学习面向多分辨率物理的可控自适应仿真

Tailin Wu[1*], **Takashi Maruyama**[1, 2*], **Qingqing Zhao**[1*], **Gordon Wetzstein**[1], **Jure Leskovec**[1]斯坦福大学, [2]NEC公司 tailin@cs.stanford.edu,49takashi@nec.com,cyanzhao@stanford.edu, gordonwz@stanford.edu,jure@cs.stanford.edu

arXiv:2305.01122v1 [cs.LG] 1 May 2023

摘要

模拟物理系统的时间演化对许多科学与工程问题至关重要。模拟此类系统面临的一项开放挑战是其多分辨率动力学：系统中一小部分区域的动态变化极为剧烈，需要非常精细的分辨率；而系统的大部分区域变化缓慢，可以采用更粗的空间尺度建模。典型的基于学习的代理模型采用统一的空间尺度，因此必须达到所需的最精细尺度，这可能会耗费大量计算量才能实现所需精度。在本研究中，我们提出面向多分辨率物理的可控自适应仿真（LAMP），这是首个能够联合学习演化模型并优化合适空间分辨率、将更多计算量分配给动态性极强区域的完整深度学习代理模型。LAMP由一个用于学习正向演化的图神经网络（GNN），以及一个基于GNN的演员-评论家组成，后者用于学习空间细化与粗化策略。我们提出了学习技术，通过以误差和计算成本的加权和为目标函数来优化LAMP，使其能够在推理时适应误差与计算权衡之间相对重要性的变化。我们在一维非线性偏微分方程基准测试和一项具有挑战性的二维基于网格的模拟中评估了该方法。结果表明，LAMP优于最先进的深度学习代理模型，并能通过自适应权衡计算量来改善长期预测误差：在一维非线性偏微分方程中，平均误差降低33.7%；在二维基于网格的模拟中，性能优于MeshGraphNets和经典自适应网格细化（AMR）。包含数据和代码的项目网站位于：http://snap.stanford.edu/lamp。

1 引言

模拟物理系统随时间的演化对科学与工程至关重要（Lynch，2008；Carpanese，2021；Sircombe 等人，2006；Courant 等人，1967；Lelievre 与 Stoltz，2016）。通常，物理系统具有多分辨率特性：系统中一小部分区域动态变化剧烈，需要非常精细的分辨率才能准确模拟；而系统的大部分区域变化缓慢。相关示例包括天气预报中的灾害预测（Majumdar 等人，2021）、核聚变中等离子体流体的破坏性不稳定性（Kates-Harbeck 等人，2019）、喷气发动机设计中边界附近的空气动力学（Athanasopoulos 等人，2009），以及更常见的示例，如布料上的褶皱（Pfaff 等人，2021）和流经圆柱体时边界附近的流体（Vlachas 等人，2022）。鉴于此类系统通常规模庞大，至关重要的是，模拟这些系统不仅要准确，还要尽可能降低计算成本。采用统一的空间分辨率，对动力学特性迥异的区域给予相近的关注，会在变化缓慢的区域浪费大量计算资源，同时可能无法充分模拟动态区域。

为了加速物理模拟，近年来，基于深度学习（DL）的代理模型已成为一种颇具前景的替代方案，可用于辅助经典求解器（Um et al., 2020）或取代经典求解器（Li et al., 2021）。这些模型能够减少计算量，并通过提高空间分辨率（Um

*同等贡献。

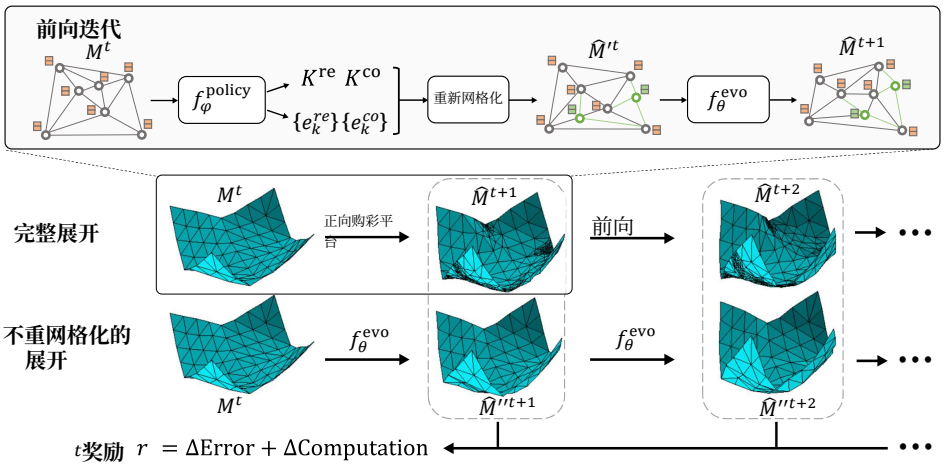


图 1：LAMP 示意图。前向迭代（上方方框）首先使用策略 $f_\varphi^{\text{policy}}$来决定边的数量 K^{re} 和 K^{co}，以及完整网格中哪些边需要细化或粗化，然后执行重新网格化和插值。演化模型 f_θ^{evo}应用于更新后的网格 $\hat{}M'^t$，以预测下一个时间步的状态 $\hat{}M^{t+1}$。与不进行重新网格化的多步展开相比，我们将误差和计算量（网格大小）的降低作为奖励来学习策略。更多详情请参见第 3.2 节。

et al., 2020; Kochkov et al., 2021）或时间分辨率（Li et al., 2021），或借助潜在表征（Sanchez-Gonzalez et al., 2020; Wu et al., 2022）来加速仿真。然而，当前基于深度学习的代理模型通常假设空间分辨率均匀或固定，却没有学习 如何将计算量最佳地分配给最需要的空间区域。因此，它们可能不足以应对上述多分辨率挑战。尽管经典求解器已有自适应网格细化（AMR）等自适应方法（Soner et al., 2003; Cerveny et al., 2019），但这些方法也面临与经典求解器相似的挑战（例如，速度慢）。因此，需要一种基于深度学习的代理模型，既能学习演化过程，也能学会将计算量分配到所需区域。

在本研究中，我们提出了可控自适应多分辨率物理模拟学习（LAMP），这是首个完全基于深度学习的代理模型，能够联合学习演化模型，并优化适当的空间分辨率，将更多计算资源分配给动态性较高的区域。我们的核心洞见是，将误差和计算量明确设为联合优化目标后，模型便能学会自适应地确定演化系统所需的最佳局部空间分辨率。为实现这一目标，LAMP 包含一个基于图神经网络（GNN）的演化模型，用于学习前向演化；以及一个基于 GNN 的演员-评论家，用于学习局部细化和粗化空间网格的离散动作策略，该策略以局部状态和系数 β 为条件，其中该系数用于权衡误差与计算量的相对重要性。策略（演员）既输出细化和粗化动作的数量，也输出要细化或粗化的边；评论家则评估当前策略的期望奖励。整个系统采用交替方式训练，在使用监督损失训练演化模型与通过强化学习（RL）训练演员-评论家之间迭代进行。总体而言，单个演化模型与演员-评论家实例可协同优化物理模拟中的误差和计算量，并能在推理时适应这两项指标的相对重要性。

我们在非线性偏微分方程的一维基准测试（用于检验模型在同一族偏微分方程之间的泛化能力）以及具有挑战性的折纸二维基于网格的模拟上评估了模型。在一维场景中，我们证明，与最先进的基于深度学习的代理模型相比，我们的模型在长期演化误差方面降低了 33.7%，并且能够通过自适应权衡计算量来改善长期预测误差。在二维基于网格的模拟中，与最先进的 Mesh-GraphNets + 经典自适应网格细化（AMR）相比，我们的模型表现更优。

2 问题设定与相关工作

我们考虑物理系统的数值仿真，并沿用 (Pfaff 等人，2021) 中引入的记号。时间 t 的系统状态被离散化为基于网格的状态 $M^t =$

(V^t, E^t) $t = 0$, 1, 2, ..., 其中 E^t 是网格的边, V^t 是时间t各节点上的状态。每个节点 $i \in V$ 包含网格空间坐标 u_i 和动态特征 q_i。请注意，这种物理系统表示方式非常通用。它涵盖欧拉系统（Wu 等人，2022），其中网格固定而节点上的场 q_i 不断变化；也涵盖拉格朗日系统（Sanchez-Gonzalez 等人，2020；Pfaff 等人，2021），其中物理空间中的网格坐标也在动态移动（对于这种情况，网格坐标 u_i还会附带一个额外的世界坐标 x_i ）。在预测过程中，仿真器 f（经典或学习得到的）会根据前一时间步的预测 $\hat{}M^t$，自回归地预测下一时间步的系统状态 $\hat{}M^{t+1}$：

$$\hat{M}^{t+1} = f(\hat{M}^t), t = 0, 1, 2, ... \tag{1}$$

其中，$\hat{}M^0 = M^0$为初始状态。在预测过程中，网格节点处的动态特征 V^t以及网格拓扑 E^t 都可能发生变化。通常通过比较长期预测结果与真实值来计算误差：对于度量指标 ℓ（例如，MSE、RMSE），误差为：$= \ell(\hat{}M^t, M^t)$；计算成本（以浮点运算次数或 FLOPs 衡量）通常随网格规模（例如，节点数量）增长。任务是以较低误差并满足计算成本约束，将系统状态向未来长期演化。

大多数经典求解器都使用固定网格 E^t，其拓扑不会随时间变化。例如，网格 $E^t \equiv E^0$ 可以是二维或三维规则网格，也可以是在仿真开始时预先生成的不规则网格（Geuzaine & Remacle，2009）。经典的自适应网格细化（AMR）（Narain 等人，2012）通过根据局部状态变化的启发式规则，自适应地细化或粗化网格分辨率，以应对多分辨率挑战。由于这些方法基于经典求解器，因此可能无法充分发挥深度学习带来的诸多优势（GPU 加速、对空间和时间分辨率的要求较低、显式前向计算等）。相比之下，我们的 LAMP 是一种基于深度学习的替代模型，能够受益于深度学习框架带来的诸多优势（例如，加速）。此外，由于它直接针对误差与计算量的线性组合进行优化，因此有望直接优化误差与计算量之间的权衡，使其更接近真实的帕累托前沿。

基于深度学习的代理模型虽然相较于经典求解器实现了加速，但通常仍在固定网格上运行（Li 等人，2021；Sanchez 等人，2020；Wu 等人，2022；Zhao 等人，2022；Han 等人，2022），尚未利用物理模拟中典型的多分辨率特性。一个重要的例外是 MeshGraphNets（Pfaff 等人，2021），它既学习如何演化状态 V^t，又通过监督学习来学习改变 E^t 的空间自适应策略。然而，由于它采用监督学习，且真实网格由使用 AMR 的经典求解器提供，因此在误差与计算量之间的权衡方面无法超越 AMR，并且推理时必须与经典求解器交互，才能执行自适应网格细化。相比之下，我们的 LAMP 直接针对目标函数进行优化，利用强化学习来学习细化和粗化策略，有望超越经典 AMR，实现更优的误差与计算量权衡。此外，单个训练好的 LAMP 在推理时能够适应误差与计算量相对重要性 β 的完整范围，因此比采用固定策略的 MeshGraphNets 更灵活。Yang 等人（2021）的另一项开创性工作使用强化学习来学习自适应重新网格化，其与我们的方法有显著区别。首先，他们的方法针对特定的有限元方法（FEM）进行评估，无法直接应用于更通用的仿真，例如我们实验中的布料模拟。除此之外，我们的方法是首个联合学习重新网格化与演化的方法。其次，Yang 等人（2021）的目标是降低误差，而我们的目标是学习一种可控的权衡，在降低误差和降低计算成本之间进行取舍。第三，Yang 等人（2021）的动作是在矩形网格的面 上进行细化，而我们的动作是在三角网格的边 上进行细化和粗化。第四，LAMP 无需将经典求解器纳入循环，因此能显著缩短训练时间。

3 方法

本节将详细介绍我们的 LAMP 方法。我们首先在第 3.1 节介绍其架构。随后介绍其学习方法（第 3.2 节），包括学习目标函数和训练，以及使其能够学会适应误差和计算量重要性变化的技术。LAMP 的高层示意图如图 1 所示。

3.1 模型架构

LAMP 的模型架构由两个部分组成：一个用于更新网格拓扑的演员-评论家，以及一个用于演化定义在网格上的状态的演化模型。下面将逐一详细介绍。

演员-评论家。演员-评论家由一个策略网络 $f_\varphi^{\text{policy}}$（参数为 φ）组成，该网络用于预测执行空间粗化或细化操作的概率；还包括一个价值网络 f_φ^{value}，用于评估策略网络的长期期望奖励：

$$P(A = a^t | M = M^t, \beta) = p_\varphi(a^t | M^t, \beta) = f_\varphi^{\text{policy}}(M^t, \beta) \tag{2}$$

$$\hat{v}^t = f_\varphi^{\text{value}}(M^t, \beta) \tag{3}$$

其中，a^t 是在边 E^t 上执行的（细化和粗化）动作，使其变为 $\hat{E}^{t+1}$。策略网络 $f_\varphi^{\text{policy}}$ 输出执行此类动作的概率，并可从该概率分布中采样。v^t 用于估计当前状态 M^t 下当前策略的“价值”（更多信息见下文第 3.2 节）。显式地依赖于 β（如公式 8 中的 β），使策略网络和值网络能够根据误差和计算量的重要性变化进行条件化。给定预测网格 $\hat{E}^{t+1}$、当前网格 E^t 上的当前节点特征 V^t，将执行插值 g^{interp}，以获得新网格上的节点特征（详情见附录 B）：

$$\hat{V}'^t = g^{\text{interp}}(V^t, \hat{E}^{t+1}, E^t) \tag{4}$$

现在，新的中间状态 $\hat{M}'^t = (\hat{V}'^t, \hat{E}^{t+1})$ 定义在新的网格 $\hat{E}^{t+1}$ 上。

演化模型。第二个组件是演化模型 f_θ^{evo}，它以定义在 $\hat{E}^{t+1}$ 上的中间状态 M'^t 为输入，并输出时间 $t+1$ 的节点特征预测 $\hat{V}^{t+1}$：

$$\hat{V}^{t+1} = f_\theta^{\text{evo}}(M'^t) \tag{5}$$

请注意，在此阶段，网格拓扑 $\hat{E}^{t+1}$ 保持不变，演化模型 f_θ^{evo}（参数为 θ）学习基于当前网格预测状态。

综合起来，公式 (2)(4)(5) 将系统状态从时间 t 的 M^t 演化为时间 $t+1$ 的状态 $\hat{M}^{t+1} = (\hat{V}^{t+1}, \hat{E}^{t+1})$。推理时，按照公式 (1) 自回归地执行这些公式，以给定的初始状态 M^0 预测系统的未来状态 $\hat{M}^t$，$t = 1$，2，3，.....。

图神经网络架构。演化模型 f_θ^{evo}、策略网络 $f_\varphi^{\text{policy}}$ 和价值网络 f_φ^{value} 的一项要求是，它们能够处理不断变化的网格拓扑 E^t。图神经网络（GNN）是满足这一要求的理想选择。由于我们将系统状态表示为网格，因此采用 MeshGraphNets（Pfaff 等人，2021年）作为上述三个模型的基础架构。具体而言，我们将 V^t 编码为图的节点特征，将网格拓扑 E^t 和世界边分别编码为两类边，而边特征则取决于网格坐标和世界坐标中的相对位置。基于该图，由 N 层消息传递组成的处理器网络会执行局部信息交换与聚合：

$$Z_{ij}^{(e)n+1} = \text{MLP}_\theta^{(v)}(E_{ij}^n, Z_i^{(v)n}, Z_j^{(v)n}) \tag{6}$$

$$Z_i^{(v)n+1} = \text{MLP}_\theta^{(v)}(Z_i^{(v)n}, \sum_j Z_{ij}^{(e)n+1}). \tag{7}$$

其中，$Z_i^{(v)n}$ 是第 n层节点 i 上的潜在节点向量，$Z_{ij}^{(e)n}$ 是从节点 i 到节点 j的边上第 n^{th} 层的潜在边向量。在公式 (5) 中，$Z_i^{(v)0} = \hat{V}_i'^t$和 $Z_i^{(v)N} = \hat{V}_i^{t+1}$ 分别是时间 t 和 $t+1$的输入节点特征和预测节点特征。图 1 展示了该架构。我们为演化模型使用独立的处理器，并由策略网络和价值网络共享同一个处理器。经过处理器后，潜在向量与 β 拼接，并输入下游解码器。对于演化模型 f_θ^{evo}，解码器对潜在状态进行处理，并输出节点上的预测值 $\hat{V}'^{t+1}$。对于价值网络，价值多层感知机对所有节点进行处理，再通过全局池化计算整体估计价值。对于策略网络，我们按如下方式设计动作解码器。

动作表示。为了预测策略网络的动作及其概率，我们首先需要设计动作空间。请注意，与典型的强化学习问题相比，这里的动作空间维度极高且十分复杂：(1) 网格中的每条边都可以选择细化或粗化。如果有数千条边 N_{edge}，那么可能的动作数量将达到 $2^{N_{\text{edge}}}$量级。(2) 并非所有动作都有效，许多动作组合都是无效的。例如，网格同一面上的两条边不能同时细化，也不能同时粗化。为了解决这一高维动作问题，我们采用以下动作空间设计：对于细化和粗化，策略网络首先采样整数 K^{re}，$K^{\text{co}} \in \{0, 1, 2, ...K^{\max}\}$，然后分别独立采样 K^{re} 条边进行细化，并采样 K^{co} 条边进行粗化，同时进行适当的过滤。完整采样动作记为 $a^t = (K^{\text{re}}, e_1^{\text{re}}, e_2^{\text{re}}, ...e_{K^{\text{refine}}}^{\text{re}}, K^{\text{co}}, e_1^{\text{co}}, e_2^{\text{co}}, ...e_{K^{\text{co}}}^{\text{co}})$，其中 K^{re}，$K^{\text{co}} \in \{0, 1, ...K^{\max}\}$，且$e_k^{\text{re}}$，$e_k^{\text{co}} \in E^t$，$k = 1$，2，..。采样动作 a^t 的对数概率为：

$$\log p_\varphi(a^t|M^t) = \log p_\varphi(K^{\text{re}}|M^t) + \sum_{k=1}^{K^{\text{re}}} \log p_\varphi(e_k^{\text{re}}|M^t) + \log p_\varphi(K^{\text{co}}|M^t) + \sum_{k=1}^{K^{\text{co}}} \log p_\varphi(e_k^{\text{co}}|M^t)$$

3.2 学习

LAMP 学习的最终目标是优化目标函数 Eq. (8)，如下所示：

$$L = (1 - \beta) \cdot \text{Error} + \beta \cdot \text{Computation} \tag{8}$$

适用于广泛的 β。为实现这一目标，我们首先在不进行重新网格化的情况下预训练演化模型，以获得一个合理的演化模型；随后将上述目标分解为两个阶段的交替学习（附录 B.1）：以目标函数 L^{evo} 学习演化模型，从而最小化长期演化误差；以及以目标函数 L^{policy} 学习策略，同时优化长期演化误差和计算成本。

演化学习。在此阶段，优化演化模型 f_θ^{evo}，以降低该演化模型的多步演化误差 。与前文一样，我们将由真实值求解器使用非常精细的网格模拟得到的系统在时间 $t + s$ 的状态记为 M^{t+s}，$t = 0, 1, 2, ..., s = 0, 1, ...S$；将当前 LAMP 遵循当前策略、向未来预测至视界 S 步的结果记为 $\hat{} M^{t+s}, t = 0, 1, 2, ..., s = 0, 1, 2, ..S$。此外，我们将当前演化模型在精细网格上给出的预测记为 $\hat{} M''^{t+s}, t = 0, 1, 2, ..., s = 0, 1, 2, ..S$，其中每个时间步的网格均由真实网格 E^{t+s} 提供。于是，损失为：

$$L^{\text{evo}} = L_S^{\text{evo}}[f_\varphi^{\text{policy}}, f_\theta^{\text{evo}}; \hat{M}^t] + L_S^{\text{evo}}[\mathbb{I}, f_\theta^{\text{evo}}; \hat{M}''^t] \tag{9}$$

$$= \sum_{s=1}^{S} \alpha_s^{\text{policy}} \ell(\hat{M}^{l+s}, M^{l+s}) + \sum_{s=1}^{S} \alpha_s^{\mathbb{I}} \ell(\hat{M}''^{l+s}, M^{l+s}) \tag{10}$$

本质上，我们优化演化损失的两个部分：(1) $L_s^{\text{evo}}[f_{\text{策略}\varphi}$，$f_\theta^{\text{evo}}$；$\hat{} M^t]$ 这是遵循策略网络 $f_\varphi^{\text{policy}}$和演化模型 f_θ^{evo}，从初始状态 $\hat{} M^t$ 开始演化 S 步所得到的演化损失。（其中 α_s^{policy} 是使用损失函数 ℓ计算的 s步损失的系数。）这确保演化模型 f_θ^{evo}能够适应当前策略 $f_\varphi^{\text{policy}}$，以分配适当的计算量。(2) 损失的第二部分 $L_s^{\text{evo}}[\mathbb{I}, f_\theta^{\text{evo}}; \hat{} M''^t]$，是使用真实网格并由演化模型 f_θ^{evo}进行演化所得到的演化损失，演化从精细网格的初始状态 $\hat{} M''^t$开始，持续 s 步。如果网格由真实网格提供，这会促使演化模型学会利用更多计算量来获得更好的预测误差。

学习策略。在此阶段，策略网络 $f_\varphi^{\text{policy}}$学习在每个位置更新空间分辨率（对网格进行细化或粗化），以同时改善计算量 和预测误差。由于空间细化和粗化都是离散动作，且计算量指标通常不可微，因此我们使用强化学习（RL）来学习策略。具体而言，我们将其建模为马尔可夫决策过程（MDP）：环境状态为系统状态 M^t，动作是在 E^t 的每条边上进行局部细化或粗化；我们将奖励设计为改进 误差和计算量的程度，比较的是遵循当前策略的动作与反事实 情境下的结果，在后者中，智能体从初始状态 $\hat{}$ 开始，遵循恒等策略 $\mathbb{I}$

该策略不会更新网格拓扑。具体而言，奖励为

$$r^t = (1-\beta)\cdot \Delta\text{Error} + \beta \cdot \Delta\text{Computation} \tag{11}$$

$$\Delta\text{Error} = L_S^{\text{evo}}[\mathbb{I}, f_\theta^{\text{evo}}; \hat{M}^t] - L_S^{\text{evo}}[f_\varphi^{\text{policy}}, f_\theta^{\text{evo}}; \hat{M}^t] \tag{12}$$

$$\Delta\text{Computation} = \mathcal{C}_S[\mathbb{I}, f_\theta^{\text{evo}}; \hat{M}^t] - \mathcal{C}_S[f_\varphi^{\text{policy}}, f_\theta^{\text{evo}}; \hat{M}^t] \tag{13}$$

这里，$\mathcal{C}_S[\cdot]$ 是一个替代指标，用于根据预测的网格拓扑 $\hat{}E^{t+1}$、$\hat{}E^{t+2}$.....$\hat{}E^{t+S}$，量化未来 S 步以内的“计算量”。本文使用节点数量作为衡量计算量的替代指标，因为通常对于图神经网络，计算量（以浮点运算次数计）与节点数量呈线性关系（由于网格上每个节点的边数量有上限，因此边的数量也与节点数量呈线性关系，消息传递和节点更新亦然）。

为了优化奖励 r^t，我们采用 Sutton 等人（1999年；Hafner 等人，2021年）使用的标准 REINFORCE 方法来更新策略网络 $f_\varphi^{\text{policy}}$，其目标函数如下：

$$L_\beta^{\text{actor}} = \mathbb{E}_t\left[-\log p_\varphi(a^t|M^t,\beta)\text{sg}(r^t - f_\varphi^{\text{value}}(M^t,\beta)) - \eta\cdot \text{H}[p_\varphi(a^t|M^t,\beta)]\right] \tag{14}$$

这里，H[·] 是熵，它鼓励动作具有更高的熵以增加探索，其中 η 是一个超参数。sg(·)表示停止梯度。本质上，损失 L^{policy}中的第一项鼓励提高具有较高“优势”的动作的对数概率，其中优势定义为当前动作 a^t执行后得到的当前奖励 r^t与期望奖励（价值） $f_\varphi^{\text{value}}(M^t,\ \beta)$之间的差值；后者是从当前状态 M^t开始、遵循当前策略所得到的期望奖励。我们也可以将其视为演员-评论家架构：评论家试图准确评估当前策略的期望奖励，而演员（策略）则试图超越这一期望。训练价值网络时，我们使用均方误差损失：

$$L_\beta^{\text{value}} = \mathbb{E}_t\left[(f_\varphi^{\text{value}}(M^t,\beta) - r^t)^2\right] \tag{15}$$

学习适应不断变化的 β。在总体目标函数（公式 8）中，β 规定了误差与计算量的相对重要性。$\beta = 0$ 意味着我们只关注最小化误差，不对计算量加以限制。$\beta = 1$ 意味着我们只关注最小化计算量，而不考虑演化误差。实际上，我们通常希望两者都得到改善，因此采用一个 a$\beta \in$ (0，1) 的值，侧重于其中一个指标，同时兼顾另一个指标。为了使 LAMP 能够在推理时适应不同的 β，在策略学习过程中，我们针对小批量中的不同样本，从范围 $\mathcal{B} \subseteq [0,1]$（例如，$\mathcal{B}$可以是 $[0,1]$ 或 $[0,0.5]$）内均匀采样β，并联合训练策略网络和价值网络，其中总损失 L^{policy} 是策略损失与价值损失的加权和：

$$L^{\text{policy}} = \mathbb{E}_{\beta\sim\mathcal{B}}[L_\beta^{\text{actor}} + \alpha^{\text{value}}\cdot L_\beta^{\text{value}}] \tag{16}$$

其中，α^{value} 是一个超参数，我们将其设为 0.5。这样，策略就能学习一种通用的空间粗化与细化方法，并以 β为条件。例如，对于更小的 β（更侧重于改善误差），策略网络可能会学习在动态区域进行更多细化、减少粗化，以牺牲计算量为代价来改善预测误差。

4 实验

在实验中，我们旨在回答关于所提出的 LAMP 的以下问题：

- LAMP 能否学会粗化和细化网格，将更多计算量集中于动态变化更剧烈的区域，从而提高预测准确性?
- 与最先进的深度学习代理模型相比，LAMP 能否改善误差与计算量之间的帕累托前沿?
- LAMP 能否学会根据 β 进行条件控制以改变其行为，并根据 β执行不同程度的细化和粗化?

我们在两个具有挑战性的数据集上评估 LAMP：（1）一维基准非线性偏微分方程，用于测试模型在同一族偏微分方程中的泛化能力（Brandstetter 等人，2022）；（2）由 ArcSim 求解器生成的基于网格的纸张仿真（Narain 等人，2012）。这两个数据集都具有多分辨率特征：系统的某些部分动态变化剧烈，而其他部分则变化得较为缓慢。

4.1 一维非线性偏微分方程族

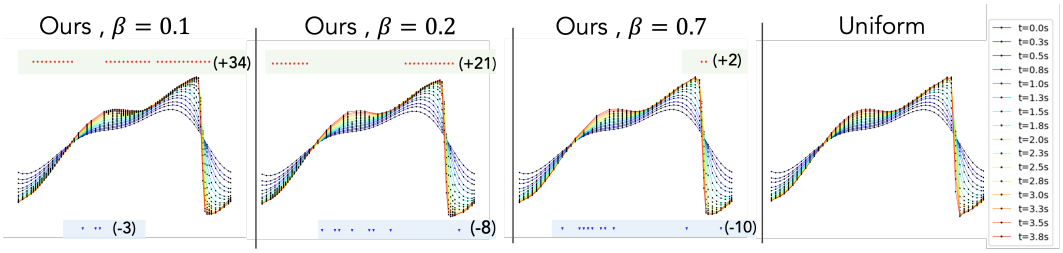


图 2：LAMP 在一维非线性偏微分方程上的滚动演化结果示例。滚动演化持续 200 个时间步，不同颜色表示系统在不同时间的状态。对于每个状态，我们还用黑点绘出了相应的网格。上方的绿色条带和下方的蓝色条带分别显示了网格新增和移除的节点，即末尾网格与初始网格之间的差异。可以看到，当 β 较小（例如，$\beta = 0.1$）时，模型更注重“误差”，会在动态程度较高的区域（激波前沿附近）进行更多细化，并减少粗化。当 β 较大（例如，$\beta = 0.7$）时，模型更侧重于降低计算量，几乎不进行细化，而选择在较静态的区域进行更多粗化。

数据与实验。在本节中，我们使用给定方程族中具有不同参数的未见方程，测试 LAMP 平衡误差与计算量的能力。我们采用 Brandstetter 等人（2022）中的一维基准，其中偏微分方程如下：

$$\left[\partial_t u + \partial_x(\alpha u^2 - \beta \partial_x u + \gamma \partial_{xx} u)\right](t,x) = \delta(t,x) \tag{17}$$

$$u(0,x) = \delta(0,x), \quad \delta(t,x) = \sum_{j=1}^{J} A_j \sin(\omega_j t + 2\pi \ell_j x / L + \phi_j) \tag{18}$$

偏微分方程的参数由 $p = (\alpha,\ \beta,\ \gamma)$ 给出。项 δ 是一个强迫项（Bar-Sinai 等，2019），其中 $J = 5$，$L = 16$，系数 A_j 和 ω_j 从 $A_j \sim U[-0.5, 0.5]$、$\omega_j \sim U[-0.4, 0.4]$、$\ell_j \in \{1,2,3\}$、$\phi_j \sim U[0, 2\pi)$ 中均匀采样。我们将空间均匀离散为 $n_x = 200$ 个点（位于 $[0,16)$），并将时间均匀离散为 $[0,4]$ 中的 $n_t = 250$ 个点。随后将空间和时间进一步下采样至分辨率 $(n_t,\ n_x) \in \{(250,\ 100)、(250,\ 50)、(250,\ 25)\}$，作为初始分辨率。我们采用基准测试中的**E2** 场景，该场景用于测试模型能否泛化到同一族偏微分方程中的新参数。具体而言，参数为 $p = (1,\ \eta,\ 0)$，其中 $\eta \sim U[0, 0.2]$。

由于 LAMP 以自回归方式模拟系统，因此它可以通过策略网络 $f_\varphi^{\text{policy}}$ 在合适的位置细化或粗化网格，然后使用演化模型f_θ^{evo}演化到下一个状态。我们使用计算量与长期演化误差这一指标评估模型。对于计算量，我们使用完整轨迹中的平均顶点数作为替代指标，因为浮点运算次数通常与网格中的顶点数呈线性关系。对于长期演化误差，我们使用滚动演化 200 步的累积均方误差，初始状态取自时间步 25 至 49。我们将 LAMP 与强基线方法进行比较，包括基于深度学习的代理模型，如卷积神经网络、傅里叶神经算子（FNO）（Li 等人，2021）以及 MP-PDE（Brandstetter 等人，2022）；后者是针对该任务的当前最先进的基于深度学习的代理模型。我们的基础神经网络架构基于 MeshGraphNets（Pfaff 等人，2021），这是用于基于网格仿真的当前最先进的基于图神经网络的模型。我们还比较了不执行重新网格化的模型消融版本（LAMP 不进行重新网格化）以及完整版本。对于所有模型，我们都以自回归方式滚动演化，预测完整 200 步轨迹中的状态，并将前 25 步作为初始步。我们进行了三组实验，初始顶点数分别为 25、50 和 100，均由 100 顶点网格下采样得到。三个基线方法都不执行重新网格化，而我们的完整模型能够在每个时间步重新网格化，以粗化或细化边。我们记录累积均方误差作为误差指标，并以完整滚动轨迹中的平均顶点数作为计算成本指标。请注意，对于所有模型，均方误差都是在具有 100 个顶点的完整真实网格上计算的，其中预测值通过线性插值映射到真实网格的位置。这样可以防止模型通过减少顶点数、只在这些顶点上做好预测来“作弊”。实验的更多细节见附录 B.2。

表1：一维非线性偏微分方程系列的结果（误差和计算成本），滚动轨迹总计 200 个时间步，并将前 25 步作为输入。与基线方法相比，LAMP 显著降低了长期预测误差；在初始顶点数较少时，效果尤为明显（初始顶点数为 25 时误差降低 71.5%，为 50 时降低 23.5%）。评估使用的是具有 100 个顶点的真实网格。这表明 LAMP 能够通过选择计算重点来降低预测误差，在计算约束更严格时尤其如此。请注意，我们将所有模型的最大顶点数限制为 100。

模型	初始顶点数	平均顶点数	误差（均方误差）
CNN	25	25.0	4.75
FNO	25	25.0	4.85
MP-PDE	25	25.0	3.69
LAMP（不进行重新网格化）	25	25.0	6.39
LAMP（我们的方法）	25	37.6	**1.05**
CNN	50	50.0	1.10
FNO	50	50.0	1.79
MP-PDE	50	50.0	0.98
LAMP（不进行重新网格化）	50	50.0	2.15
LAMP（我们的方法）	50	53.2	**0.75**
CNN	100	100.0	0.81
FNO	100	100.0	1.39
MP-PDE	100	100.0	0.88
LAMP（不进行重新网格化）	100	100.0	**0.75**
LAMP（我们的方法）	100	100.0	0.76

结果。表1展示了实验结果。可以看到，与表现最佳的基线方法（卷积神经网络、傅里叶神经算子和 MP-PDE）相比，我们的 LAMP 大幅优于所有基线方法：当初始节点数分别为 25、50 和 100 时，误差分别降低了 71.5%、23.5% 和 6.2%（平均降低 33.7%）。值得注意的是，与不进行重新网格化的消融版本相比，我们的完整 LAMP 能够显著降低误差（初始顶点数为 25 时误差降低 83.6%，初始顶点数为 50 时降低 65.1%），而平均顶点数仅适度增加（分别增加 50.4% 和 6.4%）。

这展示了 LAMP 能够通过自适应地分配计算量来降低长期预测误差。

为探究 LAMP 是否能够将计算量集中在最具动态性的区域，我们可视化了 LAMP 的一条示例轨迹，如图2和附录C.1中的图8所示。我们从 50 个顶点开始，在不同的 β 下测试模型，其中较小的 β 更侧重于降低误差。我们发现，当 β 较小时（例如，$\beta = 0.1$），LAMP 能够在动态性最高的区域增加更多节点（25 个节点），而只粗化少量节点（总共移除 3 个节点）。当 β 较大、更加侧重于减少计算量时，LAMP 的细化操作会减少，粗化操作会增加，而且只会在变化较慢的区域进行粗化。此外，我们在图3中可视化了两个不同模型在所有测试轨迹上的误差（y 轴）与顶点数量之间的关系，并展示了不同 β 下的结果。我们发现，随着 β 增大，LAMP 能够进一步减少顶点数量，而误差仅略有增加。此外，LAMP 显著改善了帕累托前沿。总之，上述结果表明，LAMP 能够将计算量集中在动态区域，并在推理时适应不同的 β。

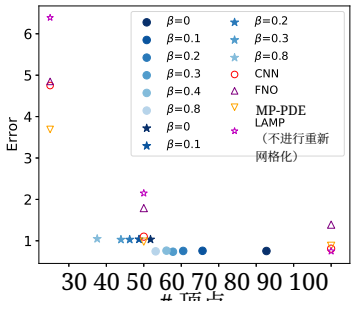


图3：不同 β 下的误差（均方误差）与平均顶点数对比。图中展示了两种 LAMP 模型：初始顶点数为 25（实心星形标记）和 50（实心圆形标记）；基线方法以空心标记表示。可以看到，LAMP 改进了帕累托前沿。

4.2 基于二维网格的仿真

这里，我们在更具挑战性的折纸仿真场景中评估 LAMP 的能力。该仿真中的纸张呈正方形，边界边长为 1。在仿真过

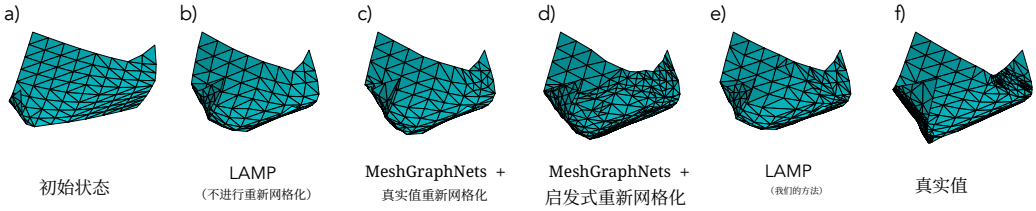


图 4：基于二维网格的纸张模拟示例结果。我们观察到，LAMP 会在高曲率区域增加分辨率以呈现细节，并粗化中部平坦区域。图 a) 对应 $t = 0$，图 b)、c)、d)、e)、f) 分别对应 LAMP（不进行重新网格化）、使用真实网格的 MeshGraphNets、使用启发式重新网格化的 MeshGraphNets、LAMP（我们的方法）以及 $t = 20$ 时的真实结果。更多可视化结果见图 9。

表 2：不同方法在基于二维网格的纸张仿真中的计算量与误差。借助所提出的学习式重新网格化框架，LAMP 能够以平均顶点数略微增加为代价，获得更低的滚动误差。所报告的数值为均方误差：从第 10、30 和 50 步的初始状态开始，进行 20 步学习式仿真器滚动，并在 50 条测试轨迹上取平均。

模型	初始顶点数	平均顶点数	误差（均方误差）
MeshGraphNets +真实值重新网格化	102.9	115.9	5.91e-4
MeshGraphNets +启发式网格重构	102.9	191.9	6.38e-4
LAMP（不进行重新网格化）	102.9	102.9	6.13e-4
LAMP（我们的方法）	102.9	123.1	**5.80e-4**

时，纸张的 4 个角会受到不同大小的力。使用 ArcSim 求解器（Narain 等人，2012）生成真实轨迹时，我们将边的最小和最大长度分别设为 0.01 和 0.2。我们使用计算量与长期演化误差这一指标评估模型。与第 4.1 节类似，我们使用完整轨迹中的节点平均数量作为复杂度的替代指标。我们还与两种基线方法进行了比较。第一种基线方法是采用真实网格重构（GT）的 MeshGraphNets，其重新网格化由 ArcSim 自适应各向异性网格重构组件提供。Pfaff 等人（2021年）使用了这种方法，其误差低于学习式网格重构。第二种基线方法是采用启发式网格重构的 MeshGraphNets +，其启发式方法根据局部曲率细化边（附录 B.4）。

如表 2 所示，我们的模型能够在高曲率区域增加更多分辨率，并取得比无网格重构消融实验和基线方法更好的滚动预测准确率。我们可以看到，LAMP 的表现优于基线方法和无网格重构消融实验。具体而言，LAMP 的表现优于强基线方法“MeshGraphNets + 真实值重新网格化”。这表明，LAMP 可以在使用真实值重新网格化的 MeshGraphNets 基础上进一步改进，从而学习到更好的网格重构策略，使演化模型能够以更忠实的方式演化系统。此外，“MeshGraphNets +启发式重新网格化”基线方法的误差更大，说明这一直观的基线方法并非最优。最后，LAMP 的表现优于其无网格重构消融版本，表明重新网格化十分必要，能够显著降低预测误差。实验的更多细节见附录 B.3。在附录 C.2 的图 4 和图 9 中，我们可以看到，LAMP 学会在折叠部分附近动态性更强的区域（高曲率区域）增加更多网格，这展现了 LAMP 将计算量分配至最需要区域的能力。

5 结论

在这项工作中，我们提出了 LAMP，这是首个完全基于深度学习的代理模型。它能联合学习物理系统的演化，并优化计算任务向动态程度最高区域的分配。在一维和二维数据集上，我们展示了该方法能够自适应地执行细化或粗化操作，其长期预测误差优于强有力的深度学习代理模型基线方法。我们希望该方法能为更高效、更准确地仿真物理系统提供实用工具。

6 致谢

我们感谢 Ian Fischer 和 Xiang Fu 参与讨论并为稿件提供反馈。我们也衷心感谢以下机构的支持：美国国防高级研究计划局，项目编号 HR00112190039（TAMI）、N660011924033（MCS）；美国陆军研究办公室，项目编号 W911NF-16-1-0342（多学科大学研究计划）、W911NF-16-1-0171（国防大学研究仪器计划）；美国国家科学基金会，项目编号 OAC-1835598（CINES）、OAC-1934578（HDR）、CCF-1918940（Ex- peditions）；美国国立卫生研究院，项目编号 3U54HG010426-04S1（人体生物分子图谱计划）；斯坦福数据科学倡议、吴蔡神经科学研究所、亚马逊、Docomo、GSK、日立、英特尔、摩根大通、瞻博网络、KDDI、NEC 和东芝。

本文内容完全由作者负责，不一定代表资助机构的官方观点。

参考文献

MichaelAthanasopoulos、HassanUgail 和 Gabriela Gonz´alez Castro。利用偏微分方程进行飞机几何形状的参数化设计。《工程软件进展》，40(7):479–486，2009。

Yohai Bar-Sinai、Stephan Hoyer、Jason Hickey 和 Michael P Brenner。学习偏微分方程的数据驱动离散化方法。《美国国家科学院院刊》，116(31):15344–15349，2019。

Jean-Paul Berrut 和 Lloyd N. Trefethen。重心拉格朗日插值。《*SIAM*评论》，46:501– 517，2004年。

Johannes Brandstetter、Daniel E. Worrall 和 Max Welling。消息传递神经偏微分方程求解器。发表于国际学习表征会议，2022年。网址：https://openreview. net/forum?id=vSix3HPYKSU。

Francesco Carpanese。用于托卡马克实验模拟和实时解读的自由边界平衡与输运求解器的开发。技术报告，洛桑联邦理工学院，2021年。

Jakub Cerveny、Veselin Dobrev 和 Tzanio Kolev。高阶有限元的非协调网格细化。《*SIAM*科学计算杂志》，41(4):C367–C392，2019年。

Richard Courant、KurtFriedrichs 和 Hans Lewy。关于数学物理中的偏差分方程。《*IBM*研究与开发杂志》，11(2):215–234，1967年。

Stefan Elfwing、Eiji Uchibe 和 Kenji Doya。用于强化学习中神经网络函数逼近的 Sigmoid加权线性单元。《神经网络》，107:3–11，2018年。

Christophe Geuzaine 和 Jean-François Remacle。Gmsh：一款内置前处理和后处理功能的三维有限元网格生成器。《国际工程数值方法杂志》，79(11):1309–1331，2009年。

Danijar Hafner、Timothy P Lillicrap、Mohammad Norouzi 和 Jimmy Ba。使用离散世界模型掌握 Atari。在国际学习表征会议，2021年。网址：https://openreview.net/forum?id=0oabwyZbOu。

Xu Han、Han Gao、Tobias Pfaff、Jian-Xun Wang 和 Liping Liu。通过时间注意力预测网格降阶空间中的物理现象。在国际学习表征会议，2022年。网址：https://openreview.net/forum?id=XctLdNfCmP。

Julian Kates-Harbeck、Alexey Svyatkovskiy 和 William Tang。通过深度学习预测受控聚变等离子体中的破坏性不稳定性。《自然》，568(7753):526–531，2019年。

Dmitrii Kochkov、Jamie A Smith、Ayya Alieva、Qing Wang、Michael P Brenner 和 Stephan Hoyer。机器学习加速计算流体力学。《美国国家科学院院刊》，118(21)，2021年。

Tony Lelievre 和Gabriel Stoltz。分子动力学中的偏微分方程和随机方法。《数值数学年刊》，25:681–880，2016年。

Zongyi Li、Nikola Borislavov Kovachki、Kamyar Azizzadenesheli、Burigede liu、Kaushik Bhattacharya、Andrew Stuart 和 Anima Anandkumar。用于参数化偏微分方程的傅里叶神经算子。载于国际学习表征会议，2021年。URL https://openreview.net/forum?id=c8P9NQVtmnO。

Peter Lynch。《计算机天气预报与气候建模的起源》。计算物理学杂志，227(7):3431–3444，2008年。

Sharanya J Majumdar、Juanzhen Sun、Brian Golding、Paul Joe、Jimy Dudhia、Olivier Caumont、Krushna Chandra Gouda、Peter Steinle、B´eatrice Vincendon、Jianjie Wang 等。高影响天气的多尺度预报：当前状况与未来挑战。美国气象学会公报，102(3):E635–E659，2021年。

Rahul Narain、Armin Samii 和 James F O'brien。用于布料仿真的自适应各向异性重网格。《*ACM* 图形学汇刊》（*TOG*），31(6):1–10，2012年。

Tobias Pfaff、Meire Fortunato、Alvaro Sanchez-Gonzalez 和 Peter W. Battaglia。利用图网络学习基于网格的仿真。载于 国际学习表征会议，2021年。

Alvaro Sanchez、Dmitrii Kochkov、Jamie Alexander Smith、Michael Brenner、Peter Battaglia 和 Tobias Joachim Pfaff。《偏微分方程的潜在场动力学学习》。《神经信息处理系统进展》，2020 年。

Alvaro Sanchez-Gonzalez、Jonathan Godwin、Tobias Pfaff、Rex Ying、Jure Leskovec 和 Peter Battaglia。利用图网络学习模拟复杂物理。载于国际机器学习会议，第 8459–8468 页。PMLR，2020年。

NJ Sircombe、TD Arber 和 RO Dendy。激光-等离子体耦合中的动力学效应：Vlasov 理论与计算。载于《物理学杂志 *IV*（会议录）》，第 133 卷，第 277–281 页。EDP Sciences，2006年。

Halil Mete Soner、Wolfgang Bangerth、Rolf Rannacher、Hans Foellmer 和 LCG Rogers。微分方程的自适应有限元方法。Springer Science & Business Media，2003年。

Richard S Sutton、David McAllester、Satinder Singh 和 Yishay Mansour。基于函数逼近的强化学习策略梯度方法。《神经信息处理系统进展》，第12卷，1999年。

Kiwon Um、Robert Brand、Yun Raymond Fei、Philipp Holl 和 Nils Thuerey。求解器在环：从可微物理中学习，以与迭代偏微分方程求解器交互。《神经信息处理系统进展》，33：6111–6122，2020年。

Pantelis R Vlachas、Georgios Arampatzis、Caroline Uhler 和 Petros Koumoutsakos。通过学习复杂系统的有效动力学进行多尺度模拟。《自然》机器智能，4(4):359–366，2022年。

Tailin Wu、Takashi Maruyama 和 Jure Leskovec。通过潜在全局演化学习加速偏微分方程。《神经信息处理系统进展》，36，2022 年。

Jiachen Yang、Tarik Dzanic、Brenden Petersen、Jun Kudo、Ketan Mittal、Vladimir Tomov、Jean- Sylvain Camier、Tuo Zhao、Hongyuan Zha、Tzanio Kolev 等。用于自适应网格细化的强化学习。*arXiv* 预印本 *arXiv:2103.01342*，2021年。

Qingqing Zhao、David B. Lindell 和 Gordon Wetzstein。使用图网络学习求解偏微分方程约束逆问题。国际机器学习会议，2022。

附录

A 模型架构

这里我们详细介绍 LAMP 的架构，作为第 3.1 节的补充。所有实验中均使用此架构，只有少数超参数（例如潜在维度、消息传递步数）会根据问题的维度（一维、二维）而有所不同。

A.1 架构

在本小节中，我们首先详细介绍演化模型$f_{\text{evo}\theta}$、策略网络 $f_{\varphi}^{\text{policy}}$和价值网络 $f_{\varphi}^{\text{value}}$共用的基础架构。随后介绍它们各自的特点，并阐述我们采用动作表示的优势。

基础架构。我们采用 MeshGraphNets Pfaff 等人（2021）作为演化模型 f_{θ}^{evo}、策略网络 $f_{\text{policy}\varphi}$和价值网络 $f_{\text{value}}{}^{\varphi}$的基础架构。这三个模型都包含一个编码器和一个处理器。对于编码器，我们分别对节点特征和边特征进行编码，并使用多层感知机将其提升至潜在维度，得到 Z_{ij}^{e}, Z_{i}^{v}。处理器由 N 个消息传递层组成。对于每个消息传递层，我们首先使用当前边特征和相连节点特征，通过 $Z_{ij}^{(e)n+1} = \text{MLP}_{\theta}^{(v)}(E_{ij}^{n}, Z_{i}^{(v)n}, Z_{j}^{(v)n})$更新边特征；然后使用当前节点特征和相连边特征，通过 $Z_{i}^{(v)n+1} = \text{MLP}_{\theta}^{(v)}(Z_{i}^{(v)n}, \sum_{j} Z_{ij}^{(e)n+1})$更新节点特征。根据模型及其预期功能，不同模型会采用不同的模块来预测输出，具体如下所述。

演化模型。对于演化模型，输出对应每个节点：将节点上最后一个潜在向量输入解码器，以预测下一个时间步 $t+1$ 的输出 $\hat{V}_{i}^{t+1} = Z_{i}^{(v)N}$ （见公式 6）。

策略网络和价值网络。 $f_{\varphi}^{\text{policy}}$和 $f_{\varphi}^{\text{value}}$共用同一个处理器 $f_{\varphi}^{\text{processor}}$，其中包含 N 个消息传递层。最终的节点特征会附加上用于可控性的 β。 $f_{\varphi}^{\text{policy}}$以编码后的图为输入，由两部分组成：首先通过全局平均池化获得整张图的全局潜在表示，然后将其输入一个多层感知机$^{(k)}$，再接一个针对特定动作的线性层，以预测应执行不同数量动作的概率分布；此外，还会在每条边上应用一个针对特定动作的多层感知机$^{(a)}$，以预测在该边上执行某个动作（即分割、粗化）的概率。 $f_{\varphi}^{\text{value}}$以同一个编码后的图为输入，并通过全局平均池化获得该图的全局潜在特征，然后将其输入一个线性层以预测准确度奖励，再输入另一个线性层以预测计算奖励；最终预测值为 $v =$ 损失 $+\beta \cdot$ 计算。

A.2 一维和二维实验中使用的架构超参数

此处详细介绍了一维和二维实验所使用的架构，并在表3中概述了超参数。

A.2.1 一维非线性偏微分方程

对于一维实验，我们的演化模型 f_{θ}^{evo}在处理器中包含 $N = 3$ 个消息传递层，潜在维度为 64。它使用 SiLU 激活函数（Elfwing等人，2018）。 $f_{\varphi}^{\text{policy}}$和 $f_{\varphi}^{\text{value}}$所用的共享处理器包含 $N = 3$ 个消息传递层，潜在维度为 64。

A.2.2 基于二维网格的仿真

对于 f_{θ}^{evo}，我们将 MeshGraphNets（Pfaff 等人，2021）的消息传递层数设为 8。我们还使用包含 56 个单元的多层感知机（MLP）来建模消息传递函数，并在其后接入 SiLU激活函数（Elfwing等人，2018）。在这里，我们在全部 8 层中共享相同的消息传递函数，因为我们在实验中发现，共享相同的消息传递函数比为每一层使用独立的多层感知机具有更好的

性能。 $f_\varphi^{\text{policy}}$和 f_φ^{value}的共享处理器包含 $N = 3$ 个消息传递层，潜在维度为 64。

B 实验详情

本节介绍一维和二维数据集的实验详情。首先，我们解释 LAMP 若干设计选择背后的理由及其优势。随后，我们在附录 B.1 中总结训练流程并列出超参数表，接着分别介绍一维（附录 B.2）和二维（附录 B.3）的具体训练细节。最后，我们在附录 B.4 中详细介绍基线方法。

将 r^t 用作价值目标。与典型的强化学习场景（如电子游戏或机器人）不同，在那些场景中，一个回合总会结束且奖励有界，而物理仿真具有以下两个鲜明特点：(1) 滚动演化可以无限期地向未来进行；(2) 随着滚动演化持续进行，预测状态与真实状态之间的误差最终会发散，因此误差无界。基于这两个特点，基于无限视界误差的价值目标（例如 Dreamer v2（ Hafner 等人，2021年）中使用的目标）并不合理，因为误差不会有界。在我们的实验中，我们也观察到类似现象：价值目标和价值预测都持续无限增长。因此，我们使用 S 步滚动演化中的平均奖励作为价值目标，以衡量我们关注的滚动演化窗口内的误差和计算量改进，并且这种做法表现得稳定得多。

我们动作空间定义的优势。我们已在第3.1节介绍了动作表示，并在第3.1节描述了动作空间的定义。与在每条边上独立采样相比，上述动作空间设计具有以下优势：

- 动作空间从 $2^{N_{\text{边}}}$降至 $N_{\text{边}}{}^{K_{\text{最大值}}}$，其中 $N_{\text{边}}$表示边的数量。在 $N_{\text{边}} \sim 1000$ 和 $K_{\text{最大值}} \sim 10 \ll N_{\text{边}}$的情况下，动作空间的维度差异十分显著，例如， $2^{1000} = 10^{300}$ 与 $1000^{10} = 10^{30}$。因此，在较小的动作空间中，更容易将奖励归因于恰当的动作。
- 与每条边独立执行动作相比，现在最多只能执行 $K \leq K$个细化或粗化动作。因此，需要专注于少数能够最大程度改善目标函数的动作。
- 对 K 进行采样也会让策略更具“可控性”，因为现在 K 明确取决于 β，而根据 β 来学习应采取多少细化或粗化动作，要比确定具体应采取哪些独立动作容易得多。

B.1 训练流程概述

这里介绍学习演化模型和策略的训练流程。训练分为两个阶段。第一阶段是单独对演化模型进行预训练，不进行重新网格化；第二阶段则交替使用强化学习来学习策略，并对演化模型进行微调。一维和二维实验的详细超参数表见表3。我们所有的模型均在 NVIDIA A100 80GB GPU 上训练。

预训练。在预训练阶段，演化模型在不进行重新网格化的情况下训练；损失通过将演化模型展开 S 步来计算，定义为 loss =（1 步损失）$\times 1$ +（2 步损失）$\times 0.1$ + ... +（S步损失）$\times 0.1$（数值 S 见表3）。我们对后续步骤采用较小的权重，以使训练更加稳定（因为在训练初期，演化模型可能会产生较大的误差；若后续步骤的权重过大，可能导致较大误差，并使训练稳定性下降）。预训练持续若干轮（见表3），之后进入下一阶段。

策略与演化模型的联合训练。在此阶段，演员-评论家与演化模型以交替方式进行训练。具体而言，在策略学习阶段，演化模型被冻结，演员-评论家通过强化学习进行学习（参见第 3.2 节中的“学习策略”部分），训练 J^{policy} 步；而在演化学习阶段，演员-评论家被冻结，

演化模型根据第 3.2 节中的“演化学习”部分进行学习，训练 J^{evo} 步。这两个阶段交替进行。

对演员-评论家与演化采用交替训练策略的原因如下。在学习演员-评论家时，强化学习的回报 是基于当前 策略和演化模型的期望奖励。如果同时也优化演化模型，那么奖励函数就会不断变化，这可能会使策略学习更加困难。因此，我们采用交替训练策略，这种策略也广泛应用于许多其他场景，例如 GAN 训练。

在接下来的两个小节中，我们将详细介绍一维和二维问题的动作空间及具体设置。

表3：模型架构和训练所用的超参数。

超参数名称	一维数据集	二维数据集
模型架构的超参数:		
时间捆绑步数	25	1
f_θ^{evo}：潜在维度	64	56
f_θ^{evo}：激活函数	SiLU	SiLU
f_θ^{evo}：编码器多层感知机层数	3	4
f_θ^{evo}：处理器消息传递层数	3	8
$f_\varphi^{\text{policy}}$: 潜在维度	64	56
$f_\varphi^{\text{policy}}$：激活函数	ELU	ELU
$f_\varphi^{\text{policy}}$：编码器多层感知机层数	2	2
$f_\varphi^{\text{policy}}$：处理器的消息传递层数	3	3
$f_\varphi^{\text{policy}}$: 多层感知机$^{(k)}$: 多层感知机层数	2	2
$f_\varphi^{\text{policy}}$: 多层感知机$^{(k)}$: 激活函数	ELU	ELU
$f_\varphi^{\text{policy}}$: 多层感知机$^{(a)}$: 多层感知机层数	3	3
$f_\varphi^{\text{policy}}$: 多层感知机$^{(a)}$: 激活函数	ELU	ELU
训练超参数:		
β 采样范围 $\mathcal{B}$	$[0, 0.5]$	$\{0\}$
损失函数	MSE	L2
$\alpha_s^{\text{策略}}$, 适用于 $s = 1,2, \ldots$ (式。10)	{1,1,1,...}	{1,1,1,...}
$\alpha_s^{\mathbb{I}}$, 适用于 $s = 1,2, \ldots$ （方程 10)	{1,1,1,...}	{1,1,1,...}
演化模型预训练的轮数	50	100
预训练期间多步损失的滚动演化步数 S 至	4	1
演员-评论家与演化模型联合训练的轮数	30	30
J^{policy}：联合训练期间更新演员-评论家的步数	200	200
J^{evo}：联合训练期间更新演化模型的步数	100	100
批次大小	128	64
预训练期间演化模型的学习率	10^{-3}	10^{-3}
策略学习期间演化模型的学习率	10^{-4}	10^{-4}
价值网络学习率	10^{-4}	10^{-4}
策略网络学习率	5×10^{-4}	5×10^{-4}
优化器	Adam	Adam
值损失的系数	0.5	0.5
$K^{\max}$：粗化或细化的最大动作数	20	20
最大梯度范数	2	20
优化器调度器	余弦	余弦
输入噪声幅度	0	10^{-2}
S：视界	4	6
权重衰减	0	0
η：熵系数	10^{-2}	2×10^{-2}

B.2 一维非线性偏微分方程

一维问题的动作空间由分割和粗化操作组成。粗化操作最初定义在所有边上，随后根据预测的粗化操作数量以及每条边被粗化的概率进行采样。在采样得到的边中，如果两条边共享同一个顶点，则只粗化最右侧的边。

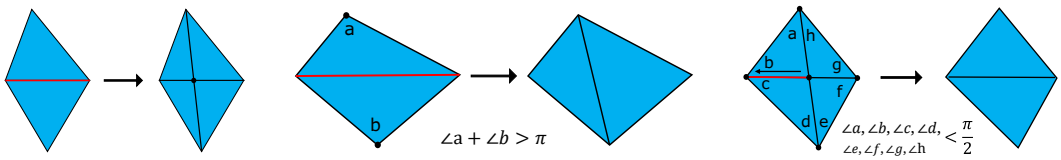
在预训练期间，为了让演化模型适应网格大小的变化，我们会随机进行顶点丢弃：随机抽取 10% 的小批量进行丢弃；如果某个小批量被选中进行节点丢弃，则对其中的每个样本，随机丢弃网格中 0–30% 的节点。

此处的插值 g^{interp} （见公式 4）使用重心插值（Berrut & Trefethen，2004）。在细化过程中，新添加节点的节点特征值是其两个相邻节点的节点特征的线性组合，具体取决于新添加节点及其相邻节点的坐标。

B.3 基于二维网格的仿真

在本小节中，我们详细介绍重新网格化动作的定义、无效重新网格化动作、如何基于 Narain 等人（2012年）生成二维网格数据集，以及如何在生成的数据上预训练演化模型，并介绍用于修正不同网格的顶点配置之间不一致问题的插值方法。

动作空间。二维网格仿真中定义了三种重新网格化操作：分割、翻转和粗化。强化学习的动作空间定义为分割操作集与粗化操作集的笛卡尔积（翻转会自动执行）。每个集合中的元素表示网格中的边，策略函数最多会从中选择 $K^{\max}$ 个元素作为要分割或粗化的边。翻转操作不属于强化学习动作的原因是，我们会翻转所有满足某些条件的边（将在下文说明）。本段其余部分将详细介绍相应的重新网格化操作。二维情况下，分割操作可用于网格中与两个三角形面共边的任意边，也可用于网格的边界边。分割操作会生成 4 个三角形（图 5a）。翻转操作同样用于与两个面共边的边，但还需满足一个附加条件：位于边两侧的顶点处角度之和大于 π：另见图 5b。在我们的重新网格化函数中，所有满足该条件的边被翻转了，这就是翻转操作不属于强化学习动作空间的原因。最后，粗化操作的条件相对严格。其中一个条件是，粗化边的源节点或目标节点之一必须为度为4；另一个条件是，与度为4的节点相连的所有面都必须具有锐角，但该节点周围的角除外。另见图5c。我们根据上述条件筛选来自 $f_\varphi^{\text{policy}}$的采样动作，以获得可分割和粗化的有效边集合。



(a) 分裂操作。 (b) 翻转操作。 (c) 粗化操作。

图5：分割、翻转和粗化操作的示意图。这些操作作用于边。

无效的重新网格化。如第3.1节所述，同一网格面上的两条边不能同时细化，也不能同时粗化。这是因为这样做可能会产生无效网格，其中包含四边形等非三角形面。例如，考虑一个三角形面 ABC，并分割边 AB 和 AC。设 AB 和 AC 的中点分别为 D 和 E，则会产生新边 DC 和 EB。设 DC 和 EB 的交点为 F，则会形成四边形 ADFE，这违反了所有面都必须是三角形的要求。为避免这种情况，每个面的最多一条边可以执行重新网格化或粗化操作。

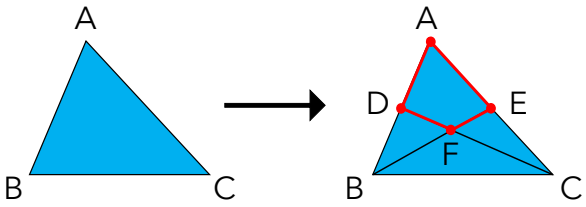


图 6：边上的无效分割操作。同时分割同一个三角形面中的两条边会得到一个四边形。

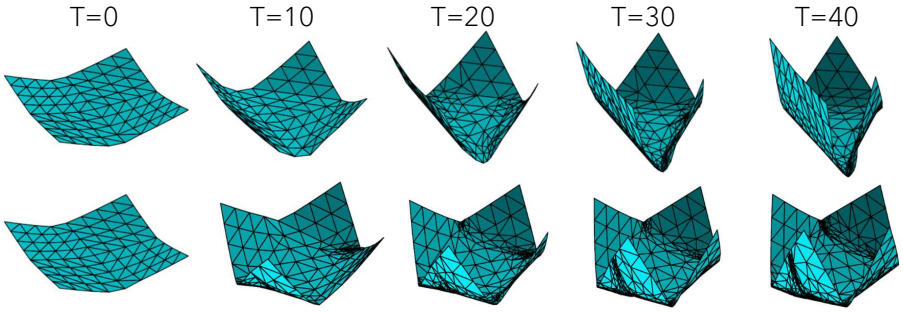


图 7：使用不同配置生成的部分轨迹。随着时间推移，网格中的高曲率区域会添加更精细的三角形面。

生成数据。为生成真实数据，我们使用基于二维三角形网格的 AMR 模拟器 Narain 等人（2012）。该模拟器采用自适应各向异性核方法，使模拟布料自动贴合几何形状和动态细节；另见图7。在我们的实验中，每条轨迹包含 325 个网格（不包括初始网格）。每个网格由顶点和面组成，每个顶点都具有二维坐标和三维世界坐标。相邻两个网格之间的时间间隔设为 0.04。边长范围设为 0.01 至 0.2。请注意，各自重新网格化动作的决策是基于二维坐标作出的。

对于训练数据集和测试数据集，我们生成了 1050 个配置，用于指定施加在网格 4 个角上的力的方向和大小，并基于生成的配置生成了 1000 条轨迹作为训练数据，使用其中 50 条轨迹作为测试数据。在训练我们的演化模型时，我们将数据下采样为原始数据的一半；对于每条轨迹，我们从初始网格开始每隔一个网格取一个。下采样的原因是，我们观察到，使用完整轨迹训练的模型在滚动演化过程中，均方根误差的累积值在迭代超过 160 步后出现了爆炸。
请注意，所有数据集均已预先生成，并将在训练期间加载，因此训练 LAMP 时不需要真实值求解器。

演化模型的预训练。我们使用 MeshGraphNets（Pfaff 等人，2021）对演化函数 f_θ^{evo}进行建模。模型的输入采用图表示，其中顶点配有根据当前和过去时间步的网格信息计算出的三维速度，而边的特征则由边界顶点之间的相对距离及其大小组成。由于网格拓扑在前向迭代过程中会发生变化，我们对过去时间步的网格进行重心插值，以获得与当前时间步顶点相对应的插值顶点坐标。
在评估我们的模型时，我们使用滚动演化均方根误差（RMSE），对每条轨迹中的所有空间坐标、所有网格节点、所有时间步，以及测试数据集中的全部 50 条轨迹取平均值。采用最佳参数设置时，我们的滚动演化均方根误差达到 4.45×10^{-2}。我们发现，添加噪声有助于提升模型性能；我们以 0.01 的尺度向顶点坐标添加了噪声。

插值。由于轨迹中的每个时间步都可能具有不同的网格拓扑，因此无法直接比较一个时间步的节点特征与另一个时间步的节点特征；如果待比较的网格属于不同轨迹，即使是在同一时间步，也无法直接比较。在比较不同网格上的节点特征时，我们会进行重心插值（Berrut & Trefethen，2004）。对于作为演化模型输入的二维仿真速度，我们将时间步

$t-1$ 的网格插值到 t 的网格上，并计算两者之差。为计算滚动演化误差指标，我们始终在每一步将预测网格插值到真实网格上，并在真实网格上计算这些指标。

B.4 基线方法

这里我们补充说明二维基于网格的模拟中所用基线方法的细节。所有基线方法都将预训练的演化函数 f_θ^{evo}用作各自前向模型的一部分。下文主要介绍负责选择要分割和粗化的边的函数细节，这些函数对应于 LAMP 前向模型中的策略函数 $f_\varphi^{\text{policy}}$。

MeshGraphNets + 启发式重新网格化。在每个步骤中，我们不再让 $f_\varphi^{\text{policy}}$推断要分割和粗化的边的概率，而是计算网格边上的局部曲率，并利用曲率筛选出要分割的边。这里，边上的局部曲率定义为该边两个端点处未加权法向量之间的夹角（节点处的法向量定义为其周围各面法向量的平均值）。筛选网格中的边时，我们首先选择曲率超过预设阈值的边；该阈值在实验中设为 0.1。接着，我们检查分割后边的长度是否不小于预设的最小长度，该长度设为 0.04，以避免产生过小的面。违反上述任一条件的边都不会被选为待分割的边。

MeshGraphNets + 真实值重新网格化。在这个基线中，我们使用经典求解器提供的真实网格配置作为预测网格的网格配置。具体而言，在f_θ^{evo}预测出下一时间步的网格后，我们将预测网格插值到真实网格上。我们使用与 LAMP 滚动演化实验相同的初始条件生成所有真实网格。

LAMP（不进行重新网格化）。此基线不涉及任何负责重新网格化的函数，我们只递归应用f_θ^{evo}来演化网格。因此，轨迹中所有网格的网格拓扑均同构。

C 补充结果可视化

C.1 一维非线性偏微分方程

本节展示一维非线性偏微分方程实验中额外随机采样的结果，如图8所示。与图2类似，我们看到，LAMP 能够在动态变化较大的位置增加更多顶点，同时在较静态的区域移除更多顶点。此外，随着 β 越来越侧重于降低计算成本，LAMP 的分割操作减少，粗化操作增多。

C.2 基于二维网格的仿真

在本节中，我们补充展示了二维基于网格的模拟中的随机采样结果，如图 9 所示。我们可以看到，LAMP（第五列）学会在曲率较高的位置增加更多边，因此其滚动演化性能优于不进行重新网格化的情况。

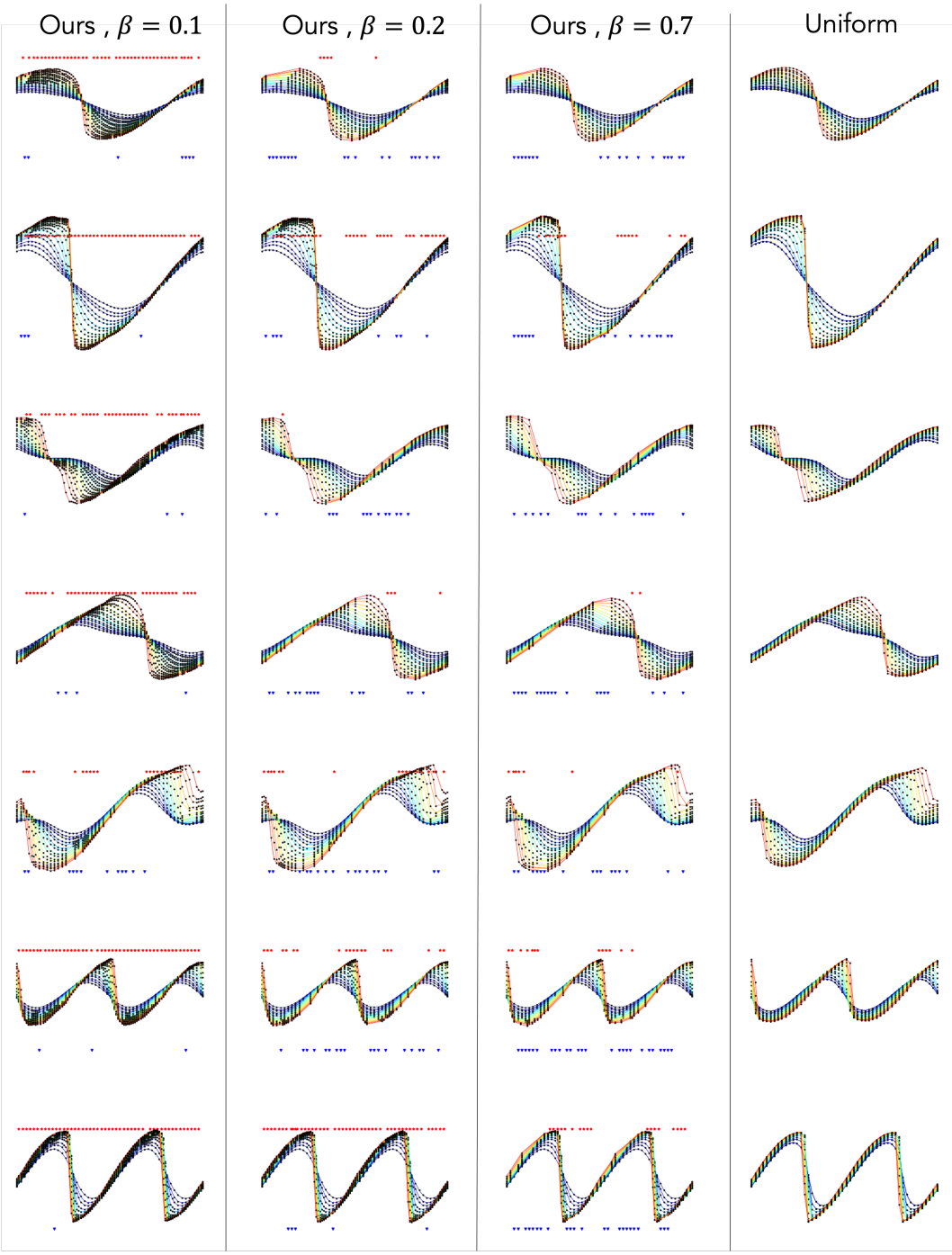


图 8：LAMP 在一维非线性偏微分方程上的其他滚动演化结果示例，采用不同的 β。初始状态位于 50 个顶点上，这些顶点由 100 个顶点均匀下采样得到。滚动演化持续 200 个时间步，不同颜色表示系统在不同时间的状态。上方的红点和下方的蓝点分别表示网格中新增和移除的节点，用于对比最终网格和初始网格。

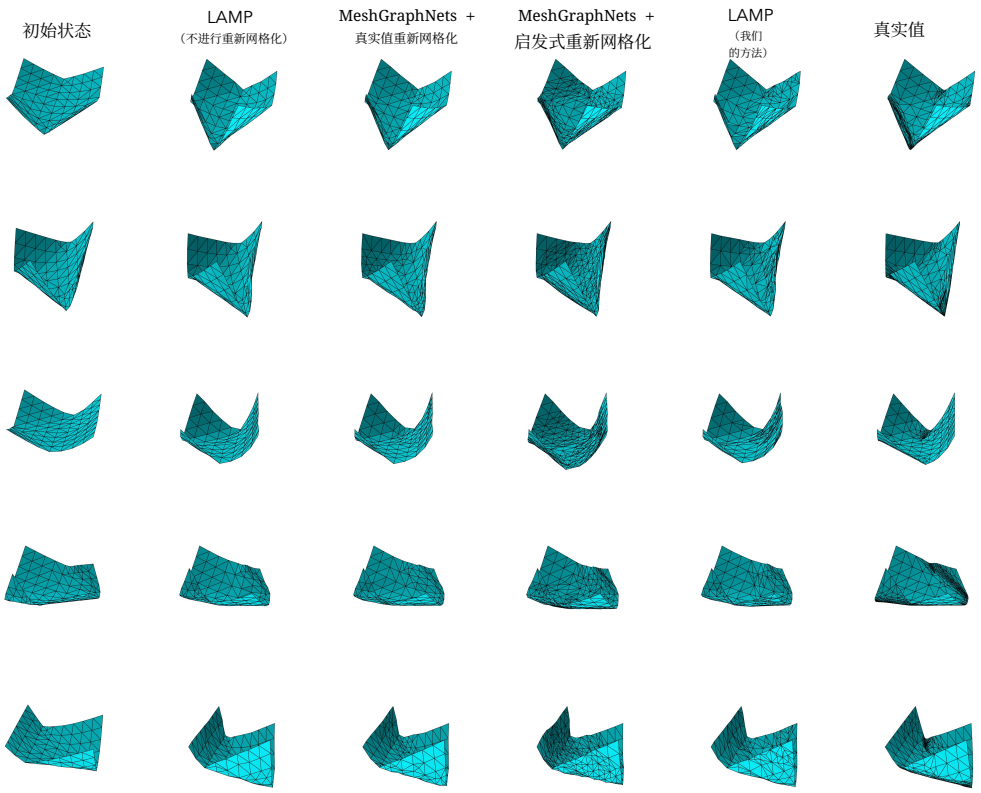


图 9：在 $t = 20$ 下，二维纸张折叠的更多滚动演化结果示例。我们可以看到，我们的 LAMP（第五列）学会在曲率较高的位置增加更多边，并学会在平坦区域进行粗化（“LAMP（我们的方法）”的第三行和第四行，可以看到网格中部发生了粗化），因此其展开性能优于不进行重新网格化的情况以及其他基线方法。