

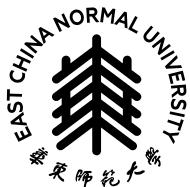
2022 届研究生硕士学位论文

分类号: \_\_\_\_\_

学校名称: 华东师范大学

密 级: \_\_\_\_\_

学 号: 00000000001



华东師範大學

East China Normal University

硕士学位论文

MASTER'S DISSERTATION

论文题目: 你的论文标题——如强人  
工智能的实现方法

院 系: 数 学 科 学 学 院

专 业: 智 能 数 学

研究方向: 人 工 智 能

指导老师: XX 教 授

学位申请人: XXX

2022 年 10 月 01 日

本页留白

Dissertation for master degree in 2022

University Code: 10269

Student ID: 00000000001

# East China Normal University

**Title:** The title of your paper— such as the implementation of strong artificial intelligence

**Department:** School of Mathematical Sciences

**Major:** Intelligent mathematics

**Research Direction:** Artificial Intelligence

**Supervisor:** XX (Associate Professor)

**Candidate:** XXX

本页留白

## 华东师范大学学位论文原创性声明

郑重声明：本人呈交的学位论文《你的论文标题——如强人工智能的实现方法》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名：\_\_\_\_\_

日期： 年 月 日

## 华东师范大学学位论文著作权使用声明

《你的论文标题——如强人工智能的实现方法》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的著作权归本人所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和学校指定的相关机构送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

- ( ) 1. 经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文 \*，于 年 月 日解密，解密后适用上述授权。
- ( ) 2. 不保密，适用上述授权。

导师签名：\_\_\_\_\_

本人签名：\_\_\_\_\_

年 月 日

\* “涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

本页留白

### XXX 硕士学位论文答辩委员会成员名单

姓名	职称	单位	备注
XXX	副教授	华东师范大学数学科学学院	主席
XXX	副教授	华东师范大学数学科学学院	
XX	教授	华东师范大学数学科学学院	

本页留白

## 摘要

摘要内容

**关键词:** 论文关键词；用中文分号分隔；如强人工智能；强化学习

本页留白

## ABSTRACT

Summary content

**Keywords:** Paper keywords; Separated by semicolons; Such as strong artificial intelligence; Reinforcement learning

本页留白

# 目录

摘要	i
<b>Abstract</b>	iii
<b>第一章 论文主要部分的写法</b>	1
1.1 论文的语言及表述 . . . . .	1
1.2 论文题目的写法 . . . . .	1
1.3 摘要的写法 . . . . .	1
1.4 引言的写法 . . . . .	2
1.5 正文的写法 . . . . .	2
1.6 结论的写法 . . . . .	2
<b>第二章 模型与算法描述</b>	3
2.1 Navier-Stokes/Darcy 耦合模型 . . . . .	3
2.2 神经网络与深度学习算法 . . . . .	6
2.3 Navier-Stokes/Darcy 耦合模型的神经网络求解方法 . . . . .	10
2.4 损失函数的合理性证明 . . . . .	13
<b>第三章 正问题的数值实验</b>	19
3.1 实验一：真解数值实验 . . . . .	19
3.2 实验二：带参数的数值实验 . . . . .	22
3.3 实验三：稳态实验 . . . . .	23
3.4 实验四：BJS 交界面实验 . . . . .	26
<b>第四章 Navier-Stokes/Darcy 耦合模型的反问题实验</b>	29
4.1 区域反问题实验 . . . . .	29
4.2 物理量反问题实验 . . . . .	31
4.3 方程参数反问题实验 . . . . .	31
<b>第五章 总结与展望</b>	35
<b>参考文献</b>	37
<b>致谢</b>	39

本页留白

# 第一章 论文主要部分的写法

研究生学位论文撰写，除表达形式上需要符合一定的格式要求外，内容方面上也要遵循一些共性原则。

通常研究生学位论文只能有一个主题（不能是几块工作拼凑在一起），该主题应针对某学科领域中的一个具体问题展开深入、系统的研究，并得出有价值的研究结论。学位论文的研究主题切忌过大，例如，“中国国有企业改制问题研究”这样的研究主题过大，因为“国企改制”涉及的问题范围太广，很难在一本研究生学位论文中完全研究透彻。

## 1.1 论文的语言及表述

除国际研究生外，学位论文一律须用汉语书写。学位论文应当用规范汉字进行撰写，除古汉语研究中涉及的古文字和参考文献中引用的外文文献之外，均采用简体汉字撰写。

国际研究生一般应以中文或英文书写学位论文，格式要求同上。论文须用中文封面。

研究生学位论文是学术作品，因此其表述要严谨简明，重点突出，专业常识应简写或不写，做到立论正确、数据可靠、说明透彻、推理严谨、文字凝练、层次分明，避免使用文学性质的或带感情色彩的非学术性语言。

论文中如出现一个非通用性的新名词、新术语或新概念，需随即解释清楚。

## 1.2 论文题目的写法

论文题目应简明扼要地反映论文工作的主要内容，力求精炼、准确，切忌笼统。论文题目是对研究对象的准确、具体描述，一般要在一定程度上体现研究结论，因此，论文题目不仅应告诉读者这本论文研究了什么问题，更要告诉读者这个研究得出的结论。例如：“在事实与虚构之间：梅乐、卡彭特、沃尔夫的新闻观”就比“三个美国作家的新闻观研究”更专业、更准确。

## 1.3 摘要的写法

论文摘要是对论文研究内容的高度概括，应具有独立性和自含性，即应是一篇简短但意义完整的文章。通过阅读论文摘要，读者应该能够对论文的研究方法及结论有一个整体性的了解，因此摘要的写法应力求精确简明。论文摘要应包括

对问题及研究目的的描述、对使用的方法和研究过程进行的简要介绍、对研究结论的高度凝练等，重点是结果和结论。

论文摘要切忌写成全文的提纲，尤其要避免“第1章……；第2章……；……”这样的陈述方式。

## 1.4 引言的写法

一篇学位论文的引言大致包含如下几个部分：1、问题的提出；2、选题背景及意义；3、文献综述；4、研究方法；5、论文结构安排。

- 问题的提出：要清晰地阐述所要研究的问题“是什么”。<sup>1</sup>
- 选题背景及意义：论述清楚为什么选择这个题目来研究，即阐述该研究对学科发展的贡献、对国计民生的理论与现实意义等。
- 文献综述：对本研究主题范围内的文献进行详尽的综合述评，“述”的同时一定要有“评”，指出现有研究状态，仍存在哪些尚待解决的问题，讲出自己的研究有哪些探索性内容。
- 研究方法：讲清论文所使用的学术研究方法。
- 论文结构安排：介绍本论文的写作结构安排。

## 1.5 正文的写法

本部分是论文作者的研究内容，不能将他人研究成果不加区分地掺和进来。已经在引言的文献综述部分讲过的内容，这里不需要再重复。各章之间要存在有机联系，符合逻辑顺序。

## 1.6 结论的写法

结论是对论文主要研究结果、论点的提炼与概括，应精炼、准确、完整，使读者看后能全面了解论文的意义、目的和作品内容。结论是最终的、总体的结论，不是正文各章小结的简单重复。结论应包括论文的核心观点，主要阐述作者的创造性工作及所取得的研究成果在本领域中的地位、作用和意义，交代研究工作的局限，提出未来工作的意见或建议。同时，要严格区分自己取得的成果与指导教师及他人的学术成果。

在评价自己的研究工作成果时，要实事求是，除非有足够的证据表明自己的研究是“首次”、“领先”、“填补空白”的，否则应避免使用这些或类似词语。

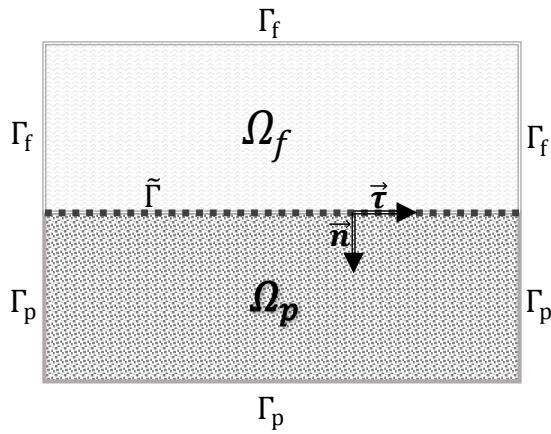
---

<sup>1</sup>选题时切记要有“问题意识”，不要选不是问题的问题来研究。

## 第二章 模型与算法描述

### 2.1 Navier-Stokes/Darcy 耦合模型

“Navier-Stokes/Darcy 耦合模型”是一种描述普通介质和多孔介质共存的物理场景中的流体运动规律的偏微分方程模型。首先引入一个简化场景，如图2.1所示：



图示有界区域  $\Omega_f \subset \mathbb{R}^d$  为自由流 (free fluid) 区域，有界区域  $\Omega_p \subset \mathbb{R}^d$  为多孔介质流 (porous media flow) 区域， $d=2$ 或 $3$  代表空间维度。 $\Omega_f \cap \Omega_p = \emptyset$ ，而两区域相邻边界  $\tilde{\Gamma} = \bar{\Omega}_f \cap \bar{\Omega}_p$  称为交界面 (interface)。在交界面  $\tilde{\Gamma}$  上任意一点的单位法向量 (由自由流区域  $\Omega_f$  指向多孔介质流区域  $\Omega_p$ ) 记为  $\vec{n}$ ；任意一点的切空间的正交基记为  $\vec{t}_i, i = 1, \dots, d-1$  (对于如图所示  $d=2$  的情况，任意一点的切空间的正交基即为其切向量本身，记为  $\vec{r}$ )。 $\Omega_f, \Omega_p$  除交界面之外的其他边界记为  $\Gamma_f = \partial\Omega_f / \tilde{\Gamma}, \Gamma_p = \partial\Omega_p / \tilde{\Gamma}$ 。除此之外，模型将关注一段时间范围内的流体运动，此时间范围记作  $T = [t_0, t_1] \subset \mathbb{R}$ 。

图 2.1: NS/D 简化示意图

#### 2.1.1 Navier-Stokes 方程组

自由流区域  $\Omega_f$  的流体运动由 Navier-Stokes 方程组控制：

$$\begin{cases} \rho \left( \frac{\partial \vec{u}_f}{\partial t} + (\vec{u}_f \cdot \nabla) \vec{u}_f \right) - \nabla \cdot \mathbf{T}(\vec{u}_f, p_f) = \rho \vec{g}_f & \text{in } \Omega_f \times T \\ \nabla \cdot \vec{u}_f = 0 & \text{in } \Omega_f \times T \end{cases} \quad (2.1)$$

方程中符号涉及到的物理单位列出如下：

- $\vec{u}_f : m \cdot s^{-1}$
- $p_f : kg \cdot m^{-1} \cdot s^{-2}$

- $\rho : kg \cdot m^{-3}$
- $\mu : kg \cdot m^{-1} \cdot s^{-1}$
- $\vec{g}_f : m \cdot s^{-2}$

其中:  $\rho$  为流体密度, 由于假设流体是不可压缩的, 所以看做一个常量。

$\vec{u}_f(\vec{x}; t) \in \mathcal{C}^2(\Omega_f \times T; \mathbb{R}^d)$ ;  $\vec{x} \in \Omega_f$ ,  $t \in T$  代表流体速度场函数,  $d$  代表空间维数;  $t$  代表时间。 $p_f \in \mathcal{C}^1(\Omega_f \times T; \mathbb{R})$  代表流体的压力函数。 $\vec{g}_f \in \mathcal{C}(\Omega_f \times T; \mathbb{R}^d)$  代表流体受到的合外加速度场函数, 实际情况下一般会取重力场。

$\mathbf{T}(\vec{u}_f, p_f) = 2\mu\mathbf{D}(\vec{u}_f) - p_f\mathbf{I}$  被称为柯西应力张量 (Cauchy stress tensor), 其中  $\mu$  为流体动黏滞度 (Dynamic viscosity), 看作一个常数参数;  $\mathbf{D}(\vec{u}_f) = \frac{1}{2}(\nabla \vec{u}_f + \nabla \vec{u}_f^T)$  被称为形变张量 (deformation tensor),  $\mathbf{I}$  为单位矩阵。

通常该方程组还需要补充初边界条件才能进行求解, 本文默认使用狄利克雷边界条件, 表示如下:

$$\begin{cases} \vec{u}_f = \vec{u}_f^{\Gamma_f} & \text{on } \Gamma_f \times T, \\ \vec{u}_f = \vec{u}_f^{t_0} & \text{in } \Omega_f \times \{t_0\}. \end{cases} \quad (2.2)$$

### 2.1.2 Darcy 方程组

多孔介质流区域  $\Omega_p$  中流体运动由以下方程组描述:

$$\begin{cases} S_0 \frac{\partial \phi}{\partial t} + \nabla \cdot \vec{u}_p = f_p & \text{in } \Omega_p \times T, \\ \vec{u}_p = -\mathbf{K} \nabla \phi & \text{in } \Omega_p \times T. \end{cases} \quad (2.3)$$

方程中符号涉及到的物理单位列出如下:

- $S_0$ : 常数
- $\vec{u}_p : m \cdot s^{-1}$
- $f_p : s^{-1}$
- $\phi = z + \frac{p_p}{g\rho} : m$
- $p_p : kg \cdot m^{-1} \cdot s^{-2}$
- $\rho : kg \cdot m^{-3}$
- $g : m \cdot s^{-2}$

$$\bullet \quad \mathbf{K} = \frac{\rho g \kappa \mathbf{I}}{\mu} : m \cdot s^{-1}$$

$$\bullet \quad \kappa : m^2$$

$$\bullet \quad \mu : kg \cdot m^{-1} \cdot s^{-1}$$

其中:  $S_0$  为质量比储存相关系数 (specific mass storativity coefficient)。 $\phi = z + \frac{p_p}{g\rho}$  称为“测压头 (pressure head)”, “测压头”本来意义是液柱的高度, 它能够代表液柱对容器底部的压力大小, 在流体力学实验中就是与流体压力相关的一个量。这里的  $p_p$  就代表着多孔介质区域中流体的动态压力 (dynamic pressure), 或者称为 Darcy 压力,  $g$  为重力加速度,  $\rho$  为流体密度,  $z$  是一种相对深度, 称为海拔标头 (elevation head), 可以看作一种常数参数。因此  $\phi$  与  $p_p$  就是一种线性关系, 对方程本身来说区别不大。 $\mathbf{K} = \frac{\rho g \kappa \mathbf{I}}{\mu}$  为导水率 (Hydraulic conductivity), 它代表了多孔介质中流体流动的难易程度, 其中  $k$  为渗透率 (permeability),  $\mu$  为动态粘度 (Dynamic viscosity)。

方程组中第二个式子即为 Darcy 定律, 它的物理含义基本上就是说多孔介质中流速与压力的梯度是负线性相关的, 这在直观上很容易理解: 压力没有变化, 那流体是平静的没有运动; 存在压力差, 那么流体就会相应运动。本方程组同样需要补充初边值条件才能进行求解, 本文也形式化地表示如下:

$$\begin{cases} \phi = \phi_{\Gamma_p} & \text{on } \Gamma_p \times T, \\ \phi = \phi^{t_0} & \text{in } \Omega_p \times \{t_0\}. \end{cases} \quad (2.4)$$

### 2.1.3 交界面条件

除了控制两个区域流体运动的方程组之外, 本模型中还有一项重要约束就是耦合这两区域流体运动的交界面条件了, (2.5) 为分别从物理学的三个角度构建的交界面条件:

$$\begin{aligned} \vec{u}_f \cdot \vec{n} &= \vec{u}_p \cdot \vec{n} \quad \text{on } \tilde{\Gamma} \\ -(\mathbf{T}(\vec{u}_f, p_f) \vec{n}) \cdot \vec{n} &= g\rho(\phi - z) \quad \text{on } \tilde{\Gamma} \\ -(\mathbf{T}(\vec{u}_f, p_f) \vec{n}) \cdot \vec{\tau}_i &= \alpha \sqrt{\frac{g\rho\mu}{(\mathbf{K}\vec{\tau}_i) \cdot \vec{\tau}_i}} (\vec{u}_f - \vec{u}_p) \cdot \vec{\tau}_i, i = 1, \dots, (d-1) \quad \text{on } \tilde{\Gamma} \end{aligned} \quad (2.5)$$

其中第一个公式代表了交界面的质量守恒定律, 直观含义即为两区域在交界面法线方向的流速相等; 第二个公式为交界面的压力平衡条件; 第三个公式被称为 Beavers-Joseph 交界面条件<sup>[1]</sup> (简称 BJ 条件), 其中  $\alpha$  在实践中是一个需要通过实验确定的常数参数。这三个交界面条件共同约束着本模型中两区域流体运动进行着的相互作用。

以上公式涉及到的一些符号约定如下：

$\nabla = (\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_d})^T$  为空间梯度算子（只对空间维度求梯度）。

$\vec{u}_f \cdot \nabla = \sum_{i=1}^d u_{fx_i} \cdot \frac{\partial}{\partial x_i}$ , 其中  $x_i$  表示空间维度第  $i$  个分量,  $u_{fx_i} \in \mathcal{C}^2(\Omega_f \times T; \mathbb{R})$  为对应分量的速度大小(函数), 也就是说,  $\vec{u}_f = \overrightarrow{(u_{fx_1}, \dots, u_{fx_d})}$ 。

$\Delta = \nabla^2 = \nabla \cdot \nabla = \sum_{i=1}^d \frac{\partial^2}{\partial x_i^2}$  为空间拉普拉斯算子。

当算子与函数由“.”连接时代表以向量点积的形式作用, 如  $\nabla \cdot \vec{u}_f = \sum_{i=1}^d \frac{\partial u_{fx_i}}{\partial x_i}$  等, 所以“ $\nabla \cdot$ ”本质上代表的是散度算子; 当算子直接连接函数且函数为向量函数时, 则代表数乘或矩阵作用, 如  $\Delta \vec{u}_f = (\sum_{i=1}^d \frac{\partial^2 u_{fx_1}}{\partial x_i^2}, \dots, \sum_{i=1}^d \frac{\partial^2 u_{fx_d}}{\partial x_i^2})_{1 \times d}$ ;  $\nabla \vec{u}_f = \left( a_{ij} = \frac{\partial u_{fx_j}}{\partial x_i} \right)_{d \times d}$ 。

## 2.2 神经网络与深度学习算法

神经网络是深度学习中一种函数拟合的工具, 它可以拟合各种所需要的未知函数。

### 2.2.1 神经网络模型

神经网络模型最初是受到大脑中神经元结构的启发建立的。神经元有三个特点, 1. 它具有多个输入, 一个输出; 2. 神经元的输入通过突触传递, 其粗细不一, 意味着传递信号的能力不同; 3. 输入可以叠加, 但是输出有一定范围且存在阈值, 也就是说输出与输入并不是线性关系。

首先建立单个神经元的数学模型: 根据特点 1, 把神经元看作一个定义在向量空间中的实值函数  $e(\vec{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ , 其中  $\vec{x} = (x_1, \dots, x_n)$  作为  $n$  维向量代表神经元有  $n$  个输入; 根据特点 2, 定义权重向量  $\vec{w} = (w_1, \dots, w_n) \in \mathbb{R}^n$ , 其中  $w_i$  代表输入  $x_i$  对应的信号权重; 再根据特点 3, 引入一个有界非线性函数(后面称其为激活函数)  $\sigma : \mathbb{R} \mapsto \mathbb{R}$  以及一个阈值参数(后面称其为偏置)  $b \in \mathbb{R}$ , 最后得到单个神经元的数学模型表示为公式 (2.6):

$$e(\vec{x}) = \sigma(\vec{w} \cdot \vec{x} + b) \quad (2.6)$$

下面定义单层全连接神经网络模型, 它是由共享同一个输入的多个神经元组合而成, 设输入为  $\vec{x} \in \mathbb{R}^n$ , 且被  $m$  个神经元  $e_i(\vec{x}) = y_i, i = 1, \dots, m$  共享, 那么整个单层神经网络模型可以看作一个  $m$  维向量值函数  $\vec{l}(\vec{x}) = (e_1(\vec{x}), \dots, e_m(\vec{x})) : \mathbb{R}^n \mapsto \mathbb{R}^m$ , 同时一般规定一层神经网络中所有神经元的激活函数  $\sigma_i$  都相同, 记作  $\sigma_i = \sigma, i = 1, \dots, m$ , 所以最终单层模型表示为公式 (2.7)

$$\vec{l}(\vec{x}) = \sigma(\vec{x}\mathbf{W} + \vec{b}) \quad (2.7)$$

其中权重矩阵为：

$$\mathbf{W} = \begin{pmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nm} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

$w_{ij}$  为第  $j$  个神经元函数中权重向量的第  $i$  个分量。

偏置向量  $\vec{b} = (b_1, \dots, b_m) \in \mathbb{R}^m$ ,  $b_i$  为  $e_i$  的偏置, 还要注意此写法中激活函数  $\sigma$  作用在向量上指的是其直接作用在向量的每一个分量上。

最经典的全连接神经网络 (Fully Connected Neural Network) 模型就是由多个单层全连接网络模型进行叠加而组成的,  $s$  层全连接网络可表达为

$$\begin{aligned} \mathcal{N}(\vec{x}, \theta) &= (\vec{l}_s \circ \cdots \circ \vec{l}_1)(\vec{x}) \\ &= \sigma_s(\cdots \sigma_2(\sigma_1(\vec{x}\mathbf{W}_1 + \vec{b}_1)\mathbf{W}_2 + \vec{b}_2) \cdots \mathbf{W}_s + \vec{b}_s) \end{aligned} \quad (2.8)$$

其中  $\circ$  为函数复合运算,  $\vec{l}_i(\vec{x}_i), i = 1, \dots, s$  为单层全连接网络,  $\mathbf{W}_i, \vec{b}_i$  为  $\vec{l}_i(\vec{x}_i)$  的权重矩阵和偏置向量; 不同的  $\vec{l}_i(\vec{x}_i)$  的权重和偏置可以不同, 输入与输出维度也可能不同, 但是要求  $\vec{l}_i(\vec{x}_i)$  的输出维度与  $\vec{l}_{i+1}(\vec{x}_{i+1})$  的输入维度相同, 这样才能进行函数复合, 也就是说, 设  $n_k$  为第  $k$  层网络的输入维度, 则  $\mathbf{W}_k \in \mathbb{R}^{n_k \times n_{k+1}}, b_k \in \mathbb{R}^{n_{k+1}}, k = 1, \dots, s-1$ ; 集合  $\theta = \{w_{ij;k}, b_{i;k} \mid \mathbf{W}_k = (w_{ij;k}), \vec{b}_k = (\dots, b_{i;k}, \dots), k = 1, \dots, s\}$  统称为该神经网络的参数, 对于相同的输入参数发生改变则神经网络的输出也就发生了改变; 易得  $\theta \in \Theta := \mathbb{R}^{\sum_{k=1}^{s-1} n_k \times n_{k+1} + n_{k+1}}$ 。因此, 一个全连接神经网络可以看作就是具有上述公式结构的拥有参数  $\theta$  的向量值函数。值得注意的是, 在实践中神经网络的激活函数一般都为连续函数或者满足  $\sigma \in L^p(\mathbb{R}; \mathbb{R})$ , 因此这里不妨假设  $\mathcal{N}(\cdot, \theta) \in L^p(\mathbb{R}^n; \mathbb{R}^m), 1 \leq p < \infty$ , 同时最后一层的激活函数往往不设置或者看作规定为恒等函数:  $\sigma_s(x) = x$ , 这样做以避免神经网络只能输出有限范围的值。

## 2.2.2 损失函数

神经网络模型的结构及其大量的参数使得其拥有一种强大的函数拟合能力。所谓其函数拟合能力, 首先要定义一种衡量神经网络  $\mathcal{N}(\cdot, \theta)$  与待拟合函数  $\vec{f}$  的“距离”的标准; 假设  $\vec{f}, \mathcal{N}(\cdot, \theta) \in L^p(\mathbb{R}^n; \mathbb{R}^m)$ , 最自然的方式就是使用函数空间  $L^p$  的范数进行构造:

$$d(\mathcal{N}(\cdot, \theta), \vec{f}) = \|\mathcal{N}(\cdot, \theta) - \vec{f}\|_{L^p} = \left( \int_X \left\| \mathcal{N}(\cdot, \theta) - \vec{f} \right\|_{L^p}^p d\vec{x} \right)^{\frac{1}{p}} \quad (2.9)$$

其中  $X \subseteq \mathbb{R}^n$  为  $\vec{f}(\vec{x})$  的定义域, 这里假设  $X$  是有界的;  $\|\cdot\|_{l^p}$  是向量值域空间  $\mathbb{R}^m$  的  $p$  范数。对于函数  $\vec{f}(\vec{x})$  以及一个预先指定的极小量  $0 < \varepsilon \ll 1$  (如  $1^{-10}$ ), 若存在一个神经网络  $\mathcal{N}(\cdot, \theta)$ , 使得  $d(\mathcal{N}(\cdot, \theta), \vec{f}) < \varepsilon$ , 则称该神经网络  $\mathcal{N}(\cdot, \theta)$  拥有对函数  $\vec{f}(\vec{x})$  的拟合能力。

事实上大名鼎鼎的通用近似定理 (Universal approximation theorem, 其实它不是具体某一个定理, 而是代表历史上对神经网络拟合能力的刻画越来越强的一组定理)<sup>[2-4]</sup> 指出, 在满足一定的条件下的神经网络对给定的函数空间内 (如  $L^p$  空间) 的任意函数均拥有很强的拟合能力, 关于它的细节将在第三章中描述。

然而在实践中所关心的是如何找到一个成功拟合了所需函数的神经网络。标准做法是, 对于待拟合函数  $\vec{f}$ , 先通过经验确定一个神经网络的结构  $\mathcal{N}(\cdot, \theta)$ , 结构就是指其网络层数, 每层的激活函数以及权重矩阵和偏置向量的形状, 但是具体的参数值待定。随后将问题转化为对该神经网络参数的优化问题 (2.10):

$$\min_{\theta \in \Theta} \|\mathcal{N}(\cdot, \theta) - \vec{f}\|_{L^p}. \quad (2.10)$$

接下来的目标就是使用各种优化算法对该问题进行求解, 这个过程通常被称为“神经网络的训练”。

但是在实践中, 计算机无法处理无限多个数据, 因此不能直接对上述优化问题中的积分进行计算, 而是使用了一种数值积分的算法进行逼近: 首先对定义域  $X$  进行均匀随机采样, 得到一个有限集合  $\hat{X} \subset X \subseteq \mathbb{R}^n$ , 然后令  $\mathcal{T}_{\hat{X}} = \{\hat{x}, \vec{f}(\hat{x}) \mid \hat{x} \in \hat{X}\}$ , 通常称集合  $\mathcal{T}_{\hat{X}}$  为“训练集”; 另一方面, 在大多数问题中待拟合函数  $\vec{f}$  本就是未知的, 只能采样到其部分输入输出作为训练集, 此时也往往假设采样到的分布为均匀随机的。随后计算  $\frac{\int_X 1 dx}{|\hat{X}|} \sum_{\hat{x}, \vec{f}(\hat{x}) \in \mathcal{T}_{\hat{X}}} \sum_{i=1}^m |\mathcal{N}(\cdot, \theta)_{\hat{x}_i} - \vec{f}_{x_i}(\hat{x})|^p$  作为对积分  $\int_X \|\mathcal{N}(\cdot, \theta) - \vec{f}(\vec{x})\|_{l_p}^p d\vec{x}$  的数值估计; 在均匀随机采样的假设下可以看出估计的期望即为原积分的值。此时原优化问题变为如下 (2.11) 形式:

$$\min_{\theta \in \Theta} \left( \frac{\int_X 1 dx}{|\hat{X}|} \sum_{\hat{x}, \vec{f}(\hat{x}) \in \mathcal{T}_{\hat{X}}} \sum_{i=1}^m |\mathcal{N}(\cdot, \theta)_{\hat{x}_i} - \vec{f}_{x_i}(\hat{x})|^p \right)^{\frac{1}{p}}. \quad (2.11)$$

作为目标函数, 不妨假设  $\int_X 1 dx = 1$ , 记  $N_{\mathcal{T}_{\hat{X}}} = |\mathcal{T}_{\hat{X}}| = |\hat{X}|$ , 同时去掉最外层的  $(\cdot)^{\frac{1}{p}}$  以简化计算量。此时该问题的目标函数变为 (2.12):

$$\mathcal{L}(\theta; \mathcal{T}_{\hat{X}}) = \frac{1}{N_{\hat{X}}} \sum_{\hat{x}, \vec{f}(\hat{x}) \in \hat{X}} \sum_{i=1}^m |\mathcal{N}(\cdot, \theta)_{\hat{x}_i} - \vec{f}_{x_i}(\hat{x})|^p \quad (2.12)$$

该目标函数在神经网络的训练中被称为损失函数。

针对不同的问题，也有很多其他的衡量神经网络与待拟合函数间距离的方式，其对应的损失函数的形式也会有不同。确定了损失函数以后接下来就是对神经网络进行训练了。

### 2.2.3 神经网络的训练算法

神经网络训练算法的目的是得到优化问题 (2.10) 的近似解  $\theta^* = \arg \min_{\theta \in \Theta} \mathcal{L}(\theta; \mathcal{T}_{\hat{X}})$ ，虽然最优化理论提供了众多优化问题的求解算法，但是对于上述神经网络的训练问题，其训练集中样本的数量往往非常大，同时神经网络的参数量也不小，因此大部分涉及到二阶微分或者线搜索的优化算法在该问题上将会产生较大的计算开销。因此目前常用的求解算法来源于相对基础的梯度下降法，同时为了进一步减少计算开销，提出了算法1：小批量随机梯度下降法（minibatch gradient descent）来进行神经网络的训练，具体表示如下：

---

#### 算法 1：小批量随机梯度下降法（minibatch gradient descent）

---

**输入**：训练集  $\mathcal{T}_{\hat{X}}$ ，神经网络  $\mathcal{N}(\cdot, \theta)$  的结构，学习率  $lr$ （一般取 0.01），容许误差  $0 < \varepsilon \ll 1$ ，训练阈值参数  $n_T$ ，学习率调整误差阈值  $0 < \epsilon_{lr} \ll 1$ ，学习率调整耐心阈值  $n_{lr}$ ，学习率调整衰减率  $r_{lr}$ （一般取 0.5）

**输出**： $\hat{\theta}^*$  的近似解  $\theta_k$

- 1 对参数  $\theta$  进行初始化，如常值初始化，Xavier 初始化<sup>[5]</sup>、Kaiming 初始化<sup>[6]</sup>等，记为  $\theta_0$ ；令  $k = 0$ ；
- 2 **while**  $\|\mathbf{g}_k\| \leq \varepsilon$  不成立 **do**
- 3     **while**  $\mathcal{T}_{\hat{X}}$  还未被全部采样过一遍或者此循环次数  $\leq n_T$  **do**
- 4         从训练集中按一定规则进行一次小批量采样得到  $\hat{\mathcal{T}}_{\hat{X}_k} \subset \mathcal{T}_{\hat{X}}$ 。可以采用顺序采样，随机采样等方法；每次采样的样本数量取决于计算机的硬件能力；
- 5         将  $\theta_k$  代入基于  $\hat{\mathcal{T}}_{\hat{X}_k}$  的损失函数  $L(\theta_k; \hat{\mathcal{T}}_{\hat{X}_k})$  中并计算其输出关于  $\theta_k$  的梯度值  $\mathbf{g}_k = \nabla_{\theta} L(\theta_k; \hat{\mathcal{T}}_{\hat{X}_k})$ ；
- 6         令  $\theta_{k+1} = \theta_k - lr \cdot \mathbf{g}_k$ ；
- 7     **end** 学习率自适应调整策略：
- 8     **if** 损失函数  $L(\theta_k; \hat{\mathcal{T}}_{\hat{X}_k})$  的值下降幅度  $< \epsilon_{lr}$  的次数  $> n_{lr}$  **then**
- 9          $lr = lr \times r_{lr}$
- 10     **end**
- 11 **end**

---

目前在上述算法的基础上又衍生出一批效果更好的训练算法，如 Rmsprop 算法，Adam 优化器等，这些算法主要是对上述算法 1 中参数  $\theta_k$  的更新策略进行修改，比如考虑进多个历史梯度数据等，算法的其他部分不变，这里不再赘述。

## 2.3 Navier-Stokes/Darcy 耦合模型的神经网络求解方法

本节将介绍求解上述 Navier-Stokes/Darcy 耦合模型的双通道神经网络求解方法。求解的主要思想就是使用神经网络来拟合模型中偏微分方程的各个未知函数。针对模型的双区域结构，本文相应提出了双通道的神经网络结构，同时针对模型的方程形式提出相应的损失函数与求解算法。

### 2.3.1 网络结构

如图2.2 所示，整个结构由两个全连接神经网络组成： $\mathcal{N}_f(\cdot, \theta_f), \mathcal{N}_p(\cdot, \theta_p)$ ，分别对应拟合 Navier-Stokes/Darcy 模型中的自由流区域方程组与多孔介质流区域方程组中的未知函数；其中 Navier-Stokes 方程组中包含未知函数  $\vec{u}_f, p_f$ ，需要将它们的值域合并得到函数  $\vec{F}_f = (u_{fx_1}, \dots, u_{fx_d}, p_f)$  作为网络  $\mathcal{N}_f(\cdot, \theta_f)$  的待拟合函数，同理令  $\vec{F}_p = (u_{px_1}, \dots, u_{px_d}, \phi)$  作为  $\mathcal{N}_p(\cdot, \theta_p)$  的待拟合函数。因此  $\mathcal{N}_f(\cdot, \theta_f)$  的网络输入和输出层神经元个数均为  $d+1$ ，至于其隐藏层的具体设置（层数，激活函数，神经元个数等）在第三章数值实验中将分别说明。

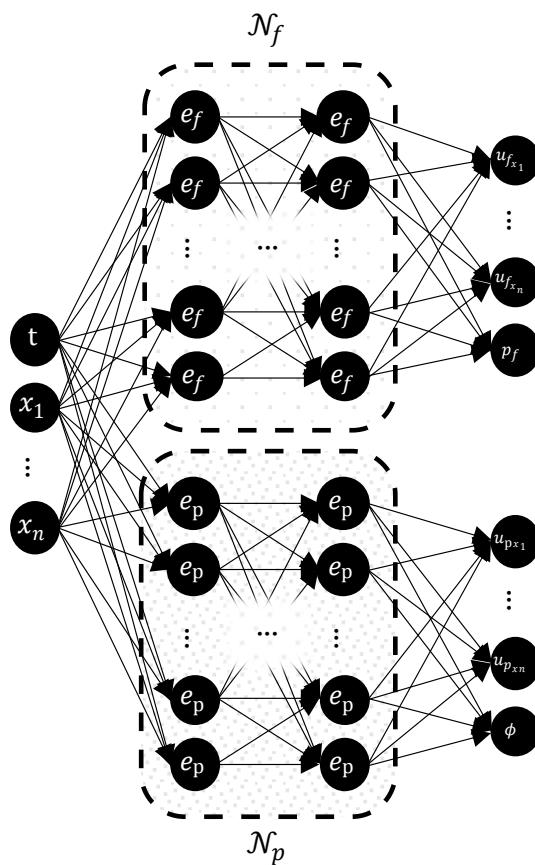


图 2.2: 网络结构示意图

### 2.3.2 损失函数

流体力学模型求解问题的已知条件中，关于待求解函数  $\vec{u}_f, p_f, \vec{u}_p, \phi$  本身的数据只有初边值条件，无法对全时空定义域进行均匀随机采样，因此无法完全使用传统神经网络损失函数的构造方法；另一方面，偏微分物理方程本身的约束是求解的重要条件，因此这里需要将物理方程编码进损失函数参与到神经网络的优化中去，做法如下：

设有抽象形式的物理方程  $\mathcal{P}(\vec{u}) = \vec{f}$  in  $\Omega$ ，其中  $\mathcal{P}$  是抽象的偏微分算子， $\vec{u} \in \mathcal{W}^{k,p}(\Omega; \mathbb{R}^m)$  代表待解函数， $\vec{f} \in L^p(\Omega; \mathbb{R}^m)$  代表源项， $\Omega \subset \mathbb{R}^n$  为方程定义区域。

这里  $\mathcal{W}^{k,p}(\Omega; \mathbb{R}^m) = \{\vec{u} \in L^p(\Omega; \mathbb{R}^m) \mid D^\alpha(\vec{u}) \in L^p(\Omega; \mathbb{R}), 0 \leq |\alpha| < \infty\}$  为 Sobolev 空间，其中  $\alpha = (\alpha_1, \dots, \alpha_n); \alpha_i \in \mathbb{N}^+ \cup \{0\}, i = 1, \dots, n$  叫做多重指标 (Multi-Index)， $|\alpha| = \sum_{\alpha_i \in \alpha} \alpha_i$ ， $D^\alpha = \frac{\partial^{\alpha_1 \dots \alpha_n}}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}}$  为相应的弱导数意义下的偏微分算子。

对应设有以  $\vec{u}$  作为待拟合函数的神经网络  $\mathcal{N}_{\vec{u}}(\cdot, \theta)$ ，与 2.2.2 类似，定义距离函数  $\mathcal{D}(\mathcal{P}, \vec{f}, \mathcal{N}_{\vec{u}}(\cdot, \theta)) = \|\mathcal{P}(\mathcal{N}_{\vec{u}}(\cdot, \theta)) - \vec{f}\|_{L^p(\Omega; \mathbb{R}^m)}$ ，只不过这里的距离衡量的是神经网络对物理方程的满足程度。同样需要进一步地利用数值估计将距离函数进行转换，具体的转换方法与 2.2.2 节所述相同，最后得到编码了物理方程的损失函数：

$$\mathcal{L}(\theta; \mathcal{T}_\Omega) = \frac{1}{N_\Omega} \sum_{\hat{x}, \vec{f}(\hat{x}) \in \Omega} \sum_{i=1}^m |\mathcal{P}(\mathcal{N}_{\vec{u}}(\cdot, \theta)))_{x_i}(\hat{x}) - f_{x_i}(\hat{x})|^p \quad (2.13)$$

根据上述的抽象做法，接下来针对本文耦合模型中各个物理方程来构造相应的损失函数。对于  $\mathcal{N}_f(\cdot, \theta_f), \mathcal{N}_p(\cdot, \theta_p)$ ，首先将它们对应物理方程中待解函数的部分记为：

$$\begin{aligned} (u_{f_{x_1}}^{\mathcal{N}}, \dots, u_{f_{x_d}}^{\mathcal{N}}, p_f^{\mathcal{N}}) &= (\vec{u}_f^{\mathcal{N}}; p_f^{\mathcal{N}}) = \mathcal{N}_f(\cdot, \theta_f) \\ (u_{p_{x_1}}^{\mathcal{N}}, \dots, u_{p_{x_d}}^{\mathcal{N}}, \phi^{\mathcal{N}}) &= (\vec{u}_p^{\mathcal{N}}; \phi^{\mathcal{N}}) = \mathcal{N}_p(\cdot, \theta_p) \end{aligned} \quad (2.14)$$

也就是说，一个多维输出神经网络也可以看作多个单输出网络的组合，只是这些网络是共享参数的。这样可以将拆分形式的 (2.14) 自然代入原物理方程约束中作为拟合约束的一部分，Navier-Stokes/Darcy 模型有三部分方程约束，分别按上述方法进行损失函数的构造可得：

$$\begin{aligned}
 \mathcal{L}_f(\theta_f; \mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}}) = & \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \sum_{i=1}^d |\rho \left( \frac{\partial u_{f_{x_i}}^{\mathcal{N}}}{\partial t} + (\vec{u}_f^{\mathcal{N}} \cdot \nabla) u_{f_{x_i}}^{\mathcal{N}} \right) \\
 & - (\nabla \cdot \mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}} \cdot))_{x_i}(\hat{x}) - \rho \vec{g}_{f_{x_i}}(\hat{x})|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\Omega_f \times T}} |(\nabla \cdot \vec{u}_f^{\mathcal{N}})(\hat{x}) - 0|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Gamma_f \times T}}} \sum_{\hat{x}, \vec{u}_f^{\Gamma_f}(\hat{x}) \in \mathcal{T}_{\Gamma_f \times T}} \sum_{i=1}^d |u_{f_{x_i}}^{\mathcal{N}}(\hat{x}) - u_{f_{x_i}}^{\Gamma_f}|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Omega_f \times \{t_0\}}}} \sum_{\hat{x}, \vec{u}_f^{t_0}(\hat{x}) \in \mathcal{T}_{\Omega_f \times \{t_0\}}} \sum_{i=1}^d |u_{f_{x_i}}^{\mathcal{N}}(\hat{x}) - u_{f_{x_i}}^{t_0}|^p
 \end{aligned} \tag{2.15}$$

$$\begin{aligned}
 \mathcal{L}_p(\theta_p; \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}}) = & \frac{1}{N_{\mathcal{T}_{\Omega_p \times T}}} \sum_{\hat{x}, f_p(\hat{x}) \in \mathcal{T}_{\Omega_p \times T}} \left| S_0 \frac{\partial \phi^{\mathcal{N}}(\hat{x})}{\partial t} + (\nabla \cdot \vec{u}_p^{\mathcal{N}})(\hat{x}) - f_p(\hat{x}) \right|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Omega_p \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\Omega_p \times T}} \sum_{i=1}^d |u_{p_{x_i}}^{\mathcal{N}}(\hat{x}) + (\mathbf{K} \nabla \phi^{\mathcal{N}})_{x_i}(\hat{x})|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Gamma_p \times T}}} \sum_{\hat{x}, \phi_{\Gamma_p}(\hat{x}) \in \mathcal{T}_{\Gamma_p \times T}} |\phi^{\mathcal{N}}(\hat{x}) - \phi_{\Gamma_p}(\hat{x})|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\Omega_p \times \{t_0\}}}} \sum_{\hat{x}, \phi^{t_0}(\hat{x}) \in \mathcal{T}_{\Omega_p \times \{t_0\}}} |\phi^{\mathcal{N}}(\hat{x}) - \phi^{t_0}(\hat{x})|^p
 \end{aligned} \tag{2.16}$$

$$\begin{aligned}
 & \mathcal{L}_{\tilde{\Gamma}}(\theta_f, \theta_p; \mathcal{T}_{\tilde{\Gamma} \times T}) \\
 = & \frac{1}{N_{\mathcal{T}_{\tilde{\Gamma} \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\tilde{\Gamma} \times T}} |(\vec{u}_f^{\mathcal{N}} \cdot \vec{n})(\hat{x}) - (\vec{u}_p^{\mathcal{N}} \cdot \vec{n})(\hat{x})|^p \\
 & + \frac{1}{N_{\mathcal{T}_{\tilde{\Gamma} \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\tilde{\Gamma} \times T}} |-(\mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}}) \vec{n}) \cdot \vec{n})(\hat{x}) - g \rho (\phi^{\mathcal{N}}(\hat{x}) - z)|^p \\
 & + \sum_{i=1}^{d-1} \frac{1}{N_{\mathcal{T}_{\tilde{\Gamma} \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\tilde{\Gamma} \times T}} \left| -(\mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}}) \vec{n}) \cdot \vec{\tau}_i)(\hat{x}) - \alpha \sqrt{\frac{g \rho \mu}{(\mathbf{K} \vec{\tau}_i) \cdot \vec{\tau}_i}} ((\vec{u}_f^{\mathcal{N}} - \vec{u}_p^{\mathcal{N}}) \cdot \vec{\tau}_i)(\hat{x}) \right|^p
 \end{aligned} \tag{2.17}$$

上述构造函数 (2.15) (2.16) (2.17) 之和即为最终构造的损失函数 (2.18) :

$$\begin{aligned}
 & \mathcal{L}_{total}(\theta_f, \theta_p; \mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}}, \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}}, \mathcal{T}_{\tilde{\Gamma} \times T}) \\
 = & \mathcal{L}_f(\theta_f; \mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}}) + \mathcal{L}_p(\theta_p; \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}}) \\
 & + \mathcal{L}_{\tilde{\Gamma}}(\theta_f, \theta_p; \mathcal{T}_{\tilde{\Gamma} \times T})
 \end{aligned} \tag{2.18}$$

### 2.3.3 求解算法

可以看出交界面条件所对应的损失函数同时受到两个通道神经网络参数的共同影响，这就使得训练算法能够使两神经网络进行共同训练。基于上述网络模型与损失函数的构造，结合神经网络的训练算法 1，本文给出算法 2——耦合物理信息的神经网络（Coupled-Physics-informed neural networks, cPINNs）求解算法：

---

#### 算法 2: cPINNs 算法

**输入**：耦合模型区域  $\Omega_f \times T, \Gamma_f \times T, \Omega_f \times \{t_0\}, \Omega_p \times T, \Gamma_p \times T, \Omega_p \times \{t_0\}, \tilde{\Gamma}$  及其对应的源项函数信息，双通道神经网络模型  $\mathcal{N}_f(\cdot, \theta_f), \mathcal{N}_p(\cdot, \theta_p)$  的隐藏层具体结构，学习率  $lr$ （一般取 0.01），容许误差  $0 < \varepsilon \ll 1$ ，Adam 优化算法参数

**输出**： $\theta_{f_k}, \theta_{p_k}$

- 1 分别对耦合模型区域  $\Omega_f \times T, \Gamma_f \times T, \Omega_f \times \{t_0\}, \Omega_p \times T, \Gamma_p \times T, \Omega_p \times \{t_0\}, \tilde{\Gamma}$  及其对应的源项函数进行均匀随机采样，得到训练集  
 $\mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}}, \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}}, \mathcal{T}_{\tilde{\Gamma}}$ ;
- 2 对参数进行随机初始化，得到  $\theta_{f_0}, \theta_{p_0}$ ；令  $k = 0$ ；
- 3 **while**  $\|\mathbf{g}_k\| \leq \varepsilon$  不成立 **do**
- 4     **while** 所有的训练集还未被全部采样过一遍 **do**
- 5         对各区域训练集按数据量等比例进行随机采样，得到  
 $\hat{\mathcal{T}}_{\Omega_f \times T}, \hat{\mathcal{T}}_{\Gamma_f \times T}, \hat{\mathcal{T}}_{\Omega_f \times \{t_0\}}, \hat{\mathcal{T}}_{\Omega_p \times T}, \hat{\mathcal{T}}_{\Gamma_p \times T}, \hat{\mathcal{T}}_{\Omega_p \times \{t_0\}}, \hat{\mathcal{T}}_{\tilde{\Gamma}}$ ;
- 6         将参数代入基于第三步采样训练集的损失函数  $\mathcal{L}(\theta_{f_k}, \theta_{p_k}; \hat{\mathcal{T}}_{\Omega_f \times T}, \dots, \hat{\mathcal{T}}_{\tilde{\Gamma}})$   
                中并计算其输出关于  $\theta_f, \theta_p$  的梯度值  
 $(\mathbf{g}_{f_k}, \mathbf{g}_{p_k}) = \nabla_{\theta} \nabla_{\theta} \mathcal{L}_{total}(\theta_f, \theta_p; \mathcal{T}_{\Omega_f \times T}, \dots, \mathcal{T}_{\tilde{\Gamma} \times T})$ ;
- 7         令  $\theta_{k+1} = \theta_k - lr \cdot \mathbf{g}_k$ ；
- 8         使用 Adam 优化算法，令  $\theta_{f_{k+1}}, \theta_{p_{k+1}} = \text{Adam}(\theta_{f_k}, \theta_{p_k}; \mathbf{g}_{f_k}, \mathbf{g}_{p_k})$ ；
- 9     **end**
- 10 **end**

---

上述算法流程如图 2.3 所示。

### 2.4 损失函数的合理性证明

与众多传统算法不同，神经网络模型最初是由于实验效果的突出而得到进一步的发展与研究的，并且近些年才应用在解偏微分方程领域上，而并非由理论推导而来，因此其可解释性一直是业界关注的要点。

上述求解算法的损失函数构造基于对真解的物理方程约束的逼近，而非直接与真解距离的刻画。利用万能表示定理虽然能直接说明网络对真解的拟合能力，但还是需要特别指出网络拟合物理方程本身的能力合理性，即存在上述并行结构的全连接神经网络

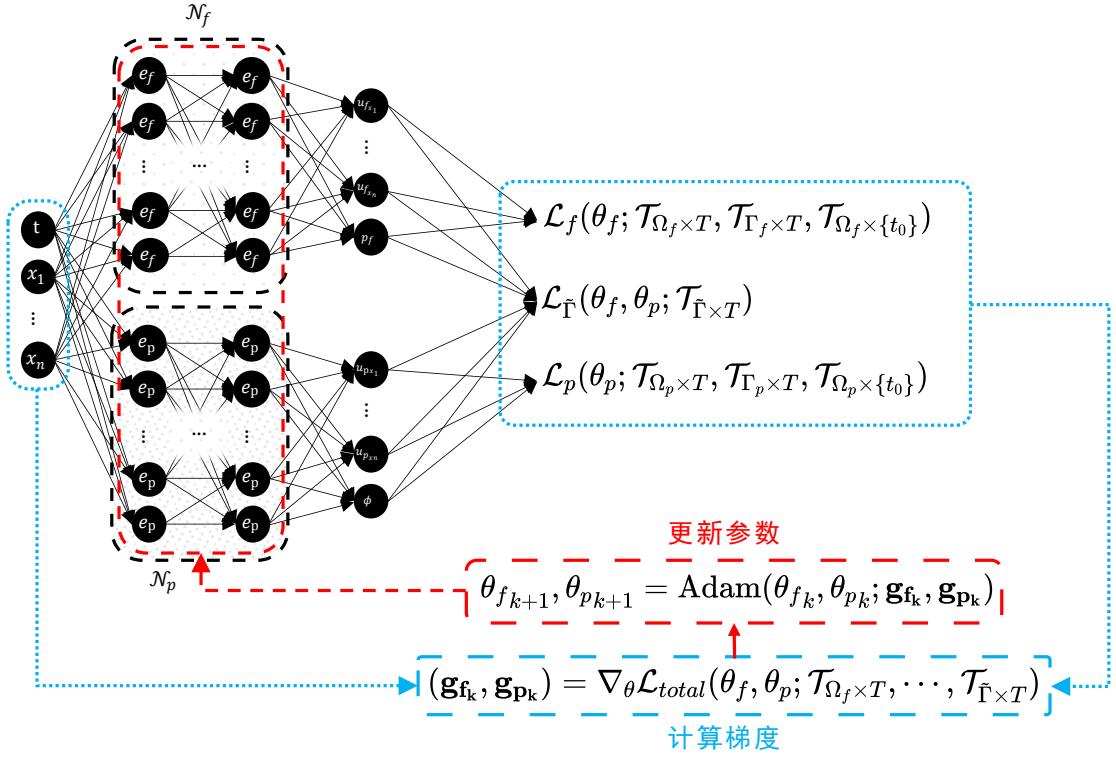


图 2.3: cPINNs 算法流程图示

定理的证明主要使用文献<sup>[2]</sup> 中定理 3 (及其讨论章节中的推论) 的结果, 这里重述如下:

**定理 2.1.** 令  $\mathfrak{N} = \{\mathcal{N}_{one}(\vec{x}, \theta) = \vec{w}_2 \cdot \sigma_1(\vec{x}\mathbf{W}_1 + \vec{b}_1) \mid \mathbf{W}_1 \in \mathbb{R}^{n \times m}; \vec{b}_1, \vec{w}_2 \in \mathbb{R}^m; m \in \mathbb{N}^+\}$ , 该集合为全体具有一层隐藏层且单个输出的全连接网络, 其激活函数  $\sigma_1$  为作用在每个分量上的非常值有界函数, 若  $\sigma_1 \in C^k(\mathbb{R}; \mathbb{R})$  是至少  $k$  次连续可微的,  $\Omega \in \mathbb{R}^n$  为有界区域且满足一种分段条件 (*The Segment Condition*)<sup>[7]</sup>, 则集合  $\mathfrak{N}$  在  $\mathcal{W}^{k,p}(\Omega; \mathbb{R})$  上稠密。

其中定理 2.1 中提到的分段条件定义如下:

**定义 2.1** (分段条件 (*The Segment Condition*)<sup>[7]</sup>). 若  $\forall x \in \partial\bar{\Omega} \subset \mathbb{R}^{d-1}$ ,  $\exists$  邻域  $U_x$ ,  $\exists 0 \neq \varepsilon_x \in \mathbb{R}^d$ , s.t.  $\forall y \in U_x \cap \bar{\Omega}$ ,  $\forall t \in (0, 1)$ ,  $y + t\varepsilon_x \in \Omega$ , 则称  $\Omega$  满足分段条件。

容易看出除非特殊的数学构造, 实际中的流体力学物理模型通常都会满足该条件。

此外, 还需要假设存在  $\vec{u}_f^*, \vec{u}_p^* \in \mathcal{W}^{2,p}(\Omega_{f/p} \times T; \mathbb{R}^{d+1})$ ;  $p_f^*, \phi^* \in \mathcal{W}^{2,p}(\Omega_{f/p} \times T; \mathbb{R})$  为耦合模型物理模型的真解。

**定理 2.2.** 在上述假设的基础上, 存在全连接神经网络  $\mathcal{N}_f^*(\cdot, \theta_f^*)$ ,  $\mathcal{N}_p^*(\cdot, \theta_p^*)$ , 使得其导出的对应函数满足本文耦合模型物理方程约束。

证明. 首先将 NS 方程组 (2.1) 中第一个公式按分量展开。首先注意到:

$$\begin{aligned}
 & \nabla \cdot \mathbf{T}(\vec{u}_f, p_f) \\
 &= \nabla \cdot (2\mu \mathbf{D}(\vec{u}_f) - p_f \mathbf{I}) \\
 &= \nabla \cdot (\mu(\nabla \vec{u}_f + \nabla \vec{u}_f^T) - p_f \mathbf{I}) \\
 &= \mu \nabla \cdot (\nabla \vec{u}_f + \nabla \vec{u}_f^T) - \nabla \cdot (p_f \mathbf{I}) \\
 &= \mu \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_d} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial u_{fx_1}}{\partial x_1} + \frac{\partial u_{fx_1}}{\partial x_1} & \cdots & \frac{\partial u_{fx_1}}{\partial x_d} + \frac{\partial u_{fx_d}}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_{fx_d}}{\partial x_1} + \frac{\partial u_{fx_1}}{\partial x_d} & \cdots & \frac{\partial u_{fx_d}}{\partial x_d} + \frac{\partial u_{fx_d}}{\partial x_d} \end{pmatrix} - \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_d} \end{pmatrix} \cdot \begin{pmatrix} p_f \\ \vdots \\ p_f \end{pmatrix} \\
 &= \mu \begin{pmatrix} \sum_{j=1}^d \left( \frac{\partial^2 u_{fx_j}}{\partial x_1 \partial x_j} + \frac{\partial^2 u_{fx_1}}{\partial x_j^2} \right) \\ \vdots \\ \sum_{j=1}^d \left( \frac{\partial^2 u_{fx_j}}{\partial x_d \partial x_j} + \frac{\partial^2 u_{fx_d}}{\partial x_j^2} \right) \end{pmatrix} - \begin{pmatrix} \frac{\partial p_f}{\partial x_1} \\ \vdots \\ \frac{\partial p_f}{\partial x_d} \end{pmatrix} \\
 &= \mu \begin{pmatrix} \frac{\partial}{\partial x_1} (\nabla \cdot \vec{u}_f) + \Delta u_{fx_1} \\ \vdots \\ \frac{\partial}{\partial x_d} (\nabla \cdot \vec{u}_f) + \Delta u_{fx_d} \end{pmatrix} - \nabla p_f \quad (\text{根据方程第二项: } \nabla \cdot \vec{u}_f = 0) \\
 &= \mu \Delta \vec{u}_f - \nabla p_f
 \end{aligned} \tag{2.19}$$

因此:

$$\begin{aligned}
 & \rho \left( \frac{\partial \vec{u}_f}{\partial t} + (\vec{u}_f \cdot \nabla) \vec{u}_f \right) - \nabla \cdot \mathbf{T}(\vec{u}_f, p_f) \\
 &= \rho \left( \frac{\partial \vec{u}_f}{\partial t} + (\vec{u}_f \cdot \nabla) \vec{u}_f \right) - \mu \Delta \vec{u}_f - \nabla p_f \\
 &= \rho \begin{pmatrix} \frac{\partial u_{fx_1}}{\partial t} \\ \vdots \\ \frac{\partial u_{fx_d}}{\partial t} \end{pmatrix} + \rho \begin{pmatrix} \sum_{j=1}^d u_{fx_j} \frac{\partial u_{fx_1}}{\partial x_j} \\ \vdots \\ \sum_{j=1}^d u_{fx_j} \frac{\partial u_{fx_d}}{\partial x_j} \end{pmatrix} - \mu \begin{pmatrix} \sum_{j=1}^d \frac{\partial^2 u_{fx_1}}{\partial x_j^2} \\ \vdots \\ \sum_{j=1}^d \frac{\partial^2 u_{fx_d}}{\partial x_j^2} \end{pmatrix} + \begin{pmatrix} \frac{\partial p_f}{\partial x_1} \\ \vdots \\ \frac{\partial p_f}{\partial x_d} \end{pmatrix} = \rho \begin{pmatrix} g_{fx_1} \\ \vdots \\ g_{fx_d} \end{pmatrix}
 \end{aligned} \tag{2.20}$$

根据假设 (真解存在), 记精确解为  $u_{fx_1}^*, \dots, u_{fx_d}^*, p_f^* \in \mathcal{W}^{k,p}(\Omega_f; \mathbb{R})$ , 再根据 Hornik<sup>[2]</sup> 定理 3 以及对区域的分段条件假设得, 对于  $\forall \varepsilon > 0$ , 存在神经网络  $\mathcal{N}_{u_{fx_1}^*}^\varepsilon, \dots, \mathcal{N}_{u_{fx_d}^*}^\varepsilon, \mathcal{N}_{p_f^*}^\varepsilon \in \mathfrak{N}$ , 使得如下成立:

$$\max_{0 \leq |\alpha| \leq 2} \sup_{(\vec{x}, t) \in \Omega_f \times T} |D_{\vec{x}, t}^\alpha u_{fx_i}^* - D_{\vec{x}, t}^\alpha \mathcal{N}_{u_{fx_i}^*}^\varepsilon| < \varepsilon, \quad i = 1, \dots, d \tag{2.21}$$

$$\max_{0 \leq |\alpha| \leq 1} \sup_{(\vec{x}, t) \in \Omega_f \times T} |D_{\vec{x}, t}^\alpha p_f^* - D_{\vec{x}, t}^\alpha \mathcal{N}_{p_f^*}^\varepsilon| < \varepsilon \tag{2.22}$$

因此,对于其中任一分量方程:

$$\rho \left( \frac{\partial u_{f_{x_i}}}{\partial t} + \sum_{j=1}^d u_{f_{x_j}} \frac{\partial u_{f_{x_i}}}{\partial x_j} \right) - \mu \sum_{j=1}^d \frac{\partial^2 u_{f_{x_i}}}{\partial x_j^2} + \frac{\partial p_f}{\partial x_i} = \rho g_{f_{x_i}} \quad (2.23)$$

将  $\mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon, \mathcal{N}_{p_f^*}^\varepsilon$  代入上述分量公式 (2.23) 对应的损失函数可得

$$\begin{aligned} & \mathcal{L}_{NS_1}(\theta_{u_{f_{x_i}}^*}^*, \theta_{p_f^*}^*; \mathcal{T}_{\Omega_f \times T}) \\ &= \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \left| \rho \left( \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial t} + \sum_{j=1}^d \mathcal{N}_{u_{f_{x_j}}^*}^\varepsilon \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j} \right) - \mu \sum_{j=1}^d \frac{\partial^2 \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j^2} + \frac{\partial \mathcal{N}_{p_f^*}^\varepsilon}{\partial x_i} - \rho g_{f_{x_i}} \right|^p \\ &= \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \left| \left( \rho \left( \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial t} + \sum_{j=1}^d \mathcal{N}_{u_{f_{x_j}}^*}^\varepsilon \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j} \right) - \mu \sum_{j=1}^d \frac{\partial^2 \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j^2} + \frac{\partial \mathcal{N}_{p_f^*}^\varepsilon}{\partial x_i} - \rho g_{f_{x_i}} \right) \right. \\ &\quad \left. - \underbrace{\left( \rho \left( \frac{\partial u_{f_{x_i}}^*}{\partial t} + \sum_{j=1}^d u_{f_{x_j}}^* \frac{\partial u_{f_{x_i}}^*}{\partial x_j} \right) - \mu \sum_{j=1}^d \frac{\partial^2 u_{f_{x_i}}^*}{\partial x_j^2} + \frac{\partial p_f^*}{\partial x_i} - \rho g_{f_{x_i}} \right)}_{=0} \right|^p \\ &\leq \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \left( \rho \left| \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial t} - \frac{\partial u_{f_{x_i}}^*}{\partial t} \right|^p + \rho \left| \sum_{j=1}^d \mathcal{N}_{u_{f_{x_j}}^*}^\varepsilon \frac{\partial \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j} - \sum_{j=1}^d u_{f_{x_j}}^* \frac{\partial u_{f_{x_i}}^*}{\partial x_j} \right|^p \right. \\ &\quad \left. + \mu \sum_{j=1}^d \left| \frac{\partial^2 \mathcal{N}_{u_{f_{x_i}}^*}^\varepsilon}{\partial x_j^2} - \frac{\partial^2 u_{f_{x_i}}^*}{\partial x_j^2} \right|^p + \left| \frac{\partial \mathcal{N}_{p_f^*}^\varepsilon}{\partial x_i} - \frac{\partial p_f^*}{\partial x_i} \right|^p \right) \quad (\cdot^p \text{ 的凸性, } (1 \leq p < \infty)) \\ &\leq \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \left( \rho \varepsilon^p + \rho d^p \left( \max_{j=1}^d \sup_{(\vec{x}, t) \in \Omega_f \times T} \left| u_{f_{x_j}}^* + \frac{\partial u_{f_{x_i}}^*}{\partial x_j} \right|^p \varepsilon + \varepsilon^2 \right)^p + \mu d \varepsilon^p + \varepsilon^p \right) \end{aligned} \quad (2.24)$$

(根据 (2.21), (2.22))

$$\leq C \varepsilon \quad (1 \leq p < \infty)$$

同理, 对其他 NS 方程组, Darcy 方程组以及初边值与交界面方程按分量展开, 同时对于精确解  $u_{p_{x_1}}^*, \dots, u_{p_{x_d}}^*, \phi^* \in \mathcal{W}^{k,p}(\Omega_f; \mathbb{R})$  也对应存在神经网络  $\mathcal{N}_{u_{p_{x_1}}^*}^\varepsilon, \dots, \mathcal{N}_{u_{p_{x_d}}^*}^\varepsilon, \mathcal{N}_{\phi^*}^\varepsilon \in \mathfrak{N}$  满足:

$$\max_{0 \leq |\alpha| \leq 2} \sup_{(\vec{x}, t) \in \Omega_p \times T} |D_{\vec{x}, t}^\alpha u_{p_{x_i}}^* - D_{\vec{x}, t}^\alpha \mathcal{N}_{u_{p_{x_i}}^*}^\varepsilon| < \varepsilon, \quad i = 1, \dots, d \quad (2.25)$$

$$\max_{0 \leq |\alpha| \leq 1} \sup_{(\vec{x}, t) \in \Omega_p \times T} |D_{\vec{x}, t}^\alpha \phi^* - D_{\vec{x}, t}^\alpha \mathcal{N}_{\phi^*}^\varepsilon| < \varepsilon \quad (2.26)$$

同理对于其他损失函数将其重写为分量形式:

$$\sum_{i=1}^d \frac{\partial u_{f_{x_i}}}{\partial x_i} = 0 \quad (2.27)$$

$$u_{f_{x_i}} = u_{f_{x_i}}^{\Gamma_f} \quad (2.28)$$

$$u_{f_{x_i}} = u_{f_{x_i}}^{t_0} \quad (2.29)$$

$$S_0 \frac{\partial \phi}{\partial t} + \left( \sum_{i=1}^d \frac{\partial \vec{u}_p}{\partial x_i} \right) = f_p \quad (2.30)$$

$$u_{p_{x_i}} = -(\kappa \frac{\partial \phi}{\partial x_i}) \quad (2.31)$$

$$\phi = \phi_{\Gamma_p} \quad (2.32)$$

$$\phi = \phi^{t_0} \quad (2.33)$$

按类似的方法容易得到相应神经网络对于这些方程组各分量方程的损失函数也满足类似的估计，最后将上述神经网络按输出维度进行拼接，得到  $\mathcal{N}_f^\varepsilon = (\mathcal{N}_{u_{f_{x_1}}^*}^\varepsilon, \dots, \mathcal{N}_{u_{f_{x_d}}^*}^\varepsilon, \mathcal{N}_{p_f^*}^\varepsilon); \mathcal{N}_p^\varepsilon = (\mathcal{N}_{u_{p_{x_1}}^*}^\varepsilon, \dots, \mathcal{N}_{u_{p_{x_d}}^*}^\varepsilon, \mathcal{N}_{\phi^*}^\varepsilon)$ ，此时总的损失函数即为这些分量方程对应损失函数的有限求和，于是得到：

$$\mathcal{L}_{total}(\theta_f^*, \theta_p^*; \mathcal{T}_{\Omega_f \times T}, \dots) \leq \tilde{C}\varepsilon \quad (2.34)$$

其中  $\tilde{C}$  是仅与物理方程中各参数  $\rho, \mu, S_0, g, \kappa, \alpha$  以及区域维度  $d$  有关的有界常数。因此，此损失函数构造是合理的。  $\square$

本页留白

## 第三章 正问题的数值实验

本章使用了四个数值实验来测试 cPINNs 算法的可行性。同时，本章利用第一个数值实验对网络结构的不同配置进行了一些探索，为接下来的实验设置提供参考；第二个数值实验中物理方程带有参数项  $\kappa$ ，以测试算法对不同参数物理方程模型的求解能力；后两个数值实验分别为稳态形式的耦合模型以及交界面为 BJS 形式的耦合模型正问题实验，本算法只需对损失函数中的时间项以及初值条件项进行轻微改动便可对两者进行迁移，进一步验证了 cPINNs 算法的通用性。

### 3.1 实验一：真解数值实验

本节使用一个具有解析解的数值算例进行网络结构探索实验。其中  $\Omega_f = (0, 1) \times (1, 2)$ ,  $\Omega_p = (0, 1) \times (0, 1)$ ,  $T = (0, 1)$ ,  $\tilde{\Gamma} = (0, 1) \times \{1\}$ ,  $\vec{n} = \overrightarrow{(0, -1)}^T$ ,  $\vec{\tau} = \overrightarrow{(0, 1)}^T$ ; 对应的解析解由公式 3.1 给出；算例的物理方程参数为:  $\rho = \mu = S_0 = g = \kappa = \alpha = 1$ ,  $\mathbf{K} = \frac{\rho g \kappa \mathbf{I}}{\mu} = \mathbf{I}$ , 另外 Navier-Stokes 方程的加速度场函数与 Darcy 方程组中的源项  $\vec{g}_f, f_p$  可由解析解计算导出。容易验证其满足 (2.1), (2.3) 与 (2.5) 条件:

$$\left\{ \begin{array}{ll} u_{f_{x_1}}^* = (x^2(y-1)^2 + y) \frac{\cos(t)}{3} & \text{in } \Omega_f \times T \\ u_{f_{x_2}}^* = \left(-\frac{2}{3}x(y-1)^3 + 2 - \pi \sin(\pi x)\right) \frac{\cos(t)}{3} & \text{in } \Omega_f \times T \\ p_f^* = (2 - \pi \sin(\pi x)) \sin\left(\pi \frac{y}{2}\right) \frac{\cos(t)}{3} & \text{in } \Omega_f \times T \\ u_{p_{x_1}}^* = \pi^2(-y - \cos(\pi y) + 1) \frac{\cos(t)}{3} \cos(\pi x) & \text{in } \Omega_p \times T \\ u_{p_{x_2}}^* = (\pi \sin(\pi x) - 2)(\pi \sin(\pi y) - 1) \frac{\cos(t)}{3} & \text{in } \Omega_p \times T \\ \phi^* = (2 - \pi \sin(\pi x))(1 - y - \cos(\pi y)) \frac{\cos(t)}{3} & \text{in } \Omega_p \times T \end{array} \right. \quad (3.1)$$

网络结构采用如下设置:  $\mathcal{N}_f, \mathcal{N}_p$  均采用全连接神经网络，各隐藏层网络的神经元个数均为 32，激活函数将分别测试标准逻辑斯谛函数 (Standard Logistic Function)  $\text{logistic}(x) = (1 + e^{-x})^{-1}$ , 双曲正切函数 (Hyperbolic Tangent Function)  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$  以及工程上常用的整流线性单位函数 (Rectified Linear Unit, ReLU)  $\text{ReLU}(x) = \max(0, x)$ ; 网络的隐藏层数分别使用 1, 2, 4, 6, 8 进行测试。

训练策略上，容许误差  $\varepsilon = 1 \times 10^{-10}$ , 训练阈值参数  $n_T = 30$ ; 初始学习率设置为  $1 \times 10^{-3}$ , 自适应调整策略中学习率调整误差阈值  $\epsilon_{lr} = 1 \times 10^{-8}$ , 学习

率调整耐心阈值  $n_{lr} = 10$ , 学习率调整衰减率  $r_{lr} = 0.5$ , 采取 Adam 参数更新策略。神经网络参数的初始化方法采用 Kaiming 初始化。

各训练集数据的采集策略为: 对耦合模型的各区域  $\Omega_f \times T, \Gamma_f \times T, \Omega_f \times \{t_0\}, \Omega_p \times T, \Gamma_p \times T, \Omega_p \times \{t_0\}, \tilde{\Gamma} \times T$  进行均匀随机采样生成对应训练集, 采样的数据点比例为  $50 : 4 : 1 : 50 : 4 : 1 : 5$ , 共采集  $3 \times 10^5$  个样本数据点。

表展示了不同的网络结构配置下的实验结果, 其中相对误差由 (3.2) 定义, 注意其数值计算类似于损失函数同样是按照均匀采样离散化后再计算的。

$$\begin{aligned}\text{error}_u^{rel} &= \frac{\sqrt{\|\vec{u}_f^N - \vec{u}_f^*\|_{L^2}^2 + \|\vec{u}_p^N - \vec{u}_p^*\|_{L^2}^2}}{\sqrt{\|\vec{u}_f^*\|_{L^2}^2 + \|\vec{u}_p^*\|_{L^2}^2}} \\ \text{error}_p^{rel} &= \frac{\sqrt{\|p_f^N - p_f^*\|_{L^2}^2 + \|\phi^N - \phi^*\|_{L^2}^2}}{\sqrt{\|p_f^*\|_{L^2}^2 + \|\phi^*\|_{L^2}^2}}\end{aligned}\quad (3.2)$$

表 3.1: 实验一结果

网络层数	激活函数	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
1	ReLU	$7.2315 \times 10^1$	$1.0311 \times 10^2$
1	logistic	$5.5137 \times 10^1$	$9.2287 \times 10^1$
1	tanh	$5.4759 \times 10^1$	$9.0141 \times 10^1$
2	ReLU	$6.2912 \times 10^{-1}$	$2.9390 \times 10^0$
2	logistic	$3.3533 \times 10^{-1}$	$8.3449 \times 10^{-1}$
2	tanh	$2.3089 \times 10^{-1}$	$6.1142 \times 10^{-1}$
4	ReLU	$5.7582 \times 10^{-1}$	$1.2959 \times 10^0$
4	logistic	$2.8264 \times 10^{-2}$	$1.0619 \times 10^{-1}$
4	tanh	$2.7503 \times 10^{-2}$	$9.8046 \times 10^{-2}$
6	ReLU	$4.9802 \times 10^{-1}$	$7.4808 \times 10^{-1}$
6	logistic	$3.0148 \times 10^{-2}$	$1.0793 \times 10^{-1}$
6	tanh	$2.7495 \times 10^{-2}$	$1.0071 \times 10^{-1}$
8	ReLU	$5.1132 \times 10^{-1}$	$9.7828 \times 10^{-1}$
8	logistic	$2.7799 \times 10^{-2}$	$9.9716 \times 10^{-2}$
8	tanh	$2.6945 \times 10^{-2}$	$9.6993 \times 10^{-2}$

由此可推测当每层神经元数量有限时, 若网络层数太少, 由于参数量不够会导致拟合不成功; 另一方面拟合效果不会随着层数的增加而无限制提高下去。虽然激活函数 ReLU 在深度学习实践中很常见, 但在此问题中效果不如 logistic

与 tanh; logistic, tanh 的拟合效果接近，但是在实验中观察到使用 tanh 时训练网络的收敛速度快于 logistic，如图 3.1 所示，推测是前者的梯度更加陡峭所致。

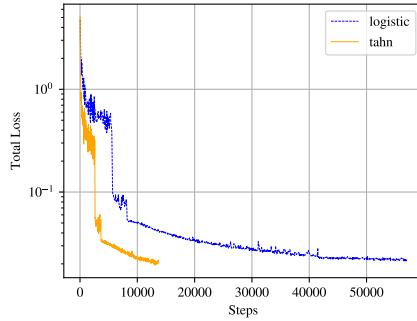


图 3.1: logistic 与 tanh 实验效果的比较

以网络隐藏层数设为 4，激活函数选为 tanh 时的拟合实验为例，此时网络已经基本能够拟合出真解，数值真解与网络拟合结果对比如图 3.2, 3.3 所示。此时再增加网络层数，最终的拟合效果提升有限，出于平衡拟合效果与网络计算量的考虑，选取这一配置作为接下来的实验配置。

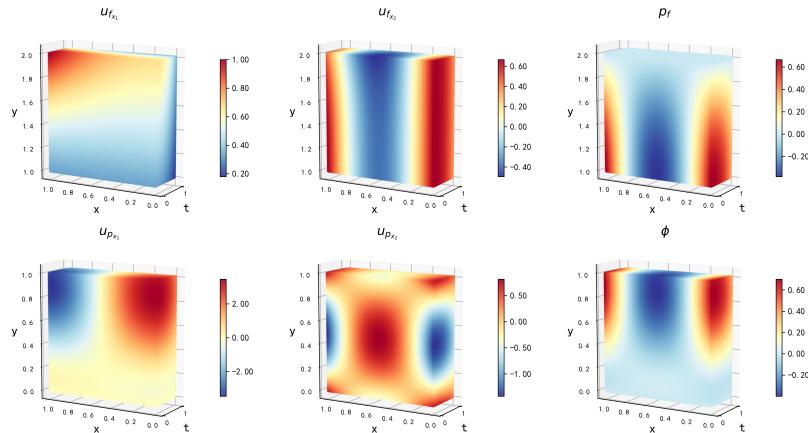


图 3.2: 实验一的数值真解

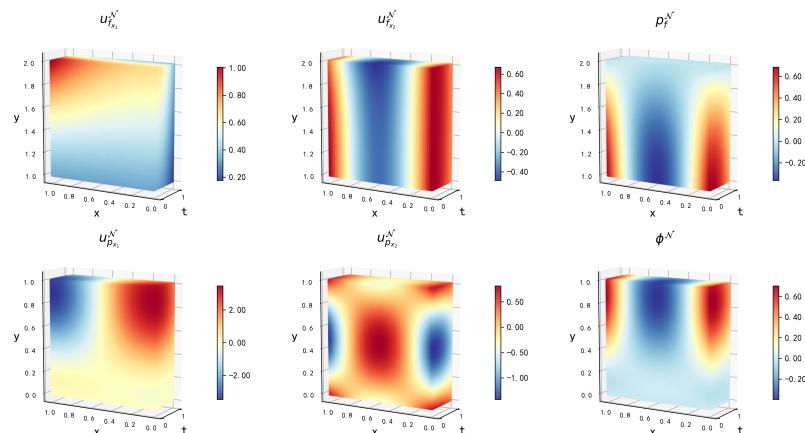


图 3.3: 实验一的神经网络拟合结果

### 3.2 实验二：带参数的数值实验

本节使用另一个带有参数  $\kappa$  的解析解测试本算法神经网络的拟合能力。这里参数  $\kappa$  同时也影响了物理方程参数中的  $\mathbf{K} = \frac{\rho g \kappa \mathbf{I}}{\mu}$ , 在下述实验中将取 1.0, 0.10, 0.010,  $\vec{g}_f, f_p$  同样由真解计算得到, 其余方程参数以及各物理方程区域设置与实验一相同。真解公式列出如下:

$$\begin{cases} u_{f_{x_1}}^* = (1 - 2x)(y - 1) \cos(t) & \text{in } \Omega_f \times T \\ u_{f_{x_2}}^* = (x(x - 1) + (y - 1)^2) \cos(t) & \text{in } \Omega_f \times T \\ p_f^* = \frac{1}{\kappa} (x(1 - x)(y - 1) + \frac{y^3}{3} - y^2 + y) \cos(t) & \text{in } \Omega_f \times T \\ u_{p_{x_1}}^* = -(-x(y - 1) + (1 - x)(y - 1)) \cos(t) & \text{in } \Omega_p \times T \\ u_{p_{x_2}}^* = -(x(1 - x) + y^2 - 2y + 1) \cos(t) & \text{in } \Omega_p \times T \\ \phi^* = \frac{1}{\kappa} (x(1 - x)(y - 1) + \frac{y^3}{3} - y^2 + y) \cos(t) & \text{in } \Omega_p \times T \end{cases} \quad (3.3)$$

网络结构采取实验一中隐藏层数为 4, 激活函数为  $\tanh$  的设置, 此外本实验的训练策略, 各训练集的采集策略均与实验一相同。在实验中发现当  $\kappa = 0.10, 0.010$  时由于压力数值解的范围的扩大会导致网络训练的收敛速度受到影响, 为此在训练中针对网络输出中代表两区域压力值的分量相应地乘上了调整系数 10.0, 100.0 以帮助收敛。最后实验结果由 3.2 给出:

表 3.2: 实验二结果

$\kappa$	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
1.0	$4.1811 \times 10^{-3}$	$1.0159 \times 10^{-2}$
$1 \times 10^{-1}$	$5.2204 \times 10^{-3}$	$1.5440 \times 10^{-2}$
$1 \times 10^{-2}$	$8.7052 \times 10^{-3}$	$3.9801 \times 10^{-2}$
$1 \times 10^{-3}$	$9.9511 \times 10^{-3}$	$4.3519 \times 10^{-2}$
$1 \times 10^{-6}$	$7.5172 \times 10^{-2}$	$6.1193 \times 10^{-2}$

以参数  $\kappa = 0.1$  为例, 其真解与拟合结果如图 3.4, 3.5 所示,

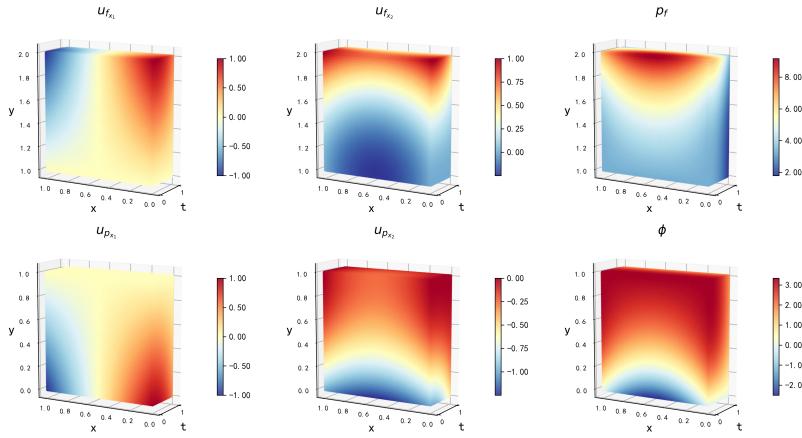


图 3.4: 实验二的数值真解

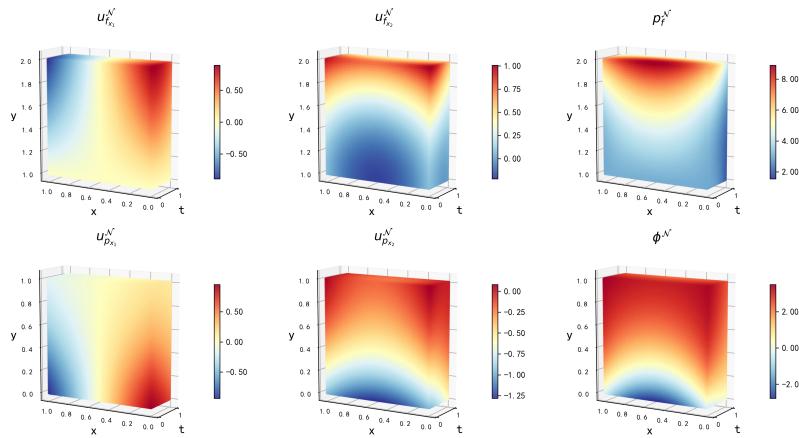


图 3.5: 实验二的神经网络拟合结果

### 3.3 实验三：稳态实验

本节针对稳态形式的 Navier-Stokes/Darcy 耦合模型进行求解。所谓稳态形式，即去掉了物理方程中的时间项，同时需要去掉各物理区域的初值条件。此时物理模型描述的是一类状态稳定的流体运动，该耦合模型方程如 (3.4) 所示：

$$\left\{ \begin{array}{ll} \rho(\vec{u}_f \cdot \nabla) \vec{u}_f - \nabla \cdot \mathbf{T}(\vec{u}_f, p_f) = \rho \vec{g}_f & \text{in } \Omega_f \\ \nabla \cdot \vec{u}_f = 0 & \text{in } \Omega_f \\ \vec{u}_f = \vec{u}_f^{\Gamma_f} & \text{on } \Gamma_f, \\ \nabla \cdot \vec{u}_p = f_p & \text{in } \Omega_p, \\ \vec{u}_p = -\mathbf{K} \nabla \phi & \text{in } \Omega_p. \\ \phi = \phi_{\Gamma_p} & \text{on } \Gamma_p \end{array} \right. \quad (3.4)$$

注意稳态耦合模型的交界面条件保持不变，因其本就没有时间项。

第四章中两个实验的解析解都可以很容易修改为其对应的稳态形式，方程

如 (3.5), (3.6) 所示, 其空间区域以及对应物理方程各参数同实验一。

$$\left\{ \begin{array}{ll} u_{f_{x_1}}^* = \frac{1}{3}(x^2(y-1)^2 + y) & \text{in } \Omega_f \times T \\ u_{f_{x_2}}^* = \frac{1}{3}(-\frac{2}{3}x(y-1)^3 + 2 - \pi \sin(\pi x)) & \text{in } \Omega_f \times T \\ p_f^* = \frac{1}{3}(2 - \pi \sin(\pi x)) \sin(\pi \frac{y}{2}) & \text{in } \Omega_f \times T \\ u_{p_{x_1}}^* = \frac{\pi^2}{3}(-y - \cos(\pi y) + 1) \cos(\pi x) & \text{in } \Omega_p \times T \\ u_{p_{x_2}}^* = \frac{1}{3}(\pi \sin(\pi x) - 2)(\pi \sin(\pi y) - 1) & \text{in } \Omega_p \times T \\ \phi^* = \frac{1}{3}(2 - \pi \sin(\pi x))(1 - y - \cos(\pi y)) & \text{in } \Omega_p \times T \end{array} \right. \quad (3.5)$$

$$\left\{ \begin{array}{ll} u_{f_{x_1}}^* = (1 - 2x)(y - 1) & \text{in } \Omega_f \times T \\ u_{f_{x_2}}^* = (x(x - 1) + (y - 1)^2) & \text{in } \Omega_f \times T \\ p_f^* = \frac{1}{\kappa}(x(1 - x)(y - 1) + \frac{y^3}{3} - y^2 + y) & \text{in } \Omega_f \times T \\ u_{p_{x_1}}^* = -(-x(y - 1) + (1 - x)(y - 1)) & \text{in } \Omega_p \times T \\ u_{p_{x_2}}^* = -(x(1 - x) + y^2 - 2y + 1) & \text{in } \Omega_p \times T \\ \phi^* = \frac{1}{\kappa}(x(1 - x)(y - 1) + \frac{y^3}{3} - y^2 + y) & \text{in } \Omega_p \times T \end{array} \right. \quad (3.6)$$

容易验证两个解析解均满足上述公式 (3.4) 以及原交界面条件 (2.5)。

要使本文提出的神经网络求解算法拟合上述真解, 只需相应构造出方程 (3.4) 对应的损失函数进行替换, 构造方法与第二章所描述的完全相同, 同时删除初值条件的损失即可, 最后得到公式 (3.7) (3.8):

$$\begin{aligned} & \mathcal{L}_f(\theta_f; \mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}}) \\ &= \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x}, \vec{g}_f(\hat{x}) \in \mathcal{T}_{\Omega_f \times T}} \sum_{i=1}^d \left| \rho(\vec{u}_f^N \cdot \nabla) u_{f_{x_i}}^N - (\nabla \cdot \mathbf{T}(\vec{u}_f^N, p_f^N \cdot))_{x_i}(\hat{x}) - \rho \vec{g}_{f_{x_i}}(\hat{x}) \right|^p \\ &+ \frac{1}{N_{\mathcal{T}_{\Omega_f \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\Omega_f \times T}} |(\nabla \cdot \vec{u}_f^N)(\hat{x}) - 0|^p \\ &+ \frac{1}{N_{\mathcal{T}_{\Gamma_f \times T}}} \sum_{\hat{x}, \vec{u}_f^{\Gamma_f}(\hat{x}) \in \mathcal{T}_{\Gamma_f \times T}} \sum_{i=1}^d \left| u_{f_{x_i}}^N(\hat{x}) - u_{f_{x_i}}^{\Gamma_f}(\hat{x}) \right|^p \end{aligned} \quad (3.7)$$

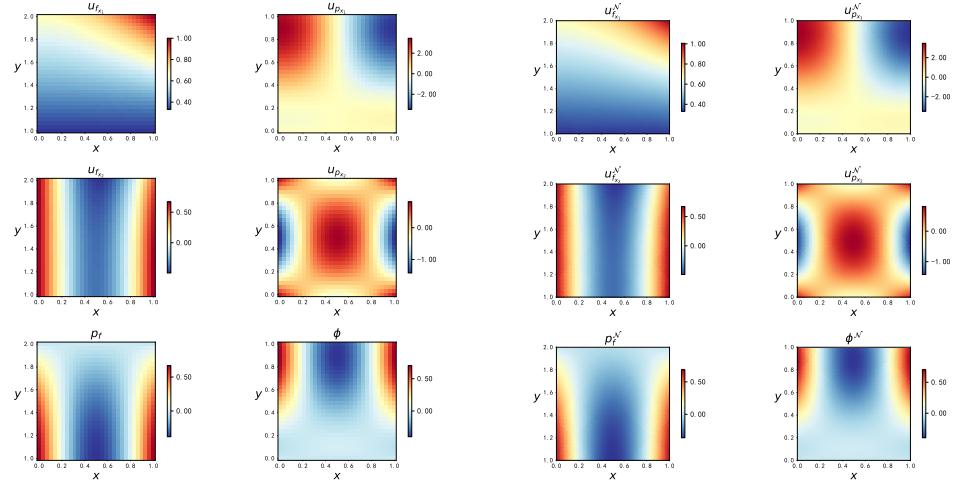
$$\begin{aligned}
 & \mathcal{L}_p(\theta_p; \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}}) \\
 &= \frac{1}{N_{\mathcal{T}_{\Omega_p \times T}}} \sum_{\hat{x}, f_p(\hat{x}) \in \mathcal{T}_{\Omega_p \times T}} |(\nabla \cdot \vec{u}_p^{\mathcal{N}})(\hat{x}) - f_p(\hat{x})|^p \\
 &\quad + \frac{1}{N_{\mathcal{T}_{\Omega_p \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\Omega_p \times T}} \sum_{i=1}^d \left| u_{p_{x_i}}^{\mathcal{N}}(\hat{x}) + (\mathbf{K} \nabla \phi^{\mathcal{N}})_{x_i}(\hat{x}) \right|^p \\
 &\quad + \frac{1}{N_{\mathcal{T}_{\Gamma_p \times T}}} \sum_{\hat{x}, \phi_{\Gamma_p}(\hat{x}) \in \mathcal{T}_{\Gamma_p \times T}} |\phi^{\mathcal{N}}(\hat{x}) - \phi_{\Gamma_p}(\hat{x})|^p
 \end{aligned} \tag{3.8}$$

实验的网络结构、训练策略以及各训练集的采集策略均与实验二相同（数据采集比例相对不变），实验结果见表 3.3。

表 3.3: 稳态实验结果

真解公式	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
公式 (3.5)	$2.7499 \times 10^{-2}$	$6.9892 \times 10^{-2}$
公式 (3.6), $\kappa = 1.0$	$7.0199 \times 10^{-3}$	$2.8815 \times 10^{-2}$
公式 (3.6), $\kappa = 1 \times 10^{-1}$	$7.0020 \times 10^{-3}$	$2.5232 \times 10^{-2}$
公式 (3.6), $\kappa = 1 \times 10^{-2}$	$9.6692 \times 10^{-3}$	$4.7237 \times 10^{-2}$
公式 (3.6), $\kappa = 1 \times 10^{-3}$	$1.1323 \times 10^{-2}$	$7.5991 \times 10^{-2}$

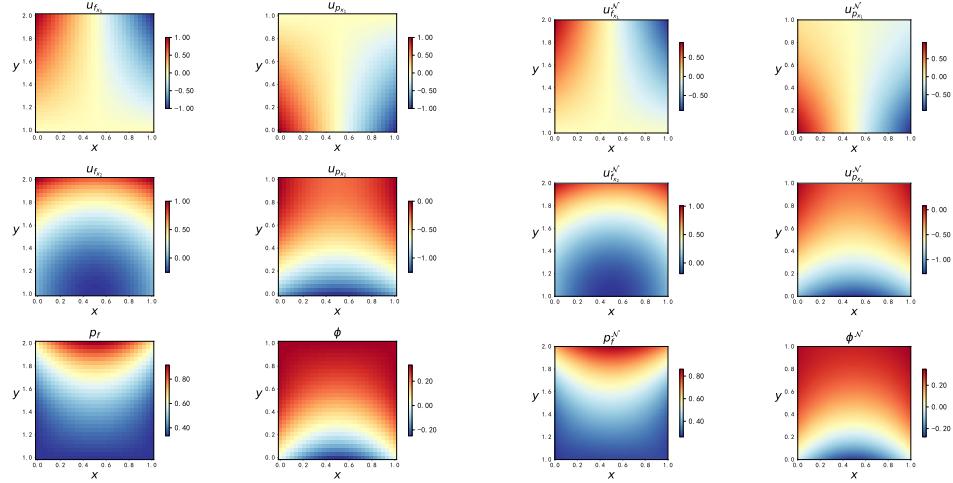
图 3.6 3.7 展示了部分真解与拟合结果的直观对比情况。虽然稳态实验中网络的训练无需考虑时间项，但同时已知条件也缺少了初值信息，因此网络拟合难度也不亚于非稳态实验。从结果可以看出算法求解依然是成功的。



(a) 数值真解

(b) 神经网络拟合结果

图 3.6: 公式 (3.5) 的真解与拟合结果对比图



(a) 数值真解

(b) 神经网络拟合结果

图 3.7: 公式 (3.6) 的真解与拟合结果对比图

### 3.4 实验四：BJS 交界面实验

关于 Navier-Stokes/Darcy 耦合模型的 BJ 交界面条件，存在一种公认的简化形式，在实践中观察到多孔介质流区域中  $\vec{u}_p \cdot \tau_i$  这一项的大小要远远小于其他项，因此是可以被忽略的。这就形成了著名的 BJS (Beavers-Joseph-Saffman) 交界面条件<sup>[8]</sup>:

$$-(\mathbf{T}(\vec{u}_f, p_f) \vec{n}) \cdot \vec{\tau}_i = \alpha \sqrt{\frac{g\rho\mu}{(\mathbf{K}\vec{\tau}_i) \cdot \vec{\tau}_i}} \vec{u}_f \cdot \vec{\tau}_i, i = 1, \dots, (d-1) \quad \text{on } \tilde{\Gamma} \quad (3.9)$$

本算例将对采用如上交界面条件的耦合模型进行求解，其对应的解析解如

下, 容易验证其满足公式 (2.1), (2.3) 与 (3.9):

$$\begin{cases} u_{f_{x_1}}^* = -\sin(\pi y) \cos(t) \cos(\pi x) & \text{in } \Omega_f \times T \\ u_{f_{x_2}}^* = \sin(\pi x) \cos(t) \cos(\pi y) & \text{in } \Omega_f \times T \\ p_f^* = \sin(\pi x) \cos(t) & \text{in } \Omega_f \times T \\ u_{p_{x_1}}^* = -\pi y \cos(t) \cos(\pi x) & \text{in } \Omega_p \times T \\ u_{p_{x_2}}^* = -\sin(\pi x) \cos(t) & \text{in } \Omega_p \times T \\ \phi^* = y \sin(\pi x) \cos(t) & \text{in } \Omega_p \times T \end{cases} \quad (3.10)$$

类似于上一节的稳态实验, 将 cPINNs 迁移到此耦合模型只需要将求解算法中 BJ 交界面条件对应的损失函数替换为 BJS 交界面条件即可, 这里 BJS 交界面条件的损失函数如下:

$$\begin{aligned} & \mathcal{L}_{\tilde{\Gamma}}(\theta_f, \theta_p; \mathcal{T}_{\tilde{\Gamma} \times T})_3 \\ &= \sum_{i=1}^{d-1} \frac{1}{N_{\mathcal{T}_{\tilde{\Gamma} \times T}}} \sum_{\hat{x} \in \mathcal{T}_{\tilde{\Gamma} \times T}} \left| -((\mathbf{T}(\vec{u}_f^N, p_f^N) \vec{n}) \cdot \vec{\tau}_i)(\hat{x}) \right. \\ & \quad \left. - \alpha \sqrt{\frac{g\rho\mu}{(\mathbf{K}\vec{\tau}_i) \cdot \vec{\tau}_i}} (\vec{u}_f^N \cdot \vec{\tau}_i)(\hat{x}) \right|^p \end{aligned} \quad (3.11)$$

其余的网络结构等配置也与实验三相同, 实验结果如 3.4 所示, 算法依然能够拟合出数值解:

表 3.4: 稳态实验结果

真解公式	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
公式 (3.10)	$4.3153 \times 10^{-3}$	$8.7145 \times 10^{-3}$

真解与拟合结果对比图如 3.8, 3.9 所示, 可以看出各物理量的数值分布几乎相同:

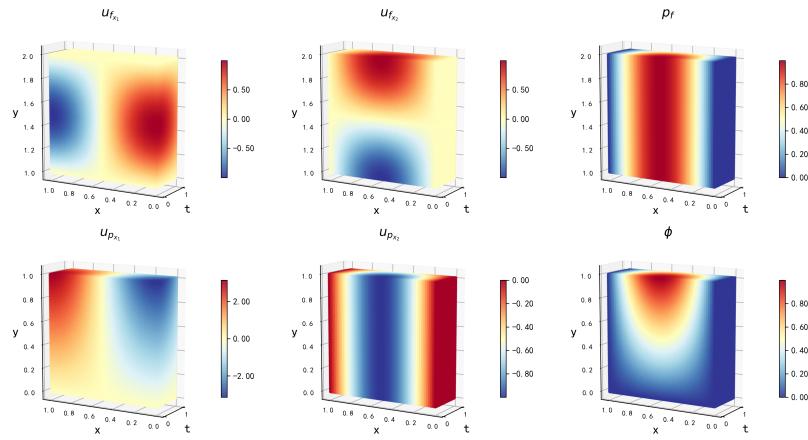


图 3.8: BJS 实验公式 (3.10) 的数值真解

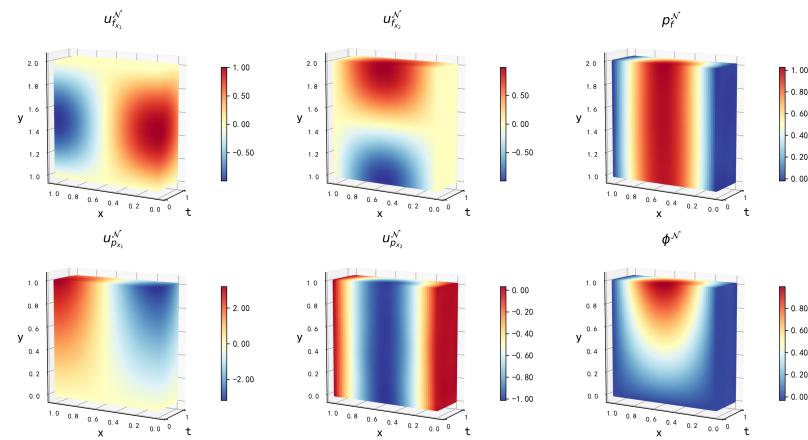


图 3.9: BJS 实验公式 (3.10) 的神经网络拟合结果

以上实验结果说明网络对不同的数值算例均有一定的拟合能力。

## 第四章 Navier-Stokes/Darcy 耦合模型的反问题实验

传统的数值求解方法如有限元、格子波尔兹曼方法等，当流体力学物理模型的方程发生改变时，这些方法往往要进行较大幅度的改动或重新设计，而本文提出的神经网络求解算法只需要对损失函数做出细微的调整便可以在不同方程中进行迁移。

另一方面，流体力学领域还存在各种各样的反问题，比如物理模型的初边值条件是未知的，取而代之的是已知内部部分区域或部分物理量的数值真解，以此反推整个区域的流体运动情况；或者，物理模型的方程本身具有一些未知参数，需要通过真实的数值结果进行反推。这类问题在工程应用中具有很大意义，然而各种传统方法对此类问题的求解具有一定的难度，在本文神经网络求解的框架下，却很容易对该类反问题尝试进行求解。

因此，本章设计了三个数值实验，分别为区域反问题实验、物理量反问题实验以及方程参数反问题实验，通过这些实验证明 cPINNs 对各种类型反问题的求解潜力。

### 4.1 区域反问题实验

本节将计算一个区域反问题的算例，即求解时不提供初边值条件，而是选择空间区域中的一部分真解作为已知条件。该问题对应的神经网络求解算法的修改为将初边值区域的损失函数替换为自定义选择区域的物理量损失即可。算例将采取实验一 (3.1) 的解析解，而自定义区域设置为  $\Omega_{id} = \{(x, y) \mid x \in [0.15, 0.25] \cup [0.45, 0.55] \cup [0.75, 0.85]; y \in [0.15, 0.25] \cup [0.45, 0.55] \cup [0.75, 0.85] \cup [1.15, 1.25] \cup [1.45, 1.55] \cup [1.75, 1.85]\}\}$ ，这是一种类似网格的子区域，如图 4.1 中实心部分所示：

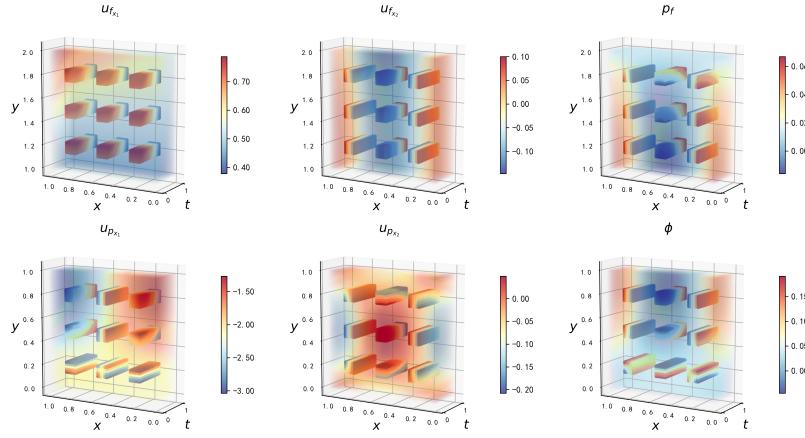


图 4.1: 反问题的区域设置

而相应地两区域对应的损失函数删掉了模型的初边值条件，添加上  $\Omega_{id}$  对应的真值条件损失。网络结构等训练配置同上节保持不变，实验结果如表 4.1 所示：

表 4.1: 区域反问题实验结果

真解公式	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
公式 (3.1)	$2.7232 \times 10^{-2}$	$6.0616 \times 10^{-2}$

拟合结果如图 4.2 所示：

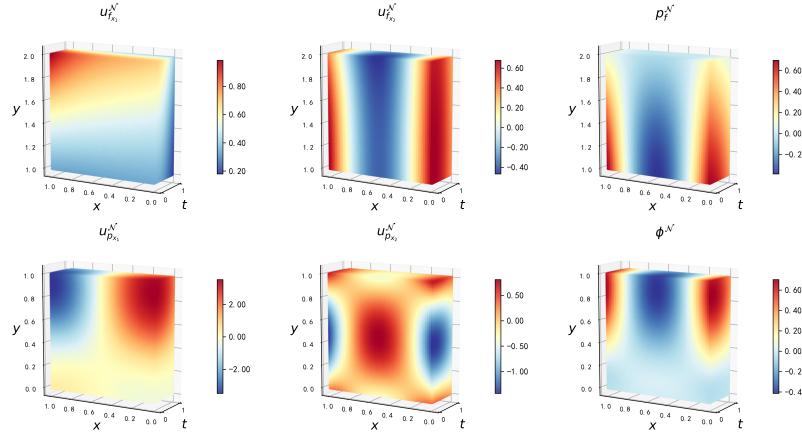


图 4.2: 区域反问题实验结果图示

可以看出对于上述区域设置，网络还是能够较好的还原出各物理量真解的。值得一提的是，在实验中发现并非任意的自定义区域均能使算法成功求解反问题，比如当区域选择过小，或者位置过于特殊时，算法虽然能够收敛，但网络拟合出的解析解将与真解差距过大，这提示该类反问题并非总是适定的（解的存在可能不唯一）。

## 4.2 物理量反问题实验

本节针对物理量反问题进行数值实验。所谓物理量反问题就是在不提供初边值条件的前提下，只提供部分物理量的全部信息来推断其他物理量的数值解。如隐藏型流体力学算法 (Hidden fluid mechanics, HFM)<sup>[9]</sup> 就是只提供输运方程中给定物的物理量信息来反推出流场中的速度与压力场。由于本文耦合模型不涉及到输运问题，此物理量反问题实验将利用实验一 (3.1) 的真解，在求解上只提供两区域中代表压力的  $p_f^*, \phi^*$  作为已知条件，目标是反推出模型中的速度场数值解。

类似区域反问题实验，cPINNs 迁移到此问题同样需要去除损失函数中的初边值部分，另外加入两区域压力物理量的损失部分。训练配置与上一个实验保持相同。实验结果见表 4.2：

表 4.2: 物理量反问题实验结果

真解公式	$\text{error}_u^{rel}$	$\text{error}_p^{rel}$
公式 (3.1)	$8.5319 \times 10^{-2}$	$2.1002 \times 10^{-1}$

同样地，拟合出的数值结果可视化展示如图 4.3：

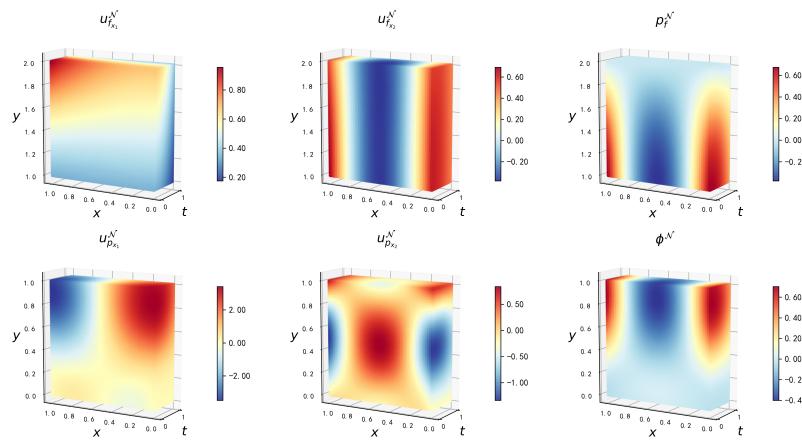


图 4.3: 物理量反问题的拟合效果

## 4.3 方程参数反问题实验

Navier-Stokes/Darcy 耦合模型物理公式中有一些参数如  $\kappa, \alpha$  需要通过实验确定，这就引出了关于未知方程参数的反问题，即已知全部或部分真解信息，反推出物理方程中的未知参数问题。本文的神经网络算法框架也能够尝试解决此类问题，具体做法为，将物理方程中的未知参数作为网络训练参数的一部分参与

优化算法中进行更新即可，实践中现有的深度学习编程框架<sup>1</sup>也不难做到这一点。

本节利用实验二的真解算例来测试神经网络算法解决方程参数反问题的能力。实验令已知条件为全部数值真解，而把耦合模型方程中的参数  $\kappa, \alpha$  设为未知量，加入到网络待训练参数中去。此时算法 2 cPINNs 中的损失函数变为公式(4.1)，注意这里还将  $\mathcal{L}_f(\theta_f; \mathcal{T}_{\Omega_f \times T}, \mathcal{T}_{\Gamma_f \times T}, \mathcal{T}_{\Omega_f \times \{t_0\}})$  与  $\mathcal{L}_p(\theta_p; \mathcal{T}_{\Omega_p \times T}, \mathcal{T}_{\Gamma_p \times T}, \mathcal{T}_{\Omega_p \times \{t_0\}})$  中的初边值条件损失修改为全部数值真解条件的损失（对应训练集为  $\mathcal{T}_{\Omega_f \times T}^*$ ,  $\mathcal{T}_{\Omega_p \times T}^*$ ,  $\mathcal{T}_{\Gamma \times T}^*$ ）。最后迁移后的 cPINNs 算法流程见图 4.4。

$$\begin{aligned} & \mathcal{L}_{total}(\theta_f, \theta_p, \kappa, \alpha; \mathcal{T}_{\Omega_f \times T}^*, \mathcal{T}_{\Omega_p \times T}^*, \mathcal{T}_{\Gamma \times T}^*) \\ &= \mathcal{L}_f(\theta_f; \mathcal{T}_{\Omega_f \times T}^*) + \mathcal{L}_p(\theta_p, \kappa; \mathcal{T}_{\Omega_p \times T}^*) + \mathcal{L}_{\Gamma}(\theta_f, \theta_p, \alpha; \mathcal{T}_{\Gamma \times T}^*) \end{aligned} \quad (4.1)$$

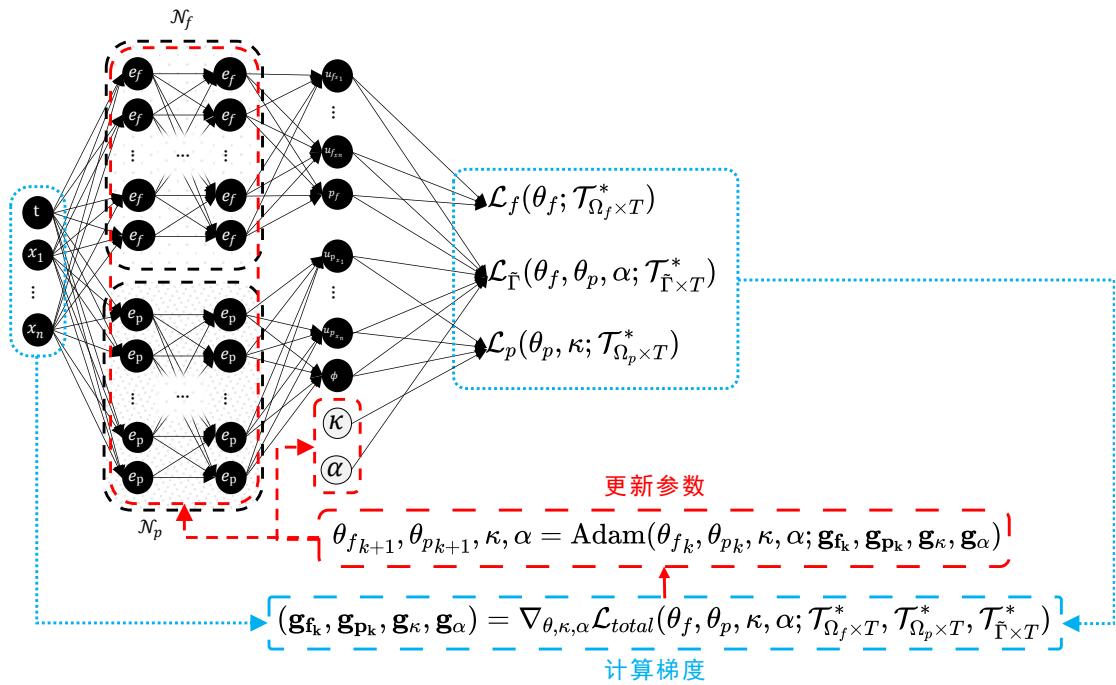


图 4.4: cPINNs 算法流程图示

物理方程的其他参数设置以及训练的网络结构等参数配置与上节相同。拟合结果见表 4.3，可以看出算法对各个参数的拟合效果都很好。

<sup>1</sup>本文算法的实现主要基于 pytorch 编程库

表 4.3: 方程参数反问题实验结果

真解		拟合值	
$\kappa$	$\alpha$	$\kappa$	$\alpha$
1.0	1.0	$0.99989 \times 10^0$	$1.00001 \times 10^0$
$1 \times 10^{-1}$	1.0	$0.99993 \times 10^{-1}$	$1.00003 \times 10^0$
$1 \times 10^{-2}$	1.0	$0.10032 \times 10^{-2}$	$1.00011 \times 10^0$
$1 \times 10^{-3}$	1.0	$0.10102 \times 10^{-3}$	$1.00105 \times 10^0$
$1 \times 10^{-6}$	1.0	$0.10379 \times 10^{-6}$	$1.03007 \times 10^0$

本页留白

## 第五章 总结与展望

本文基于 PINN 的思路提出了 Navier-Stokes/Darcy 耦合模型的神经网络求解算法 cPINNs。文章针对两个物理区域提出使用双通道的神经网络结构进行数值拟合，并利用交界面条件构造的损失函数使两个网络能够协同训练，达到求解的目的，同时利用万能表示定理一定程度上证明了损失函数构造的合理性，同时本文通过若干数值实验的测试，在一定程度上探索并验证了深度学习技术在耦合物理模型偏微分方程正反问题领域的求解能力。

在数值实验中文章还针对不同的网络层数与激活函数进行了简单探索，平衡了网络参数量与准确性的关系。本文提出的算法针对耦合模型的稳态形式以及不同的交界面条件，只需按第二章所描述同样的方法重新构造损失函数便可进行求解，说明算法具有一定的可扩展性。针对初始条件复杂的区域反问题和物理量反问题，算法只要对涉及到已知物理量真解的损失函数部分进行替换就可进行迁移；对于方程自身存在未知参数的反问题，将未知参数等同于网络自身参数的一部分即可求解。实验效果表明算法在这些反问题的解决上也拥有较高的潜力。

受时间与技术所限，文章测试的网络结构配置数量有限，也没有对更多的激活函数或网络结构如卷积神经网络（CNN），循环神经网络（RNN）等进行深入探索，此外训练算法的设置更多来源于实验中的经验，而严谨地对不同训练算法对网络拟合效果影响进行探究也是以后可以关注的方向之一。另一方面，相信本文算法对于更多更复杂的区域或方程模型数值求解问题上也会拥有潜力，值得未来进行探索。

本页留白

## 参考文献

- [1] BEAVERS G S, JOSEPH D D. Boundary conditions at a naturally permeable wall [J]. *Journal of fluid mechanics*, 1967, 30(1): 197-207.
- [2] HORNIK K. Approximation capabilities of multilayer feedforward networks[J]. *Neural networks*, 1991, 4(2): 251-257.
- [3] HORNIK K, STINCHCOMBE M, WHITE H. Multilayer feedforward networks are universal approximators[J]. *Neural networks*, 1989, 2(5): 359-366.
- [4] HORNIK K, STINCHCOMBE M, WHITE H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks[J]. *Neural networks*, 1990, 3(5): 551-560.
- [5] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks[C]//Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010: 249-256.
- [6] HE K, ZHANG X, REN S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1026-1034.
- [7] ADAMS R A, FOURNIER J J. Sobolev Spaces (Pure and applied mathematics; v. 140)[M]. Elsevier, 2003.
- [8] SAFFMAN P G. On the boundary condition at the surface of a porous medium[J]. *Studies in applied mathematics*, 1971, 50(2): 93-101.
- [9] RAISSI M, YAZDANI A, KARNIADAKIS G E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations[J]. *Science*, 2020, 367(6481): 1026-1030.

本页留白

## 致谢

这篇论文能够顺利完成离不开我的导师的辛勤教学与耐心指导。

XXX  
于华东师范大学  
2022 年 10 月