

傅里叶神经算子(Fourier Neural Operator, FNO)是当前算子学习领域中受到关注最多的工作之一. FNO不仅是一个表示能力比较强的神经网络模型, 而且当训练数据来源于网格点时, 可以利用FFT提高模型训练和推理的效率. 然而, 当训练数据并不是网格点时, 无法利用FFT进行加速. 这使得FNO难以高效地求解不规则区域上的问题. 针对于FNO现存的局限, 我们借鉴有限体积法cut cell离散格式构造损失函数, 使得在求解不规则计算区域上的方程时也可以使用规则的直角网格, 从而可以利用FFT进行加速.

1 背景介绍

1.1 Fourier neural operator

FNO¹ 学习两个函数空间之间的映射记两个函数空间分别为 $\mathcal{A} = \mathcal{A}(D; \mathbb{R}^{d_a})$ 和 $\mathcal{U} = \mathcal{U}(D; \mathbb{R}^{d_u})$, 其中 $D \subset \mathbb{R}^{d_x}$ 是计算区域, $d_x, d_a, d_u \in \mathbb{N}_+$, FNO构造映射 G_θ 以学习映射 $G^\dagger : \mathcal{A} \rightarrow \mathcal{U}$.

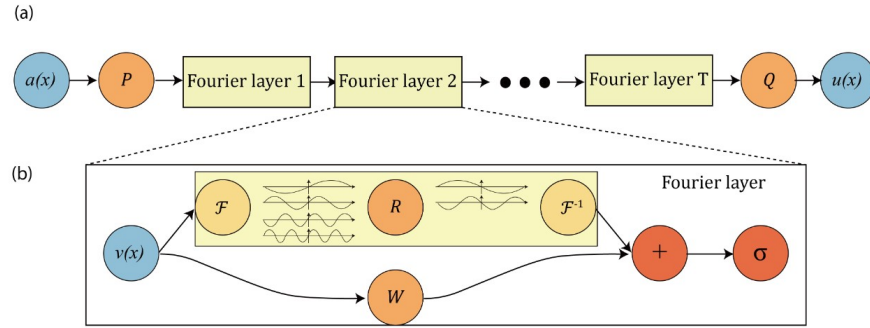


Figure 1: 傅里叶神经算子

如图1所示, FNO接受 \mathcal{A} 中的函数 a 作为输入, 首先通过全连接层 \mathcal{P} 进行升维得到输出 v_0 ($v_0 \in \mathbb{R}^{d_v}$), 然后通过 L 个傅里叶层进行非线性变换, 依次得到输出 v_1, \dots, v_L ($v_1, \dots, v_T \in \mathbb{R}^{d_v}$), 最后通过全连接层 \mathcal{Q} 进行降维得到最终输出 u . 每个傅里叶层的构造为

$$v_{\ell+1}(\mathbf{x}) := \sigma\left(\mathcal{W}v_\ell(\mathbf{x}) + \mathcal{K}v_\ell(\mathbf{x})\right), \quad (1)$$

其中 \mathcal{W} 是线性变换算子, \mathcal{K} 是积分变换算子. 从而网络 \mathcal{G}_θ 的构造为:

$$\mathcal{G}_\theta = \mathcal{Q} \circ \sigma(\mathcal{W}_L + \mathcal{K}_L) \circ \dots \circ \sigma(\mathcal{W}_1 + \mathcal{K}_1) \circ \mathcal{P} \quad (2)$$

傅里叶层1中积分变换算子 \mathcal{K} 的定义为:

$$[\mathcal{K}v_\ell](\mathbf{x}) := \int_D \kappa(\mathbf{x}, \mathbf{y}) v_\ell(\mathbf{y}) d\mathbf{y}, \quad (3)$$

其中 $\kappa(\mathbf{x}, \mathbf{y})$ 是核函数. FNO通过学习核函数 $\kappa(\mathbf{x}, \mathbf{y})$ 逼近映射 \mathcal{G} . 引入傅里叶变换和逆变换

$$(\mathcal{F}f)_j(\boldsymbol{\lambda}) = \int_D f_j(\mathbf{x}) \exp(-2i\pi\langle \mathbf{x}, \boldsymbol{\lambda} \rangle) d\mathbf{x}, \quad (\mathcal{F}^{-1}f)_j(\mathbf{x}) = \int_D f_j(\boldsymbol{\lambda}) \exp(2i\pi\langle \mathbf{x}, \boldsymbol{\lambda} \rangle) d\boldsymbol{\lambda}, \quad (4)$$

积分变换(??)等价于

$$[\mathcal{K}v_\ell](\mathbf{x}) := \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}(v_\ell))(\mathbf{x}) \quad (5)$$

¹Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint, arXiv:2010.08895 (2020).

离散情形下, 采用神经网络直接参数化 κ 的傅里叶变换 $R_\phi = \mathcal{F}(\kappa)$, 即将 R_ϕ 参数化为一个大小为 $n_k \times d_v \times d_v$ 的复值张量, 其中 n_k 是傅里叶级数的模式数目 (保留前 n_k 个模式). 这样 R 与 $\mathcal{F}v_\ell$ 的内积为

$$(R \cdot (\mathcal{F}v_\ell))_{k,l} = \sum_{j=1}^{d_v} (R)_{k,l,j} (\mathcal{F}v_\ell)_{k,j}.$$

如果训练数据来源于网格点时, 可以利用FFT加速计算.

1.2 Physics-informed neural operator (PINO)

PINO² 根据方程信息构造损失函数. 在计算损失函数时, 需要计算神经网络关于输入坐标点的导数. PINO采用了两种方式计算导数, 其中一种是差分近似, 另一种是根据链式法则计算, 根据(2)中FNO的构造

$$\mathcal{G}_\theta = \mathcal{Q} \circ \sigma(\mathcal{W}_L + \mathcal{K}_L) \circ \cdots \circ \sigma(\mathcal{W}_1 + \mathcal{K}_1) \circ \mathcal{P},$$

忽略线性变换 \mathcal{W}_L , 有

$$u(\mathbf{x}) = \mathcal{Q} \circ \mathcal{K}_L v_L(\mathbf{x}) = \mathcal{Q} \circ \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}v_{L-1})(\mathbf{x}) = \mathcal{Q} \circ \left(\frac{1}{n_k} \sum_{k=1}^{n_k} \sum_j (R)_{k, \cdot, j} (\mathcal{F}v_{L-1})_{k,j} \exp(2i\pi \langle \boldsymbol{\lambda}^k \mathbf{x} \rangle) \right). \quad (6)$$

根据链式法则,

$$\frac{\partial u}{\partial x_m} = \nabla \mathcal{Q} \cdot \left(\frac{1}{n_k} \sum_{k=1}^{n_k} \sum_j (R)_{k, \cdot, j} (\mathcal{F}v_{L-1})_{k,j} 2i\pi \lambda_m^k \exp(2i\pi \langle \boldsymbol{\lambda}^k \mathbf{x} \rangle) \right). \quad (7)$$

当计算网格是均匀网格时, (7)的计算也可以通过FFT进行加速. 高阶导数的计算也是类似的.

(7)在计算导数时只考虑了最后一个傅里叶层, 正确性尚且存疑. 不过目前已通过一些数值实验证实, 按照(7)计算导数, 算法也能基本收敛.

1.3 Geometry-aware Fourier neural operator (Geo-FNO)

当计算网格不是均匀网格时, 无法利用FFT进行加速. 这使得FNO难以高效求解不规则区域上的问题. 对此, Geo-FNO³ 尝试通过引入坐标变换以拓宽算法的适用范围. 对于一些非矩形区域 D , 若存在到矩形区域 D^c 的双射, 则记 D^c 到 D 的映射为 $\phi: D^c \rightarrow D$. 构造全连接网络 g_θ 学习逆映射 ϕ^{-1} , 逆映射 ϕ^{-1} 可以将 D 上的非均匀网格 $\{\mathbf{x}^l\}$ 映射到 D^c 上的均匀网格 $\{\boldsymbol{\xi}^l\}$, 然后使用 $\{\boldsymbol{\xi}^l\}$ 作为FNO的部分输入, 从而可以通过FFT进行加速.

根据上述设计, Geo-FNO的构造为

$$\mathcal{G}_\theta = \mathcal{Q} \circ \sigma(\mathcal{W}_L + \mathcal{K}_L(\phi^{-1})) \circ \cdots \circ \sigma(\mathcal{W}_1 + \mathcal{K}_1(\phi^{-1})) \circ \mathcal{P} \quad (8)$$

其中积分变换

$$[\mathcal{K}(\phi^{-1})v_\ell](\mathbf{x}) = [\mathcal{K}v_\ell](\boldsymbol{\xi}).$$

实际在应用Geo-FNO时, 往往不需要构造全连接网络 g_θ 学习逆映射 ϕ^{-1} . 因为当网格生成以后, 逆映射 ϕ^{-1} 的结果已知, 即为 D^c 上的均匀网格 $\{\boldsymbol{\xi}^i\}$ 的坐标, 将其直接将代入(8)即可 ((8)已经和原始坐标无关了), 网络的输出值即为解在网格点上的值.

²Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint arXiv:2111.03794 (2021).

³Z. Li, D. Huang, B. Liu, A. Anandkumar, Fourier neural operator with learned deformations for PDEs on general geometries, arXiv preprint arXiv:2207.05209 (2022).

1.4 Geometry-aware Physics-informed neural operator (Geo-PINO)

将PINO和Geo-FNO组合可得到Geo-PINO. 不同于Geo-FNO, 在应用Geo-PINO时, 需要构造全连接网络 g_{θ} 学习逆映射 ϕ^{-1} . 这是因为在计算导数值时需要计算 ϕ^{-1} 的输出关于输入的导数:

$$\begin{aligned}\frac{\partial u}{\partial x_m} &= \sum_j \frac{\partial u}{\partial \xi_j} \frac{\partial \xi_j}{\partial x_m}, \\ \frac{\partial^2 u}{\partial x_m^2} &= \sum_{j_1} \sum_{j_2} \frac{\partial^2 u}{\partial \xi_{j_1} \partial \xi_{j_2}} \frac{\partial \xi_{j_2}}{\partial x_m} \frac{\partial \xi_{j_1}}{\partial x_m} + \sum_{j_1} \frac{\partial u}{\partial \xi_{j_1}} \frac{\partial^2 \xi_{j_1}}{\partial x_m^2}.\end{aligned}\quad (9)$$

尽管在Geo-FNO的文章中提到Geo-PINO是后续的拓展工作, 但目前并没有相关的论文刊出. 原因可能在于Geo-PINO的求解精度并不高. 接下来将通过数值实验予以验证.

尽管Geo-PINO可以求解参数化方程, 为便于讨论, 在这个实验中只考虑一个确定的Poisson方程

$$\begin{cases} -\Delta u = r_{\Omega}, & \text{in } \Omega, \\ u = r_{\Gamma}, & \text{in } \Gamma. \end{cases}\quad (10)$$

计算区域为 $\Omega := [-1, 1]^2 \setminus \Omega_c$, 其中 Ω_c 是以 $[0, 0]$ 为圆心, 半径为0.2的圆. 真解设置为 $u = \cdot$. 构造两个网络 g_{θ_1} 和 f_{θ_2} , g_{θ_1} 用于将 Ω 上的原始网格映射到矩形区域 $\Omega_c = [0, 1]^2$ 上的计算网格. f_{θ_2} 用于进一步将计算网格点映射到真解. 两个网络的训练分开进行, 训练所用的损失函数分别为:

$$L[g_{\theta_1}] := \sum_l (g_{\theta_1}(\mathbf{x}^l) - (\boldsymbol{\lambda}^l))^2. \quad (11)$$

$$L[f_{\theta_2}] := \sum_{\boldsymbol{\lambda}^l \in S(\Omega)} (-\Delta f(\boldsymbol{\lambda}^l) - r_{\Omega}(\boldsymbol{\lambda}^l))^2 + \sum_{\boldsymbol{\lambda}^l \in S(\Gamma)} (f(\boldsymbol{\lambda}^l) - r_{\Gamma}(\boldsymbol{\lambda}^l))^2 \quad (12)$$

进行两组实验, 在两组实验中网络 g_{θ_1} 均采用全连接网络构造, 而网络 f_{θ_2} 则分别采用全连接网络和FNO构造. 训练10000个epoch. 图2展示了训练过程损失函数和误差的变化情况, 从中可以看到无论网络 f_{θ_2} 的构造如何, 算法始终难以收敛, 损失函数和近似解的误差难以下降. 可见训练网络的困难程度比较大.

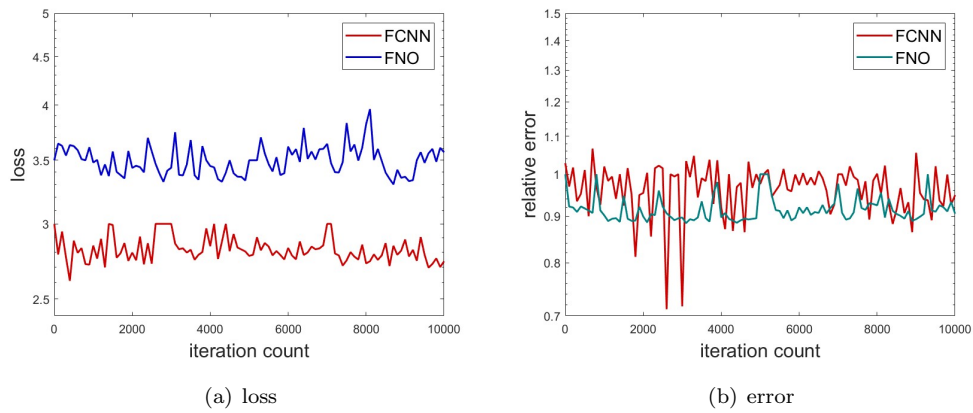


Figure 2: 求解过程中的损失函数和误差变化情况.

求解精度并不理想的原因是否是因为 g_{θ_1} 没有很好地学习所需的坐标变换? 为探究原因, 设计第二个实验.

事实上, 存在解析式的映射 ϕ^{-1} , 将 Ω 映射到 Ω_c 上:

$$\begin{cases} r = (x_1^2 + x_2^2)^{0.5}, \\ \theta = \arctan(x_2, x_1), \\ r_u = \min\{1.0/|\cos(\theta)|, 1.0/|\sin(\theta)|\}, \\ x_1 = \theta/(2\pi), \\ x_2 = (r - 0.2)/(r_u - 0.2). \end{cases} \quad (13)$$

用 ϕ^{-1} 替换网络 g_{θ_1} , 只训练网络 f_{θ_2} . 训练个epoch以后, 求解精度为, 仍然不理想. 可见之前求解精度不理想并不是因为网络 g_{θ_1} 没有准确逼近 ϕ^{-1} .

影响精度的原因另有其他. 有可能是坐标变换的引入本身就增加了训练网络 f_{θ_2} 的困难程度. 为此设计三个实验, 将两个网络连接起来(这样可以通过反向传播直接计算近似解关于原始空间坐标的导数). 并一同进行训练, 训练所用的损失函数为

$$\begin{aligned} L[g_{\theta_1}, f_{\theta_2}] &:= \sum_{x^l \in S(\Omega)} (-\Delta f_{\theta_2}(g_{\theta_1}(x^l)) - r_{\Omega}(x^l))^2 + \sum_{x^l \in S(\Gamma)} (f_{\theta_2}(g_{\theta_1}(x^l)) - r_{\Gamma}(x^l))^2 \\ &+ \beta \sum_{x^l \in S(\Omega) \cup S(\Gamma)} (g_{\theta_1}(x^l) - (\lambda^l))^2. \end{aligned} \quad (14)$$

调整惩罚项系数 β , 探究近似解的相对误差 e_u 和变换后坐标的相对误差 e_{λ} 的变化, 结果见表. 从中可以看到, 两个误差项始终难以同时下降到理想的范围之内. 可见坐标变换的引入反而不利于提升近似解的精度.

Table 1: 求解得到的近似解误差和变换后坐标误差的变化

β	e_u	e_{λ}
1e0	8.881e-03	2.933e-01
1e1	7.377e-02	2.624e-01
1e2	3.451e-01	1.901e-01
1e3	1.049e+00	5.980e-02
1e4	1.216e+00	1.335e-02

2 Extended Fourier Neural Operator

2.1 含参问题

考虑含参边值问题

$$\begin{cases} -\nabla \cdot (\mathbf{A} \nabla u) + \mathbf{B} \cdot \nabla u = r_{\Omega}, & \text{in } \Omega \subseteq \bar{\Omega} \subset \mathbb{R}^{d_{\mathbf{x}}}, \\ u = r_D, & \text{in } \Gamma_D, \\ (\mathbf{A} \nabla u) \cdot \mathbf{n} = r_N, & \text{in } \Gamma_N. \end{cases} \quad (15)$$

其中 $d_{\mathbf{x}} \in \mathbb{N}_+$, u 是未知变量, $\mathbf{A} \in \mathbb{R}^{d_{\mathbf{x}} \times d_{\mathbf{x}}}$ 和 $\mathbf{B} \in \mathbb{R}^{d_{\mathbf{x}}}$ 是可变参数, 其元素分别记为 $a_{j_1, j_2} = (\mathbf{A})_{j_1, j_2}$ 和 $b_{j_1} = (\mathbf{B})_{j_1}$, $j_1, j_2 = 1 \cdots, d_{\mathbf{x}}$, \mathbf{n} 是单位外法向, r_{Ω} , r_D 和 r_N 分别是控制方程, Dirichlet和Neumann边界条件的右端项, Γ_D 和 Γ_N 分别为Dirichlet和Neumann边界, $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$. 计算区域 Ω 的形状是可变的, 且存在一个矩体上界 $\bar{\Omega}$.

将计算区域的形状参数化为形状参数

$$s_{\Omega}(\mathbf{x}; \Omega) = \begin{cases} 1, & \text{in } \Omega, \\ 0, & \text{in } \bar{\Omega} \setminus \Omega. \end{cases} \quad (16)$$

这样形状参数可以和问题(15)中的其他可变参数拼接到一起. 拓展问题(15)中可变参数和真解的定义为

$$\mathbf{P} = \begin{cases} [a_{1,1} & a_{1,2} \cdots a_{d_{\mathbf{x}}, d_{\mathbf{x}}} & b_1 \cdots b_{d_{\mathbf{x}}} & s_{\Omega}]^{\top}, & \text{in } \Omega, \\ \mathbf{0}, & \text{in } \bar{\Omega} \setminus \Omega, \end{cases} \quad \tilde{u} = \begin{cases} u, & \text{in } \Omega, \\ 0, & \text{in } \bar{\Omega} \setminus \Omega, \end{cases} \quad (17)$$

分别属于空间 $\mathcal{P} = \mathcal{P}(\bar{\Omega}; \mathbb{R}^{d_p})$ 和 $\mathcal{U} = \mathcal{U}(\bar{\Omega}; \mathbb{R})$, 其中 $d_p = d_{\mathbf{x}} \times d_{\mathbf{x}} + d_{\mathbf{x}} + 1$. 这样存在映射 $G^{\dagger} : \mathcal{P} \rightarrow \mathcal{U}$. 目标是构造神经网络 G_{θ} , 学习映射 G^{\dagger} .

由于不影响接下来的讨论, 为简洁起见, 仍然将 \tilde{u} 记为 u ,

2.2 网络输入和输出

采用FNO构造网络 G_{θ} , 学习映射 G^{\dagger} . 网络接受参数 \mathbf{P} 的离散形式作为输入, 输出解 u 的离散形式. 以 $d_{\mathbf{x}} = 3$ 为例, 取一个覆盖计算区域 Ω 的矩体 $\tilde{\Omega}$, 将 $\tilde{\Omega}$ 划分为 $n_1 \times n_2 \times n_3$ ($n_1, n_2, n_3 \in \mathbb{N}_+$) 个单元, 记各个单元为 c^{i_1, i_2, i_3} , $i_1 = 1, \dots, n_1$, $i_2 = 1, \dots, n_2$, $i_3 = 1, \dots, n_3$. 取每个单元的中心, 生成网格点集 $S_{\mathbf{x}} := \{\{\mathbf{x}^{i_1, i_2, i_3}\}_{i_1=1}^{n_1}\}_{i_2=1}^{n_2}\}_{i_3=1}^{n_3}$. 同时, 在 \mathbf{P} 所属的空间 \mathcal{P} 中采样得到集合 $S_{\mathbf{P}} = \{\mathbf{P}^{i_p}\}_{i_p=1}^{n_p}$. 网络 G_{θ} 接受参数集合 $S_{\mathbf{P}}$ 在网格点集 $S_{\mathbf{x}}$ 上的值作为输入, 输出在这些参数设置下网格集 $S_{\mathbf{x}}$ 上的解 u 的预测值.

记输入为 $\mathbf{I} \in \mathbb{R}^{n_p \times d_i \times n_1 \times n_2 \times n_3}$, 其元素为 $p^{i_p, i_1, i_2, i_3} = (\mathbf{I})^{i_p, i_1, i_2, i_3}$, 输出为 $\mathbf{O} \in \mathbb{R}^{n_p \times d_o \times n_1 \times n_2 \times n_3}$, 其元素为 $u^{i_p, i_1, i_2, i_3} = (\mathbf{O})^{i_p, i_1, i_2, i_3}$, 其中 $d_i = d_p$ 和 $d_o = 1$ 分别是输入和输出通道数目.

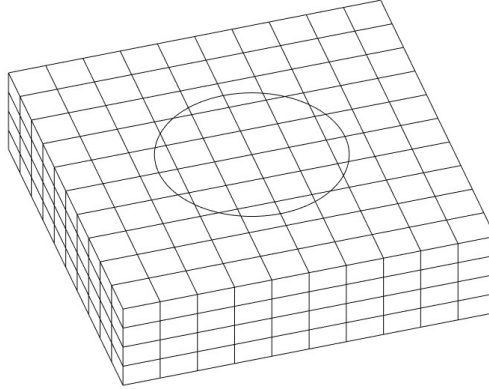


Figure 3: 计算区域和网格示例.

2.3 损失函数

对于落在计算区域内的每个网格点 $\mathbf{x}_{i_1, i_2, i_3}$, 借鉴有限体积法 cut cell 的离散格式度量该点方程的残差. 首先, 对于每个参数设置 \mathbf{P}^{i_p} 下的问题(15), 将其中控制方程在点 $\mathbf{x}^{i_1, i_2, i_3}$ 所属的单元 c^{i_1, i_2, i_3} 上进行积分, 有

$$\begin{aligned} \int_{c^{i_1, i_2, i_3}} (-\nabla \cdot (\mathbf{A}^{i_p} \nabla u) + \mathbf{B}^{i_p} \cdot \nabla u) d\mathbf{x} &= \int_{c^{i_1, i_2, i_3}} \left(\sum_{j_1=1}^3 \left(- \sum_{j_2=1}^3 a_{j_1, j_2}^{i_p} \frac{\partial^2 u}{\partial x_{j_2} \partial x_{j_1}} + b_{j_1}^{i_p} \frac{\partial u}{\partial x_{j_1}} \right) \right) d\mathbf{x} \\ &= \sum_{k=1}^{n_f^{i_1, i_2, i_3}} \int_{f_k^{i_1, i_2, i_3}} \left(\sum_{j_1=1}^3 \left(- \sum_{j_2=1}^3 a_{j_1, j_2}^{i_p} \frac{\partial u}{\partial x_{j_2}} + b_{j_1}^{i_p} u \right) n_{j_1} \right) d\mathbf{x} \\ &= \int_{c^{i_1, i_2, i_3}} r_{\Omega} d\mathbf{x}, \end{aligned} \quad (18)$$

其中 $f_k^{i_1, i_2, i_3}$ 是组成单元 c^{i_1, i_2, i_3} 边界的面, $k = 1, \dots, n_f^{i_1, i_2, i_3}$, $n_f^{i_1, i_2, i_3}$ 是面的数目.

通过数值积分, 可将上式近似为

$$\sum_k \left(\sum_{j_1} \left(- \sum_{j_2} a_{j_1, j_2}^{i_p} \frac{\partial u}{\partial x_{j_2}} + b_{j_2}^{i_p} u \right) n_{j_1} \right)_k D_k^{i_1, i_2, i_3} \approx r^{i_1, i_2, i_3} V^{i_1, i_2, i_3}, \quad (19)$$

其中 $(\square)_k^{i_1, i_2, i_3}$ 表示函数 \square 在面 $f_k^{i_1, i_2, i_3}$ 中心 $\mathbf{x}_k^{i_1, i_2, i_3}$ 的值, $D_k^{i_1, i_2, i_3}$ 是面 $f_k^{i_1, i_2, i_3}$ 的面积, V^{i_1, i_2, i_3} 是单元 $c_k^{i_1, i_2, i_3}$ 的体积.

将(19)左右两端相减, 可用于估计点 $\mathbf{x}^{i_1, i_2, i_3}$ 处方程的残差, 其中 $(u)_k^{i_1, i_2, i_3}$ 和 $\left(\frac{\partial u}{\partial x_{j_2}} \right)_k^{i_1, i_2, i_3}$ 将通过计算插值函数 \hat{u}^{i_1, i_2, i_3} 的值及其导数进行估计. 插值函数 \hat{u}^{i_1, i_2, i_3} 的构造为二次多项式

$$\hat{u}^{i_1, i_2, i_3}(\mathbf{x}) = \sum_{s_1=0}^2 \sum_{s_2=0}^2 \sum_{s_3=0}^2 \hat{a}^{s_1, s_2, s_3} \phi^{s_1, s_2, s_3}(\mathbf{x}), \quad \phi^{s_1, s_2, s_3}(\mathbf{x}) = x_1^{s_1} x_2^{s_2} x_3^{s_3}, \quad (20)$$

其中系数 \hat{a}^{s_1, s_2, s_3} 根据点 $\mathbf{x}^{i_1, i_2, i_3}$ 和它邻点处解的预测值或边界条件值计算得到.

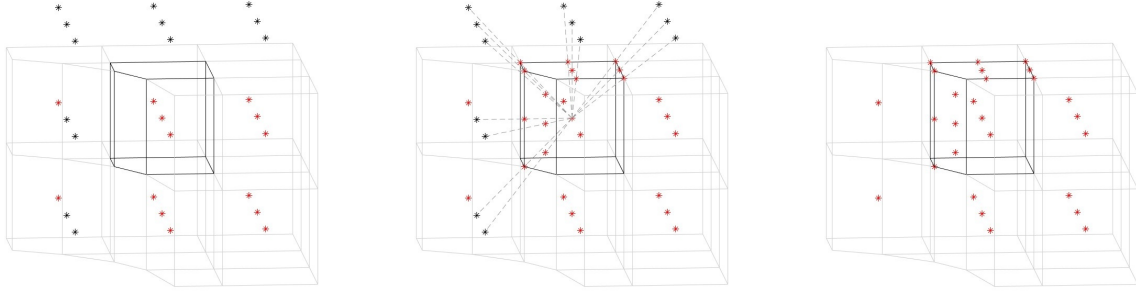


Figure 4: 经边界切割的单元和插值节点的选取(红点位于计算区域内部和边界, 黑点位于计算区域外部).

为计算系数 \hat{a}^{s_1, s_2, s_3} , 首先选取插值节点 $S_i^{i_1, i_2, i_3} = \{ \{ \{ \hat{\mathbf{x}}^{t_1, t_2, t_3} \}_{t_1=0}^2 \}_{t_2=0}^2 \}_{t_3=0}^2$, 然后构造相应的线性方程组并求解得到. 插值节点的选取方式如下: 若点 $\mathbf{x}^{i_1-1+t_1, i_2-1+t_2, i_3-1+t_3}$ 位于计算区域内部, 则作为插值节点 $\hat{\mathbf{x}}^{t_1, t_2, t_3}$, 反之, 则连接点 $\mathbf{x}^{i_1, i_2, i_3}$ 与点 $\mathbf{x}^{i_1-1+t_1, i_2-1+t_2, i_3-1+t_3}$, 取连线与计算区域边界 $\partial\Omega$ 的交点作为插值节点 $\hat{\mathbf{x}}^{t_1, t_2, t_3}$. 根据插值节点, 构造线性方程组

$$\Psi \hat{\mathbf{A}} = \mathbf{V}[G_\theta], \quad (21)$$

其中

$$\Psi = \begin{bmatrix} \psi_{0,0,0}^{0,0,0} & \cdots & \psi_{0,0,0}^{2,2,2} \\ \vdots & \ddots & \vdots \\ \psi_{2,2,2}^{0,0,0} & \cdots & \psi_{2,2,2}^{2,2,2} \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} \hat{a}^{0,0,0} \\ \vdots \\ \hat{a}^{2,2,2} \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} v^{0,0,0}[G_\theta] \\ \vdots \\ v^{2,2,2}[G_\theta] \end{bmatrix},$$

$$\psi_{t_1, t_2, t_3}^{s_1, s_2, s_3} = \begin{cases} \phi^{s_1, s_2, s_3}(\hat{\mathbf{x}}^{t_1, t_2, t_3}), & \hat{\mathbf{x}}^{s_1, s_2, s_3} \in (\Omega \cup \Gamma_D), \\ (A^{i_p} \nabla \phi^{s_1, s_2, s_3} \cdot \mathbf{n})(\hat{\mathbf{x}}^{t_1, t_2, t_3}), & \hat{\mathbf{x}}^{s_1, s_2, s_3} \in \Gamma_N, \end{cases}$$

$$v^{s_1, s_2, s_3}[G_\theta] = \begin{cases} u^{i_1-1+s_1, i_2-1+s_2, i_3-1+s_3}[G_\theta], & \hat{\mathbf{x}}^{s_1, s_2, s_3} \in \Omega, \\ r_D(\mathbf{x}^{s_1, s_2, s_3}), & \hat{\mathbf{x}}^{s_1, s_2, s_3} \in \Gamma_D, \\ r_N(\mathbf{x}^{s_1, s_2, s_3}), & \hat{\mathbf{x}}^{s_1, s_2, s_3} \in \Gamma_N. \end{cases}$$

由于线性方程组(21)的右端项 \mathbf{V} 中存在一些未知变量, 无法直接求解得到插值系数 $\hat{\mathbf{A}}$. 但是考虑到 $\hat{\mathbf{A}} = \Psi^{-1}\mathbf{V}$, 可以将 $\hat{\mathbf{A}}$ 写作 \mathbf{V} 的函数. 具体地, 记 $\hat{b}_{t_1, t_2, t_3}^{s_1, s_2, s_3} = (\Psi^{-1})_{t_1, t_2, t_3}^{s_1, s_2, s_3}$, 有

$$\hat{a}^{s_1, s_2, s_3} = \sum_{t_1, t_2, t_3} \hat{b}_{t_1, t_2, t_3}^{s_1, s_2, s_3} v^{t_1, t_2, t_3} [G_{\theta}]. \quad (22)$$

将(22)代入插值函数(20), 得到

$$\begin{aligned} \hat{u}^{i_1, i_2, i_3}(\mathbf{x}) &= \sum_{s_1, s_2, s_3} \left(\sum_{t_1, t_2, t_3} \hat{b}_{t_1, t_2, t_3}^{s_1, s_2, s_3} v^{t_1, t_2, t_3} [G_{\theta}] \right) \phi^{s_1, s_2, s_3}(\mathbf{x}) \\ &= \sum_{t_1, t_2, t_3} \left(\sum_{s_1, s_2, s_3} \hat{b}_{t_1, t_2, t_3}^{s_1, s_2, s_3} \phi^{s_1, s_2, s_3}(\mathbf{x}) \right) v^{t_1, t_2, t_3} [G_{\theta}]. \end{aligned} \quad (23)$$

令 $\hat{c}^{t_1, t_2, t_3}(\mathbf{x}) = \sum_{s_1, s_2, s_3} \hat{b}_{t_1, t_2, t_3}^{s_1, s_2, s_3} \phi^{s_1, s_2, s_3}(\mathbf{x})$, $S_v^{i_1, i_2, i_3} [G_{\theta}] = \{\{\{v^{t_1, t_2, t_3}\}_{t_1=0}^2\}_{t_2=0}^2\}_{t_3=0}^2$ 有

$$\hat{u}^{i_1, i_2, i_3}(\mathbf{x}; S_v^{i_1, i_2, i_3} [G_{\theta}]) = \sum_{t_1, t_2, t_3} \hat{c}^{t_1, t_2, t_3}(\mathbf{x}) v^{t_1, t_2, t_3} [G_{\theta}]. \quad (24)$$

从而插值函数 \hat{u}^{i_1, i_2, i_3} 可以利用插值节点 $S_i^{i_1, i_2, i_3}$ 处的预测值或边界值进行构造, 并进一步被用于估计单元中心点 $\mathbf{x}^{i_1, i_2, i_3}$ 处的残差

$$\begin{aligned} R^{i_p, i_1, i_2, i_3} [G_{\theta}] &= - \sum_k \left(\sum_{j_1} \sum_{j_2} a_{j_1, j_2}^{i_p} n_{j_1} \right)_k^{i_1, i_2, i_3} \frac{\partial \hat{u}^{i_1, i_2, i_3}}{\partial x_{j_2}} (\mathbf{x}_k^{i_1, i_2, i_3}; S_v^{i_1, i_2, i_3} [G_{\theta}]) D_k^{i_1, i_2, i_3} \\ &\quad + \sum_k \left(\sum_{j_1} b_{j_1}^{i_p} n_{j_1} \right)_k^{i_1, i_2, i_3} \hat{u}^{i_1, i_2, i_3} (\mathbf{x}_k^{i_1, i_2, i_3}; S_v^{i_1, i_2, i_3} [G_{\theta}]) D_k^{i_1, i_2, i_3} \\ &\quad - r^{i_1, i_2, i_3} V^{i_1, i_2, i_3}. \end{aligned} \quad (25)$$

所有样本点的残差进一步构成训练网络所用的损失函数

$$L[G_{\theta}] = \sum_{i_p} \sum_{i_1, i_2, i_3} (R^{i_p, i_1, i_2, i_3} [G_{\theta}])^2. \quad (26)$$

由于边界条件被吸收到控制方程中, 损失函数(26)仅包含与控制方程相关的残差. 这样就降低了优化损失函数的困难. 训练神经网络, 最小化损失函数, 得到方程的近似解.

2.4 N-S方程

当控制方程为N-S方程时, 由于含有连续性方程, 损失函数的构造略有不同, 以二维N-S方程为例:

$$\begin{cases} \mathcal{L}_1 := -\nu \Delta u_1 + u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} + \frac{\partial p}{\partial x_1} = 0, \\ \mathcal{L}_2 := -\nu \Delta u_2 + u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} + \frac{\partial p}{\partial x_2} = 0, \\ \mathcal{L}_3 := \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = 0. \end{cases} \quad (27)$$

其中 u_1, u_2 分别是 x_1, x_2 方向的流速, p 是压力, $\nu = 1/\text{Re}$ 是黏性系数, Re 是雷诺数.

构造神经网络 G_{θ} 学习问题参数到解的映射 $G^{\dagger} : \mathbf{P} \mapsto [u_1 \ u_2 \ p]^{\top}$. 方程残差的计算方式为:

$$\begin{cases} R_1^{i_1, i_2, i_3} [G_{\theta}] &:= \sum_k \left(\sum_{j_1} \left(- \sum_{j_2} a_{j_1, j_2} \frac{\partial u_1}{\partial x_{j_2}} + b_{j_2} u_1 \right) n_{j_1} + p n_1 \right)_k^{i_1, i_2, i_3} D_k^{i_1, i_2, i_3} \\ R_2^{i_1, i_2, i_3} [G_{\theta}] &:= \sum_k \left(\sum_{j_1} \left(- \sum_{j_2} a_{j_1, j_2} \frac{\partial u_2}{\partial x_{j_2}} + b_{j_2} u_2 \right) n_{j_1} + p n_2 \right)_k^{i_1, i_2, i_3} D_k^{i_1, i_2, i_3} \\ R_3^{i_1, i_2, i_3} [G_{\theta}] &:= \sum_k \left((u_1)_k^{i_1, i_2, i_3} n_1^{i_1, i_2, i_3} + (u_2)_k^{i_1, i_2, i_3} n_2^{i_1, i_2, i_3} \right) D_k^{i_1, i_2, i_3} \end{cases} \quad (28)$$

其中 $a_{1,1} = a_{2,2} = \nu$, $a_{1,2} = a_{2,1} = 0$, $b_1 = u_1^{i_1, i_2, i_3}$, $b_2 = u_2^{i_1, i_2, i_3}$.

在计算 $R_3^{i_1, i_2, i_3}$ 时, $(u_1)_k^{i_1, i_2, i_3}$ 和 $(u_2)_k^{i_1, i_2, i_3}$ 的插值方式采用

$$(u_1)_k^{i_1, i_2, i_3} = \sum_{(s_1, s_2, s_3) \in I_k^{i_1, i_2, i_3}} \hat{w}_1^{s_1, s_2, s_3} v_1^{s_1, s_2, s_3}, \quad (u_2)_k^{i_1, i_2, i_3} = \sum_{(s_1, s_2, s_3) \in I_k^{i_1, i_2, i_3}} \hat{w}_2^{s_1, s_2, s_3} v_2^{s_1, s_2, s_3}, \quad (29)$$

其中 $I_k^{i_1, i_2, i_3}$ 是点 $(\mathbf{x})_k^{i_1, i_2, i_3}$ 的邻点的指标集. 而 $v_1^{s_1, s_2, s_3}$ 和 $v_2^{s_1, s_2, s_3}$ 则根据 $R_1^{s_1, s_2, s_3}$ 和 $R_2^{s_1, s_2, s_3}$ 计算, 考虑到当 $R_1^{s_1, s_2, s_3} = 0$ 和 $R_2^{s_1, s_2, s_3} = 0$ 时有

$$\begin{aligned} u_1^{s_1, s_2, s_3} &= \sum_{(t_1, t_2, t_3) \in I^{s_1, s_2, s_3}} \hat{a}_1^{t_1, t_2, t_3} u_1^{t_1, t_2, t_3} + \hat{b}_1^{t_1, t_2, t_3} p^{t_1, t_2, t_3}, \\ u_2^{s_1, s_2, s_3} &= \sum_{(t_1, t_2, t_3) \in I^{s_1, s_2, s_3}} \hat{a}_2^{t_1, t_2, t_3} u_2^{t_1, t_2, t_3} + \hat{b}_2^{t_1, t_2, t_3} p^{t_1, t_2, t_3}, \end{aligned} \quad (30)$$

其中 I^{s_1, s_2, s_3} 是点 $\mathbf{x}^{s_1, s_2, s_3}$ 的邻点的指标集. 利用(30)估计 $v_1^{s_1, s_2, s_3}$ 和 $v_2^{s_1, s_2, s_3}$, 并进一步利用(29)插值, 从而计算残差 $R_3^{i_1, i_2, i_3}$. 最后, 所有采样点的残差按照类似于(26)的形式构成损失函数.

3 数值实验

3.1 Linear elasticity equation

(非瞬态)线性弹性方程的位移形式(displacement form), 也称为纳维尔(Navier)方程, 被定义为

$$\begin{cases} (\lambda + \mu)(u_{1,11} + u_{2,21} + u_{3,31}) + \mu(u_{1,11} + u_{1,22} + u_{1,33}) = f_1, \\ (\lambda + \mu)(u_{1,12} + u_{2,22} + u_{3,32}) + \mu(u_{2,11} + u_{2,22} + u_{2,33}) = f_2, \\ (\lambda + \mu)(u_{1,13} + u_{2,23} + u_{3,33}) + \mu(u_{3,11} + u_{3,22} + u_{3,33}) = f_3, \end{cases} \quad (31)$$

其中 u_1, u_2, u_3 是位移, f_1, f_2, f_3 是体积单位体力(body force), λ 和 μ 是拉梅参数(Lamé parameters), 被定义为

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{(1+\nu)}, \quad (32)$$

这里, E 和 ν 分别是杨氏模量(Young's modulus)和泊松比(Poisson's ratio).

求解带有孔洞的支架上的线弹性方程, 计算区域如图6所示控制方程中参数 λ 和 μ 分别设置为1.0和0.5. 边界条件设置为混合边界条件, 其中 $\{-1\} \times [-1, 1] \times [-1, 1]$ 上设置为Dirichlet边界 $[u_1 \ u_2 \ u_3]^\top = [0 \ 0 \ 0]^\top$, 其余边界上设置为Neumann边界条件

$$\begin{cases} (\lambda + \mu)(u_{1,1} + u_{2,2} + u_{3,3})n_1 + \mu(u_{1,1}n_1 + u_{1,2}n_2 + u_{1,3}n_3) = 0, \\ (\lambda + \mu)(u_{1,1} + u_{2,2} + u_{3,3})n_2 + \mu(u_{2,1}n_1 + u_{2,2}n_2 + u_{2,3}n_3) = 0, \\ (\lambda + \mu)(u_{1,1} + u_{2,2} + u_{3,3})n_3 + \mu(u_{3,1}n_1 + u_{3,2}n_2 + u_{3,3}n_3) = \begin{cases} 0, & \text{on } \{1\} \times [-1, 1] \times [-0.2, 0.2], \\ 1, & \text{otherwise.} \end{cases} \end{cases} \quad (33)$$

采用FNO构造网络, 网络包含4个Fourier层. 设置 $k_{\max} = 12$, $d_v = 32$, 每个Fourier层的权重张量 R 是一个大小为 $12 \times 32 \times 32$ 的复值张量. 网络含有2,376,449个参数. 采用Adam训练网络, 初始学习率为0.001, 每经过5,000个epoch后学习率衰减为之前的0.98倍. 在计算区域中采样 $10 \times 10 \times 10$ 个网格点, 去除落在计算区域外的点, 构成训练集. 图5展示了训练过程中误差的下降曲线, 经过1,000,000个epoch后, 解的相对误差下降到了 $1e-03$ 附近. 观察收敛曲线, 可以看到, 而在训练前期, 收敛则比较困难, 随着训练进行, 收敛速度反而越来越快. 或许需要设计更合理的训练方式, 加快收敛速度. 图6展示了某些参数设置下求解得到的近似解, 基本与参考解相符.

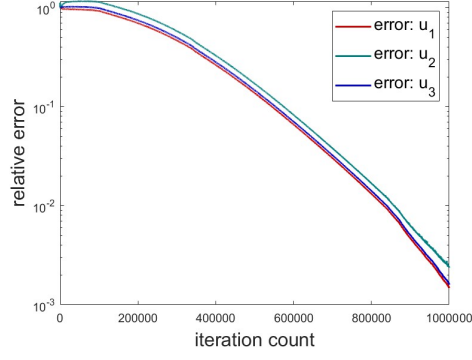


Figure 5: 训练过程中近似解误差的变化情况.

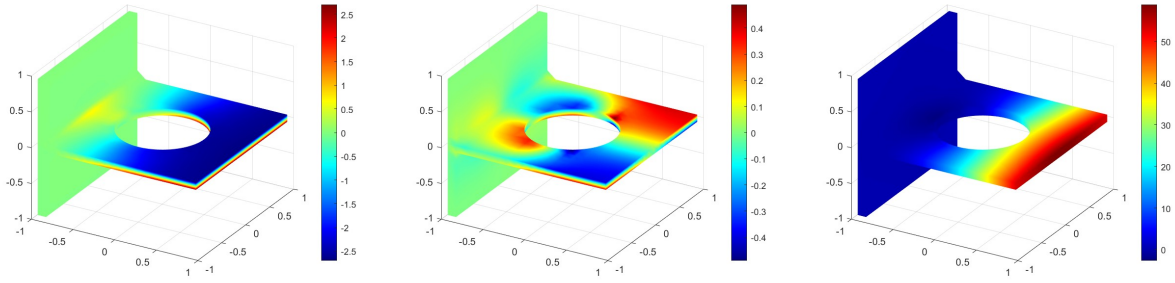


Figure 6: 求解得到的近似解.

3.2 Cylinder flow

考虑含参圆柱绕流问题. 计算区域为如图7所示的2D通道 $\Omega := [-1.5, 1.5] \times [-0.5, 0.5]$, 其中有一圆心为 c , 半径为0.2的圆柱 Ω_c . 调整圆柱的位置, 探究流场的变化. 控制方程为N-S方程, 雷诺数 Re 设置为100. 边界条件为:

$$\begin{cases} u_1 = \begin{cases} u_i, & \text{on } \Gamma_i, \\ 0, & \text{on } \Gamma_w \cup \Gamma_c, \end{cases} \\ u_2 = 0, & \text{on } \Gamma_i \cup \Gamma_w \cup \Gamma_c, \\ p = 0, & \text{on } \Gamma_o, \end{cases} \quad (34)$$

其中 $\Gamma_i = \{-1.5\} \times [-0.5, 0.5]$, $\Gamma_o = \{1.5\} \times [-0.5, 0.5]$, $\Gamma_w = [-1.5, 1.5] \times \{-0.5, 0.5\}$, $\Gamma_c = \partial\Omega_c$, u_i 是入口处的流速. u_i 和 c 共同构成了问题的参数 $\mathbf{P} = [u_i \ c]^\top$.

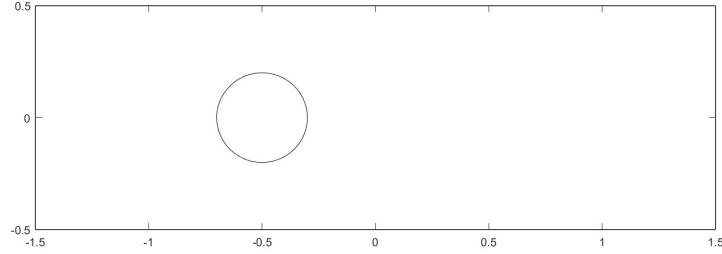


Figure 7: 计算区域

采用FNO构造网络, 网络包含4个Fourier层. 设置 $k_{\max} = 12$, $d_v = 32$, 每个Fourier层的权重张量 R 是一个大小为 $12 \times 32 \times 32$ 的复值张量. 网络含有2,376,449个参数. 采用Adam训练网络, 初始学习率为0.001, 每经过500个epoch后学习率衰减为之前的0.95倍. 在 $[0.5, 1.5]$ 之间采样30个点, 作为入口处的流速, 在参数空间 $[-1.0, 1.0] \times [-0.2, 0.2]$ 中采样30个点, 作为圆柱的中心. 并对于每个参数设置, 在计算区域中采样 60×20 个网格点, 去除落在圆柱内的点, 构成训练集. 图8展示了训练过程中误差的下降曲线, 经过5,000个epoch后, 解的相对误差下降到了 $1e-02$ 以下. 图9展示了某些参数设置下求解得到的近似解, 基本与参考解相符.

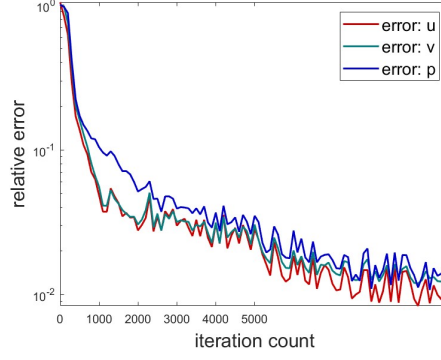


Figure 8: 训练过程中近似解误差的变化情况.

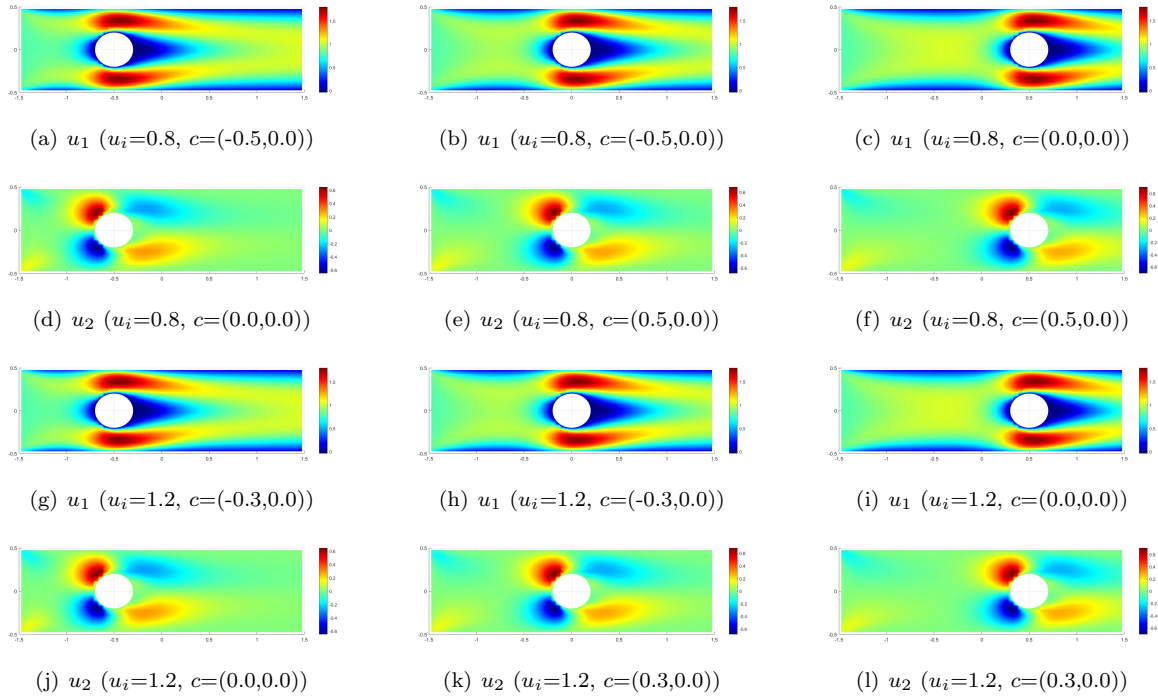


Figure 9: 求解得到的近似解.

3.3 Heat sink

3D含参heat sink算例的计算区域为如图10所示的3D通道 $\Omega = [-1.0, 1.0] \times [-0.5, 0.5] \times [-0.5, 0.5]$ ，其中有一暖气片，由底座和三片翅片组成。设置底座的大小为 $0.36 \times 0.6 \times 0.2$ ，三片翅片的大小为 $0.36 \times 0.08 \times 0.37$ ，移动暖气片的位置，探究流场的变化。控制方程为Stokes方程，雷诺数 Re 设置为100。入口处的流速为1m/s，出口处的压力为0Pa，壁面处的流速为0。

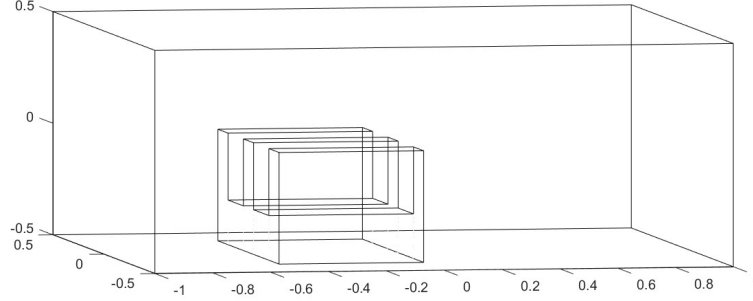


Figure 10: 计算区域

采用FNO构造网络，网络包含4个Fourier层。设置 $k_{\max} = 12$ ， $d_v = 32$ ，网络含有2,376,449个参数。采用Adam训练网络，初始学习率为0.001，每经过500个epoch后学习率衰减为之前的0.95倍。在 $[-0.18, 0.18] \times \{0\} \times \{-0.5\}$ 之间采样10个点，作为底座的中心。并对于每个参数设置，在计算区域中采样 $40 \times 20 \times 10$ 个网格点，去除落在暖气片内的点，构成训练集。图11展示了训练过程中误差的下降曲线，经过10,000个epoch后，速度分量 u_1 ， u_3 的解的相对误差下降到了 $1e-02$ 附近，而 u_2 的相对误差则稍高一些。不过这主要是由于训练集中的样本数目较少所致，增加采样以后，预期测试集中的近似解的误差还将进一步下降。图12展示了测试集中某些参数设置下求解得到的近似解，基本与参考解相符。

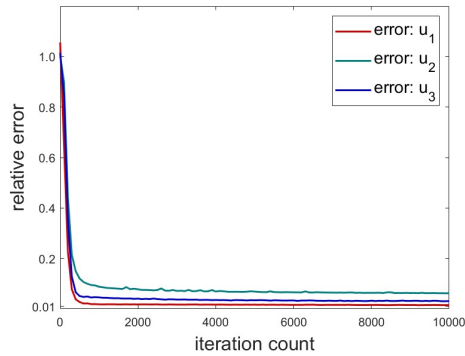


Figure 11: 训练过程中的误差变化

目前这个算例只包含流场部分，接下来会将温度场也耦合进来。问题所包含的参数只有暖气片的位置，接下来会将暖气片的形状(如翅片的长宽高，翅片的数目)也加入问题参数，探索模型的能力边界。

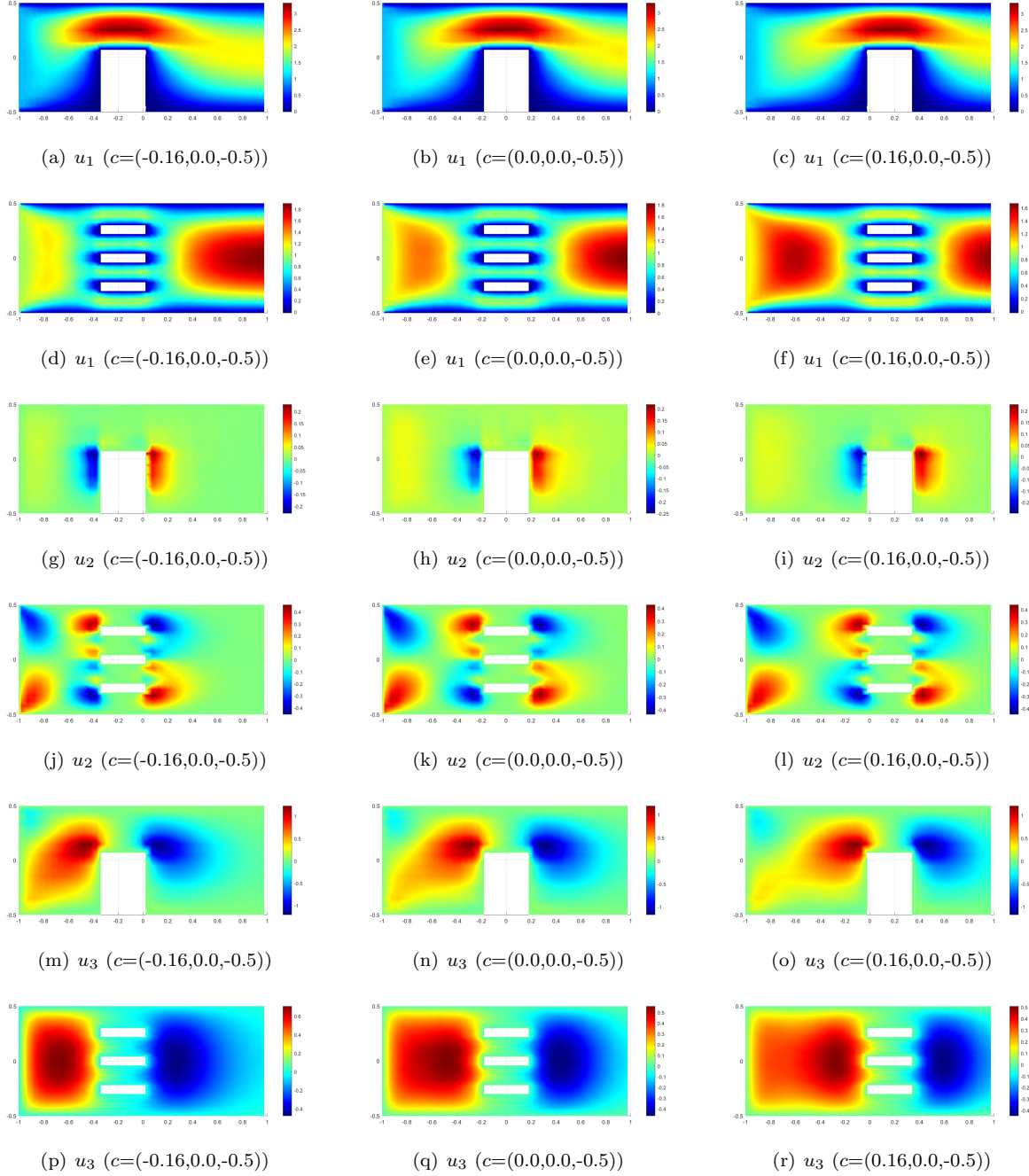


Figure 12: 求解得到的近似解.