arXiv:2302.02076v1 [math.OC] 4 Feb 2023

AONN：用于参数化最优控制问题整体求解的伴随导向神经网络方法*

PENGFEI YIN†，GUANGQIANG XIAO‡，KEJUN TANG§，和 CHAO YANG¶

摘要。 由偏微分方程（PDEs）控制的参数化最优控制问题广泛存在于科学与工程应用中。对于这类问题，传统的基于网格的数值方法通常需要针对不同的参数设置反复求解偏微分方程，这在计算上代价高昂，尤其是对于具有高维参数空间的问题。尽管近期提出的神经网络方法使得能够同时获得不同参数对应的最优解，但在处理具有复杂约束的问题时仍面临挑战。本文提出 AONN，即一种伴随导向神经网络方法，以克服现有方法在求解参数化最优控制问题时的局限性。在 AONN 中，神经网络作为控制、伴随变量和状态函数的参数化代理模型，用于一次性获得最优解。为降低训练难度并处理复杂约束，我们引入了一种受经典直接-伴随循环（DAL）方法启发的迭代训练框架，从而避免了卡鲁什–库恩–塔克（KKT）系统所产生的惩罚项。训练完成后，可通过神经网络的前向传播快速计算特定参数对应的最优解，并可进一步用于分析最优解的参数化性质。通过一系列涉及各种类型参数的问题的数值实验，验证了 AONN 的有效性和效率。

关键词 。 参数化最优控制、偏微分方程约束优化、深度神经网络、伴随方法

美国数学学会学科分类。 49M41,49M05,65N21

1. 引言。 最优控制建模在众多应用领域中发挥着重要作用，例如航空学 [56], 机械工程 [52], 血流动力学 [43], 微电子学 [39], 油藏模拟 [57], 以及环境科学 [48]。特别是，为求解偏微分方程约束最优控制问题，需要找到一个最优控制函数，使其能够最小化由偏微分方程（PDE）支配的系统的给定代价泛函。求解偏微分方程约束最优控制问题的常用方法包括直接-伴随循环（DAL）方法 [32, 21]，该方法通过迭代求解伴随系统；牛顿共轭梯度法 [47] ，该方法利用海森矩阵信息；半光滑牛顿法[58, 10] ，该方法纳入控制约束和状态约束；以及交替方向乘子法 [8] ，该方法专为凸优化设计。在实践中，代价泛函和偏微分方程系统通常涉及不同的物理参数或几何参数配置，从而形成参数化最优控制建模。这些参数通常来源于某些期望的分布，例如材料属性、边界条件、控制约束和计算域 [22, 23, 48, 35, 46, 39, 31]。

上述大多数方法无法直接应用于参数化最优控制问题。主要原因是，除了最优控制建模中求解相关偏微分方程本身已十分耗时之外，参数的存在还引入了额外的显著复杂性，使参数化最优控制问题比非参数化问题更具挑战性 [18]。求解参数化最优控制问题的一种高效方法是降阶模型（ROM） [40, 43, 48, 36], ，它依赖代理模型进行参数化模型降阶；如果解位于低维子空间中，则能够提供高效且稳定的近似 [18]。然而，由于空间域与参数域相互耦合，降阶模型中的离散化仍受维数灾难的困扰，因此无法获得参数化最优控制问题的整体求解方案 [22, 43, 40]。

基于深度学习的数值方法在求解偏微分方程[41, 42, 12, 16, 59, 44, 45]方面正受到越来越多的关注。近年来，基于深度学习的方法在求解偏微分方程约束最优控制问题方面取得了多项进展。例如，有研究提出一种物理信息神经网络（PINN）方法，将代价泛函加入标准的物理信息神经网络损失函数中，以求解最优控制问题 [34, 29]。与此同时，研究者提出了基于深度学习的代理模型 [56, 30] 和算子学习方法 [55, 20]，无需大量计算即可快速推理得到最优控制解。尽管这些方法在求解最优控制问题方面取得了成功，但很少有方法能够直接应用于参数化最优控制建模。近期的一项研究 [11], 提出了一种扩展的物理信息神经网络，通过将参数作为神经网络输入，从而结合 KKT 条件与神经网络。在这

*已提交给编辑，日期。
†北京大学数学科学学院（pengfeiyin@pku.edu.cn）。
‡北京大学前沿交叉学科研究院（2001213226@stu.pku.edu.cn）。§中国深圳鹏城实验室（tangkj@pcl.ac.cn）。
¶北京大学数学科学学院（chao yang@pku.edu.cn）。

种方式下，对于约束简单的参数化最优控制问题，可以得到参数连续取值范围内的最优解。然而，该方法难以推广以求解更复杂的参数化最优控制问题，尤其是在控制函数带有额外不等式约束的情况下[2, 3]。在此类情形下，必须在损失函数中引入过多惩罚项才能拟合复杂的 KKT系统，这使得优化十分困难 [26]。关于上述方法的更详细讨论，请参见第 4节。

为应对求解参数化最优控制问题的挑战并避免维数灾难，我们提出了 AONN——一种结合经典 DAL 方法与深度学习技术优点的伴随导向神经网络方法。在 AONN 中，我们构建了三个输入中增添参数的神经网络，并将其整合到 DAL 方法框架中，从而一次性近似控制函数、伴随函数和状态函数。首先，借助随机采样，神经网络使经典 DAL 框架能够同时求解参数化问题，而无需对耦合的空间域和参数域进行离散化。其次，与基于 PI NN 的惩罚方法不同， [42, 29, 34, 11], 引入 DAL 后，无需通过各种惩罚项直接求解复杂的 KKT 系统。数值结果将表明，AONN 能够高精度地求解一系列参数化最优控制问题。

本文其余部分安排如下。在第 2 节中，我们介绍问题设置。随后，我们将在第 3 节介绍 AONN 框架。第 4 节将讨论 AONN 与若干近期提出的方法之间的进一步比较。接着，我们在第 5 节给出数值结果，以展示所提出的 AONN 方法的效率。最后在第 6 节总结全文。

2. 问题设置。令 $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^D$ 表示一个由有限个参数组成的向量。令 $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^d$ 表示依赖于 $\boldsymbol{\mu}$ 的空间域，该域有界、连通，且其边界为 $\partial\Omega(\boldsymbol{\mu})$； $\in\Omega(\boldsymbol{\mu})$ 表示空间变量$\mathbf{x}$ 。考虑如下参数化最优控制问题

$$\mathrm{OCP}(\boldsymbol{\mu}): \quad \begin{cases} \min\limits_{(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))\in Y\times U} J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu}), \\ \text{s.t.} \quad \mathbf{F}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu})=0 \ \text{ in } \Omega(\boldsymbol{\mu}), \text{ and } \ u(\mathbf{x},\boldsymbol{\mu})\in U_{ad}(\boldsymbol{\mu}), \end{cases} \tag{2.1}$$

其中， $J\colon Y\times U\times\mathcal{P}\mapsto\mathbb{R}$ 是一个参数依赖的目标泛函， Y 和 U 是定义在 $\Omega(\boldsymbol{\mu})$ 上的两个适当函数空间，其中 $y\in Y$ 和 $u\in U$ 分别为状态函数和控制函数。y 和 u 均依赖于 $\mathbf{x}$ 和 $\boldsymbol{\mu}$。为简化记号，记 $y(\boldsymbol{\mu})=y(\mathbf{x}\ \boldsymbol{\mu})$，以及 $u(\boldsymbol{\mu})=u(\mathbf{x}\ \boldsymbol{\mu})$。在最优控制问题 $\boldsymbol{\mu}$ (2.1) 中，F 表示控制方程，例如本文所研究的参数依赖偏微分方程，其中包括偏微分算子 F_I 和边界算子 F_B （示例见第 5 节）。可容许集 $U_{ad}(\boldsymbol{\mu})$ 是 U 的一个参数依赖的有界闭凸子集，它为 u 提供了额外的不等式约束，例如盒约束 $U_{ad}(\boldsymbol{\mu})=\{u(\boldsymbol{\mu})\in U\colon u_a(\boldsymbol{\mu})\le u(\boldsymbol{\mu})\le u_b(\boldsymbol{\mu})\}$。

由于最优控制问题$(\boldsymbol{\mu})$ (2.1)是一个约束最小化问题，因此$(y^*(\boldsymbol{\mu})\ u^*(\boldsymbol{\mu}))$作为2.1的极小化解，其必要条件为以下KKT系统 [52, 9, 19]:

$$\begin{cases} J_y(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})-\mathbf{F}_y^*(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})p^*(\boldsymbol{\mu})=0, \\ \mathbf{F}(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})=0, \\ (\mathrm{d}_u J(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu}),v(\boldsymbol{\mu})-u^*(\boldsymbol{\mu}))\ge 0,\ \forall v(\boldsymbol{\mu})\in U_{ad}(\boldsymbol{\mu}), \end{cases} \tag{2.2}$$

其中， $p^*(\boldsymbol{\mu})$是伴随函数，也称为拉格朗日乘子；$\mathrm{F}_y^*(y(\boldsymbol{\mu})\ u(\boldsymbol{\mu});\boldsymbol{\mu})$表示F$y(y(\boldsymbol{\mu})\ u(\boldsymbol{\mu});\boldsymbol{\mu})$的伴随算子。由于 $y(\boldsymbol{\mu})$总能通过状态方程F由 $u(\boldsymbol{\mu})$唯一确定，因此(2.2)中 J 关于 u 的全导数可表示为

$$\mathrm{d}_u J(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})=J_u(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})-\mathbf{F}_u^*(y^*(\boldsymbol{\mu}),u^*(\boldsymbol{\mu});\boldsymbol{\mu})p^*(\boldsymbol{\mu}). \tag{2.3}$$

最优控制问题$(\boldsymbol{\mu})$的解满足系统(2.2)。因此，关键在于求解该KKT系统，并据此找到最优控制问题$(\boldsymbol{\mu})$的极小化解。通常，直接求解(2.2)并非易事，而求解KKT系统中涉及的参数化偏微分方程还会带来额外的计算挑战（例如参数空间的离散化）。本文着重采用深度学习方法求解(2.2)。具体而言，我们使用三个深度神经网络，分别通过高效的训练算法对 $y^*(\boldsymbol{\mu})\ u^*(\boldsymbol{\mu})$和 $p^*(\boldsymbol{\mu})$进行逼近。

3. 方法论。设 $\hat{}y(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_y)$ ， $\hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$ 和 $\hat{}p(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p)$ 是三个相互独立的深度神经网络，分别由 $\boldsymbol{\theta}_y$、 $\boldsymbol{\theta}_u$ 和 $\boldsymbol{\theta}_p$ 参数化。这里，$\mathbf{x}(\boldsymbol{\mu})$ 是神经网络的增广输入，

其定义如下：

$$\mathbf{x}(\boldsymbol{\mu}) = \left[\ x_1,\ \ \ldots,\ \ x_d,\ \ \mu_1,\ \ \ldots,\ \ \mu_D\ \right].$$

随后，我们通过最小化定义如下的三个损失函数，使用 $\hat{}y(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y), \hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$ 和 $\hat{}p(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p)$ 来近似 $y^*(\boldsymbol{\mu})$ $u^*(\boldsymbol{\mu})$ 和 $p^*(\boldsymbol{\mu})$：

$$\mathcal{L}_s(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| r_s \left(\hat{y}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_y), \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_u); \boldsymbol{\mu}_i \right) \right|^2 \right)^{\frac{1}{2}}, \tag{3.1a}$$

$$\mathcal{L}_a(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u, \boldsymbol{\theta}_p) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| r_a \left(\hat{y}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_y), \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_u), \hat{p}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_p); \boldsymbol{\mu}_i \right) \right|^2 \right)^{\frac{1}{2}}, \tag{3.1b}$$

$$\mathcal{L}_u(\boldsymbol{\theta}_u, u_{\mathsf{step}}) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_u) - u_{\mathsf{step}}(\mathbf{x}(\boldsymbol{\mu})_i) \right|^2 \right)^{\frac{1}{2}}, \tag{3.1c}$$

其中，$\{\mathbf{x}(\boldsymbol{\mu})_i\}_{i=1}^N$ 表示配置点。泛函 r_s 和 r_a 分别表示由 KKT 条件导出的状态方程和伴随方程的残差，即：

$$r_s(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \triangleq \mathbf{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}), \tag{3.2a}$$

$$r_a(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu}); \boldsymbol{\mu}) \triangleq J_y(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) - \mathbf{F}_y^*(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) p(\boldsymbol{\mu}), \tag{3.2b}$$

而 $u_{\mathsf{step}}(\mathbf{x}(\boldsymbol{\mu}))$ 是控制函数更新过程中引入的中间变量，用于处理 KKT 条件（2.2）中的第三个变分不等式，相关内容将在第 3.2 节讨论。这三个损失函数通过调整三个神经网络 $\hat{}y(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y)$、$\hat{}u(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u)$ 和 $\hat{}p(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_p)$ 的参数，尝试拟合 KKT 条件；训练过程按顺序进行。损失函数中涉及的导数可通过 TensorFlow[1] 或 PyTorch [38] 等深度学习库的自动微分功能高效计算。

3.1. 用于参数化偏微分方程的深度学习。 由于物理系统中涉及额外参数，高效求解参数化偏微分方程对于参数化最优控制建模至关重要（2.1）。为处理参数化偏微分方程，我们将参数 $\boldsymbol{\mu}$作为额外输入，与坐标$\mathbf{x}$ 一起扩充神经网络的输入空间，以处理参数依赖偏微分方程。此外，我们采用无惩罚技术 [27, 44] 在求解参数化偏微分方程时施加边界条件。接下来，我们说明如何应用无惩罚深度神经网络求解参数化状态方程（3.2a），该方法可直接推广至伴随方程（3.2b）的求解。

无惩罚方法的关键在于引入两个神经网络来近似解，其中一个神经网络 $\hat{}y_B$ 用于近似本质边界条件，另一个神经网络 $\hat{}y_I$ 则处理计算域的其余部分。这样便消除了边界条件带来的训练困难，从而提升了复杂几何下的精度和鲁棒性。对于几何形状简单的问题，我们还可以为 $\hat{}y_B$ 构造解析表达式，以进一步降低训练成本（示例见第5节）。状态方程的近似解构造如下：

$$\hat{y}\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y\right) = \hat{y}_B(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_B}) + \ell(\mathbf{x}(\boldsymbol{\mu})) \hat{y}_I\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_I}\right), \tag{3.3}$$

其中，$\boldsymbol{\theta}_y = \{\boldsymbol{\theta}_{y_B}, \boldsymbol{\theta}_{y_I}\}$汇集了两个子神经网络$\hat{}y_B$和$\hat{}y_I$的所有参数，$\ell$是一个长度因子函数，用于建立$\hat{}y_B$ 与$\hat{}y_I$之间的连接，并满足以下两个条件：

$$\begin{cases} \ell(\mathbf{x}(\boldsymbol{\mu})) > 0, & \text{in } \Omega(\boldsymbol{\mu}), \\ \ell(\mathbf{x}(\boldsymbol{\mu})) = 0, & \text{on } \partial\Omega(\boldsymbol{\mu}). \end{cases}$$

长度因子函数 ℓ 的构造细节见参考文献[44]。采用这些设置后，可以分别训练$\hat{}y_B$和$\hat{}y_I$，即先训练$\hat{}y_B$，再固定$\hat{}y_B$ 以训练$\hat{}y_I$。对于固定的 $u(\boldsymbol{\mu})$，我们有

$$\mathbf{F}(\hat{y}\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y\right), u(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} \mathbf{F}_I(\hat{y}\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y\right), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ \mathbf{F}_B(\hat{y}_B(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_B}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \end{bmatrix},$$

状态方程的残差可改写为

$$r_s(\hat{y}\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y\right), u(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} r_{s_I}(\hat{y}\left(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y\right), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ r_{s_B}(\hat{y}_B(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_B}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \end{bmatrix}.$$

然后，我们采样一组 $\{\mathbf{x}(\boldsymbol{\mu})_i\}_{i=1}^N$ 配置点，通过最小化状态损失函数（3.1a）来优化 $\boldsymbol{\theta}_y$，前提是 $\boldsymbol{\theta}_u$ 固定。

对于参数化问题，我们将参数作为神经网络的额外输入。该方法用于求解参数化正问题 [24] 和控制问题 [49]。一种典型的训练点采样方式是在 Ω 和 $\mathcal{P}$ 中分别采样数据，以获得 $\{\mathbf{x}_i\}$ 和 $\{\boldsymbol{\mu}_j\}$，然后组合成乘积数据 $\{(\mathbf{x}_i,\boldsymbol{\mu}_j)\}$，其中 $\times\mathcal{P}$位于 Ω 内。本文并非从 $\Omega(\boldsymbol{\mu})$ 的每个切片中针对固定的 $\boldsymbol{\mu}$采样，而是在空间 $\Omega_{\mathcal{P}}$中采样配置点，其中

$$\Omega_{\mathcal{P}} = \{\mathbf{x}(\boldsymbol{\mu}) : \mathbf{x} \in \Omega(\boldsymbol{\mu})\}$$

表示联合时空参数域。原因在于，对于参数化几何问题，由于空间域 $\Omega(\boldsymbol{\mu})$ 依赖于参数，采样空间无法表示为 Ω 与 $\mathcal{P}$的笛卡尔积。

3.2. 投影梯度下降。由于控制函数存在额外的不等式约束 $u \in U_{ad}(\boldsymbol{\mu})$，因此无法直接将零梯度条件 $\mathrm{d}_u J(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0$ 应用于最优解，以获得 u的更新方案。解决这一问题的一种方法是引入额外的拉格朗日乘子和一些松弛变量来处理不等式约束。然而，这会带来额外的惩罚项，可能影响优化过程 [7]。此外，不等式约束还会导致控制函数非光滑，使得通过惩罚方法[29, 15]捕捉奇异性更加困难。为避免这些问题，我们在此采用一种简单的迭代方法来处理变分不等式，而不使用拉格朗日乘子；该方法采用投影梯度下降法，据此可以得到 u的更新方案。到可容许集 $U_{ad}(\boldsymbol{\mu})$ 上的投影算子定义如下：

$$\mathbf{P}_{U_{ad}(\boldsymbol{\mu})}(u(\boldsymbol{\mu})) = \arg\min_{v(\boldsymbol{\mu})\in U_{ad}(\boldsymbol{\mu})} \|u(\boldsymbol{\mu}) - v(\boldsymbol{\mu})\|_2,$$

该步骤将 $u(\boldsymbol{\mu})$ 投影到凸集 $U_{ad}(\boldsymbol{\mu})$ 上。在实际应用中，上述投影是在有限维向量空间中实现的，即在一组配置点上对 $u(\boldsymbol{\mu})$ 进行离散化（例如，在定义域 Ω 上的网格点）。因此，由于 $U_{ad}(\boldsymbol{\mu})$ 是凸集，构造该投影十分直接。例如，如果

$$U_{ad}(\boldsymbol{\mu}) = \{u \in U : u_a(\mathbf{x}(\boldsymbol{\mu})) \le u(\mathbf{x}(\boldsymbol{\mu})) \le u_b(\mathbf{x}(\boldsymbol{\mu})), \forall \mathbf{x} \in \Omega(\boldsymbol{\mu})\} \tag{3.4}$$

为 u 提供盒约束，其中 u_a 和 u_b 分别是下界函数和上界函数，二者都依赖于 $\boldsymbol{\mu}$；$[u_1, \ldots, u_N]^\mathsf{T}$ 表示 N 个配置点 $\{\mathbf{x}(\boldsymbol{\mu})_i\}_{i=1}^N$ in $\Omega_{\mathcal{P}}$ 处的控制函数值，那么我们可以按元素方式构造投影 $\mathbf{P}_{U_{ad}(\boldsymbol{\mu})}$ [52, 19]:

$$\mathbf{P}_{U_{ad}(\boldsymbol{\mu})}(u_i) = \begin{cases} u_a(\mathbf{x}(\boldsymbol{\mu})_i), & \text{if } u_i < u_a(\mathbf{x}(\boldsymbol{\mu})_i), \\ u_i, & \text{if } u_b(\mathbf{x}(\boldsymbol{\mu})_i) \ge u_i \ge u_a(\mathbf{x}(\boldsymbol{\mu})_i), \\ u_b(\mathbf{x}(\boldsymbol{\mu})_i), & \text{if } u_i > u_b(\mathbf{x}(\boldsymbol{\mu})_i). \end{cases} \quad i = 1, \ldots, N. \tag{3.5}$$

根据上述公式即可执行投影梯度步，从而得到记为 u_{step} 的控制函数更新，其形式为

$$u_{\mathsf{step}}(\boldsymbol{\mu}) = \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right), \tag{3.6}$$

而用于更新控制函数的损失自然定义为式（3.1c），通过最小化式（u）得到对 $\hat{}$ 的近似3.1c）。

最优控制函数 $u^*(\boldsymbol{\mu})$ 满足以下变分性质：

$$u^*(\boldsymbol{\mu}) - \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u^*(\boldsymbol{\mu}) - c\mathrm{d}_u J(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu})\right) = 0, \quad \forall c \ge 0.$$

此处 $\mathrm{d}_u J$ 与全导数表达式（ $p^*\,\boldsymbol{\mu}$）中的伴随函数相关，因此我们定义控制函数的残差2.3。

$$r_v(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu}), c; \boldsymbol{\mu}) \triangleq u(\boldsymbol{\mu}) - \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right), \tag{3.7}$$

其对应的变分损失定义为

$$\mathcal{L}_v(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u, \boldsymbol{\theta}_p, c) = \left(\frac{1}{N}\sum_{i=1}^N \left| r_v\left(\hat{y}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_y), \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_u), \hat{p}(\mathbf{x}(\boldsymbol{\mu})_i; \boldsymbol{\theta}_p), c; \boldsymbol{\mu}_i\right)\right|^2\right)^{\frac{1}{2}}. \tag{3.8}$$

(3.1) 中前两个损失函数与 (3.8) 一起反映了 $y(\boldsymbol{\mu})$ $u(\boldsymbol{\mu})$ 和 $p(\boldsymbol{\mu})$ 对由 KKT系统 (2.2) 所决定的最优解的逼近程度。注意， r_v 和 $\mathcal{L}_v$ 取决于常数 c，该常数实际上是梯度下降的步长。为验证算法的收敛性，我们构造了变分损失，并且 c 通常取最后一步的步长。

3.3. AONN算法。现在，我们已将所有内容整合起来，可以介绍我们的算法了。我们的目标是通过伴随导向神经网络（AONN）高效逼近（2.1）的极小化解。AONN的整体训练过程包含三个步骤：训练 $\hat{}y(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y)$，更新 $\hat{}p(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p)$，以及细化 $\hat{}u(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$。图3.1展示了用于求解参数化最优控制问题的AONN示意图。如图A和B所示，带有增广参数化输入的三个神经网络（$\hat{}y(\mathrm{x}(\boldsymbol{\mu});\boldsymbol{\theta}_y)$, $\hat{}p(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p)$, $\hat{}u(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$）经过优化，以便每次针对相应变量迭代最小化目标泛函。具体而言，训练过程按照图D所示进行，并依据图C所示三个方程推导出的损失函数执行。

从三个初始神经网络 $\hat{}y\left(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y^0\right)$、$\hat{}p\left(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p^0\right)$ 和 $\hat{}u\left(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u^0\right)$ 开始，通过最小化 $\mathcal{L}_s\left(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u^0\right)$（见 (3.1a)）进行训练，得到状态函数 $\hat{}y\left(\mathrm{x}(\boldsymbol{\mu});\boldsymbol{\theta}_y^1\right)$，这等价于求解参数依赖状态方程。利用 $\hat{}y\left(\mathrm{x}(\boldsymbol{\mu});\boldsymbol{\theta}_y^1\right)$，我们针对伴随方程最小化损失 $\mathcal{L}_a\left(\boldsymbol{\theta}_y^1, \boldsymbol{\theta}_u^0, \boldsymbol{\theta}_p\right)$（见 (3.1b)），从而得到 $\hat{}p\left(\mathrm{x};\ \boldsymbol{\theta}_p^1\right)$，这对应于求解参数依赖伴随方程。为更新控制函数 $\hat{}u(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$，首先通过梯度下降计算 $u_{\mathsf{step}}^0(\mathrm{x}(\boldsymbol{\mu}))$，随后进行投影步骤（见 (3.6)），然后通过最小化 $\mathcal{L}_u(\boldsymbol{\theta}_u, u_{\mathsf{step}}^0(\mathrm{x}(\boldsymbol{\mu})))$（见 (3.1c)）得到 $\hat{}u\left(\mathrm{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u^1\right)$。接着，以 $\boldsymbol{\theta}_y^1, \boldsymbol{\theta}_p^1, \boldsymbol{\theta}_u^1$ 作为初始参数，开始下一次迭代。一般而言，迭代格式如下：

$$
\begin{aligned}
\text{training } \hat{y}: \quad & \boldsymbol{\theta}_y^k = \arg\min_{\boldsymbol{\theta}_y} \mathcal{L}_s\left(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u^{k-1}\right), \\
\text{updating } \hat{p}: \quad & \boldsymbol{\theta}_p^k = \arg\min_{\boldsymbol{\theta}_p} \mathcal{L}_a\left(\boldsymbol{\theta}_y^k, \boldsymbol{\theta}_u^{k-1}, \boldsymbol{\theta}_p\right), \\
\text{refining } \hat{u}: \quad & \boldsymbol{\theta}_u^k = \arg\min_{\boldsymbol{\theta}_u} \mathcal{L}_u\left(\boldsymbol{\theta}_u, u_{\mathsf{step}}^{k-1}\right),
\end{aligned}
$$

其中

$$
u_{\mathsf{step}}^{k-1}(\mathbf{x}(\boldsymbol{\mu})) = \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(\hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u^{k-1}) - c^k \mathrm{d}_u J(\hat{y}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y^k), \hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u^{k-1}); \boldsymbol{\mu})\right), \tag{3.9}
$$

and

$$
\begin{aligned}
\mathrm{d}_u J\left(\hat{y}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y^k), \hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u^{k-1}); \boldsymbol{\mu}\right) &= J_u\left(\hat{y}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y^k), \hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u^{k-1}); \boldsymbol{\mu}\right) \\
&\quad - \mathbf{F}_u^*\left(\hat{y}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y^k), \hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u^{k-1}); \boldsymbol{\mu}\right) \hat{p}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_p^k).
\end{aligned} \tag{3.10}
$$

AONN 的迭代围绕着对 $\hat{}u$ 的改进展开，借助 $\hat{}y$ 和 $\hat{}p$形成直接-伴随循环（DAL），图中以红线表示。3.1。此过程与经典 DAL 框架相似，但 AONN 与 DAL 之间存在一个关键区别：借助我们全即神经网络框架中训练好的神经网络，可以高效地计算任意参数下最优控制问题（$\boldsymbol{\mu}$）的可靠解，而 DAL 则无法做到这一点。更多详情请参见第4节的讨论。

训练过程总结于算法 3.1中，其中损失函数 $\mathcal{L}_v(\boldsymbol{\theta}_y,\ \boldsymbol{\theta}_u,\ \boldsymbol{\theta}_p,\ c)$（见（3.8））用于验证。在实际实现中，为增强稳健性，我们采用了衰减因子为 γ 的步长衰减技术。在某种程度上，AONN 方法可视为一种不精确的 DAL，因为每次迭代中，状态方程和伴随方程并未被精确求解，而是通过神经网络进行近似。因此，与前一步相比，训练轮次数增加 n_{aug} （算法 3.1第 9 行），以确保精度和收敛。值得注意的是，如果配置点始终固定，网络 $\hat{}u\left(\mathrm{x}(\boldsymbol{\mu});\boldsymbol{\theta}_u^*\right)$ 的训练可以放在 while 循环之后，因为训练状态函数只使用 u 在配置点处的值（算法 3.1第 5 行的计算）。

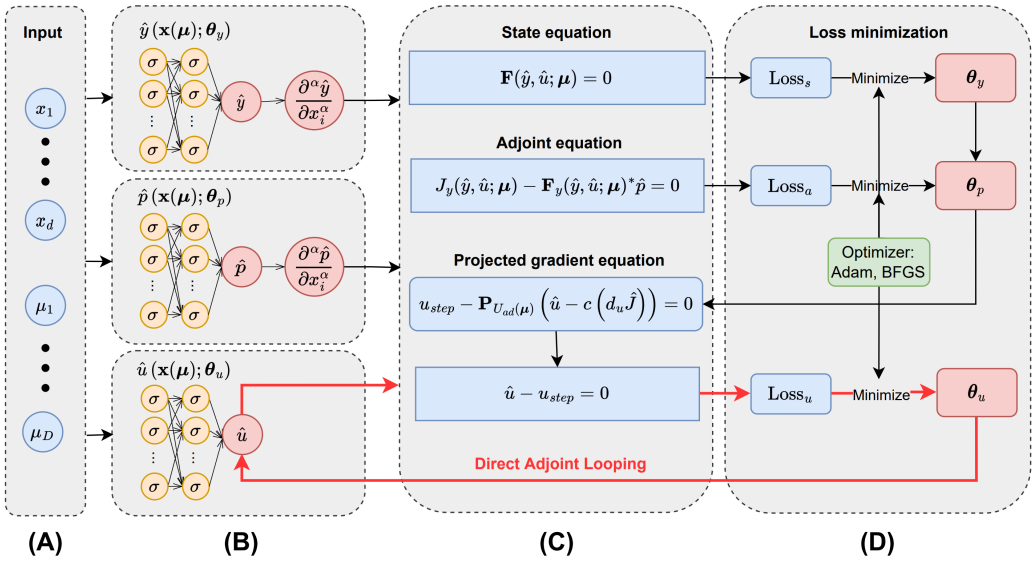


FIG. 3.1。用于求解参数化最优控制问题的 AONN 示意图。（A） *Spatial coordinates and parametersform the input of neural networks.*（B） *AONN consists of three separate neural networks* $\hat{y}$、$\hat{p}$、$\hat{u}$ *and return the approximation ofstate, adjoint and control respectively.*（C） *The state equation, the adjoint eqaution and the projected gradient equation are derivedto formulate the corresponding loss functions.*（D） *The gradients in the state PDE and the adjoint PDE are computed via automaticdifferentiation [38].* $\hat{y}$、$\hat{p}$、$\hat{u}$ *are then trained sequentially via the Adam [25] or the BFGS optimizer.*

算法 3.1AONN 求解最优控制问题$(\boldsymbol{\mu})$

输入：初始 $\boldsymbol{\theta}_y^0, \boldsymbol{\theta}_u^0, \boldsymbol{\theta}_p^0$，配置点$\{\mathbf{x}(\boldsymbol{\mu})_i\}_{i=1}^N$，衰减因子 $\gamma \in (0,1]$，初始步长 c^0、初始训练轮次数 n^0、正整数 n_{aug}和总迭代次数 N_{iter}。1: $k \longleftarrow 1$2: 当 $k < N_{\text{iter}}+1$ 时，执行 3: $\boldsymbol{\theta}_y^k \longleftarrow \arg\min_{\boldsymbol{\theta}_y} \mathcal{L}_s\left(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u^{k-1}\right)$：以 $\boldsymbol{\theta}_y^{k-1}$为初始化，对网络 $\hat{}y(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y)$ 训练 n^k 轮。4: $\boldsymbol{\theta}_p^k \longleftarrow \arg\min_{\boldsymbol{\theta}_p} \mathcal{L}_a\left(\boldsymbol{\theta}_y^k, \boldsymbol{\theta}_u^{k-1}, \boldsymbol{\theta}_p\right)$：以 $\boldsymbol{\theta}_p^{k-1}$为初始化，对网络 $\hat{}p(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_p)$ 训练 n^k 轮。5: 根据 (3.9) 和 (3.10) 计算 $u_{\text{step}}^{k-1}(\mathbf{x}(\boldsymbol{\mu}))$。6: $\boldsymbol{\theta}_u^k \longleftarrow \arg\min_{\boldsymbol{\theta}_u} \mathcal{L}_u(\boldsymbol{\theta}_u, u_{\text{step}}^{k-1})$：以 $\boldsymbol{\theta}_u^{k-1}$为初始化，对网络 $\hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$ 训练 n^k 轮。7: $c^{k+1} \longleftarrow \gamma c^k$。8: $n^{k+1} \longleftarrow n^k + n_{\text{aug}}$。9: $k \longleftarrow k+1$。10: 循环结束 11: $\hat{}y\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_y^*\right) \longleftarrow \hat{}y\left(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y^k\right)$。12: $\hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u^*) \longleftarrow \hat{}u\left(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u^k\right)$。输出：$\hat{}y\left(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_y^*\right), \hat{}u\left(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u^*\right)$。

备注3.1. 可以应用后处理步骤继续训练 $\boldsymbol{\theta}_y^*$，直至获得更精确的状态函数（或伴随函数）解。也就是说，我们可以固定 $\hat{}u\left(\mathbf{x};\ \boldsymbol{\theta}_u^*\right)$，并通过最小化状态损失（3.1a）来训练 $\boldsymbol{\theta}_y^*$。我们也可以固定 $\boldsymbol{\theta}_y^*$，$\boldsymbol{\theta}_u^*$并使用（$\boldsymbol{\theta}_p^*$）训练3.1b。初始步长 c^0 对于算法3.1的收敛至关重要。步长过大可能导致算法发散，而步长过小则可能导致收敛缓慢。

4. 与其他方法的比较。与求解确定性最优控制问题不同，OCP$(\boldsymbol{\mu})$ 中参数的存在给传统的基于网格的数值方法带来了困难。一种直接的方法是将 OCP$(\boldsymbol{\mu})$ 转化为确定性最优控制问题。对于参数的每一种实现，

OCP($\boldsymbol{\mu}$) 可简化为以下形式

$$\text{OCP}: \quad \begin{cases} \min\limits_{(y,u)\in Y\times U} J(y,u), \\ \text{s.t. } \mathbf{F}(y,u)=0 \ \text{ in } \Omega, \ \text{ and } u\in U_{ad}. \end{cases} \tag{4.1}$$

经典直接-伴随循环（DAL）方法 [32, 21] 是解决该问题的一种常用方法，它采用迭代格式，通过数值求解器（例如有限元方法）求解 KKT 系统中的子问题，从而逐步逼近最优解。在每次直接-伴随循环过程中，首先求解控制偏微分方程（4.2a），然后求解伴随偏微分方程（4.2b），以构造总梯度（4.2c），用于更新控制函数。

$$\mathbf{F}(y,u)=0, \tag{4.2a}$$
$$J_y(y,u)-\mathbf{F}_y^*(y,u)p=0, \tag{4.2b}$$
$$\mathrm{d}_u J(y,u)=J_u(y,u)-\mathbf{F}_u^*(y,u)p. \tag{4.2c}$$

尽管 DAL 有效，但由于 $\Omega_{\mathcal{P}}$ 离散化所带来的维数灾难，它无法直接处理 OCP($\boldsymbol{\mu}$) 问题。另一种策略是采用降阶模型（ROM） [36], ，它依赖于参数依赖偏微分方程的代理模型。其思路是，任意参数下的偏微分方程解都可以基于少数几个基函数来计算，这些基函数由对应于若干预先选定参数的解构造而成。然而，当参数诱导的解流形不位于低维子空间中时，使用 ROM 仍然需要付出难以承受的计算代价。

近期，一些深度学习算法被用于求解固定参数[29, 34]下的最优控制问题。通过引入两个深度神经网络，可以通过最小化以下目标泛函来逼近状态函数 y 和控制函数 u ：

$$\min_{(y,u)\in Y\times U} J(y,u)+\beta_1\mathbf{F}(y,u)^2+\beta_2\|u-\mathbf{P}_{U_{ad}}(u)\|_U, \tag{4.3}$$

其中加入了两个惩罚项，且 $\beta=(\beta_1,\ \beta_2)$是两个需要调节的参数。随着惩罚参数增大到 $+\infty$，该无约束问题的解集会趋近于约束问题的解集。然而，这种惩罚方法有一个严重缺陷。一方面，随着惩罚参数增大，越来越难以获得最优解。另一方面，如果惩罚参数较小，则约束无法得到很好的满足。为缓解这一困难，可以使用 hPI NN[29] ，它采用增广拉格朗日法求解（4.3）。然而，由于参数的存在，直接将这种方法扩展到最优控制问题($\boldsymbol{\mu}$)仍然具有挑战性。这是因为同时优化一系列参数连续变化的目标泛函极其困难。

4.1. 针对最优控制问题($\boldsymbol{\mu}$)。为处理参数化最优控制问题，我们采用一种扩展的 PINN 方法 [11] ，通过增广输入获得更准确的参数化预测。也就是说，神经网络的输入由两部分组成：空间坐标和参数。对于任意参数，其最优解均由一个深度神经网络近似，该网络通过求解参数依赖 KKT 系统（2.2）获得。特别地，当控制函数不受限制时 $u(\boldsymbol{\mu})$（例如， $U_{ad}(\boldsymbol{\mu})$为整个巴拿赫空间），KKT 系统为

$$\mathcal{F}(y(\boldsymbol{\mu}),u(\boldsymbol{\mu}),p(\boldsymbol{\mu});\boldsymbol{\mu})=\begin{bmatrix} J_y(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu})-\mathbf{F}_y^*(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu})p(\boldsymbol{\mu}) \\ \mathbf{F}(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu}) \\ \mathrm{d}_u J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu}) \end{bmatrix}=\mathbf{0}, \tag{4.4}$$

其中，总梯度 $d_u J$ 由(2.3)给出。在这类情况下，可以使用物理信息神经网络算法，通过最小化由 KKT系统导出的最小二乘损失来求得最优解。然而，要将此方法应用于控制函数 u受到某些额外约束（例如盒约束（3.4）的情况，可能需要引入拉格朗日乘子 $\lambda(\boldsymbol{\mu})=(\lambda_a(\boldsymbol{\mu})\ \lambda_b(\boldsymbol{\mu}))$，它们对应于

约束 $u(\boldsymbol{\mu}) \geq u_a(\boldsymbol{\mu})$ 和 $u(\boldsymbol{\mu}) \leq u_b(\boldsymbol{\mu})$。对于这类情况，KKT系统为

$$\mathcal{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu}), \lambda(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} J_y(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) - \mathbf{F}_y^*(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) p(\boldsymbol{\mu}) \\ \mathbf{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ \mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) - \lambda_a(\boldsymbol{\mu}) + \lambda_b(\boldsymbol{\mu}) \\ \lambda_a(\boldsymbol{\mu})(u_a(\boldsymbol{\mu}) - u(\boldsymbol{\mu})) \\ \lambda_b(\boldsymbol{\mu})(u_b(\boldsymbol{\mu}) - u(\boldsymbol{\mu})) \end{bmatrix} = \mathbf{0}, \tag{4.5a}$$

$$\text{and} \quad \begin{cases} u_a(\boldsymbol{\mu}) \leq u(\boldsymbol{\mu}) \leq u_b(\boldsymbol{\mu}), \\ \lambda_a(\boldsymbol{\mu}) \geq 0, \lambda_b(\boldsymbol{\mu}) \geq 0. \end{cases} \tag{4.5b}$$

将 PINN 框架应用于求解系统（4.5），需要处理损失函数中的若干惩罚项，包括等式项（4.5a）和不等式项（4.5b）的惩罚；这会导致解不准确，即使对于参数固定的问题也是如此，下一节将对此进行介绍。此外，额外约束 U_{ad}（$\boldsymbol{\mu}$）通常会引入不等式项和非线性项，并使最优控制函数 [2, 3], 具有更强的奇异性$\boldsymbol{\mu}$，从而限制了 PINN 在求解带控制约束的最优控制问题$(\boldsymbol{\mu})$时的应用。

4.2. PINN+在最优控制问题(μ)中的投影。为了找到更好的比较基准，我们提出通过引入投影算子来提升 PINN 的性能。这样，KKT系统（2.2）可以改写为一个表述更紧凑的条件 [37]:

$$\mathcal{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu}), c; \boldsymbol{\mu}) = \begin{bmatrix} J_y(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) - \mathbf{F}_y^*(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) p(\boldsymbol{\mu}) \\ \mathbf{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ u(\boldsymbol{\mu}) - \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right) \end{bmatrix} = \mathbf{0}, \tag{4.6}$$

其中， c 可以是任意正数。请注意，选择合适的 c 可以加快算法的收敛。例如，经典做法是选择 $c = 1/\alpha$ ，以消去投影算子中的控制函数 u ，其中 α 是 Tikhonov正则化项的系数（见第 5 节.1 中的实验）。式（4.6）避免了由控制约束引起的互补条件和不等式，从而显著降低了优化难度。本文将该方法称为 PINN+投影，它将投影策略与 KKT系统结合起来，以构造物理信息神经网络的残差损失。尽管 PINN+投影缓解了控制约束给物理信息神经网络求解带来的困难，但在非光滑最优控制问题中仍存在局限性。对于稀疏 L_1最小化等非光滑优化问题，由于 L_1范数 [9]不可微，KKT系统无法再用式（4.6）描述。相反，需要引入 L_1代价项的对偶乘子。困难源于式（4.6）中的第三个非光滑变分方程，这使得神经网络难以训练。AONN 通过利用 DAL 方法中的更新方案，避免使用式（4.6）中的隐式变分方程，从而降低了这一难度。数值结果还表明，KKT系统（4.6）不能直接用于构造神经网络的损失函数。涉及最优控制问题（$\boldsymbol{\mu}$）中 L_1最小化的结果（见第 5 节.5）有力地表明，AONN 是一个更可靠、高效的框架。

所提出的 AONN 方法兼具上述方法的所有优点，同时避免了它们的缺点。通过继承 DAL 的结构，AONN 方法可以采用交替最小化迭代的方式求解 KKT 系统，从而获得精确解。因此，该方法不需要与额外控制约束相对应的拉格朗日乘子，既能降低存储成本，也能提高精度。此外，AONN 能够准确逼近任意参数下参数化最优控制问题的最优解，并可推广至高维参数的情形。

5. 数值研究。本节展示五项数值实验的结果，以说明 AONN 的有效性，并研究不同参数设置下的各类偏微分方程约束、目标泛函和控制约束。下文首先通过求解最优控制问题来验证 AONN，随后将其用于求解$\boldsymbol{\mu}$在特定区间内连续参数变化的 OCP(+)。为进行比较，我们还采用 PINN 方法和 PINN$\boldsymbol{\mu}$投影方法求解 OCP([17])。我们采用带有正弦激活函数的残差网络模型$_{\mathcal{P}}$ 来构建 AONN 及其他基于神经网络的算法所用的神经网络。除非另有说明，否则使用准蒙特卡罗方法，通过调用 SciPy 模块 [53]从 Ω 中生成配置点。对于所有测试问题，均构造了解析长度因子函数（见（3.3））,

以使近似解自然满足狄利克雷边界条件。神经网络在配备 PyTorch 1.8.1 的 Geforce RTX 2080 GPU 上进行训练。采用带强 Wolfe 线搜索策略的 Broyden–Fletcher–Goldfarb–Shanno（BFGS）算法更新神经网络参数，以加快收敛；BFGS 的最大迭代次数设为 100。

5.1. 测试 1：半线性椭圆方程的最优控制。我们从以下非参数最优控制问题开始：

$$\begin{cases} \min\limits_{y,u} J(y,u) := \dfrac{1}{2}\|y-y_d\|^2_{L_2(\Omega)} + \dfrac{\alpha}{2}\|u\|^2_{L_2(\Omega)}, \\ \text{subject to } \begin{cases} -\Delta y + y^3 = u + f & \text{in } \Omega \\ y = 0 & \text{on } \partial\Omega, \end{cases} \\ \text{and} \quad u_a \le u \le u_b \quad \text{a.e. in } \Omega. \end{cases} \tag{5.1}$$

J 关于 u 的全导数为 $\mathrm{d}_u J(y,\ u) = \alpha u + p$，其中 p 是相应伴随方程的解：

$$\begin{cases} -\Delta p + 3py^2 = y - y_d & \text{in } \Omega, \\ p = 0 & \text{on } \partial\Omega. \end{cases} \tag{5.2}$$

我们采用与参考文献相同的设置。[14], 其中 $\Omega = (0,1)^2$，$\alpha = 0.01$，$u_a = 0$，以及 $u_b = 3$。解析最优解为

$$\begin{aligned} y^* &= \sin(\pi x_1)\sin(\pi x_2), \\ u^* &= \mathbf{P}_{[u_a,u_b]}(2\pi^2 y^*), \\ p^* &= -2\alpha\pi^2 y^*, \end{aligned} \tag{5.3}$$

其中$\mathbf{P}_{[u_a,u_b]}$是到区间上的逐点投影算子。期望状态 $y_d = 1+4\pi^4\alpha\ y^* - 3y^{*2}p^*$ 和源项给定为 $f = 2\pi^2 y^* + y^{*3} - u^*$ 为满足状态方程和伴随方程。

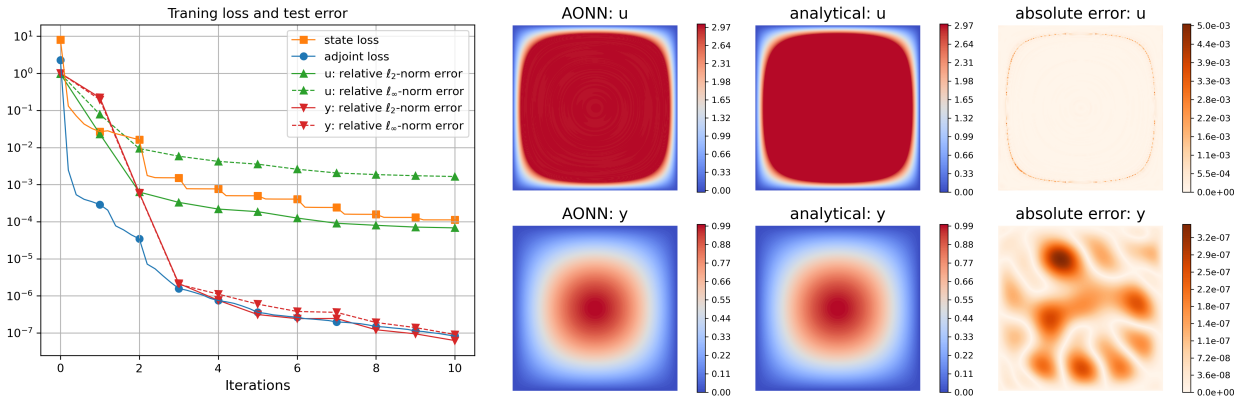
为使用 AONN 求解最优控制问题，我们构建了三个网络 $\hat{}y_I(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I})$、$\hat{}p_I(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{p_I})$ 和 $\hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$，其网络结构均由两个 ResNet 块组成，每个 ResNet 块包含两个各有 15 个神经元的全连接层，并带有一个残差连接。我们在 Ω 内随机采样 $N = 4096$ 个点，构成训练集。在 Ω 内生成大小为 256×256 的均匀网格，用于测试和可视化。我们在子问题中采用固定的步长和训练轮数，即 $c^k \equiv 1/\alpha = 100$、$n^k \equiv 500$。图 5.1(a) 展示了损失变化，以及采用 ℓ_2-范数和 ℓ_∞-范数计算的相对误差 $\|u-u^*\|/\|u^*\|$；图 5.1(b) 则评估了 AONN 解与解析解之间的差异。如文献 [14], 所报告，要在 ℓ_2 意义下达到误差 1×10^{-4}，采用有限元方法需要 7733 个自由度，而 AONN 方法仅需 781 个神经网络参数即可逼近控制函数。

5.2. 测试 2：带控制约束参数化的半线性椭圆方程最优控制。接下来，我们考虑具有控制约束参数化的同一个最优控制问题。控制约束上界 u_b 设为一个连续变量 $\boldsymbol{\mu}$，取值范围为 3 到 20，而非固定数值。因此（5.1）实际上构建了一系列最优控制问题，其最优解（5.3）取决于 $\boldsymbol{\mu}$。现在，我们验证当 $\boldsymbol{\mu}$在区间 $3 \le \boldsymbol{\mu} \le 20$内连续变化时，能否通过 AONN 获得整体求解方案。我们寻求如下问题定义的最优 $y(\boldsymbol{\mu})$ $u(\boldsymbol{\mu})$：

$$\begin{cases} \min\limits_{y(\boldsymbol{\mu}),u(\boldsymbol{\mu})} J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu})) := \dfrac{1}{2}\|y(\boldsymbol{\mu})-y_d(\boldsymbol{\mu})\|^2_{L_2(\Omega)} + \dfrac{\alpha}{2}\|u(\boldsymbol{\mu})\|^2_{L_2(\Omega)}, \\ \text{subject to } \begin{cases} -\Delta y(\boldsymbol{\mu}) + y(\boldsymbol{\mu})^3 = u(\boldsymbol{\mu}) + f(\boldsymbol{\mu}) & \text{in } \Omega \\ y(\boldsymbol{\mu}) = 0 & \text{on } \partial\Omega, \end{cases} \\ \text{and} \quad u_a \le u(\boldsymbol{\mu}) \le \boldsymbol{\mu} \quad \text{a.e. in } \Omega. \end{cases} \tag{5.4}$$

为自然满足状态方程和伴随方程中的齐次狄利克雷边界条件，定义三个神经网络来近似最优控制问题（$\boldsymbol{\mu}$）（5.4）的 AONN 解，如下：

$$\begin{aligned} \hat{y}\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I}\right) &= \ell(\mathbf{x})\hat{y}_I\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I}\right), \\ \hat{p}\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{p_I}\right) &= \ell(\mathbf{x})\hat{p}_I\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{p_I}\right), \\ \hat{u}\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_u\right) &= \hat{u}_I\left(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_u\right), \end{aligned} \tag{5.5}$$



(a) (b)

FIG. 5.1 *Test 1: training loss and test error of problem 5.1. Test error is evaluated at 256×256 grid points. (a) Loss behaviour measured in terms of* 3 1 - 3 1 *and test errors in both ℓ_2-norm and ℓ_∞-norm during training process. (b) The AONN solution and its absolute errors compared with the analytical solution.*

其中，长度因子函数由下式构成：

$$\ell(\mathbf{x}) = x_0(1-x_0)x_1(1-x_1). \tag{5.6}$$

$\hat{}y_I(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_I})$、$\hat{}p_I(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{p_I})$ 和 $\hat{}u(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u)$ 的网络结构与前一次测试相同，但输入维数为 3，且每个隐藏层中的神经元数量为 20，因此共有 1361 个待定参数。为计算损失，我们在时空参数空间 $\Omega_{\mathcal{P}}$ 中采样 $N = 20480$ 个点。我们采用与前一次测试相同的步长 c^k 和训练轮数 n^k，并进行 $N_{\text{iter}} = 20$ 次迭代，直至算法 3.1 收敛。对于 $\boldsymbol{\mu}$ 的每次实现，均在大小为 256×256 的均匀网格上计算测试误差。

在图5.2中，我们绘制了 $\boldsymbol{\mu}$的八个等距实现对应的解析解、AONN 解和 PINN 解。可以看出，按绝对误差衡量，AONN 解优于 PINN 解。仔细观察可发现，较大的误差集中在 u的奇异性位置附近，即活跃约束曲线 $\{\mathbf{x} : u(\mathbf{x}(\boldsymbol{\mu})) = \boldsymbol{\mu}\}$附近； $\boldsymbol{\mu} = 20$ 的情形除外，此时不等式约束不活跃，最优控制函数保持平滑。需要注意的是，[51, 50, 13] 可以采用自适应采样策略来提高奇异区域内的精度，这将留待未来研究。

5.3. 测试 3：采用物理参数化的纳维-斯托克斯方程最优控制问题。下一个测试案例是参数化最优控制问题

$$\min_{y(\boldsymbol{\mu}),u(\boldsymbol{\mu})} J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu})) = \frac{1}{2}\|y(\boldsymbol{\mu}) - y_d(\boldsymbol{\mu})\|^2_{L_2(\Omega)} + \frac{1}{2}\|u(\boldsymbol{\mu})\|^2_{L_2(\Omega)}, \tag{5.7}$$

满足以下稳态不可压纳维-斯托克斯（NS）方程：

$$\begin{cases} -\boldsymbol{\mu}\Delta y(\boldsymbol{\mu}) + (y(\boldsymbol{\mu})\cdot\nabla)y(\boldsymbol{\mu}) + \nabla p(\boldsymbol{\mu}) = u(\boldsymbol{\mu}) + f(\boldsymbol{\mu}) & \text{in } \Omega, \\ \operatorname{div} y(\boldsymbol{\mu}) = 0 & \text{in } \Omega, \\ y(\boldsymbol{\mu}) = 0 & \text{on } \partial\Omega, \end{cases} \tag{5.8}$$

在 $\Omega = (0,\ 1)^2$ 中，参数 $\boldsymbol{\mu}$ 表示雷诺数的倒数。请注意，文献[28, 54]研究了针对 $\boldsymbol{\mu} = 0.1$ 和 $\boldsymbol{\mu} = 1.0$ 的不含控制约束的非参数化问题。我们设定物理参数 $\boldsymbol{\mu} \in [0.1, 100]$，此外，还考虑以下针对 $u(\boldsymbol{\mu}) = (u_1(\boldsymbol{\mu})\ u_2(\boldsymbol{\mu}))$ 的约束：

$$u_1(\boldsymbol{\mu})^2 + u_2(\boldsymbol{\mu})^2 \le r^2, \tag{5.9}$$

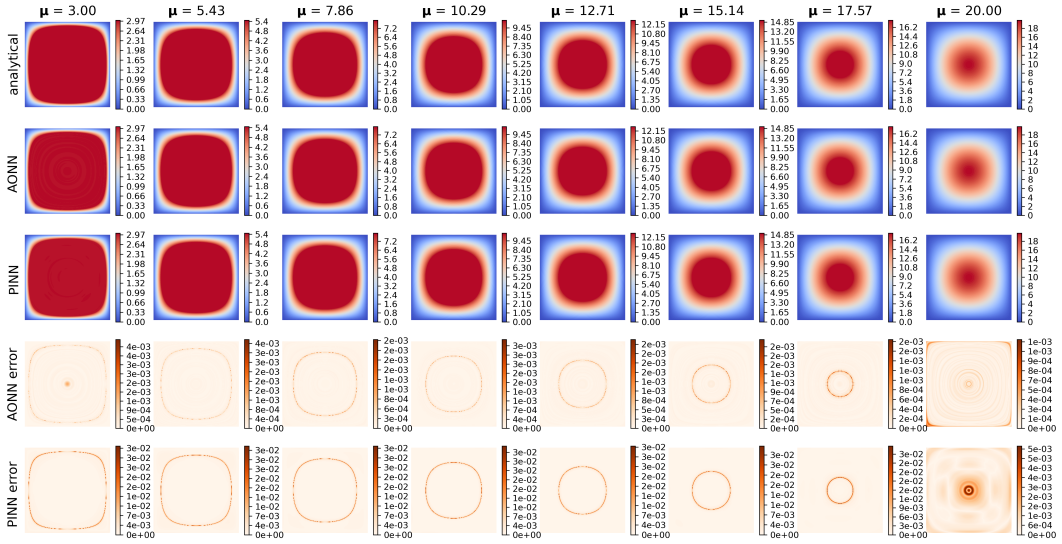


FIG. 5.2. *Test 2: the control solutions $u(\boldsymbol{\mu})$ of AONN and PINN with eight realizations of $\boldsymbol{\mu} \in [3, 20]$, and their absolute errors.*

其中 $r = 0.2$，这给该问题带来了额外的挑战。期望状态 $y_d(\boldsymbol{\mu})$ 和源项 $f(\boldsymbol{\mu})$ 预先给定，以确保上述最优控制问题$(\boldsymbol{\mu})$的解析解为

$$
\begin{aligned}
y^*(\boldsymbol{\mu}) &= e^{-0.05\boldsymbol{\mu}} \begin{pmatrix} \sin^2 \pi x_1 \sin \pi x_2 \cos \pi x_2 \\ -\sin^2 \pi x_2 \sin \pi x_1 \cos \pi x_1 \end{pmatrix}, \\
\lambda^*(\boldsymbol{\mu}) &= \left(e^{-0.05\boldsymbol{\mu}} - e^{-\boldsymbol{\mu}}\right) \begin{pmatrix} \sin^2 \pi x_1 \sin \pi x_2 \cos \pi x_2 \\ -\sin^2 \pi x_2 \sin \pi x_1 \cos \pi x_1, \end{pmatrix}.
\end{aligned}
$$

伴随方程定义为

$$
\begin{cases}
-\boldsymbol{\mu}\Delta\lambda(\boldsymbol{\mu}) - (y(\boldsymbol{\mu}) \cdot \nabla)\lambda(\boldsymbol{\mu}) + (\nabla y(\boldsymbol{\mu}))^T \lambda(\boldsymbol{\mu}) + \nabla\nu(\boldsymbol{\mu}) = y(\boldsymbol{\mu}) - y_d(\boldsymbol{\mu}) & \text{in } \Omega, \\
\operatorname{div} \lambda(\boldsymbol{\mu}) = 0 & \text{in } \Omega, \\
\lambda(\boldsymbol{\mu}) = 0 & \text{on } \partial\Omega,
\end{cases} \tag{5.10}
$$

其中，$\lambda(\boldsymbol{\mu})$ 表示伴随速度，$\nu(\boldsymbol{\mu})$ 表示伴随压力。最优压力和伴随压力 $p^*(\boldsymbol{\mu})$ $\nu^*(\boldsymbol{\mu})$ 均为零。为了满足状态方程和伴随方程，$y_d(\boldsymbol{\mu})$ 和$f(\boldsymbol{\mu})$ 取为

$$
\begin{aligned}
f(\boldsymbol{\mu}) &= -\boldsymbol{\mu}\Delta y^*(\boldsymbol{\mu}) + (y^*(\boldsymbol{\mu}) \cdot \nabla) y^*(\boldsymbol{\mu}) - u^*(\boldsymbol{\mu}), \\
y_d(\boldsymbol{\mu}) &= y^*(\boldsymbol{\mu}) - \left(-\boldsymbol{\mu}\Delta\lambda^*(\boldsymbol{\mu}) + (y^*(\boldsymbol{\mu}) \cdot \nabla)\lambda^*(\boldsymbol{\mu}) - (\nabla y^*(\boldsymbol{\mu}))^T \lambda^*(\boldsymbol{\mu})\right).
\end{aligned} \tag{5.11}
$$

容易验证，最优控制为 $u^*(\boldsymbol{\mu}) = \mathbf{P}^{B(0,r)}(\lambda^*(\boldsymbol{\mu}))$，其中 $B(0,\ r)$ 是一个以原点为中心、半径为 r的球。状态方程（5.8）和伴随方程（5.10）以及变分不等式，其中 $\mathrm{d}_u J(\boldsymbol{\mu}) = u(\boldsymbol{\mu}) - \lambda(\boldsymbol{\mu})$ 构成最优性系统。

对于 AONN，我们使用一个神经网络来近似 y，该网络由两个 ResNet 块构成，每个块包含两个具有 20 个单元的全连接层和一个残差连接，共有 1382 个参数。用于近似 p 的神经网络包含两个由具有 10 个单元的全连接层构成的 ResNet 块，共有 381 个参数。用于 λ 和 ν 的神经网络架构分别与 y 和p 相同。我们在时空参数空间 $\Omega_{\mathcal{P}}$中随机采样 $N = 20000$ 个点。算法 3.1 中的最大迭代次数设为 $N_{\text{iter}} = 300$，步长设为 $c^k \equiv c^0 = 1.0$。初始训练轮次设为 $n^0 = 200$，并在每 100 次迭代后增加 $n_{\text{aug}} = 100$。对于 PINN 方法，除额外添加一个神经网络外，其神经网络架构与 AONN 相同

对于 $\zeta(\boldsymbol{\mu})$，使其满足以下 KKT系统：

$$\begin{cases} \text{state equation } (5.8), \\ \text{adjoint equation } (5.10), \\ u_1(\boldsymbol{\mu}) - \lambda_1(\boldsymbol{\mu}) + 2u_1(\boldsymbol{\mu})\zeta(\boldsymbol{\mu}) = 0, \\ u_2(\boldsymbol{\mu}) - \lambda_2(\boldsymbol{\mu}) + 2u_2(\boldsymbol{\mu})\zeta(\boldsymbol{\mu}) = 0, \\ (u_1(\boldsymbol{\mu})^2 + u_2(\boldsymbol{\mu})^2 - r^2)\zeta(\boldsymbol{\mu}) = 0, \\ u_1(\boldsymbol{\mu})^2 + u_2(\boldsymbol{\mu})^2 \le r^2, \zeta(\boldsymbol{\mu}) \ge 0, \end{cases} \tag{5.12}$$

其中 $\zeta(\boldsymbol{\mu})$ 是控制约束 (5.9) 的拉格朗日乘子。

我们在图5.3中比较 AONN 的解与使用 PINN 得到的解，并绘制它们的绝对误差。图中展示了代表性参数 $u = (u_1, u_2)$ 下的控制函数 $\boldsymbol{\mu} = 10$。从图中可以看出，即使 PINN 的训练轮次多于 AONN，AONN 仍能得到比 PINN 更精确的最优控制函数。此外，PINN 解未能很好地满足二次约束，因为物理信息神经网络损失函数还受到更多来自 KKT系统(5.12)的惩罚项。我们在每个参数 $\|u - u^*\|/\|u^*\|$ 对应的均匀 256×256 网格上计算相对误差 $\boldsymbol{\mu}$，并在图5.4中绘制结果。对于大多数参数，AONN 解的相对误差小于 PINN 的相对误差，这表明在求解参数化最优控制问题时，AONN 比 PINN 更有效、更高效。请注意，当参数 $\boldsymbol{\mu}$ 变小 [54]时，这个问题会变得更加困难。特别是当 $\boldsymbol{\mu}$ 趋近于 0时，*AONN* 和 *PINN* 的相对误差都很大。

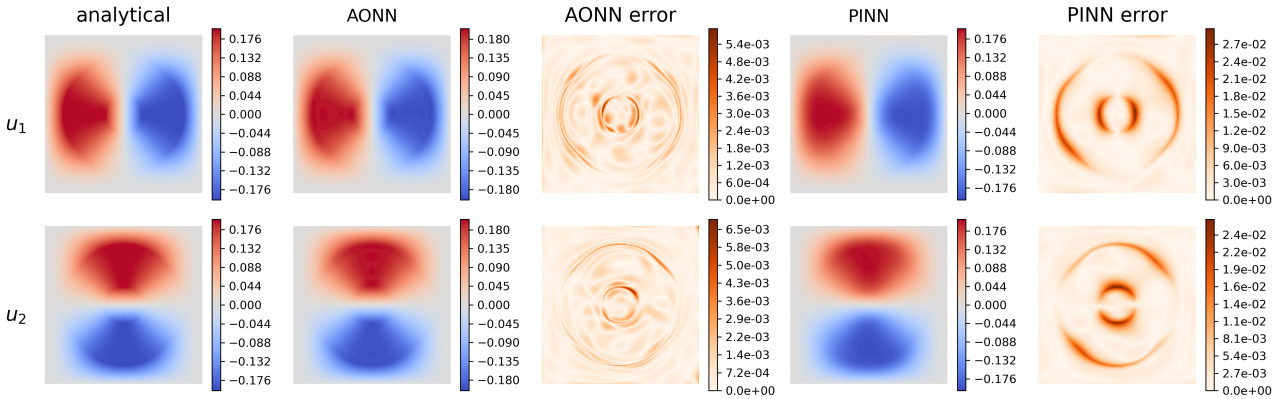


FIG. 5.3. *Test 3: optimal solutions of the state function* $y = y_1\ y_2$ *and the control function* $u = u_1\ u_2$ *obtained by AONN* (,) (,) *and PINN and their absolute errors for a given parameter* $\boldsymbol{\mu} = 10$.

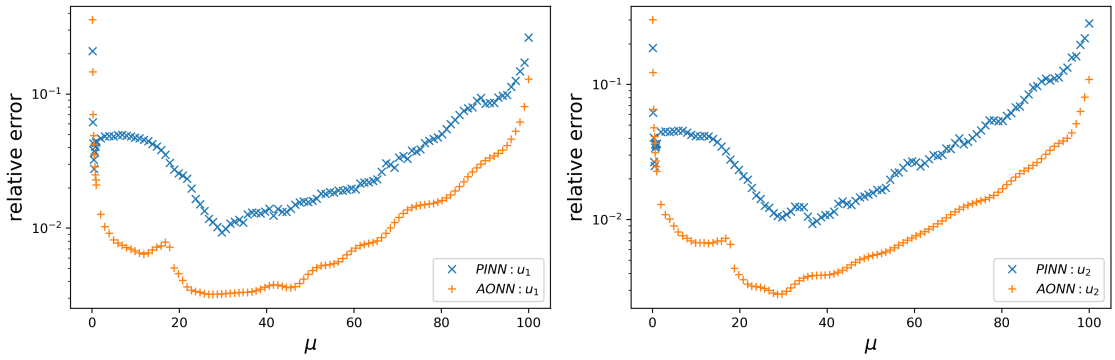


FIG. 5.4. *Test 3: the relative errors (in the* ℓ_2*-norm sense) of AONN and PINN for the two components of* $u(\boldsymbol{\mu})$ $= (u_1(\boldsymbol{\mu})\ u_2(\boldsymbol{\mu}))$. *The relative errors are computed on the* 256×256 *meshgrid for each fixed parameter* $\boldsymbol{\mu}$.

5.4. 测试 4：具有几何参数化的拉普拉斯方程最优控制。在这个测试案例中，我们将求解以下参数化最优控制问题：

$$\begin{cases} \min\limits_{y(\boldsymbol{\mu}),u(\boldsymbol{\mu})} J\left(y(\boldsymbol{\mu}),u(\boldsymbol{\mu})\right)=\dfrac{1}{2}\left\|y(\boldsymbol{\mu})-y_d(\boldsymbol{\mu})\right\|^2_{L_2(\Omega(\boldsymbol{\mu}))}+\dfrac{\alpha}{2}\left\|u(\boldsymbol{\mu})\right\|^2_{L_2(\Omega(\boldsymbol{\mu}))}, \\ \text{subject to } \begin{cases} -\Delta y(\boldsymbol{\mu})=u(\boldsymbol{\mu}) & \text{in } \Omega(\boldsymbol{\mu}), \\ y(\boldsymbol{\mu})=1 & \text{on } \partial\Omega(\boldsymbol{\mu}), \end{cases} \\ \text{and} \quad u_a\le u(\boldsymbol{\mu})\le u_b \quad \text{a.e. in } \Omega(\boldsymbol{\mu}), \end{cases} \tag{5.13}$$

其中，$\boldsymbol{\mu}=(\mu_1,\ \mu_2)$表示几何参数和期望状态参数。参数化计算域为 $\Omega(\boldsymbol{\mu})=([0,2]\times[0,1])\backslash B((1.5,\ 0.5),\ \mu_1)$，期望状态定义为

$$y_d(\boldsymbol{\mu})=\begin{cases} 1 & \text{in } \Omega_1=[0,1]\times[0,1], \\ \mu_2 & \text{in } \Omega_2(\boldsymbol{\mu})=([1,2]\times[0,1])\backslash B((1.5,0.5),\mu_1), \end{cases} \tag{5.14}$$

其中，$B((1.5,\ 0.5),\ \mu_1)$是一个半径为 μ_1、圆心为 $(1.5,\ 0.5)$ 的球。我们设定 $\alpha=0.001$，参数区间为 $\boldsymbol{\mu}\in\mathcal{P}=[0.05,0.45]\times[0.5,2.5]$。

此测试案例受到文献 [36, 23] 的启发，该文献涉及癌症局部热疗的应用。在这种情况下，预期通过热源控制，使肿瘤区域达到特定温度场，非病灶区域达到另一温度场。从矩形区域中裁出的圆形代表某个人体器官，我们旨在利用 AONN 获得针对不同期望温度场和不同器官形状的最优热源控制整体求解方案。具体而言，我们考虑一个与癌症热疗相对应的二维模型问题。该问题的一个难点是几何参数 μ_1 会导致计算域各不相同，从而给传统基于网格的数值方法带来困难。在 AONN 框架中，我们可以通过在时空参数空间中采样来求解该问题：

$$\Omega_{\mathcal{P}}=\{(x_0,\ x_1,\ \mu_1,\ \mu_2)|0\le x_0\le 2,\ 0\le x_1\le 1,\ 0.05\le\mu_1\le 0.45,\ 0.5\le\mu_2\le 2.5,\ (x_0-1.5)^2+(x_1-0.5)^2\ge\mu_1^2\}$$

计算域 $\Omega(\boldsymbol{\mu})$ 以及 40000 个训练点见图 5.5(a)和图 5.5(b)。状态神经网络 $\hat{}y$ 由 $\hat{}y(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I})=\ell(\mathbf{x}\ \boldsymbol{\mu})\hat{}y_I(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I})+1$ 构造而成，从而自然满足狄利克雷边界条件（5.13），其中长度因子函数为

$$\ell(\mathbf{x},\boldsymbol{\mu})=x_0(2-x_0)x_1(1-x_1)(\mu_1^2-(x_0-1.5)^2-(x_1-0.5)^2).$$

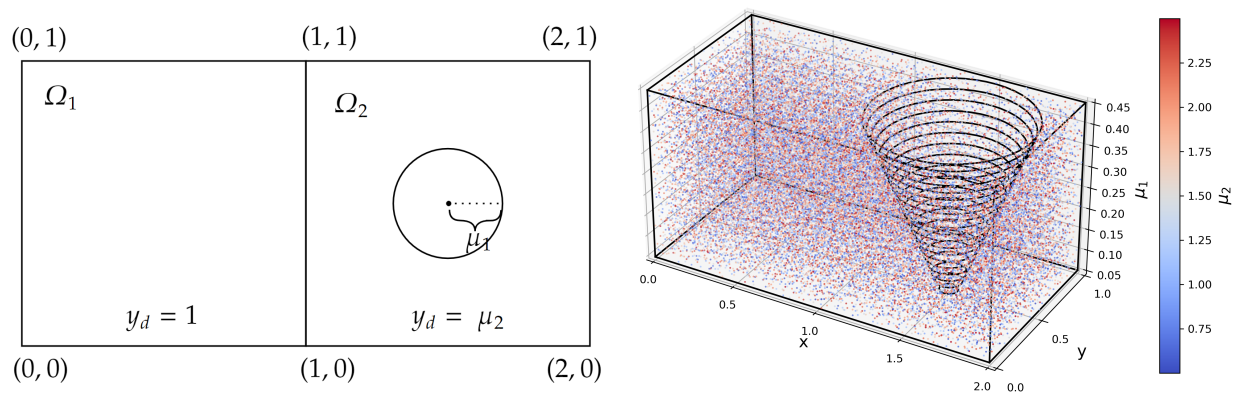
三个神经网络 $\hat{}y_I(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{y_I})$ 、$\hat{}p_I(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\theta}_{p_I})$ 和 $\hat{}u(\mathbf{x}(\boldsymbol{\mu});\ \boldsymbol{\theta}_u)$ 均由三个 ResNet 块组成，每个块包含两个具有 25 个单元的全连接层和一个残差连接。这三个神经网络的输入维数为 4，参数总数为 $3\times3401=10203$。我们取 $\gamma=0.985$，并且在训练过程中，状态函数和伴随函数的训练轮次从 200 增加到 700。PINN 和 PINN+投影方法的神经网络配置与 AONN 相同，训练轮次为 50000。对于这个测试问题，AONN算法在 300 步内收敛。请注意，我们的 AONN 方法能够针对任意参数 $\boldsymbol{\mu}$ 获得同步求解结果。为了评估 AONN 的性能，我们采用经典有限元方法，在参数固定的情况下求解最优控制问题$(\boldsymbol{\mu})$。具体而言，我们采用 dolfin-adjoint [33] 中实现的带有界支撑的有限内存BFGS算法，求解相应的最优控制问题。使用 dolfin-adjoint 求得的解可视为真实解。在这四种方法中，AONN、PINN 和 PINN+投影方法能够求解参数化最优控制问题，而 dolfin-adjoint 求解器只能求解参数固定的最优控制问题。

图5.6 展示了使用 AONN 求解参数化最优控制问题（5.13）得到的最优控制解。我们选择了几个不同的参数 $\boldsymbol{\mu}$ 进行可视化。图5.6左列对应于 $\mu_2=1$，此时最优控制恰为零，因为当 $y=y_d\equiv 1$时，期望状态可以实现。图5.6 的中列和右列表明，减小 μ_1 并增大 μ_2 可能会增大 u的幅值。图中展示了使用不同值时 dolfin-adjoint 求解器、AONN、物理信息神经网络和物理信息神经网络+投影方法得到的结果 c，如图5.7所示，其中对比了$\boldsymbol{\mu}=(0.3,2.5)$处的控制函数。dolfin-adjoint 求解器采用了包含 138604 个三角形单元的网格，经过 16 步后，最终投影梯度范数达到2.379×10^{-10}。图5.7 显示，AONN 可以

收敛到 dolfin-adjoint 求解器得到的参考解，而物理信息神经网络无法得到精确解；物理信息神经网络+投影方法的结果则严重依赖于 c 在（4.6）中的取值。当 c 不等于 $1/\alpha = 1000$时，物理信息神经网络+投影方法无法保证收敛到参考解。这也证实了变分损失（3.7）给 KKT 系统（4.6）的神经网络训练带来了很大困难，除非 $c = 1/\alpha$；在这种情况下，控制函数 u 在投影算子内部会被抵消。

$$\mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right) = \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c(\alpha u(\boldsymbol{\mu}) + p(\boldsymbol{\mu}))\right) = \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(-\frac{1}{\alpha}p(\boldsymbol{\mu})\right).$$

然而，对于下面这个非光滑测试问题， u 无法从变分损失中分离出来（对任意 c 均如此），因此 PINN+ Projection 方法失效。为展示 AONN 能够实现整体求解方案，我们首先对 $\mathcal{P}$ 取一个 100×100 网格，并选取 $\boldsymbol{\mu}$ 的若干种不同实现，使用 dolfin-adjoint 求解器求解相应的最优控制问题。然后，将网格上的参数连同空间坐标输入由算法 3.1 得到的训练神经网络，从而一次性获得最优控制问题($\boldsymbol{\mu}$)的最优解。值得注意的是，使用 dolfin-adjoint 求解器计算 100×100 网格上所有参数对应的最优解，计算开销很大，因为这需要进行 10000 次仿真。因此，我们仅在网格上选取 16 个有代表性的点，使用 dolfin-adjoint 求解器进行计算（这仍需数小时）。尽管如此，通过我们的全即神经网络框架，仍可高效地计算整体求解方案。图 5.8 展示了关于 μ_1、μ_2的三个量，其中图 5.8(a) 显示目标泛函 J，图 5.8(b) 显示目标状态的可达性，图 5.8(c) 显示最优控制 u的 L_2-范数。图 5.8 中的红点表示由 dolfin-adjoint 求解器得到的结果，其中进行了 16 次最优控制问题仿真，参数为 $(\mu_1, \mu_2) \in \{0.05, 0.1833, 0.3167, 0.45\} \times \{0.5, 1.1667, 1.8333, 2.5\}$。从图 5.8 可以看出，AONN 能够获得准确的解。



(a) (b)

FIG. 5.5. *Test 4: (a) The parametric computational domain* Ω*((there are no points inside the frustum).* $\boldsymbol{\mu}$ $\mathcal{P}$ *). (b)* $N = 40000$ *training collocation points sampled in* Ω

5.5. 测试 5：具有稀疏性参数化的半线性椭圆方程最优控制。在本测试问题中，我们再次考虑与（5.1）中相同的半线性椭圆方程控制问题。不过，这次我们考虑的是具有稀疏性参数化的稀疏最优控制问题。最优控制中的稀疏解通常通过 L_1-控制代价 [5, 6, 4] 实现，其在控制器布置问题中的应用已得到充分研究 [46]。具体而言，我们考虑如下具有 L_1-控制代价的目标泛函：

$$J(y, u) = \frac{1}{2}\|y - y_d\|_{L_2}^2 + \frac{\alpha}{2}\|u\|_{L_2}^2 + \beta\|u\|_{L_1},$$

其中， L_1项的系数 β 控制控制函数 u的稀疏性。随着 β增大，最优控制逐渐变得稀疏，最终达到零。为了连续观察这一现象，我们将 β 设为变量，求解以下参数化最优控制问题

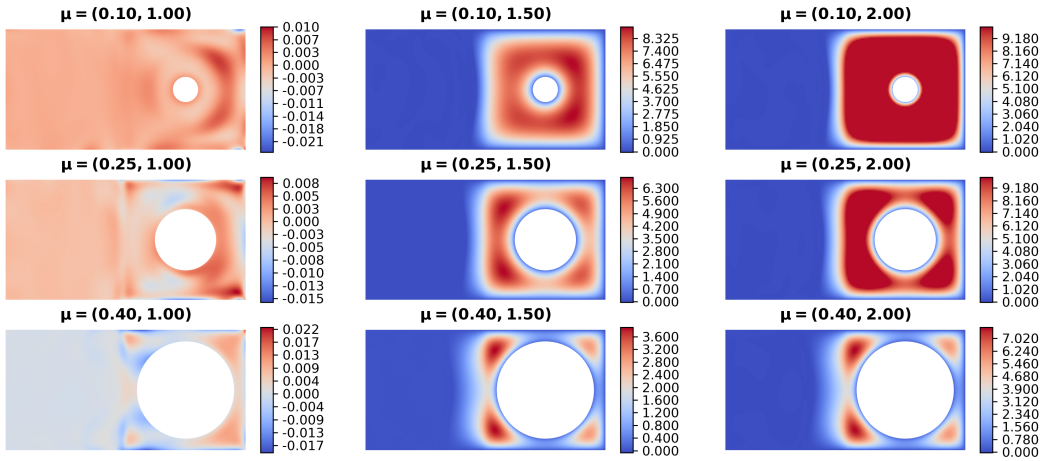


FIG. 5.6。 *Test 4: the AONN solutions* $u(\boldsymbol{\mu})$ *with several realizations of* $\boldsymbol{\mu} = (\mu_1, \mu_2)$ 。

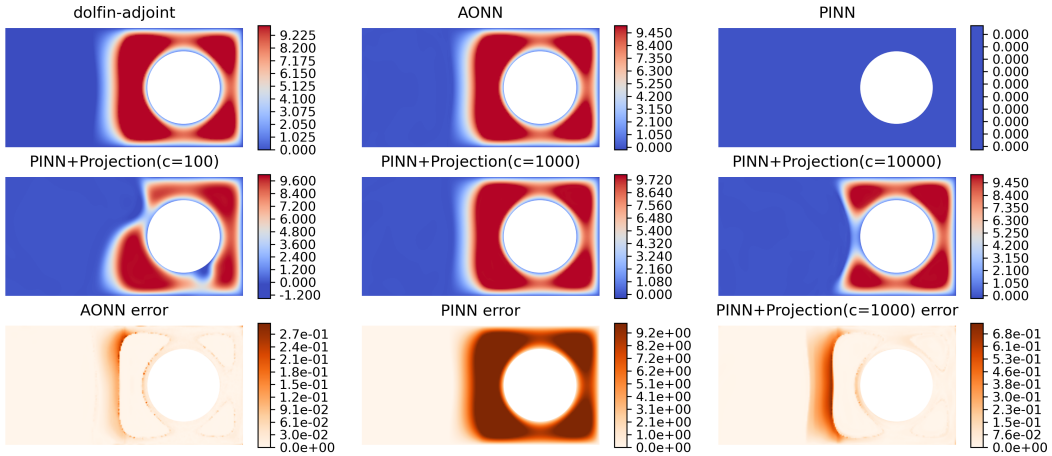


FIG. 5.7。 *Test 4: the solution obtained by the dolfin-adjoint solver for a fixed parameter* $\boldsymbol{\mu} = (0。3, 2。5)$ *the approximate solutions of* u *obtained by AONN PINN PINN+Projection (with different* $c = 100, 1000, 10000$*) and the absolute errors of the ,,, AONN solution and the PINN+Projection solution with* $c = \frac{1}{\alpha} = 1000$.

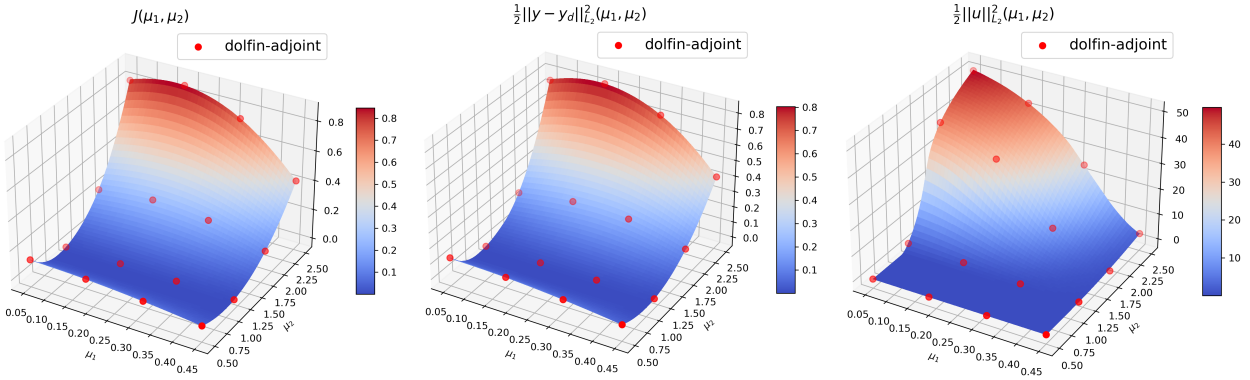


FIG. 5.8 *Test 4: several quantities as functions with respect to parameter* $\boldsymbol{\mu} = \mu_1\ \mu_2$ *obtained by AONN. Each red dot。 (,) denotes the quantity corresponding to a specific* $\boldsymbol{\mu}$ *computed from the dolfin-adjoint solver. (a) Objective value:* J *(b) Attainability of the desired state:* $\frac{1}{2}\|y - y_d\|^2_{L_2}$. *(c)* L_2*-norm of control function:* $\frac{1}{2}\|u\|^2_{L_2}$.

参数 $\boldsymbol{\mu}$,

$$\begin{cases} \min\limits_{y(\boldsymbol{\mu}),u(\boldsymbol{\mu})} J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu}) := \dfrac{1}{2}\|y(\boldsymbol{\mu})-y_d\|^2_{L_2(\Omega)} + \dfrac{\alpha}{2}\|u(\boldsymbol{\mu})\|^2_{L_2(\Omega)} + \boldsymbol{\mu}\|u(\boldsymbol{\mu})\|_{L_1(\Omega)}, \\ \text{subject to } \begin{cases} -\Delta y(\boldsymbol{\mu}) + y(\boldsymbol{\mu})^3 = u(\boldsymbol{\mu}) & \text{in } \Omega, \\ y(\boldsymbol{\mu}) = 0 & \text{on } \partial\Omega, \end{cases} \\ \text{and} \quad u_a \le u(\boldsymbol{\mu}) \le u_b \quad \text{a.e. in } \Omega. \end{cases} \tag{5.15}$$

我们固定其他参数

$$\begin{aligned} &\Omega = B(0,1), \\ &\alpha = 0.002, u_a = -12, u_b = 12, \\ &y_d = 4\sin(2\pi x_1)\sin(\pi x_2)\exp(x_1), \end{aligned}$$

并将参数的取值范围设为 $\boldsymbol{\mu} \in [0, \boldsymbol{\mu}_{max}]$。上界 $\boldsymbol{\mu}_{max} = 0.128$ 确保对于任意 $\boldsymbol{\mu} \ge \boldsymbol{\mu}_{max}$，最优控制 $u^*(\boldsymbol{\mu})$ 恒为零。我们计算广义导数

$$\mathrm{d}_u J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu});\boldsymbol{\mu}) = \alpha u(\boldsymbol{\mu}) + p(\boldsymbol{\mu}) + \boldsymbol{\mu}\ \mathrm{sign}(u(\boldsymbol{\mu})). \tag{5.16}$$

其中 p 是式（5.2）中定义的伴随方程的解，而 sign 是一个逐元素运算符，用于提取函数的符号函数。

由于最优控制函数会随不同的 $\boldsymbol{\mu}$ 而变化，通常直接求解一系列稀疏最优控制问题并不困难。例如，Eduardo Cases [4] 计算了 $\boldsymbol{\mu} = 2^i \times 10^{-3}$，$i$ =0、1、.....、8 时的最优解。在这里，我们使用 AONN 一次性计算任意 $\boldsymbol{\mu} \in [0, 0.128]$ 下的所有最优解。狄利克雷边界条件的长度因子函数选为 $\ell(\mathbf{x}) = 1 - x_0^2 - x_1^2$。AONN 对神经网络 $\hat{}y_I(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{y_I})$ 、$\hat{}p_I(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_{p_I})$ 和 $\hat{}u(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u)$ 进行训练，其配置与前一项测试相同，但输入维数为 3，因此包含 $3 \times 3376 = 10128$ 个待定参数。为此，我们从时空参数空间 $\Omega_{\mathcal{P}} = B(0,\ 1) \times [0, \boldsymbol{\mu}_{max}]$ 中按均匀分布抽取 $N = 20000$ 个点。为捕捉 $\boldsymbol{\mu}$ 边界处的信息，在 $B(0,\ 1) \times \{0\}$ 和 $B(0,\ 1) \times \{\boldsymbol{\mu}_{max}\}$ 中抽取了这 20000 个点中的 2000 个。我们进行 500 个迭代步，并在每 100 个迭代步后将训练轮次逐渐增加 $n_{\text{aug}} = 100$ 。因此，训练过程中，状态函数和伴随函数的训练轮数从 200 增至 600。步长 c^k 初始设为 $c^0 = 10$ ，并在每个迭代步后按因子 $\gamma = 0.985$ 递减。

AONN 计算得到的某些代表性 $\boldsymbol{\mu} \in [0, \boldsymbol{\mu}_{max}]$ 的最优控制如图5.9所示。AONN 的结果与文献[4], 中的结果一致：随着 $\boldsymbol{\mu}$ 增大，最优控制的稀疏性也随之增强。如图5.9所示， $\boldsymbol{\mu} = 0$ 的初始最优控制有八个峰值，随着 $\boldsymbol{\mu}$ 增大，每个峰值都会消失。为了确定控制装置的最高效布置位置，可能需要手动调整 $\boldsymbol{\mu}$ ，因此需要针对不同的 $\boldsymbol{\mu}$多次求解最优控制问题。如果已获得参数化解 $u^*(x, y, \boldsymbol{\mu})$，确定这些最优位置就很容易，而这正是 AONN 所实现的。通过在 $[-1,1] \times [-1,1] \times [0, \boldsymbol{\mu}_{max}]$上取均匀的 100^3 网格，考察 $\boldsymbol{\mu}$ 增大时 $u^*(x, y, \boldsymbol{\mu})$最后消失的位置，即可得到八个峰值的坐标。图5.10展示了八个峰值处的控制值随 $\boldsymbol{\mu}$变化的情况。我们观察到，点P_1、P_2、P_3 处的控制函数值初始为 12，并在 $\boldsymbol{\mu}$ 达到某个值后开始减小，最终降至零。 P_4处的值从一个小于 12 的数开始下降，直至降为零。点 $P_5 \sim P_8$ 处的变化则完全对称。

总之，通过这五个数值测试，我们考察了 AONN 的效率，并将其性能与 PINN、PINN+Projection 以及传统求解器进行了比较。数值结果表明，在求解参数化最优控制问题时，所提出的 AONN 方法优于 PINN+Projection 方法和 PINN 方法。PINN 方法无法为复杂约束问题获得精确解；PINN+projection 方法总体上提高了 PINN 方法的精度，但在稀疏最优控制问题等非光滑问题上存在局限性，而 AONN 方法是一个通用框架，在不同类型的参数化最优控制问题上表现更佳。

6. 结论。我们提出了 AONN，即一种伴随导向神经网络方法，用于计算参数化最优控制问题的整体求解方案。也就是说，只需进行一次求解，即可获得任意参数下的最优控制解。AONN 的核心思想是在最优性条件中采用三个神经网络，分别逼近控制函数、伴随函数和状态函数，这

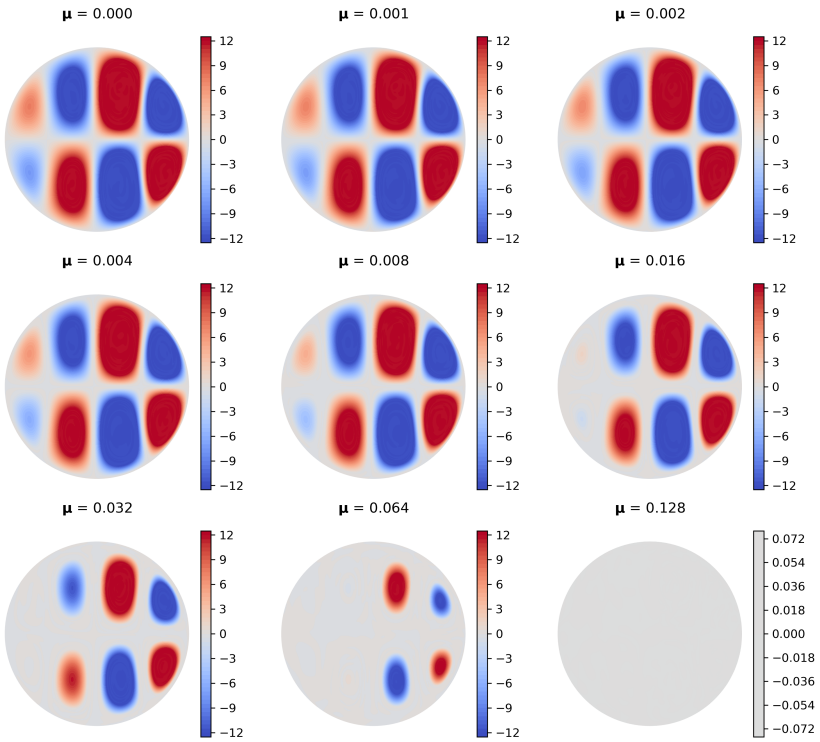


FIG. 5.9。 *Test 5: the AONN solutions* $u\ (\boldsymbol{\mu})$ *of representative values for* $\boldsymbol{\mu} = 2^i \times 10^{-3}, i = 0, 1, \ldots, 8$.

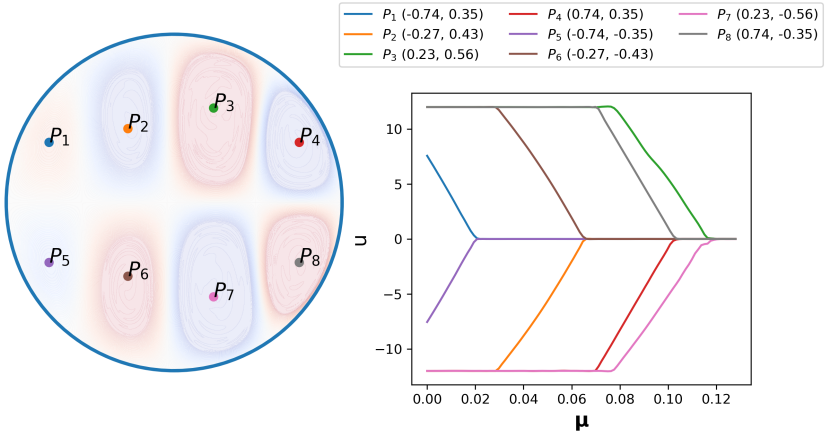


FIG. 5.10 *Test 5: the AONN solution* $u\ \boldsymbol{\mu}$ *of eight fixed peaks* $P_1 \sim P_8$ *as a function respect to* $\boldsymbol{\mu}$. *The legend on the right is* 。 0 *the coordinates of the eight points.*

使该方法能够将直接-伴随循环（DAL）方法的思想融入神经网络逼近中。通过这种方式，三个基于神经网络的参数化代理模型提供了最优解的整体表示，避免了空间和参数空间的网格生成，因此可以推广到高维问题。通过融入 DAL，AONN 还避免了复杂 Karush–Kuhn–Tucker（KKT）系统中的基于惩罚的损失函数，从而降低了神经网络的训练难度并提高了解的精度。数值实验表明，在包括控制参数、物理参数、模型参数和几何参数在内的多种应用场景中，AONN 都能高精度地整体求解参数化最优控制问题。

许多问题仍有待解决，例如，步长和缩放因子的选择仍依赖启发式方法，而求解某些复杂问题则需要更高的计算成本。未来的研究可包括分析收敛速度，以更好地理解 AONN 的性质，并引入自适应采样策略

以进一步提升鲁棒性和有效性，以及将 AONN 推广并应用于更具挑战性的问题，例如形状或拓扑优化。

致谢。 本研究部分获得国家自然科学基金委员会（#12131002）和中国博士后科学基金（2022M711730）资助。

参考文献

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems* 1603 04467, 2016 , arXiv 预印本 arXiv: . ( ).

[2] A. A. Ali and M. Hinze, *Reduced basis methods—an application to variational discretization of parametrized elliptic optimal control problems* 42 2020 271 291 , SIAM 科学计算杂志, ( ), pp. A –A .

[3] E. Bader, M. Kaercher, M. A. Grepl, and K. Veroy, *Certified reduced basis methods for parametrized distributed elliptic optimal control problems with control constraints* 38 2016 3921 3946 , SIAM 科学计算杂志, ( ), pp. A –A .

[4] E. Casas, *A review on sparse solutions in optimal control of partial differential equations*, SeMA 期刊 , 74 (2017), pp. 319– 344.

[5] E. Casas, R. Herzog, and G. Wachsmuth, *Approximation of sparse controls in semilinear elliptic equations*, 载于国际大规模科学计算会议, Springer, 2011, pp. 16–27.

[6] E. Casas, R. Herzog, and G. Wachsmuth, *Optimality conditions and error analysis of semilinear elliptic control problems with L^1 cost functional* 22 2012 795 820 , SIAM 优化杂志, ( ), pp. – .

[7] X. Chen, X. Song, Z. Chen, and B. Yu, *A multi-level admm algorithm for elliptic PDE-constrained optimization problems*, 计算与应用数学, 39 (2020), pp. 1–31.

[8] Z. Chen, X. Song, X. Zhang, and B. Yu, *A PE-ADMM algorithm for Lavrentiev-regularized state-constrained elliptic control problem* 25 2019 5 , ESAIM：控制、优化与变分法, ( ), p. .

[9] J. C. De los Reyes, *Numerical PDE-constrained optimization*, Springer, 2015.

[10] J. C. de los Reyes and F. Tröltzsch, *Optimal control of the stationary Navier–Stokes equations with mixed control-state constraints* , SIAM 控制与优化杂志, 46 2007 604 629 ( ), pp. – .

[11] N. Demo, M. Strazzullo, and G. Rozza, *An extended physics informed neural network for preliminary analysis of parametric optimal control problems* 2110 13530, 2021 , arXiv 预印本 arXiv: . ( ).

[12] W. E and B. Yu, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, 数学与统计学通讯 , 6 (2018), pp. 1–12.

[13] Z. Gao, L. Yan, and T. Zhou, *Failure-informed adaptive sampling for PINNs*, arXiv 预印本 arXiv:2210.00279, (2022).

[14] W. Gong, H. Xie, and N. Yan, *A multilevel correction method for optimal controls of elliptic equations*, SIAM 科学计算杂志, 37 (2015), pp. A2198–A2221.

[15] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, *A physics-informed deep learning framework for inver-sion and surrogate modeling in solid mechanics*, 《计算机方法在应用力学与工程中的应用》, 379 (2021), p. 113741.

[16] J. Han, A. Jentzen, and E. Weinan, *Solving high-dimensional partial differential equations using deep learning*, 美国国家科学院院刊, 115 (2018), pp. 8505–8510.

[17] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 载于 IEEE 计算机视觉与模式识别会议论文集, 2016, pp. 770–778.

[18] J. S. Hesthaven, G. Rozza, B. Stamm, et al., *Certified reduced basis methods for parametrized partial differential equations*, 第590卷, Springer, 2016.

[19] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, 第23卷, 施普林格科学与商业媒体, 2008.

[20] R. Hwang, J. Y. Lee, J. Y. Shin, and H. J. Hwang, *Solving PDE-constrained control problems using operator learning*, 载于 AAAI 人工智能会议论文集, 第36卷, 2022, pp. 4504–4512.

[21] A. Jameson, *Aerodynamic design via control theory*, 科学计算杂志, 3 (1988), pp. 233–260.

[22] M. Kärcher, *Certified reduced basis methods for parametrized PDE-constrained optimization problems*, 博士论文, Univer- sit¨atsbibliothek der RWTH Aachen, 2016.

[23] M. Kärcher, Z. Tokoutsi, M. A. Grepl, and K. Veroy, *Certified reduced basis methods for parametrized elliptic optimal control problems with distributed controls* 75 2018 276 307 , 科学计算杂志, ( ), pp. – .

[24] R. Khodayi-Mehr and M. Zavlanos, *VarNet: Variational neural networks for the solution of partial differential equations*, 载于动力学与控制学习, PMLR, 2020, pp. 298–307.

[25] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv 预印本 arXiv:1412.6980, (2014).

[26] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, *Characterizing possible failure modes in physics-informed neural networks* 34 2021 26548 26560 , 神经信息处理系统进展, ( ), pp. – .

[27] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE 神经网络汇刊, 9 (1998), pp. 987–1000.

[28] J. Löwe, G. Lube, and L. Röhe, *A Projection-Based variational multiscale method for the incompressible Navier–Stokes/Fourier model*, 载于 BAIL 2010-边界层与内层、计算与渐近方法, Springer, 2011, pp. 167–175.

[29] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, *Physics-informed neural networks with hard constraints for inverse design* 43 2021 1105 1132 , SIAM 科学计算杂志, ( ), pp. B –B .

[30] K. O. Lye, S. Mishra, D. Ray, and P. Chandrashekar, *Iterative surrogate model optimization (ISMO): an active learning algorithm for PDE constrained optimization with deep neural networks*, 《计算机方法在应用力学与工程中的应用》, 374 (2021), p. 113575.

[31] R. Milani, A. Quarteroni, and G. Rozza, *Reduced basis method for linear elasticity problems with many parameters*, 《计算机方法在应用力学与工程中的应用》, 197 (2008), pp. 4812–4829.

[32] S. Mitter and J. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer Berlin Heidelberg, 1971.
[33] S. K. Mitusch, S. W. Funke, and J. S. Dokken, *dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake*, 《开放源代码软件期刊》, 4 (2019), p. 1292. [34] S. Mowlavi and S. Nabi, *Optimal control of PDEs using physics-informed neural networks*, 《计算物理学杂志》, (2022), p. 111731. [35] F. Negri, A. Manzoni, and G. Rozza, *Reduced basis approximation of parametrized optimal flow control problems for the Stokes equations* 69 2015 319 336 , 《计算机与应用数学》, ( ), pp. – . [36] F. Negri, G. Rozza, A. Manzoni, and A. Quarteroni, *Reduced basis method for parametrized elliptic optimal control problems* 35 2013 2316 2340 , SIAM 科学计算杂志, ( ), pp. A –A . [37] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer, 1999. [38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch* 2017 , ( ). [39] R. Pinnau, C. Totzeck, and O. Tse, *The quasi-neutral limit in optimal semiconductor design*, SIAM 控制与优化杂志, 55 (2017), pp. 2603–2635.
[40] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced basis methods for partial differential equations: an introduction*, 第92卷, Springer, 2015. [41] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, 《计算物理学杂志》, 357 (2018), pp. 125–141. [42] 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*, 《计算物理学杂志》, 378 (2019), pp. 686–707. [43] G. Rozza, A. Manzoni, and F. Negri, *Reduction strategies for PDE-constrained optimization problems in haemodynamics*, 载于第六届计算科学与工程欧洲大会论文集, no. CONF, 维也纳工业大学, 2012, pp. 1748–1769. [44] H. Sheng and C. Yang, *PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries* 428 2021 110085 , 《计算物理学杂志》, ( ), p. . [45] H. Sheng and C. Yang, *PFNN-2: A domain decomposed penalty-free neural network method for solving partial differential equations* 2205 00593, 2022 , arXiv 预印本 arXiv: . ( ).
[46] G. Stadler, *Elliptic optimal control problems with L1-control cost and applications for the placement of control devices*, 《计算优化与应用》, 44 (2009), pp. 159–181. [47] J. Sternberg and M. Hinze, *A memory-reduced implementation of the Newton-CG method in optimal control of nonlinear time-dependent PDEs* 25 2010 553 571 , 《优化方法与软件》, ( ), pp. – . [48] M. Strazzullo, F. Ballarin, R. Mosetti, and G. Rozza, *Model reduction for parametrized optimal control problems in environmental marine sciences and engineering* 40 2018 1055 1079 , SIAM 科学计算杂志, ( ), pp. B –B . [49] L. Sun, H. Gao, S. Pan, and J.-X. Wang, *Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data* 361 2020 112732 , 《计算机方法在应用力学与工程中的应用》, ( ), p. . [50] K. Tang, X. Wan, and Q. Liao, *Adaptive deep density approximation for Fokker-Planck equations*, 《计算物理学杂志》, 457 (2022), p. 111080. [51] K. Tang, X. Wan, and C. Yang, *DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations* 2112 14038, 2021 , arXiv 预印本 arXiv: . ( ).
[52] F. Tröltzsch, *Optimal control of partial differential equations: theory, methods, and applications*, 第 112 卷 , 美国数学学会 , 2010.
[53] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser J. Bright et al. *SciPy 1.0: fundamental algorithms for scientific computing in python* , , , , 《自然·方法》, 17 (2020), pp. 261–272. [54] D. Wachsmuth, *Optimal control of the unsteady Navier-Stokes equations*, (2006). [55] S. Wang, M. A. Bhouri, and P. Perdikaris, *Fast PDE-constrained optimization via self-supervised operator learning*, arXiv 预印本 arXiv:2110.13297, (2021). [56] M. Xu, S. Song, X. Sun, W. Chen, and W. Zhang, *Machine learning for adjoint vector in aerodynamic shape optimization*, 《力学学报》, (2021), pp. 1–17. [57] H. Yang, F.-N. Hwang and X.-C. Cai *Nonlinear preconditioning techniques for full-space Lagrange–Newton solution of PDE-constrained optimization problems* 38 2016 2756 2778 , , , SIAM 科学计算杂志, ( ), pp. A –A . [58] H. Yang, S. Sun, and C. Yang, *Nonlinearly preconditioned semismooth Newton methods for variational inequality solution of two-phase flow in porous media* 332 2017 1 20 , 《计算物理学杂志》, ( ), pp. – . [59] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, *Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data*, 《计算物理学杂志》, 394 (2019), pp. 56–81.