINN 方法通过在总损失函数中加正则化项即可方便地施加物理定律（例如能量守恒）约束，从而改善遵守该物理定律的 ODE 的求解性能，例如二阶非线性 Duffing oscillator 能够在极简数据（含初始值的 2 个训练点加少量配置点）下快速收敛。

(3) 基于 DeepXDE 实现 PINN 方法，不仅轻代码、训练效率也高。并且 PINN 方法不依赖于大数据故可以在 CPU 或 GPU 等平台上高效运算，跨平台灵活度高。

3.2 Future Work

在本文基础上，下一步可继续探讨 PINN 方法（包括硬编码和软约束）求解刚性 ODE 和 PDE 以及对求解精度和效率至关重要的采样方法的探索 [WZT+23]。此外，PINN 与领域分解相结合可扩展至大型问题 [KKL+21]，并进一步扩展到求解积分微分方程 (IDE)、分数阶微分方程 (FDE) 和随机微分方程 (SDE) [LMMK21]。

Code Availability

论文中已有部分代码示例，详细代码将分享在 https://github.com/mikelu-shanghai/PINNtoODEwithSmallData。[7]

Mathematical Notations

Notations	Description
u	state variables of the physical system
$\boldsymbol{x}$	spatial or spatial-temporal coordinates
x	spatial coordinates
t	temporal coordinates
θ	parameters for a physical system
w	weights of neural networks
$\int$	integral operator
$\mathcal{F}$	differential operator representing the PDEs/ODEs/etc.
$\mathcal{I}$	initial conditions (operator)
$\mathcal{B}$	boundary conditions (operator)
Ω	spatial or spatial-temporal domain of the system
Θ	space of the parameters θ
W	space of weights of neural networks
$\mathcal{L}_{total}$	total loss function
$\mathcal{L}_{data}$	supervised data loss
$\mathcal{L}_{gov}$	residual loss
$\mathcal{L}_{BC}$	boundary condition loss
$\mathcal{L}_{IC}$	initial condition loss
$\mathcal{L}_{reg}$	regularization loss
$\|\cdot\|$	norm of a vector or a function

参考文献

[Bat23a] Hubert Baty. Solving higher-order lane-emden-fowler type equations using physics-informed neural networks: benchmark tests comparing soft and hard constraints. *arXiv:2307.07302*, 2023.

[7] 待论文接收后，需要的读者可与我们联系。

[Bat23b] Hubert Baty. Solving stiff ordinary differential equations using physics informed neural networks (pinns): simple recipes to improve training of vanilla-pinns. *arXiv:2304.08289*, 2023.

[BB23] Hubert Baty and Leo Baty. Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests. *arXiv:2302.12260*, 2023.

[BG21] Mohammad Mahdi Bejani and Mehdi Ghatee. A systematic review on overfitting control in shallow and deep neural networks. *Artif. Intell. Rev.*, 54(8):6391–6438, 2021.

[CLS21] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12:6136–48, 10 2021.

[CYY+22] Kwan Ho Ryan Chan, Yaodong Yu, Chong You, Haozhi Qi, John Wright, and Yi Ma. Redunet: A white-box deep network from the principle of maximizing rate reduction. *Journal of Machine Learning Research*, 23(114):1–103, 2022.

[GKLS24] Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, 05 2024.

[HLZ+23] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: A survey on problems, methods and applications, 2023.

[KKL+21] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422 – 440, 2021.

[LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[LMMK21] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[LPY+21] Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[PMR+16] Tomaso A. Poggio, Hrushikesh Narhar Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14:503 – 519, 2016.

[RPK19] 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. *Journal of Computational Physics*, 378:686–707, 2019.

[SHS+18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140 – 1144, 2018.

[WJX+22] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55:1–37, 03 2022.

[WZT+23] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[YLRK20] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLOS Computational Biology*, 16(11):1–19, 2020.