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

FNO[1] 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$D \subset \mathbb{R}^{d_{\boldsymbol{x}}}$是计算区域, $d_{\boldsymbol{x}}$, d_a, $d_u \in \mathbb{N}_+$, FNO构造映射G_θ以学习映射$G^\dagger : \mathcal{A} \to \mathcal{U}$.

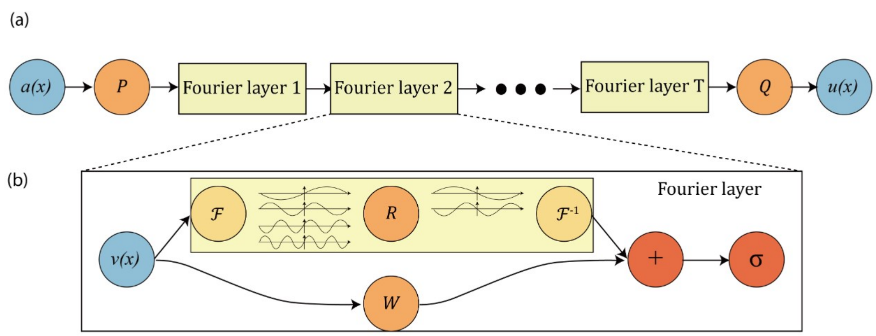


Figure 1: 傅里叶神经算子

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, $\cdots$, v_L (v_1, $\cdots$, $v_T \in \mathbb{R}^{d_v}$), 最后通过全连接层Q进行降维得到最终输出u. 每个傅里叶层的构造为

$$v_{\ell+1}(\boldsymbol{x}) := \sigma\Big(\mathcal{W}v_\ell(\boldsymbol{x}) + \mathcal{K}v_\ell(\boldsymbol{x})\Big), \tag{1}$$

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

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

傅里叶层1中积分变换算子$\mathcal{K}$的定义为:

$$\left[\mathcal{K}v_\ell\right](\boldsymbol{x}) := \int_D \kappa(\boldsymbol{x}, \boldsymbol{y}) v_\ell(\boldsymbol{y}) d\boldsymbol{y}, \tag{3}$$

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

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

积分变换(**??**)等价于

$$\left[\mathcal{K}v_\ell\right](\boldsymbol{x}) := \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}(v_\ell)(\boldsymbol{x}) \tag{5}$$

[1]Z. 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$R_\phi$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$\mathcal{F}v_t$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)

PINO[2] 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

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

根据链式法则,

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

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-FNO[3] 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$\phi: D^C \to D$. 构造全连接网络$g_{\boldsymbol{\theta}}$学习逆映射ϕ^{-1}, 逆映射ϕ^{-1}可以将D上的非均匀网格$\{\boldsymbol{x}^l\}$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

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

实际在应用Geo-FNO时, 往往不需要构造全连接网络$g_{\boldsymbol{\theta}}$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.

[2]Z. 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).

[3]Z. 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$g_{\boldsymbol{\theta}}$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:

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

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

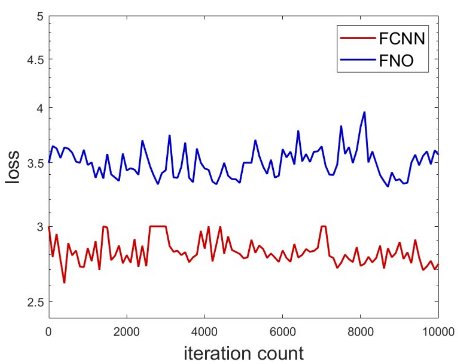
$$\begin{cases} -\Delta u = r_\Omega, & \text{in } \Omega,\\ u = r_\Gamma, & \text{in } \Gamma. \end{cases} \tag{10}$$

计算区域为$\Omega := [-1,1]^2 \setminus \Omega_c$, 其中$\Omega_c$是以$[0,0]$为圆心, 半径为0.2的圆. 真解设置为$u =$. 构造两个网络$g_{\boldsymbol{\theta}_1}$和$f_{\boldsymbol{\theta}_2}$, $g_{\boldsymbol{\theta}_1}$用于将$\Omega$上的原始网格映射到矩形区域$\Omega_c = [0,1]^2$上的计算网格. $f_{\boldsymbol{\theta}_2}$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:

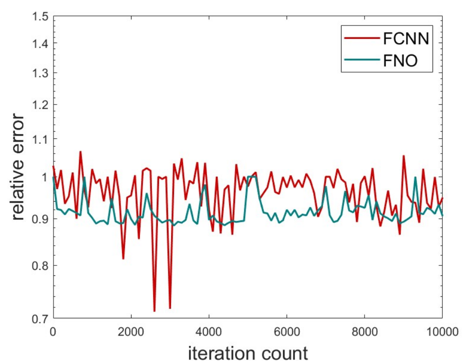
$$L[g_{\boldsymbol{\theta}_1}] := \sum_l \left(g_{\boldsymbol{\theta}_1}(\boldsymbol{x}^l) - (\boldsymbol{\lambda}^l)\right)^2. \tag{11}$$

$$L[f_{\boldsymbol{\theta}_2}] := \sum_{\boldsymbol{\lambda}^l \in S(\Omega)} \left(-\Delta f(\boldsymbol{\lambda}^l) - r_\Omega(\boldsymbol{\lambda}^l)\right)^2 + \sum_{\boldsymbol{\lambda}^l \in S(\Gamma)} \left(f(\boldsymbol{\lambda}^l) - r_\Gamma(\boldsymbol{\lambda}^l)\right)^2 \tag{12}$$

进行两组实验, 在两组实验中网络$g_{\boldsymbol{\theta}_1}$均采用全连接网络构造, 而网络$f_{\boldsymbol{\theta}_2}$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$f_{\boldsymbol{\theta}_2}$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.



(a) loss



(b) error

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$g_{\boldsymbol{\theta}_1}$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:

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

用ϕ^{-1}替换网络$g_{\boldsymbol{\theta}_1}$, 只训练网络$f_{\boldsymbol{\theta}_2}$. 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$g_{\boldsymbol{\theta}_1}$没有准确逼近$\phi^{-1}$.

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$f_{\boldsymbol{\theta}_2}$的困难程度. 为此设计第三个实验, 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

$$\begin{aligned} L[g_{\boldsymbol{\theta}_1}, f_{\boldsymbol{\theta}_2}] \quad := & \sum_{x^l \in S(\Omega)} \left(-\Delta f_{\boldsymbol{\theta}_2}(g_{\boldsymbol{\theta}_1}(\boldsymbol{x}^l)) - r_\Omega(\boldsymbol{x}^l)\right)^2 + \sum_{x^l \in S(\Gamma)} \left(f_{\boldsymbol{\theta}_2}(g_{\boldsymbol{\theta}_1}(\boldsymbol{x}^l)) - r_\Gamma(\boldsymbol{x}^l)\right)^2 \\ & + \beta \sum_{x^l \in S(\Omega) \cup S(\Gamma)} \left(g_{\boldsymbol{\theta}_1}(\boldsymbol{x}^l) - (\boldsymbol{\lambda}^l)\right)^2. \end{aligned} \tag{14}$$

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$e_{\boldsymbol{\lambda}}$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	$e_{\boldsymbol{\lambda}}$
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

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

其中$d_{\boldsymbol{x}} \in \mathbb{N}_+$, u是未知变量, $\boldsymbol{A} \in \mathbb{R}^{d_{\boldsymbol{x}} \times d_{\boldsymbol{x}}}$和$\boldsymbol{B} \in \mathbb{R}^{d_{\boldsymbol{x}}}$是可变参数, 其元素分别记为$a_{j_1,j_2} = (\boldsymbol{A})_{j_1,j_2}$和$b_{j_1} = (\boldsymbol{B})_{j_1}$, $j_1, j_2 = 1 \cdots, d_{\boldsymbol{x}}$, $\boldsymbol{n}$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, $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \varnothing$. 计算区域$\Omega$的形状是可变的, 且存在一个矩体上界$\overline{\Omega}$.

将计算区域的形状参数化为形状参数

$$s_\Omega(\boldsymbol{x}; \Omega) = \begin{cases} 1, & \text{in } \Omega, \\ 0, & \text{in } \overline{\Omega} \setminus \Omega. \end{cases} \tag{16}$$

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

$$\boldsymbol{P}=\begin{cases}[a_{1,1}\ a_{1,2}\cdots a_{d_{\boldsymbol{x}},d_{\boldsymbol{x}}}\ b_1\cdots b_{d_{\boldsymbol{x}}}\ s_\Omega]^\top, & \text{in }\Omega,\\ \boldsymbol{0}, & \text{in }\overline{\Omega}\setminus\Omega,\end{cases}\qquad \widetilde{u}=\begin{cases}u, & \text{in }\Omega,\\ 0, & \text{in }\overline{\Omega}\setminus\Omega,\end{cases}\tag{17}$$

分别属于空间$\mathcal{P}=\mathcal{P}(\overline{\Omega};\mathbb{R}^{d_p})$和$\mathcal{U}=\mathcal{U}(\overline{\Omega};\mathbb{R})$, 其中$d_p=d_{\boldsymbol{x}}\times d_{\boldsymbol{x}}+d_{\boldsymbol{x}}+1$. 这样存在映射$G^\dagger:\mathcal{P}\to\mathcal{U}$. 目标是构造神经网络$G_{\boldsymbol{\theta}}$, 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$\widetilde{u}$记为u,

2.2 网络输入和输出

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

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

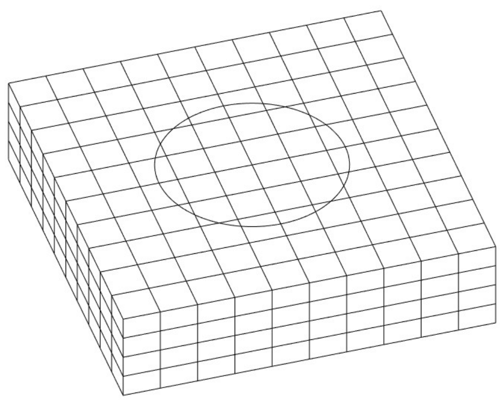


Figure 3: 计算区域和网格示例.

2.3 损失函数

对于落在计算区域内的每个网格点$\boldsymbol{x}_{i_1,i_2,i_3}$, 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$\boldsymbol{P}^{i_p}$下的问题(15), 将其中控制方程在点$\boldsymbol{x}^{i_1,i_2,i_3}$所属的单元$c^{i_1,i_2,i_3}$上进行积分, 有

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

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, $k=1,\cdots,n_f^{i_1,i_2,i_3}$, $n_f^{i_1,i_2,i_3}$是面的数目.

通过数值积分, 可将上式近似为

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

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$\boldsymbol{x}_k^{i_1,i_2,i_3}$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$\boldsymbol{x}^{i_1,i_2,i_3}$处方程的残差, 其中$(u)_k^{i_1,i_2,i_3}$和$\Big(\frac{\partial u}{\partial x_{j_2}}\Big)_k^{i_1,i_2,i_3}$将通过计算插值函数$\widehat{u}^{i_1,i_2,i_3}$的值及其导数进行估计. 插值函数$\widehat{u}^{i_1,i_2,i_3}$的构造为二次多项式

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

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

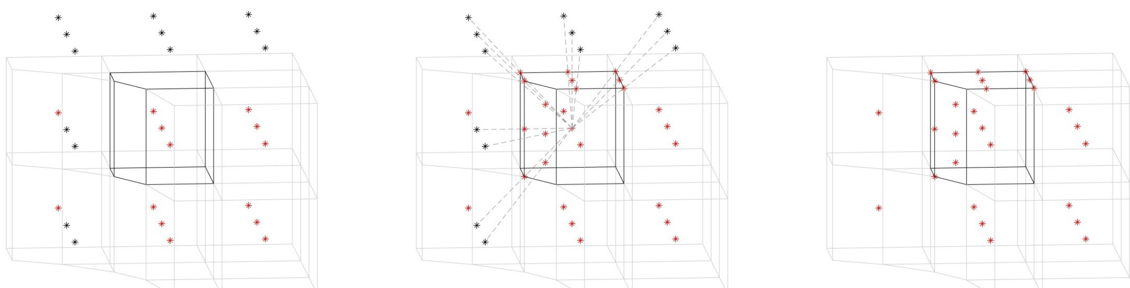


Figure 4: 经边界切割的单元和插值节点的选取(红点位于计算区域内部和边界, 黑点位于计算区域外部).

为计算系数$\widehat{a}^{s_1,s_2,s_3}$, 首先选取插值节点$S_i^{i_1,i_2,i_3}=\{\{\{\widehat{\boldsymbol{x}}^{t_1,t_2,t_3}\}_{t_1=0}^2\}_{t_2=0}^2\}_{t_3=0}^2$, 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$\boldsymbol{x}^{i_1-1+t_1,i_2-1+t_2,i_3-1+t_3}$位于计算区域内部, 则作为插值节点$\widehat{\boldsymbol{x}}^{t_1,t_2,t_3}$, 反之, 则连接点$\boldsymbol{x}^{i_1,i_2,i_3}$与点$\boldsymbol{x}^{i_1-1+t_1,i_2-1+t_2,i_3-1+t_3}$, 取连线与计算区域边界$\partial\Omega$的交点作为插值节点$\widehat{\boldsymbol{x}}^{t_1,t_2,t_3}$. 根据插值节点, 构造线性方程组

$$\boldsymbol{\Psi}\widehat{\boldsymbol{A}}=\boldsymbol{V}[G_{\boldsymbol{\theta}}], \tag{21}$$

其中

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

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

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

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

$$\widehat{a}^{s_1,s_2,s_3} = \sum_{t_1,t_2,t_3} \widehat{b}^{s_1,s_2,s_3}_{t_1,t_2,t_3} v^{t_1,t_2,t_3}[G_{\boldsymbol{\theta}}]. \tag{22}$$

将(22)代入插值函数(20), 得到

$$\begin{aligned} \widehat{u}^{i_1,i_2,i_3}(\boldsymbol{x}) &= \sum_{s_1,s_2,s_3}\Big(\sum_{t_1,t_2,t_3} \widehat{b}^{s_1,s_2,s_3}_{t_1,t_2,t_3} v^{t_1,t_2,t_3}[G_{\boldsymbol{\theta}}]\Big)\phi^{s_1,s_2,s_3}(\boldsymbol{x}) \\ &= \sum_{t_1,t_2,t_3}\Big(\sum_{s_1,s_2,s_3} \widehat{b}^{s_1,s_2,s_3}_{t_1,t_2,t_3} \phi^{s_1,s_2,s_3}(\boldsymbol{x})\Big) v^{t_1,t_2,t_3}[G_{\boldsymbol{\theta}}]. \end{aligned} \tag{23}$$

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

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

从而插值函数$\widehat{u}^{i_1,i_2,i_3}$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$\boldsymbol{x}^{i_1,i_2,i_3}$处的残差

$$\begin{aligned} R^{i_p,i_1,i_2,i_3}[G_{\boldsymbol{\theta}}] = \quad & -\sum_k \Big(\sum_{j_1}\sum_{j_2} a^{i_p}_{j_1,j_2} n_{j_1}\Big)_k^{i_1,i_2,i_3} \frac{\partial \widehat{u}^{i_1,i_2,i_3}}{\partial x_{j_2}}(\boldsymbol{x}_k^{i_1,i_2,i_3}; S_v^{i_1,i_2,i_3}[G_{\boldsymbol{\theta}}]) D_k^{i_1,i_2,i_3} \\ & + \sum_k \Big(\sum_{j_1} b^{i_p}_{j_1} n_{j_1}\Big)_k^{i_1,i_2,i_3} \widehat{u}^{i_1,i_2,i_3}(\boldsymbol{x}_k^{i_1,i_2,i_3}; S_v^{i_1,i_2,i_3}[G_{\boldsymbol{\theta}}]) D_k^{i_1,i_2,i_3} \\ & - r^{i_1,i_2,i_3} V^{i_1,i_2,i_3}. \end{aligned} \tag{25}$$

所有样本点的残差进一步构成训练网络所用的损失函数

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

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:

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

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

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

$$\begin{cases} R_1^{i_1,i_2,i_3}[G_{\boldsymbol{\theta}}] & := \sum_k \Big(\sum_{j_1}\Big(-\sum_{j_2} a_{j_1,j_2}\dfrac{\partial u_1}{\partial x_{j_2}} + b_{j_2}u_1\Big)n_{j_1} + pn_1\Big)_k^{i_1,i_2,i_3} D_k^{i_1,i_2,i_3} \\ R_2^{i_1,i_2,i_3}[G_{\boldsymbol{\theta}}] & := \sum_k \Big(\sum_{j_1}\Big(-\sum_{j_2} a_{j_1,j_2}\dfrac{\partial u_2}{\partial x_{j_2}} + b_{j_2}u_2\Big)n_{j_1} + pn_2\Big)_k^{i_1,i_2,i_3} D_k^{i_1,i_2,i_3} \\ R_3^{i_1,i_2,i_3}[G_{\boldsymbol{\theta}}] & := \sum_k \Big((u_1)_k^{i_1,i_2,i_3} n_1^{i_1,i_2,i_3} + (u_2)_k^{i_1,i_2,i_3} n_2^{i_1,i_2,i_3}\Big) D_k^{i_1,i_2,i_3} \end{cases} \tag{28}$$

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

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

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

其中$I_k^{i_1,i_2,i_3}$是点$(\boldsymbol{x})_k^{i_1,i_2,i_3}$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

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

其中I^{s_1,s_2,s_3}是点$\boldsymbol{x}^{s_1,s_2,s_3}$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

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

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(Laméparameters), 被定义为

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

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

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

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$12\times32\times32$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$10\times10\times10$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.

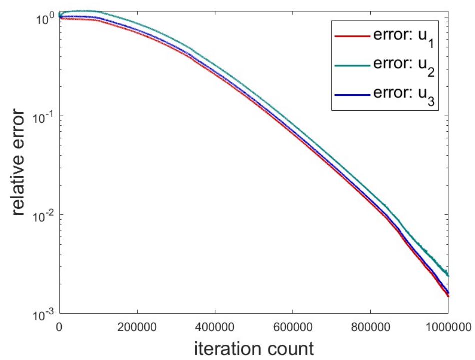


Figure 5: 训练过程中近似解误差的变化情况.

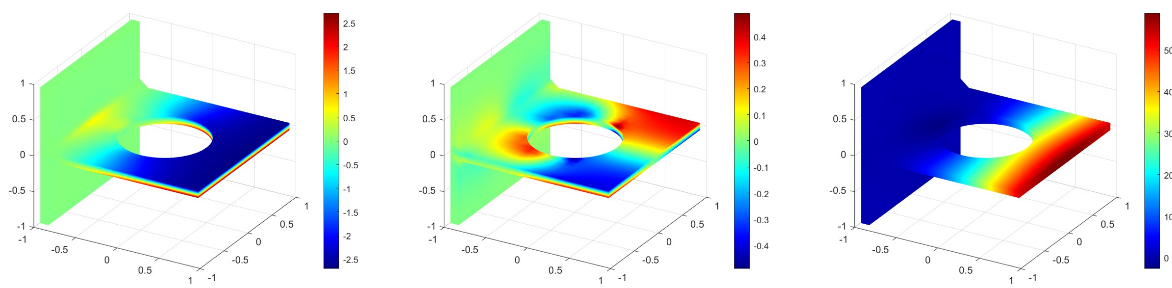


Figure 6: 求解得到的近似解.

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$\Omega_c$. 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:

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

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$\boldsymbol{P} = [u_i \ c]^\top$.

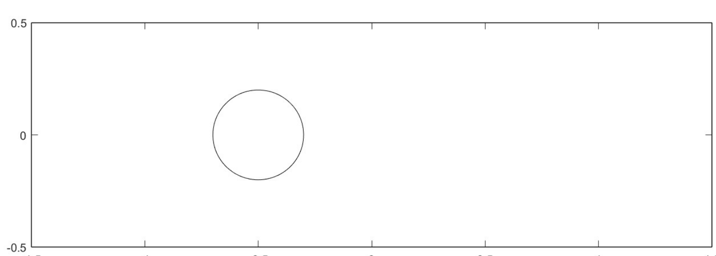


Figure 7: 计算区域

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.

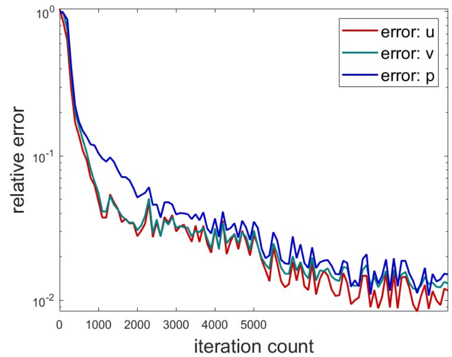
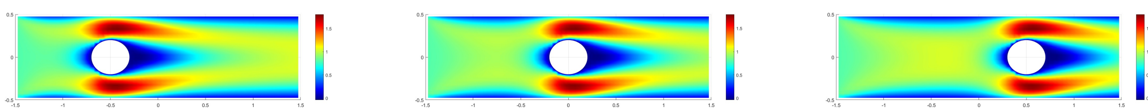


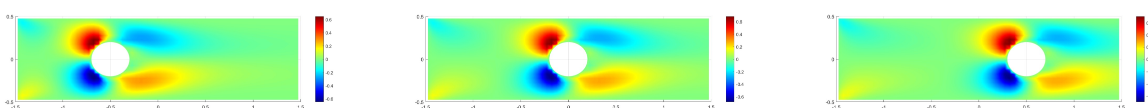
Figure 8: 训练过程中近似解误差的变化情况.



(a) u_1 (u_i=0.8, c=(-0.5,0.0))

(b) u_1 (u_i=0.8, c=(-0.5,0.0))

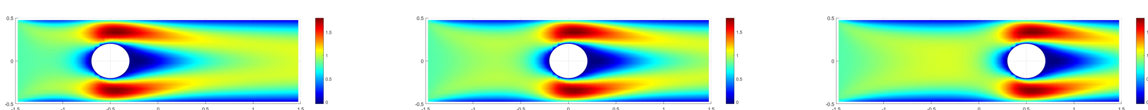
(c) u_1 (u_i=0.8, c=(0.0,0.0))



(d) u_2 (u_i=0.8, c=(0.0,0.0))

(e) u_2 (u_i=0.8, c=(0.5,0.0))

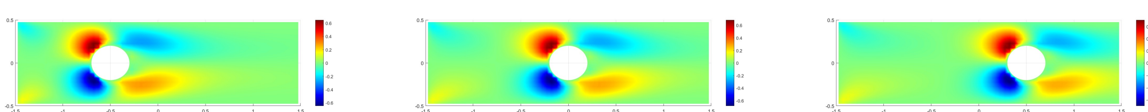
(f) u_2 (u_i=0.8, c=(0.5,0.0))



(g) u_1 (u_i=1.2, c=(-0.3,0.0))

(h) u_1 (u_i=1.2, c=(-0.3,0.0))

(i) u_1 (u_i=1.2, c=(0.0,0.0))



(j) u_2 (u_i=1.2, c=(0.0,0.0))

(k) u_2 (u_i=1.2, c=(0.3,0.0))

(l) u_2 (u_i=1.2, c=(0.3,0.0))

Figure 9: 求解得到的近似解.