arXiv:2303.03848v2 [math.NA] 5 Jun 2023

采用物理信息神经网络作为粗传播器的 Parareal*

AbdulQadirIbrahim[1][0000−0003−3452−8500]、Sebastian Götschel[1][0000−0003−0287−2120] 和 Daniel Ruprecht[1][0000−0003−1904−2473]

汉堡工业大学数学研究所计算数学教席，德国汉堡
{abdul.ibrahim,sebastian.goetschel,daniel.ruprecht}@tuhh.de

摘要。 时间并行算法为基于时间依赖微分方程模型的数值积分提供了额外的并发层。Parareal 等跨多个时间步并行化的方法，依赖计算成本低廉的粗积分器向前传播时间信息，同时由可并行化但计算成本高昂的细传播器提供精度。通常，粗方法采用低分辨率、降阶或简化模型的数值积分器。本文提出改用物理信息神经网络（PINN）。我们以计算金融领域的偏微分方程——布莱克-斯科尔斯方程为例，证明采用 PINN 粗传播器的 Parareal 比采用数值粗传播器能获得更高的加速比。训练和评估神经网络的计算模式都非常适合在 GPU 上执行。相比之下，计算强度低的基于网格的算法难以发挥良好性能。我们还表明，将粗传播器 PINN 移至 GPU，同时在 CPU 上运行数值精细传播器，可以进一步提升 Parareal 在单节点上的性能。这表明，将机器学习技术融入时间并行积分方法，并利用它们在计算模式上的差异，或许能更好地利用异构架构。

关键词： Parareal · 时间并行积分 · 物理信息神经网络· 机器学习 · 图形处理器 · 异构架构

1 引言

基于微分方程的模型在科学和工程领域无处不在。由于许多问题具有多尺度特性，通常需要满足高分辨率要求，因此这些模型往往必须在高性能计算机上运行

* 本项目获得了欧洲高性能计算联合事业（JU）根据资助协议第955701号提供的资助。JU还获得了欧盟地平线2020研究与创新计划以及比利时、法国、德国和瑞士的支持。本项目还获得了德国联邦教育与研究部（BMBF）资助，项目编号为16HPC048。本项目还获得了德国联邦教育与研究部（BMBF）根据资助协议16ME0679K提供的资助。

以应对内存需求和计算成本。空间并行已成为一种广泛使用且有效的方法，可用于并行化偏微分方程的数值算法；但仅靠这一方法，无法为极大规模并行架构提供足够的并发度。时间并行算法有助于提高数值模型的并行度。结合时空并行，可在数十万个核心上实现比单独采用空间并行更高的加速比 [24]。

Parareal [14], PFASST [4] 或 MGRIT [5] 等时间并行方法依赖串行粗层积分器，将信息向前传播至后续时间点。这些粗传播器构成了不可避免的串行瓶颈，限制了可实现的加速比。因此，粗层积分器必须尽可能快速。然而，这些方法是迭代式的，随着迭代次数增加，加速比也会降低。过于不准确的粗传播器即使计算成本低，也无法提供良好的加速比，因为所需的迭代次数会过多。因此，优秀的粗传播器需要具备至少一定程度的准确性，同时还必须尽可能快速地运行。这种权衡表明，将神经网络用作粗传播器可能很有前景：训练完成后，神经网络的评估速度非常快，同时仍能提供合理的准确性。此外，神经网络非常适合在 GPU 上运行，而基于网格的离散化则较难高效运行，因为其计算强度较低。因此，结合基于网格的组件与神经网络组件的算法，非常适合在结合 CPU 和 GPU 或其他加速器的异构系统上运行。

我们的论文有三项创新贡献：（i）首次研究了在 Parareal 中使用 PINN 作为粗传播器；（ii）表明将 PINN 用作粗传播器可以加快 Parareal 收敛并提高加速比；（iii）说明将 PINN 粗传播器移至图形处理器上可以进一步提高加速比。虽然我们以 Black-Scholes 方程（计算金融中的一种模型）来展示该方法，但这一思路也适用于其他类型的偏微分方程，而 Parareal 已被证明对这些方程有效。我们仅研究了在配备一块 GPU 的单节点上的性能。将该方法扩展到多个节点间的时间并行，并使其与空间并行相结合，则留待未来研究。本文中用于生成所示结果的代码可免费获取 [19]。

2 相关工作

使用机器学习（ML）求解微分方程已成为一个活跃的研究领域。一些论文旨在完全以神经网络取代数值求解器 [21,25]。物理信息神经网络（PINNs） [20], 将偏微分方程（PDE）的残差以及边界条件和初始条件纳入损失函数，已应用于许多领域。其中包括对 Black-Scholes 方程（1）的演示，表明 PINN 能够准确地为一系列具有复杂收益的期权定价，

而且速度显著快于传统数值方法 [23]。然而，单独使用机器学习求解微分方程，通常无法达到数值求解器所能实现的高精度。因此，出现了许多新思路：将机器学习作为经典数值方法的组成部分，而不是用它取而代之 [9]。

对于时间并行积分方法，有两个旨在将其与机器学习相结合的研究方向。一方面，有研究尝试使用机器学习技术改进时间并行算法。本文属于这一类。此前已有两篇论文研究了将神经网络用作 Parareal 的粗传播器。Yalla 和 Enquist [26] 率先探索了这一方法。他们采用一个包含 1000 个神经元的隐藏层的神经网络，并通过一个高维振子证明，与数值粗传播器相比，该方法有助于 Parareal 更快收敛。不过，论文没有报告运行时间或加速比。Agboh 等人 [1] 采用前馈深度神经网络作为粗传播器，对一个常微分方程进行积分，该方程模拟机械臂推动多个物体时产生的响应。他们还发现，与简化的解析粗模型相比，经过训练的粗传播器能够改善 Parareal 收敛。Nguyen 和 Tsai [17] 并未完全取代数值粗传播器，而是使用监督学习提高其在波传播建模中的精度。他们发现，只要训练数据包含足够有代表性的样例，这种方法就能提高 Parareal 的稳定性和精度。Gorynina 等人 [6] 研究了在使用 Parareal 的分子动力学模拟中采用机器学习得到的谱邻域分析势。

少数论文则采取相反的方向，借鉴时间并行积分方法的思想，以实现深度神经网络训练过程的并行化与加速。Günther 等人 [7]采用非线性多重网格方法来改进深度残差网络的训练过程。他们使用 MGRIT——Parareal 的多层泛化方法——在 CPU 上实现层并行训练，并报告称在 128 个核心上最高可获得 8.5 倍的加速比。Kirby 等人 [11] 将其方法扩展到多个 GPU，进一步提升了性能。类似地，Meng 等人 [16] 使用 Parareal 为一系列物理信息神经网络生成初始值，以帮助训练过程。鉴于对需要在较长时间区间上进行积分的物理信息神经网络进行训练的成本较高，他们转而将多个短时间物理信息神经网络串联起来。他们使用廉价的数值粗传播器和 Parareal 迭代来连接这些物理信息神经网络，并使每个物理信息神经网络继承其前一个网络的参数。尽管他们提到了使用物理信息神经网络作为粗传播器的可能性，但并未在论文中进一步探究这一思路。Lorin [15] 推导出神经常微分方程的时间并行变体，以改进深度残差神经网络的训练。最后，Lee 等人 [13] 使用类似 Parareal 的过程，在多个 GPU 上训练深度神经网络。

3 算法与基准问题

布莱克-斯科尔斯方程是金融市场中广泛使用的期权定价模型 [3]。它基于这样的假设：资产价格遵循

几何布朗运动，因此资产的对数收益率服从正态分布。欧式看涨期权或看跌期权的价格存在闭式解 [12], ，但对于美式期权或含有多个标的资产等更复杂的期权，则不存在闭式解。为了能够计算数值误差，我们因此聚焦于欧式看涨期权，这是一种金融衍生品，赋予买方权利但没有义务，在到期日或之前以预先确定的价格（行权价）购买标的资产。期权的价格 V 可建模为

$$f(V) = \frac{\partial V}{\partial t}(S,t) + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}(S,t) + rS\frac{\partial V}{\partial S}(S,t) - rV(S,t) = 0, \quad (1)$$

其中， S 表示标的资产的当前价值， t 表示时间， r 表示无风险利率（例如银行储蓄利率）， σ 表示标的资产的波动率。为完全确定方程 **(1)** 的解，我们在到期时间 $t = T$ 处施加终端状态，并根据期权在 $S = 0$ 处以及当 $S \to \infty$时的行为，对 S施加两个边界条件。对于看涨期权，到期时间条件为

$$V(T,S) = \max(S-K,0) \text{ for all } S. \quad (2)$$

如果标的资产变得一文不值，那么它将一直一文不值，因此期权也将一文不值。因此，

$$V(t,0) = 0 \text{ for all } t. \quad (3)$$

另一方面，如果 S 变得非常大，那么期权几乎肯定会被行权，且与 S相比，行权价可以忽略不计。因此，期权的价值基本上与标的资产本身的价值相同，并且

$$V(t,S) \sim 0 \text{ as } S \to \infty, \text{ for fixed } t. \quad (4)$$

对于欧式看涨期权，我们选取一个由 $t = 0$ 和 $T = 1$ 构成的区间，并为资产设定一个人为上界 $S = 5000€$。

3.1 Parareal

Parareal 是用于求解以下形式初值问题的迭代算法

$$V'(t) = \phi(V(t)),\ t \in [0,T],\ V(0) = V_0, \quad (5)$$

其中在我们的情形下，右端函数 ϕ 源自式 **(1)** 中空间导数的离散化。注意，式 **(1)** 中的系数不依赖于时间，因此我们可以将讨论限定于自治情形。将时间域 $[0,T]$ 分解为 N个时间片 $[T^n, T^{n+1}]$、 $n = 0$，……， $N-1$。记 $\mathcal{F}$ 为一个步长恒定且精度较高的数值时间步进算法 δt ，并将……记为

$$V_{n+1} = \mathcal{F}(V_n) \quad (6)$$

从时间片的起始时刻 T^n 处某个初值 V_n 开始，积分至终止时刻 T^{n+1}所得的结果。经典时间步进对应于串行计算式 (6) 在 $n = 0, \ldots\ldots, N-1$ 处的值。Parareal 用如下迭代取代这一串行过程

$$V_{n+1}^{k+1} = \mathcal{G}(V_n^{k+1}) + \mathcal{F}(V_n^k) - \mathcal{G}(V_n^k) \tag{7}$$

其中， $k = 1, \ldots, K$ 表示迭代次数。式 (7) 的关键在于，计算开销较大的$\mathcal{F}$ 求值可以在所有 N 个时间片上并行执行。这里，我们始终假设使用 $P = N$ 个进程，且每个进程负责一个时间片。文献 [22]中给出了 Parareal 工作流的可视化示意图和伪代码。根据 $k \to N$， V_n^k收敛到通过串行求解式 (6) 得到的同一解。不过，要实现加速，我们要求在 $K \ll N$ 次迭代内收敛。使用 P 个处理器对 $N = P$ 个时间片进行积分时，Parareal 可实现的加速比上界为

$$s_{\mathrm{bound}}(P) = \frac{1}{\left(1 + \frac{K}{P}\right)\frac{c_{\mathrm{c}}}{c_{\mathrm{f}}} + \frac{K}{P}} \tag{8}$$

其中， K 是迭代次数， c_{c} 是$\mathcal{G}$ 的运行时间， c_{f} 是$\mathcal{F}$ [22]的运行时间。由于式 (8) 忽略了开销和通信，因此它是可实现加速比的上界，而实测加速比会更低。

3.2 布莱克-斯科尔斯方程的数值求解

我们在等距网格上采用二阶中心有限差分来近似式 (1) 中的空间导数

$$0 = S_0 < S_1 < \ldots < S_N = L \tag{9}$$

其中， $S_{i+1} - S_i = \Delta S$适用于 $i = 0, \ldots\ldots, N-1$。对于内部节点，我们得到半离散初值问题

$$V_j^{'}(t) = -\frac{1}{2}\sigma^2 S_j^2 \frac{V_{j+1} - 2V_j + V_{j-1}}{\Delta S^2} - rS_j \frac{V_{j+1} - V_{j-1}}{2\Delta S} + rV_j \tag{10}$$

其中， $j = 1, \ldots\ldots,$ 。此外，还需满足资产价值为零时的边界条件 $V_0 = 0$。我们还在有限距离 L 处施加渐近边界条件 (4)，使得 $V_N = 0$。在时间离散方面，对于$\mathcal{F}$ 我们采用二阶 Crank-Nicolson 方法，对于数值$\mathcal{G}$则采用一阶隐式欧拉方法。由于这里给定的是终值条件而非初值条件，因此我们从时间 $T = 1$开始，反向求解该问题。精细方法采用 200 个时间步，粗略方法采用 100 个时间步。

3.3 物理信息神经网络（PINN）

我们使用的物理信息神经网络作为粗传播器，接收时间片 $[t_{\mathrm{start}}, t_{\mathrm{end}}] \subset [0, T]$、$t_{\mathrm{start}}$ 时的资产价格 V 和股票价值 S，并输出预测状态

$\tilde{}$ t_{end}时资产价格 V 的状态。为训练该网络，我们在时间和股票价格上定义三组配置点：$(S_i,\ t_i)\ i = 1,\ \ldots N_f$位于时空域内部，用于计算 Black-Scholes 方程 (1) 的残差 $f(V)$；$(S_i,\ t_i)\ i = 1,\ \ldots N_b$是边界上的配置点，用于计算 (2)；以及$S_i,\ i = 1,\ \ldots N_{\text{exp}}$，用于终态条件 (3)、(4)。待最小化的损失函数为

$$\text{MSE}_{\text{total}} = \text{MSE}_f + \text{MSE}_{\text{exp}} + \text{MSE}_b, \tag{11}$$

由一项用于最小化偏微分方程残差 $f(V)$ 的项组成

$$\text{MSE}_{\text{f}} = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(\tilde{V}(t_i, S_i))|^2, \tag{12}$$

边界损失项

$$\text{MSE}_{\text{b}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left|\tilde{V}(t_i, S_i) - V(t_i, S_i)\right|^2, \tag{13}$$

以及到期时的损失

$$\text{MSE}_{\text{exp}} = \frac{1}{N_{\text{exp}}} \sum_{i=1}^{N_{\text{exp}}} \left|\tilde{V}(T, S_i) - \max(S_i - K, 0)\right|^2, \tag{14}$$

对于我们的设置，我们在区域内随机生成 $N_f = 100{,}000$ 个配置点 $[0, 5000] \times [0, 1]$, $N_b = 10{,}000$ 个边界配置点$[0, 1]$ ，并生成 $N_{\text{exp}} = 10{,}000$ 个配置点，以在 $[0, 5000]$上对到期条件进行采样。计算偏微分方程损失所需的导数通过自动微分 [2]计算。我们在区域内的点上计算偏微分方程残差 (12)，在空间边界上计算边界条件损失 (13)，并在端点上计算终止损失 (14)。这三项之和构成总损失函数 (11)。图1展示了生成的配置点子集，以说明这一方法。

该神经网络包含 10 个全连接层，每层有 50 个神经元，并使用 PyTorch [18] 实现。图2展示了物理信息神经网络的原理；为便于阅读，图中采用了规模较小的网络。除输出层外，每个线性层后面都接有 ReLU激活函数。神经网络的权重使用 Kaiming [8] 初始化。本文侧重于概念验证，尚未系统地优化网络架构，但这将是未来研究的一个有趣方向。

我们使用 Adam优化器 [10] ，以 10^{-2} 的学习率进行初始训练，共训练 5000个训练轮次；随后以 10^{-3} 的学习率进行第二轮训练，共训练800个训练轮次。每个训练轮次期间都会对训练数据（配置点）进行随机打乱，以防止模型基于数据顺序而非数据中的潜在模式来改善预测结果。表1展示了三个损失函数项的变化情况。该模型的总训练时间约为30分钟。

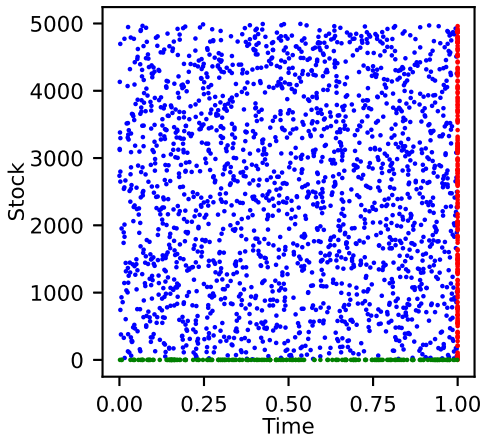

图1：随机生成的部分配点。通过最小化偏微分方程残差，强制解在内部节点处满足偏微分方程；通过边界损失使其在绿色节点处满足边界条件，并通过终止损失使其在红色节点处满足到期条件。

4 结果

数值实验在运行 OpenSUSE Leap 15.4 的系统上进行，硬件配置为 Intel Core 24 x 12th Gen Intel i9-12900K，基准时钟频率为3.2 GHz，最高睿频为5.2 GHz，配备62.6 GiB内存和 NVIDIA GeForce RTX 3060/PCIe/SSE2 GPU。实验使用 Python 3.10、PyTorch 1.13.1+cu117、mpi4py3.1.4，以及用于 GPU 运行的 numba0.55.1 实现。

Parareal 收敛。 图3展示了串行细传播器、数值粗传播器和 PINN 粗传播器随时间变化的归一化 ℓ_2 误差（左）。正如预期，细传播器最为精确，在模拟结束时的 ℓ_2 误差约为 10^{-3} 。数值粗传播器的精度明显较低。PINN 粗传播器比数值粗传播器更为精确，但同样无法达到细传播器的精度。为说明将微分方程编码到损失函数中的重要性，我们还展示了一个仅使用细传播器生成的数据进行训练、但不包含编码微分方程的项的神经网络（NN）。不含偏微分方程残差的神经网络比数值粗方法略为精确，但不如 PINN。需要注意的是，此处使用的 PINN 不需要数值生成的轨迹作为训练数据，因为损失函数（11）仅由偏微分方程残差、边界条件和到期条件组成，不包含数据失配项。

图 3（右）展示了 Parareal 的归一化 ℓ_2 误差随迭代次数的变化。对于数值方法、神经网络和物理信息神经网络这三种粗传播器，Parareal

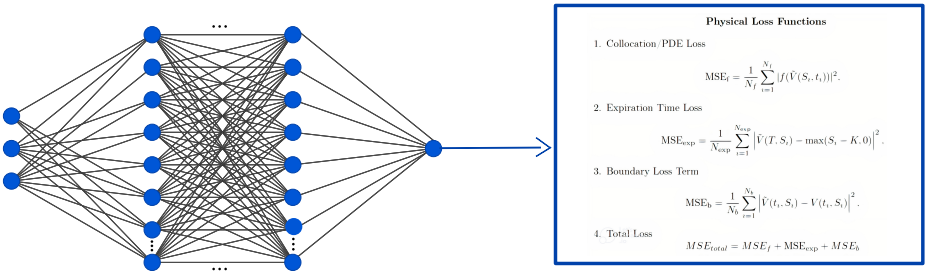

图2：PINN 的结构。网络将时间 t_{start}，t_{end}、资产价˜值 V 和股票价值 S 作为输入，并返回 t_{end} 时刻的预测资产价值 V。损失函数编码了偏微分方程、到期条件和边界条件。图由 `https://alexlenail.me/NN-SVG/index.html` 制作。

都能非常快速地收敛。尽管物理信息神经网络和神经网络比数值粗传播器略为精确，但对收敛的影响很小。经过一次迭代后，Parareal 的迭代误差小于精细方法的离散误差。经过 $K = 3$ 次迭代后，Parareal 已将精细解复现至舍入误差的精度。下文报告 K =3 的运行时间和加速比。仅进行一次迭代时，式 (8) 中的 K/P 项不那么重要，降低粗传播器的运行时间会进一步提高整体加速比。因此，$K = 3$ 的情况是切换到粗传播器所带来的改进较小的情况。

泛化。 图 4 展示了将 PINN 粗传播器与 Parareal 结合应用于方程 (1) 时，如果参数不同于 PINN 的训练参数，其收敛情况如何。随着参数与训练值的差异越来越大，粗传播器的准确性会降低。然而，只要 Parareal 收敛，就会得到正确解，因为数值细传播器始终使用正确的参数。Parareal 与 PINN 的结合具有良好的泛化能力。即使参数比训练值大十倍以上，也只需额外迭代一次即可收敛。虽然这次额外迭代会在一定程度上降低式 (8) 所示的可实现加速比，但下文给出的性能结果对模型参数的变化不应过于敏感。

Parareal 运行时间与加速比。报告的运行时间使用 Linux 中的 `time` 命令测量，并包含设置、计算和数据传输所需的时间。表 2 展示了使用 P =16 个核心、采用四种不同粗传播器配置时 Parareal 的运行时间（毫秒）。所示数据为五次运行的平均值及标准差。将数值粗传播器替换为 CPU 上的物理信息神经网络，可使 Parareal 的执行时间缩短至原来的 1/2。4；若在 GPU 上运行该物理信息神经网络，则可缩短至原来的 1/2。9。对于数值粗传播器，使用 GPU 并无性能提升，因为分辨率

训练轮次	到期	边界	残差
0	9.21×10^{2}	9.21×10^{2}	7.33×10^{3}
2000	5.58×10^{-1}	3.45×10^{-2}	2.50×10^{-2}
4000	4.11×10^{-2}	2.34×10^{-2}	5.00×10^{-3}
5000	5.92×10^{-1}	1.34×10^{-2}	4.22×10^{-3}
5300	4.19×10^{-2}	3.22×10^{-3}	1.94×10^{-4}
5500	6.46×10^{-4}	1.96×10^{-4}	5.73×10^{-5}
5800	2.92×10^{-5}	1.14×10^{-5}	3.19×10^{-4}

表1：网络训练过程中损失函数的演变。三列分别显示了与终端条件 (2)、边界条件 (3) 和 (4) 以及残差 (1) 相关的三个损失函数项的均方误差。以训练率 10^{-2} 训练 5000 轮后，又以降低后的训练率 10^{-3} 进行了 800 轮训练。

因此计算强度不够高。物理信息神经网络所提供的粗传播器速度快得多，显著减轻了 Parareal 中的串行瓶颈，并且如下文所示，将显著提升加速比。

	数值计算	PINN	相较于 CPU 数值计算的加速比
CPU	3.48 ± 0.056	1.47 ± 0.073	2.4
GPU	3.99 ± 0.651	1.21 ± 0.041	2.9
加速比	–	1.21	

表 2：粗传播器 $\mathcal{C}$ 的运行时间 c_{c} （毫秒），为五次运行的平均值加减标准差。

表 3 展示了完整 Parareal 迭代运行时间的五次运行平均值。最快的配置是在 CPU 上运行数值精细传播器，并在 GPU 上运行 PINN 粗传播器。将细传播器和粗传播器都置于 CPU 上执行，耗时约为其三倍。值得注意的是，将二者都移至 GPU 上运行，虽然比全部在 CPU 上运行略快，但仍比混合配置慢约两倍。当细传播器和粗传播器的分辨率都极高时，完整的 GPU 方案最终将会更快。然而，目前的分辨率已经产生了约为

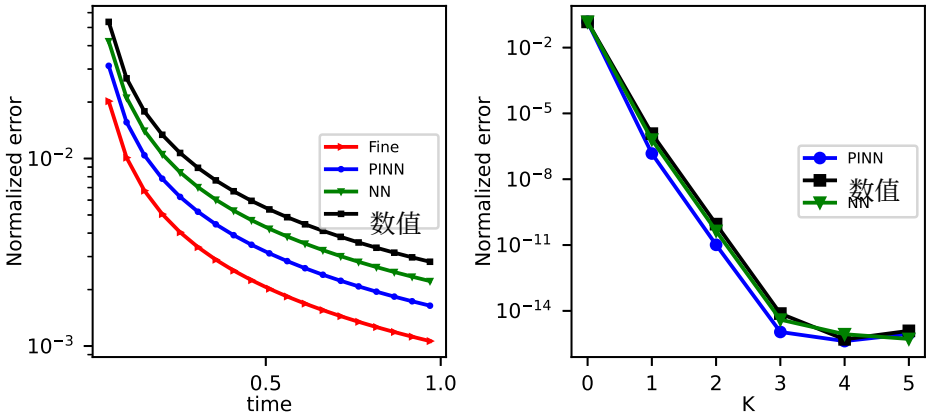

图3：粗传播器和细传播器相对于解析解的归一化 ℓ_2误差随时间的变化（左）。三种不同 Parareal 变体相对于串行细解的归一化 ℓ_2误差与迭代次数的关系（右）。黑线（方形）表示采用数值粗传播器的 Parareal，绿线（菱形）表示采用仅基于数据训练的神经网络作为粗传播器的 Parareal，蓝线（圆形）表示采用 PINN 作为粗传播器的 Parareal；后者还在损失函数中使用了微分方程的各项。所有情况下，Parareal 均使用 $P = 16$ 个时间片。

的误差，这在大多数情况下已经足够。这说明，在 Parareal 中结合数值方法和 PINN，不仅可以通过降低 PINN 的成本来提升性能，还能更好地利用同时配备 CPU 、 GPU 甚至神经网络加速器的节点。因此，可以将有限差分数值方法与神经网络不同的计算模式转化为优势。

	CPU-粗传播器	**GPU-粗传播器**
CPU-细致	128.48 ± 0.715	41.241970 ± 0.334
GPU-细致	83.2545 ± 0.356	87.45234 ± 0.253

表 3：Parareal 的运行时间（毫秒），为五次运行的平均值加减标准差。

图 5 展示了采用 PINN 和数值粗传播器的 Parareal 在 CPU（左）和 GPU（右）上的运行时间随核心数/时间片数 P 的变化。数值精细传播器始终在 CPU 上运行。在这两种

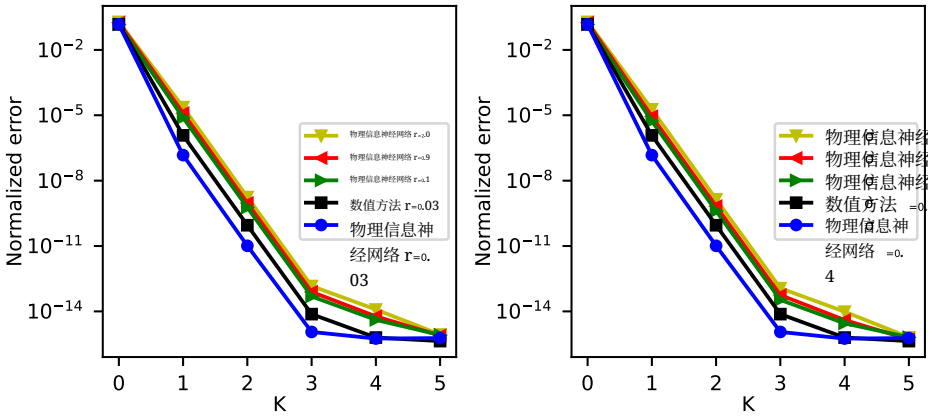

图 4：不同利率 r （左）和波动率σ （右）下 Parareal 的收敛情况。所有情况下，粗传播器均为针对$r = 0.03$ 和 $\sigma = 0.4$取值训练的物理信息神经网络。即使参数值比物理信息神经网络的训练值大十倍以上，Parareal 也只需额外迭代一次，就能收敛到与精细积分器的结果达到机器精度一致。

种情况下，随着时间片数/核心数 P 的增加，运行时间都以相似的速率下降。数值粗传播器始终比 PINN 慢，而且在 CPU 和 GPU 上的差距相近。最后，图 6 展示了采用数值、PINN-CPU 和 PINN-GPU 粗传播器的 Parareal 的加速比（左）和并行效率（右）。加速比界限 (8) 以线条表示。从数值粗传播器改用 PINN，以及将 PINN 从 CPU 移至 GPU，都能显著提升加速比。采用数值粗传播器时，Parareal 的加速比约为 $S(16) \approx 2$。将数值积分器替换为 PINN 后，加速比提升至 $S(16) \approx 3$。在 GPU 上运行该 PINN 后，加速比进一步提升至 $S(16) \approx 4.5$，是采用 CPU 上的数值粗传播器时的两倍多。加速比的提升也带来了并行效率的提高：从采用数值粗传播器时的约 30%，提升至采用 PINN-GPU 粗方法时的约 60%。处理器数量较少时，加速比的提升不那么明显，因为 (8) 中的 K/P 项影响更大。但从 $P = 2$ 个核心到 $P = 16$ 个核心，并行效率的提升幅度相当稳定。总之，这表明用在 GPU 上运行的 PINN 替换在 CPU 上运行的数值粗传播器，可以通过最大限度地降低粗传播器带来的串行瓶颈，大幅提升 Parareal 的性能。

5 讨论

Parareal 是一种时间并行方法，在廉价的粗传播器和可并行但昂贵的精细积分器之间进行迭代。为保持因果性，粗传播器

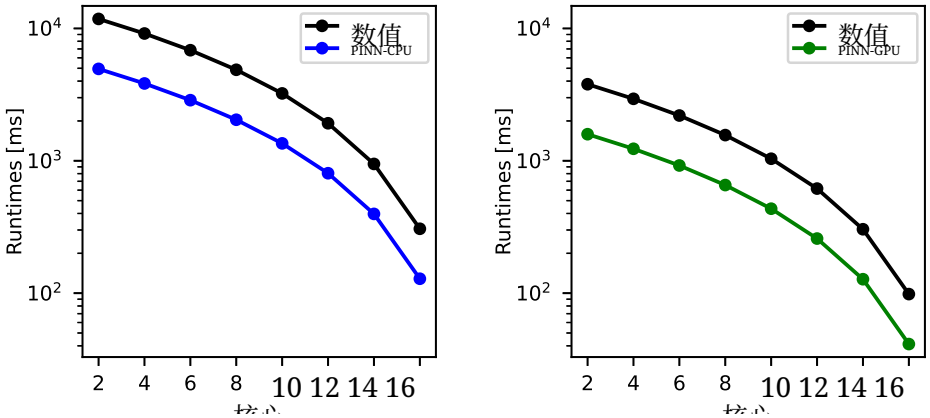

图5：Parareal（点）和串行数值精细传播器（水平线）在 CPU（左）和 GPU（右）上的运行时间（单位：毫秒）

需要串行运行，因此形成了限制可实现加速比的瓶颈。通常，粗传播器与精细传播器相似，使用数值方法构建，但采用更低阶数、更低分辨率，或在某些情况下使用简化复杂度模型。我们改为研究使用物理信息神经网络（PINN）。结果表明，PINN 的精度略高于数值粗传播器，但速度快了三倍。使用 PINN 不会影响 Parareal 的收敛速度，却能大幅减少粗传播器造成的串行瓶颈。

我们展示了，在配备一块 GPU 的单节点上，将运行于 CPU 的数值精细传播器与运行于 GPU 的 PINN 粗传播器相结合，相较于采用运行于 CPU 的数值粗传播器的原始 Parareal，可获得两倍以上的加速比。此外，我们证明，将精细传播器和粗传播器都迁移到 GPU 上，比仅将 PINN 粗方法迁移到 GPU、而将数值精细方法保留在 CPU 上更慢。其原因在于，除非精细传播器的分辨率极高，否则其较低的计算强度意味着在 GPU 上进行计算的收益很小，因此数据传输产生的开销占主导地位。相比之下，PINN 的评估非常适合在 GPU 上进行计算。我们的结果表明，利用 PINN 为时间并行方法构建粗层模型，是减少因果性所造成的串行瓶颈的一种有前景的方法。结果还表明，结合数值算法与神经网络的时间并行方法，可能有助于更好地利用异构系统。

参考文献

1. Agboh, W., Grainger, O., Ruprecht, D., Dogar, M.：用于机器人操作的学习型粗模型 Parareal。《计算与科学可视化》**23(8)**（2020）

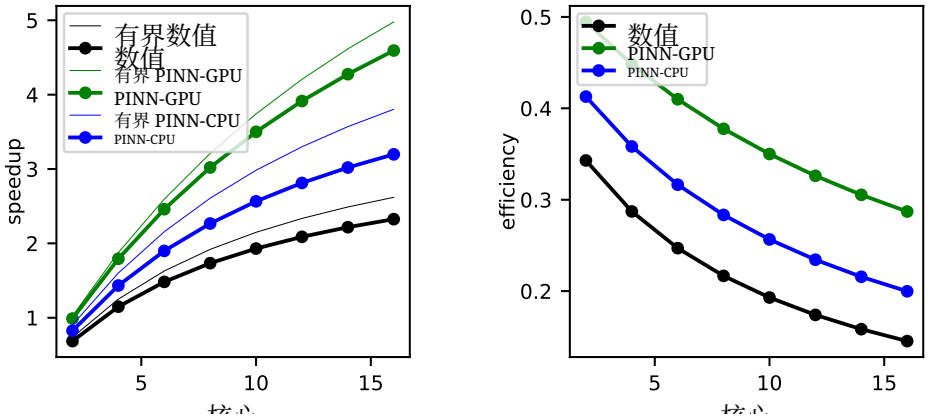

图6：Parareal 相对于 CPU 上串行数值精细传播器的加速比（左）和并行效率（右）。由于 PINN-GPU 粗传播器速度更快，它降低了 Parareal 的串行瓶颈，从而实现更高的加速比和并行效率。

2. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.：机器学习中的自动微分：综述。《机器学习研究杂志》**18**，1–43（2018） 3. Black, F., Scholes, M.：期权和公司负债的定价。《政治经济学杂志》**81**(3)，637–654（1973） 4. Emmett, M., Minion, M.L.：面向偏微分方程的高效时间并行方法。应用数学与计算科学通讯 **7**, 105 132 2012 https://doi.org/10.2140/camcos.2012.7.105 – 5 （）。. Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.：基于多重网格的时间并行积分。《SIAM科学计算杂志》**36**，C635–C 661 2014 https://doi.org/10.1137/130944230 （）。. Gorynina, O., 6 Legoll, F., Lelievre, T., Perez, D.：将机器学习力场与经验力场结合到 Parareal算法中：应用于原子尺度缺陷的扩散（2022） 7. Günther, S., Ruthotto, L., Schroder, J.B., Cyr, E.C., Gauger, N.R.：深度残差神经网络的层并行训练。《SIAM数据科学数学杂志》 **2** 1 1 23 2020 https://doi.org/10.1137/19M1247620 （），–（）。. 8 He, K., Zhang, X., Ren, S., Sun, J.：深入研究整流器：在 ImageNet 分类上超越人类水平的表现。载于：IEEE国际计算机视觉会议论文集。第1026–1034页（2015） 9. Huang, R., Li, R., Xi, Y.：通过神经网络学习最优多重网格平滑器。《SIAM科学计算杂志》第S199–S225页（2022）。 https://doi.org/10.1137/21M1430030 . 10 Kingma, D.P., Ba, J.：Adam：一种随机优化方法（2014）。 https://doi.org/10.48550/arXiv.1412.6980 . 11 . Kirby, A., Samsi, S., Jones, M., Reuther, A., Kepner, J., Gadepally, V.：利用GPU并发，通过非线性多重网格实现深度残差神经网络的层并行训练。载于：2020年IEEE高性能极限计算会议（HPEC）。I EEE（2020）。 https://doi.org/10.1109/hpec43674.2020.9286180

12. Kumar, S., Yildirim, A., Khan, Y., Jafari, H., Sayevand, K., Wei, L.：利用拉普拉斯变换求解分数阶 Black-Scholes 欧洲期权定价方程的解析解。《分数阶微积分与应用期刊》**2**(8), 1–9 (2012) 13. Lee, Y., Park, J., Lee, C.O.：模拟时间并行算法的 Parareal 神经网络。《IEEE 神经网络与学习系统汇刊》pp. 1–12 (2022 https://doi.org/10.1109/tnnls.2022.3206797). 14. Lions, J.L., Maday, Y., Turinici, G.：偏微分方程的“parareal”时间离散化。《法国科学院院报：数学系列》**332**, 661–668 (2001 https://doi.org/10.1016/S0764-4442(00)01793-6). 15. Lorin, E.：时间并行神经常微分方程的推导与分析。《数学与人工智能年刊》(2020). https://doi.org/10.1007/s10472-020-09702-6. 16. Meng, X., Li, Z., Zhang, D., Karniadakis, G.E.：PPINN：用于时变偏微分方程的 Parareal 物理信息神经网络。《计算机方法在应用力学与工程中的应用》**370**, 113250 (2020). https://doi.org/10.1016/j.cma.2020.113250 17. Nguyen, H., Tsai, R.：借助深度学习进行数值波传播。《计算物理学杂志》**475**, 111828 (2023年). https://doi.org/10.1016/j.jcp.2022.111828 18. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.：Pytorch：一种命令式、高性能深度学习库 (2019). https://doi.org/10.48550/arXiv.1912.01703. 19. Qadir, I.A., Götschel, S., Ruprecht, D.：采用物理信息神经网络作为粗传播器的 Parareal (2023 https://doi.org/10.5281/ZENODO.7695133). 20. Raissi, M., Perdikaris, P., Karniadakis, G.E.：物理信息深度学习（第一部分）：非线性偏微分方程的数据驱动解 (2017). https://doi.org/10.48550/arXiv.1711.10561. 21. Ranade, R., Hill, C., Pathak, J.：Discretizationnet：一种使用有限体积离散化求解纳维–斯托克斯方程的机器学习求解器。《计算机方法在应用力学与工程中的应用》**378**, 113722 (2021). https://doi.org/10.1016/j.cma.2021.113722. 22. Ruprecht, D.：共享内存流水线 Parareal，第 669–681 页。Springer International Publishing (2017 https://doi.org/10.1007/978-3-319-64203-1_48). 23. Sirignano, J., Spiliopoulos, K.：DGM：一种用于求解偏微分方程的深度学习算法。《计算物理学杂志》**375**, 1339–1364 (2018) 24. Speck, R., Ruprecht, D., Krause, R., Emmett, M., Minion, M.L., Winkel, M., Gibbon, P.：大规模时空并行 N 体求解器。载于：《国际高性能计算、网络、存储与分析会议论文集》。第 92:1–92:11 页。SC ’12，IEEE 计算机学会出版社，洛斯阿拉米托斯，加利福尼亚州，美国 (2012 https://doi.org/10.1109/SC.2012.6). 25. Stender, M., Ohlsen, J., Geisler, H., Chabchoub, A., Hoffmann, N., Schlaefer, A.：Up-Net：一种用于参数化时空动力学的通用基于深度学习的时间步进器。可见于 SSRN 4053304 (2022) 26. Yalla, G.R., Engquist, B.：使用插值和神经网络求解多尺度动力系统的时间并行算法。载于：《高性能计算研讨会论文集》。第 9:1–9:12 页。HPC ’18，国际计算机仿真学会 (2018)